Azure pour les architectes

Troisième édition

Créer des applications sécurisées, évolutives et à haute disponibilité sur le Cloud

Ritesh Modi, Jack Lee et Rithin Skaria

Packt
www.packt.com

Azure pour les architectes

Troisième édition

Créer des applications sécurisées, évolutives et à haute disponibilité sur le Cloud

Ritesh Modi, Jack Lee et Rithin Skaria

Packt›

Azure pour les architectes Troisième édition

Auteurs : Ritesh Modi, Jack Lee et Rithin Skaria

Réviseurs techniques : Melony Qin et Sanjeev Kumar

Directeurs de la rédaction : Aditya Datar et Afzal Shaikh

Rédactrice responsable des commandes : Shrilekha Inani

Éditeurs de production : Ganesh Bhadwalkar et Deepak Chavan

Comité de rédaction : Vishal Bodwani, Ben Renow-Clarke, Edward Doxey, Joanne Lovell, Arijit Sarkar et Dominic Shakeshaft

Première édition : octobre 2017

Deuxième édition : janvier 2019

Troisième édition : juin 2020

Référence de la production : 3260620

ISBN : 978-1-83921-586-5

Publié par Packt Publishing Ltd.

Livery Place, 35 Livery Street

Birmingham B3 2PB, UK

Table des matières

Chapitre 2 : Disponibilité, évolutivité et surveillance de la solution Azure 23

Chapitre 5 : Conception des stratégies, des verrous et des balises pour les déploiements Azure 145

Chapitre 6 : Gestion des coûts des solutions Azure 167

Chapitre 8 : Architecture d'applications sécurisées sur Azure 225

Chapitre 12 : Solutions d'événements de Big Data Azure 389

Chapitre 13 : Intégration de DevOps Azure 421

Chapitre 17 : Concevoir des solutions IoT 575

Préface

À propos de

Cette section présente brièvement les auteurs, ce que couvre ce livre, les compétences techniques dont vous aurez besoin pour commencer, ainsi que le matériel et les logiciels requis pour concevoir des solutions avec Azure.

À propos de Azure pour les architectes, troisième édition

Grâce à sa prise en charge de la haute disponibilité, de l'évolutivité, de la sécurité, des performances et de la récupération d'urgence, Azure a été largement adopté pour créer et déployer différents types d'applications en toute simplicité. Mise à jour selon les derniers événements, cette troisième édition de *Azure pour les architectes* vous aide à vous familiariser avec les concepts fondamentaux de la conception d'une architecture sans serveur, y compris les conteneurs, les déploiements Kubernetes et les solutions Big Data. Vous apprendrez à concevoir des solutions telles que les fonctions sans serveur, vous découvrirez des modèles de déploiement pour les conteneurs et Kubernetes, et vous découvrirez le traitement Big Data à grande échelle à l'aide de Spark et Databricks. À mesure que vous avancerez, vous allez implémenter DevOps à l'aide d'Azure DevOps, travailler avec des solutions intelligentes à l'aide d'Azure Cognitive Services et intégrer la sécurité, la haute disponibilité et l'évolutivité dans chaque solution. Enfin, vous approfondirez les concepts de sécurité Azure, tels que OAuth, OpenConnect et les identités gérées.

Lorsque vous atteindrez la fin de ce livre, vous aurez acquis la confiance nécessaire pour concevoir des solutions Azure intelligentes basées sur des conteneurs et des fonctions sans serveur.

À propos des auteurs

Ride Modi est un ancien spécialiste de la haute technologie Microsoft. Il est reconnu en tant que directeur régional Microsoft pour ses contributions aux produits, services et communautés Microsoft. Il est architecte du Cloud, auteur de publications, conférencier et leader, reconnu pour ses contributions dans les datacenters, Azure, Kubernetes, les blockchains, les services cognitifs, le DevOps, l'intelligence artificielle et l'automatisation. Il est l'auteur de huit livres.

Ritesh s'est exprimé lors de nombreuses conférences nationales et internationales et est un auteur publié dans le magazine MSDN. Il possède plus de 10 ans d'expérience dans la création et le déploiement de solutions d'entreprise pour les clients et détient plus de 25 certifications techniques. Durant son temps libre, il aime écrire, jouer avec sa fille, regarder des films et apprendre de nouvelles technologies. Il vit actuellement à Hyderabad, en Inde. Suivez-le sur Twitter : **@automationnext**.

Jack Lee est un consultant senior certifié Azure et un responsable de la pratique Azure, passionné par le développement logiciel, le Cloud et les innovations DevOps. Jack a été reconnu comme MVP Microsoft pour ses contributions à la communauté technologique. Il a livré des présentations à plusieurs groupes d'utilisateurs et conférences, y compris le Global Azure Bootcamp de Microsoft Canada. Jack est un mentor et un juge expérimenté en hackathons. Il est également président d'un groupe d'utilisateurs qui se concentre sur Azure, le DevOps et le développement de logiciels. Il

est le co-auteur de *Analyse des données du cloud avec Microsoft Azure*, publié par Packt Publishing. Suivez Jack sur Twitter : **@jlee_consulting**.

Rithin Skaria est un spécialiste de l'open source comptant plus de 7 ans d'expérience dans la gestion de charges de travail open source dans Azure, AWS et OpenStack. Il travaille actuellement chez Microsoft et participe à plusieurs activités de la communauté open source réalisées au sein de Microsoft. Il est un formateur Microsoft certifié, ingénieur et administrateur de la fondation Linux, développeur et administrateur d'applications Kubernetes, ainsi qu'administrateur OpenStack certifié. Il détient également quatre certifications Azure dans ces domaines (architecture de solution, administration Azure, DevOps et sécurité). Il détient aussi une certification pour l'administration d'Office 365. Il a joué un rôle essentiel dans plusieurs déploiements open source, ainsi que dans l'administration et la migration de ces charges de travail vers le cloud. Il a également co-écrit *Linux sur Azure pratique*, publié par Packt Publishing. Retrouvez-le sur LinkedIn : **@rithin-skaria**.

À propos des réviseurs

Melony Qin est une femme travaillant dans les matières STEM. Actuellement responsable de programme chez Microsoft, elle est membre de l'**Association for Computing Machinery** (**ACM**) et du **Project Management Institute** (**PMI**). Elle a contribué à l'informatique sans serveur, au traitement du Big Data, au DevOps, à l'intelligence artificielle, au Machine Learning et à l'IoT avec Microsoft Azure. Elle détient toutes les certifications Azure (à la fois en applications, en infrastructure, en données et en pistes d'IA), ainsi que les titres de **Certified Kubernetes Administrator** (**CKA**) et de **Certified Kubernetes Application Developer** (**CKAD**). Elle travaille principalement sur ses contributions au **logiciel open source** (**OSS**), à DevOps, à Kubernetes, au sans serveur, à l'analyse de Big Data et à l'Internet des objets sur Microsoft Azure dans la communauté. Elle est l'auteure et co-auteure de deux livres, *Infrastructure Microsoft Azure* et *L'atelier Kubernetes*, tous deux publiés par Packt Publishing. Vous pouvez la contacter sur Twitter : **@MelonyQ**.

Sanjeev Kumar est un architecte de solutions Cloud pour SAP sur Azure chez Microsoft. Il travaille actuellement à Zurich, en Suisse. Il exploite la technologie SAP depuis plus de 19 ans. Il travaille avec les technologies de Cloud public depuis environ 8 ans, et s'est concentré sur Microsoft Azure les 2 dernières années.

Dans son rôle consultatif sur l'architecture Cloud SAP sur Azure, Sanjeev Kumar a travaillé avec un certain nombre des meilleurs services financiers et entreprises de fabrication au monde. Ses domaines d'intérêt comprennent l'architecture et la conception du Cloud pour aider les clients à migrer leurs systèmes SAP vers Azure et à adopter les meilleures pratiques Azure pour les déploiements SAP, en particulier en implémentant l'infrastructure en tant que code et le DevOps. Il a également travaillé

dans les domaines de la conteneurisation et des microservices à l'aide de Docker et d'Azure Kubernetes Service, du traitement de données de streaming à l'aide d'Apache Kafka et du développement d'applications full stack à l'aide de Node. js. Il a travaillé sur diverses initiatives de développement de produits couvrant IaaS, PaaS et SaaS. Il s'intéresse également aux sujets émergents de l'intelligence artificielle, du Machine Learning et du traitement et de l'analyse de données à grande échelle. Il écrit sur les sujets liés à SAP sur Azure, DevOps et l'infrastructure en tant que code sur LinkedIn, où vous pouvez le trouver sur ce compte : **@sanjeevkumarprofile**.

Objectifs d'apprentissage

À la fin de ce livre, vous serez en mesure :

- de comprendre les composants de la plateforme Azure Cloud ;
- d'utiliser des modèles de conception dans le Cloud ;
- d'utiliser les directives de sécurité d'entreprise dans le cadre de votre déploiement Azure ;
- de concevoir et d'implémenter des solutions sans serveur et d'intégration ;
- de créer des solutions de données efficaces sur Azure ;
- de comprendre les services de conteneur sur Azure.

Audience

Si vous êtes un architecte Cloud, un ingénieur DevOps ou un développeur qui souhaite en savoir plus sur les aspects architecturaux clés de la plateforme Cloud Azure, ce livre est fait pour vous. Une compréhension de base de la plateforme Cloud Azure vous aidera à saisir plus efficacement les concepts abordés dans ce livre.

Approche

Cet ouvrage aborde chaque sujet avec des explications pas à pas sur les concepts essentiels, des exemples pratiques et des questions d'auto-évaluation. En fournissant un équilibre entre la théorie et l'expérience pratique du travail grâce à des projets attrayants, cet ouvrage vous aidera à comprendre comment les architectes travaillent dans le monde réel.

Configurations matérielles requises

Pour une expérience optimale, nous recommandons la configuration suivante :

- Minimum 4 Go de RAM
- Minimum de 32 Go de mémoire libre

Configurations logicielles requises

- Visual Studio 2019
- Dernière version de Docker pour Windows
- Module AZ PowerShell 1.7 et versions supérieures
- Dernière version CLI Azure
- Abonnement Azure
- Windows Server 2016/2019
- Windows 10 dernière version - 64 bits

Conventions

Les mots en code dans le texte, les noms de tables de base de données, les noms de dossier, les noms de fichiers, les extensions de fichiers, les chemins d'accès, les URL factices et les entrées utilisateur sont indiquées comme suit :

La configuration DSC n'est toujours pas prise en charge dans Azure Automation. Elle est disponible sur certaines machines locales. Elle doit être chargée dans les configurations DSC Azure Automation. Azure Automation fournit une cmdlet de commande **`Import-AzureRMAutomationDscConfiguration`** pour importer la configuration dans Azure Automation :

```
Import-AzureRmAutomationDscConfiguration -SourcePath "C:\DSC\AA\DSCfiles\
ConfigureSiteOnIIS.ps1" -ResourceGroupName "omsauto" -AutomationAccountName
"datacenterautomation" -Published -Verbose
```

Téléchargez les ressources

Le code de ce livre est également hébergé sur GitHub à l'adresse suivante : https://github.com/PacktPublishing/Azure-for-Architects-Third-Edition.

Notre vaste catalogue de livres et de vidéos propose également d'autres codes, disponibles à l'adresse suivante : https://github.com/PacktPublishing Passez-les en revue !

1

Prise en main d'Azure

Des innovations technologiques surviennent régulièrement et bouleversent l'environnement ainsi que l'écosystème qui les entourent. Si nous revenons un peu en arrière, les années 70 et 80 furent l'ère des ordinateurs centraux. Ces machines très imposantes, qui occupaient quasiment des salles entières, réalisaient à elles seules toutes les tâches informatiques. Étant donné que la technologie était difficile à obtenir et fastidieuse à utiliser, de nombreuses entreprises commandaient des ordinateurs centraux un mois à l'avance, le délai requis pour disposer d'un ordinateur central opérationnel.

Par la suite, la demande pour l'informatique personnelle et l'Internet s'est accrue au début des années 1990. Les ordinateurs ont donc réduit leur encombrement et sont devenus beaucoup plus accessibles au grand public. Les innovations continues au niveau de l'informatique personnelle et d'Internet ont tous deux bouleversé l'industrie de l'informatique. De plus en plus d'individus ont commencé à faire l'acquisition d'ordinateurs de bureau, grâce auxquels ils pouvaient exécuter plusieurs programmes à la fois et se connecter à Internet. L'essor d'Internet a également favorisé le développement des déploiements client-serveur. Désormais quiconque doté d'une connexion à Internet dans le monde pouvait avoir accès à des serveurs centralisés hébergeant des applications et services. C'est à cette époque également que la technologie du serveur a gagné beaucoup d'importance ; Windows NT a été lancé durant cette période et a été rapidement suivi par Windows 2000 et Windows 2003 au début du siècle.

Les années 2000 ont été marquées par l'apparition et l'adoption de dispositifs portables, notamment les smartphones, équipés d'une multitude d'applications. Ces applications étaient capables de se connecter à des serveurs centralisés via Internet pour assurer la continuité des activités professionnelles. Les utilisateurs n'avaient plus besoin des navigateurs pour accomplir ces tâches ; tous les serveurs étaient auto-hébergés ou hébergés à l'aide d'un fournisseur de services, comme un **fournisseur d'accès Internet** (**FAI**).

Les utilisateurs n'avaient que peu de contrôle sur leurs serveurs. Plusieurs clients, ainsi que leurs déploiements, étaient intégrés à un même serveur, sans que ces derniers n'en aient connaissance.

Cependant, un autre événement a marqué le milieu et la fin des années 2000. Ce fut l'avènement du Cloud computing, ce qui a une nouvelle fois bouleversé toute l'industrie IT. Les utilisateurs ont mis du temps à adopter le Cloud et se sont tournés vers cette solution avec circonspection, d'une part car cette technologie en était encore à ses balbutiements et était donc très peu mature, et d'autre part en raison des différentes idées négatives que tout un chacun se faisait à son sujet.

Pour mieux comprendre les technologies révolutionnaires, nous aborderons les sujets suivants dans ce chapitre :

- Cloud computing
- **Infrastructure en tant que service** (**IaaS**), **Plateforme en tant que service** (**PaaS**) et **Logiciel en tant que service** (**SaaS**)
- Présentation d'Azure
- **Azure Resource Manager** (**ARM**)
- Virtualisation, conteneurs et Docker
- Interaction avec le Cloud intelligent

Cloud computing

Aujourd'hui, le cloud computing est l'une des technologies d'avenir les plus prometteuses et chaque entreprise ou organisation, quelle que soit sa taille, l'a adoptée dans le cadre de sa stratégie IT. Il serait aujourd'hui difficile de discuter de solutions pertinentes sur une stratégie informatique sans aborder la technologie du Cloud computing.

Le cloud computing, ou tout simplement le Cloud, fait référence à la disponibilité des ressources sur Internet. Ces ressources sont mises à la disposition d'utilisateurs sur Internet en tant que services. Par exemple, le stockage est disponible à la demande via Internet, afin de permettre aux utilisateurs de stocker leurs fichiers, documents, etc. Ici, le stockage est un service fourni par un fournisseur Cloud.

Un fournisseur Cloud est une entreprise ou un ensemble d'entreprises qui fournit des services Cloud à d'autres entreprises et aux consommateurs. Ils hébergent et gèrent des services pour le compte de l'utilisateur. Ils sont responsables de la mise en place des services et assurent leur intégrité. Il s'agit de grands datacenters, répartis dans le monde entier qui sont gérés par des fournisseurs de Cloud afin de satisfaire aux besoins informatiques des utilisateurs.

Les ressources Cloud se composent de services d'hébergement sur une infrastructure à la demande, tels que les infrastructures informatiques, les réseaux et les espaces de stockage. L'IaaS est la composante essentielle du Cloud.

Les avantages du Cloud Computing

Le Cloud n'a jamais enregistré des taux d'adoption si élevés et cette tendance n'a de cesse de s'affirmer en raison de plusieurs avantages, par exemple :

- **Modèle basé sur votre consommation :** les clients n'ont pas besoin d'acheter du matériel et des logiciels pour les ressources Cloud. Il n'y a pas de dépense en capital pour l'utilisation d'une ressource Cloud ; les clients paient simplement pour la durée durant laquelle ils utilisent ou réservent une ressource.
- **Accès global :** les ressources Cloud sont disponibles globalement via Internet. Les clients peuvent accéder à leurs ressources à la demande où qu'ils se trouvent.
- **Ressources illimitées :** la capacité de mise à l'échelle de la technologie Cloud est illimitée ; les clients peuvent mettre en service autant de ressources qu'ils le souhaitent, sans aucune contrainte. Ce modèle est également appelé l'extensibilité illimitée.
- **Services gérés :** le fournisseur Cloud fournit de nombreux services qu'il gère pour les clients. Cela élimine toute charge technique et financière pour le client.

Pourquoi le cloud computing ?

Pour comprendre la nécessité du Cloud computing, nous devons adopter le point de vue du secteur.

Flexibilité et agilité

Au lieu de créer une application monolithique volumineuse à l'aide d'une méthodologie de déploiement de type Big-Bang, les applications comprennent aujourd'hui des services plus petits en appliquant le paradigme des microservices. Les microservices permettent de créer des services de manière indépendante et autonome, qui peuvent évolués indépendamment, sans affecter l'intégralité de l'application. Ils permettent d'apporter des modifications en production avec une flexibilité et une agilité considérables, en vue d'accélérer les processus et de les améliorer. De nombreux microservices peuvent être réunis pour créer une application et fournir des solutions intégrées pour les clients. Ces microservices doivent être accessibles et présenter des points de terminaison bien définis pour l'intégration. Le nombre d'intégrations avec l'approche de microservices est très élevé par rapport aux applications monolithiques traditionnelles. Ces intégrations ajoutent de la complexité dans le développement et le déploiement des applications.

Rapidité, normalisation et cohérence

Il s'ensuit que la méthodologie des déploiements doit également être modifiée pour s'adapter aux besoins de ces services, à savoir des changements et des déploiements fréquents. Des processus permettant d'apporter des modifications de manière prévisible et cohérente doivent être collaborateurs pour effectuer les changements et les déploiements fréquents. Des processus agiles automatisés doivent être utilisés de sorte que les petites modifications puissent être déployées et testées de manière isolée.

Rester dans la course

Enfin, les cibles de déploiement doivent être redéfinies. Non seulement les cibles de déploiement doivent être pouvoir être créées en quelques secondes, mais l'environnement créé doit également être cohérent entre les versions, et présenter des binaires, des runtimes, des infrastructures et une configuration appropriés. Les machines virtuelles ont été utilisées avec des applications monolithiques. Toutefois, les microservices nécessitent plus d'agilité, de flexibilité et une option plus légère que les machines virtuelles. La technologie des conteneurs est le mécanisme privilégié des cibles de déploiement de ces services. Nous aborderons ce sujet plus en détail plus loin dans ce chapitre.

Évolutivité

Voici certains principes importants de l'utilisation des microservices : ils disposent d'une capacité de mise à l'échelle illimitée de manière isolée, d'une haute disponibilité mondiale, d'une récupération d'urgence avec un point de récupération quasi nul et des objectifs de temps. Ces qualités de microservices nécessitent une infrastructure capable d'évoluer de manière illimitée. Aucune contrainte ne doit être appliquée aux ressources. Même si c'est le cas, il est également important qu'une organisation ne paie pas les ressources à l'avance si celles-ci ne sont pas utilisées.

Rentabilité

Le principe fondamental du Cloud computing consiste à payer les ressources consommées et à les utiliser de manière optimale en augmentant et en diminuant automatiquement le nombre de ressources et la capacité. Ces exigences émergentes font du Cloud la plateforme de prédilection pour évoluer facilement, obtenir une haute disponibilité, résister aux sinistres, apporter des modifications facilement et réaliser des déploiements automatisés, prévisibles et cohérents de manière rentable.

Paradigmes de déploiement dans Azure

Il existe trois modèles de déploiement différents qui sont disponibles dans Azure ; les voici :

- IaaS
- PaaS
- SaaS

La différence entre ces trois modèles de déploiement réside dans le niveau de contrôle exercé par les clients via Azure. La *Figure 1.1* indique les différents niveaux de contrôle dans chacun de ces modèles de déploiement :

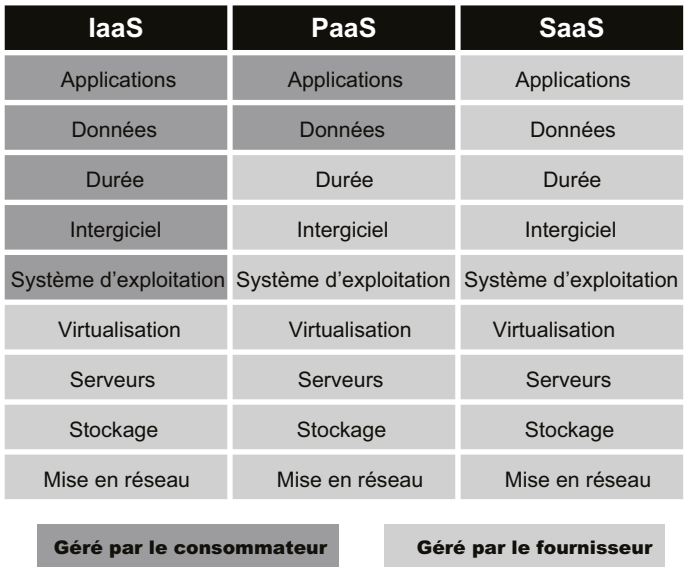


Figure 1.1 : services Cloud : IaaS, PaaS et SaaS

La *Figure 1.1* précédente met en évidence le fait que les clients ont davantage de contrôle lorsqu'ils utilisent les déploiements IaaS et que ce niveau de contrôle se réduit à mesure que nous passons des déploiements PaaS à SaaS.

IaaS

IaaS est un modèle de déploiement qui permet aux clients de mettre en service leur propre infrastructure sur Azure. Azure fournit plusieurs ressources d'infrastructure et les clients peuvent les mettre en service à la demande. Les clients sont responsables du maintien et de la gouvernance de leur propre infrastructure. Azure assurera la maintenance de l'infrastructure physique sur laquelle ces ressources d'infrastructure virtuelle sont hébergées. Dans le cadre de cette approche, les clients requièrent une gestion et des opérations actives dans l'environnement Azure.

PaaS

PaaS élimine les tâches de déploiement et de contrôle de l'infrastructure pour le client. Il s'agit d'une abstraction de niveau supérieur par rapport à IaaS. Dans cette approche, les clients apportent leur propre application, code et données, et les déploient sur la plateforme fournie par Azure. Ces plateformes sont gérées et régies par Azure et les clients sont les seuls responsables de leurs applications. Les clients effectuent uniquement des activités liées à leur déploiement d'applications. Ce modèle fournit des options plus rapides et plus simples pour le déploiement d'applications par rapport à IaaS.

SaaS

SaaS est une abstraction de niveau supérieur par rapport à PaaS. Dans cette approche, le logiciel et ses services sont disponibles pour la consommation des clients. Les clients n'apportent que leurs données à ces services et n'ont aucun contrôle sur ces derniers. Maintenant que nous avons évoqué les types de services dans Azure, intéressons-nous de plus près à Azure et découvrons-le en détail.

Présentation d'Azure

Azure fournit tous les avantages du Cloud tout en restant ouvert et flexible. Azure prend en charge une vaste gamme de systèmes d'exploitation, de langages, d'outils, de plateformes, d'utilitaires et de cadres. Par exemple, il prend en charge Linux et Windows, SQL Server, MySQL et PostgreSQL. Il prend en charge la plupart des langages de programmation, y compris C#, Python, Java, Node.js et Bash. Il prend en charge les bases de données NoSQL, telles que MongoDB et Cosmos DB, ainsi que des outils d'intégration continue, comme Jenkins et Azure DevOps Services (anciennement **Visual Studio Team Services** [**VSTS**]). Tout l'intérêt de cet écosystème est de permettre aux clients d'avoir la liberté de choisir leur propre langage, plateforme et système d'exploitation, base de données, stockage, et outils et utilitaires. Les clients ne doivent pas être limités du point de vue de la technologie, et doivent au contraire être en mesure de construire et de se concentrer sur leur solution métier, c'est pourquoi Azure leur fournit un environnement technologique de classe mondiale.

Azure est très compatible avec la pile technologique choisie par le client. Par exemple, Azure prend en charge tous les environnements de base de données courants (open source et commerciaux). Azure fournit des services PaaS Azure SQL, MySQL et Postgres. Il fournit l'écosystème Hadoop et propose HDInsight, un service PaaS Apache basé intégralement sur Hadoop. Il fournit également Hadoop sur des **machines virtuelles** (**VM**) Linux, pour les clients qui préfèrent l'approche de l'IaaS. Azure fournit également le service de cache Redis et prend en charge d'autres environnements de base de données populaires, comme Cassandra, Couchbase, Oracle et bien d'autres encore, en tant qu'implémentation IaaS.

Le nombre de services augmente chaque jour dans Azure et la liste de services la plus à jour se trouve à l'adresse https://azure.microsoft.com/services.

Azure fournit également un paradigme de Cloud Computing unique, à savoir le Cloud hybride. Le Cloud hybride fait référence à une stratégie de déploiement dans le cadre de laquelle un sous-ensemble de services est déployé sur un Cloud public, tandis que d'autres services sont déployés dans un Cloud privé ou un datacenter sur site. Une connexion **VPN** (**Virtual Private Network**) est établie entre les Cloud public et privé. Azure fournit aux clients la flexibilité nécessaire pour diviser et déployer leur charge de travail à la fois sur un Cloud public et sur un datacenter sur site.

Azure dispose de datacenters dans le monde entier et les combine en régions. Chaque région possède plusieurs datacenters, afin d'assurer une reprise après sinistre efficace et rapide, le cas échéant. À l'heure actuelle, il existe 58 régions à travers le monde. Cela procure aux clients la flexibilité nécessaire pour déployer leurs services sur les sites et régions de leur choix. Ils peuvent également combiner ces régions afin de déployer une solution résistante aux sinistres et proche de leurs clients.

> **Remarque**
>
> En Chine et en Allemagne, Azure Cloud Services diffère selon son utilisation, générale ou gouvernementale. Cela signifie que les services Cloud sont conservés dans des datacenters distincts.

Azure, un Cloud intelligent

Azure fournit l'infrastructure et les services nécessaires pour ingérer des milliards de transactions grâce au traitement informatique à grande échelle (hyperscale). Il fournit plusieurs pétaoctets de stockage pour les données, ainsi qu'une multitude de services interconnectés qui peuvent se transmettre des données. Compte tenu de ces capacités, les données peuvent être traitées pour générer des idées et des connaissances utiles. L'analyse des données permet d'obtenir plusieurs types d'information, comme suit :

- **Descriptive** : ce type d'analyse fournit des détails sur ce qui se passe ou sur ce qui s'est passé dans le passé.
- **Prédictive** : ce type d'analyse fournit des détails sur ce qui va se passer dans un avenir proche ou plus lointain.
- **Prescriptive** : ce type d'analyse fournit des détails sur ce qui doit être fait afin d'améliorer la situation actuelle ou d'éviter certains événements futurs.
- **Cognitive** : ce type d'analyse met en œuvre des actions déterminées dans le cadre de l'analyse prédictive, et ce de manière automatisée.

Dresser des conclusions à partir de données est une bonne chose, mais il est tout aussi important de prendre des mesures concrètes en conséquence. Azure fournit une plateforme enrichie permettant d'ingérer de vastes volumes de données, de les traiter et de les transformer, de les stocker et de les générer et de les afficher dans des tableaux de bord en temps réel. Il est également possible d'agir automatiquement sur ces informations. Ces services sont à la disposition de chaque client d'Azure et ils offrent un écosystème riche pour créer des solutions dont ils constitueront la base. Les entreprises créent de nombreuses applications et des services qui révolutionnent leurs secteurs, de par la mise à disposition de services intelligents Azure, qui peuvent être aisément combinés afin d'apporter une valeur significative aux clients finaux. Azure veille à ce que des services non viables d'un point de vue commercial, qui ne peuvent pas être implémentés par les petites et moyennes entreprises, puissent désormais être utilisés et déployés en quelques minutes.

Azure Resource Manager

Azure Resource Manager (**ARM**) est la plateforme technologique et le service d'orchestration de Microsoft qui relie tous les composants abordés précédemment. Il regroupe les fournisseurs de ressources, les ressources et les groupes de ressources Azure afin de former une plateforme Cloud cohésive. Il met les services Azure à la disposition des abonnements, assure la disponibilité des types de ressources aux groupes de ressources, rend les ressources et les API des ressources accessibles au portail et à d'autres clients et permet l'authentification de l'accès aux ressources. Il permet également d'activer des fonctionnalités, telles que le marquage, l'authentification, le **contrôle d'accès en fonction du rôle** (**RBAC**), le verrouillage des ressources, et l'application des stratégies pour les abonnements et les groupes de ressources. Il fournit également des fonctions de déploiement et de gestion à l'aide du portail Azure, d'Azure PowerShell et des outils d'**interface de ligne de commande** (**CLI**).

L'architecture ARM

L'architecture ARM et de ses composants est illustrée dans la *Figure 1.2*. Comme nous pouvons le constater, l'**abonnement Azure** comprend plusieurs groupes de ressources. Chaque groupe de ressources contient des instances de ressource qui sont créées à partir des types de ressources disponibles dans le fournisseur de ressources :

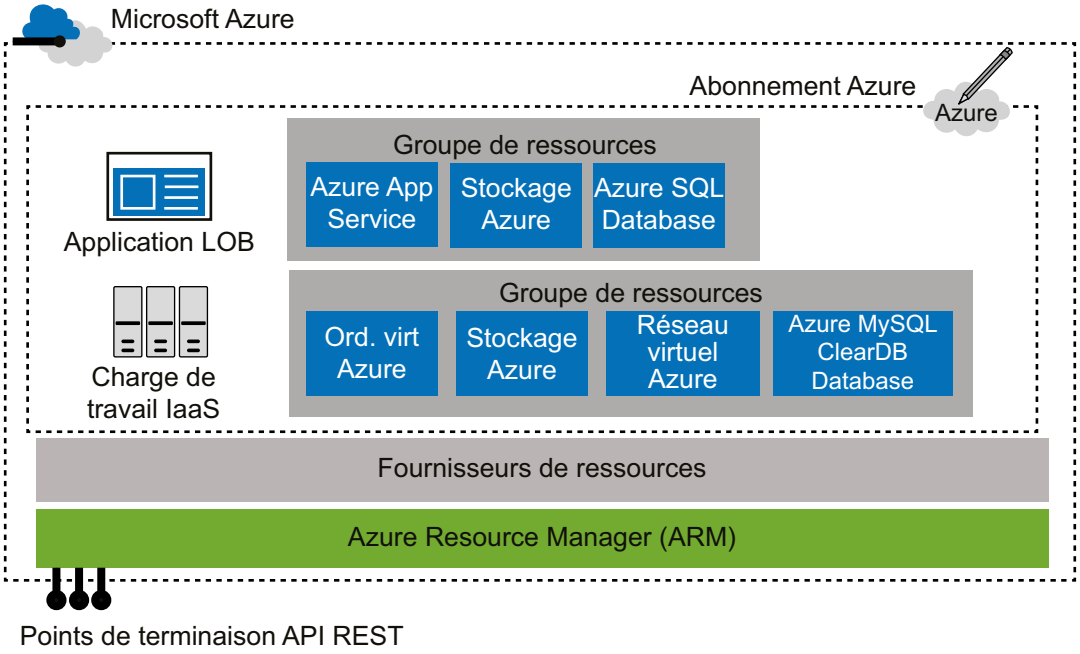


Figure 1.2 : l'architecture ARM

Pourquoi ARM ?

Avant ARM, l'infrastructure utilisée par Azure était connue sous le nom d'**Azure Service Manager** (**ASM**). Il est important de présenter brièvement cette infrastructure pour bien comprendre l'émergence d'ARM et l'obsolescence lente et progressive d'ASM.

Limitations d'ASM

ASM comporte des contraintes intrinsèques. Par exemple, les déploiements ASM sont lents et contraignants ; les opérations sont bloquées si une opération antérieure est déjà en cours. Voici certaines des limitations ASM :

- **Parallélisme** : le parallélisme constitue un défi dans ASM. Il n'est pas possible d'exécuter plusieurs opérations en parallèle. Dans ASM, les opérations sont linéaires et exécutées les unes après les autres. Si plusieurs transactions sont exécutées en même temps, des erreurs d'opération parallèles surviendront ou bien les transactions seront bloquées.
- **Ressources** : les ressources dans ASM sont fournies et gérées indépendamment les unes des autres, et il n'y a aucun lien entre elles. Tout regroupement des services et des ressources ou toute configuration globale n'est pas possible.
- **Services Cloud** : les services Cloud constituent les unités de déploiement dans ASM. Ils sont tributaires de groupes d'affinités et ne sont pas évolutifs, de par leur conception et l'architecture.

Il n'est pas possible d'assurer aux ressources des rôles distincts dans ASM. Les clients sont soit des administrateurs de service, soit des coadministrateurs dans l'abonnement. Ils disposent d'un contrôle complet sur les ressources ou bien n'ont accès à aucune d'elles. ASM ne fournit aucune prise en charge du déploiement. Les déploiements sont effectués manuellement ou via l'écriture d'une programmation procédurale dans .NET ou PowerShell. Les API d'ASM ne concordent pas entre les ressources.

Avantages d'ARM

ARM offre des avantages distincts d'ASM, comme suit :

- **Regroupement** : ARM permet le regroupement des ressources dans un conteneur logique. Ces ressources peuvent être gérées ensemble et suivent un cycle de vie commun, en tant que groupe. Cela facilite l'identification des ressources connexes.
- **Cycles de vie communs** : les ressources d'un groupe suivent le même cycle de vie. Ces ressources peuvent évoluer et être gérées ensemble, en tant qu'unité.
- **Contrôle d'accès en fonction du rôle (RBAC)** : différents rôles et autorisations peuvent être attribués aux ressources, pour personnaliser l'accès de chaque client. Les clients ne bénéficient que des autorisations qui leur ont été attribuées.

- **Prise en charge du déploiement** : ARM fournit une prise en charge du déploiement en termes de modèles permettant d'implémenter DevOps et l'**infrastructure en tant que code** (**IaC**). Ces déploiements sont plus rapides, cohérents et prévisibles.
- **Technologie supérieure** : le coût et la facturation des ressources peuvent être gérés en tant qu'unité. Chaque groupe de ressources peut fournir ses propres informations en termes d'utilisation et de coût.
- **Facilité de gestion** : ARM fournit des fonctions avancées comme la sécurité, la surveillance, l'audit et le marquage des fonctions, afin d'optimiser la gestion des ressources. Les ressources peuvent être interrogées grâce à des balises. Les balises fournissent également des informations sur le coût et la facturation sur les ressources qui leur sont associées.
- **Migration** : la migration et la mise à jour des ressources au sein des groupes de ressources et entre eux sont simplifiées.

Concepts ARM

Avec ARM, tout ce qui se trouve dans Azure est une ressource. Une machine virtuelle, les interfaces réseau, une adresse IP publique, les comptes de stockage, les réseaux virtuels sont tout autant d'exemples de ressources. ARM s'appuie sur des concepts liés aux fournisseurs de ressources et aux consommateurs de ressources. Azure fournit des ressources et des services par le biais de plusieurs fournisseurs de ressources, qui sont consommés et déployés dans des groupes.

Fournisseurs de ressources

Voici les services qui assurent la fourniture des types de ressources via ARM. Le concept principal de la solution ARM s'articule autour des fournisseurs de ressources. Ces fournisseurs sont des conteneurs de types de ressources. Les types de ressources sont regroupées en fournisseurs de ressources. Ces derniers sont responsables du déploiement et de la gestion des ressources. Par exemple, un type de ressource de machine virtuelle est fourni par un fournisseur de ressources appelé **Microsoft.Compute/virtualMachines**. Les opérations d'API **REST** (**Representational State Transfer**) sont versionnées pour les distinguer. Le nommage des versions se base sur leurs dates de lancement par Microsoft. Il est nécessaire qu'un fournisseur de ressource soit disponible dans un abonnement afin de déployer une ressource. Les fournisseurs de ressources ne sont pas tous disponibles pour un abonnement préconfiguré. Si une ressource n'est pas disponible dans l'abonnement, vous devez vous assurer que le fournisseur de ressources requis est disponible dans chaque région. Si celui-ci est disponible, le client peut s'inscrire explicitement dans l'abonnement.

Types de ressources

Les types de ressources constituent une véritable spécification de ressource définissant l'interface API publique et sa mise en œuvre. Ils implémentent les tâches et opérations prises en charge par la ressource. Tout comme les fournisseurs de ressources, la mise en œuvre interne des types de ressource évolue au fil du temps, lesquels disposent de plusieurs versions du schéma et de l'interface API publique. Les noms de version se basent sur la date de lancement par Microsoft de leur version d'évaluation ou de **disponibilité générale** (**DG**). Les types de ressources deviennent disponibles pour un abonnement une fois qu'un fournisseur de ressources y est inscrit. En outre, les types de ressources ne sont pas tous disponibles dans toutes les régions Azure. La disponibilité d'une ressource dépend de la disponibilité et de l'inscription d'un fournisseur de ressources dans une région Azure et doit prendre en charge la version d'API nécessaire à sa mise en service.

Groupes de ressources

Les groupes de ressources constituent des unités de déploiement dans ARM. Il s'agit de conteneurs regroupant plusieurs instances de ressources dans une limite de gestion et de sécurité. Un groupe de ressources est nommé de manière unique dans un abonnement. Les ressources peuvent être configurées sur différentes régions Azure mais appartiennent au même groupe de ressources. Les groupes de ressources fournissent des services supplémentaires à toutes les ressources qu'ils contiennent. Les groupes de ressources fournissent des services de métadonnées, telles que le balisage, qui permet la catégorisation des ressources, la stratégie de gestion des ressources, le contrôle RBAC, la protection des ressources contre les suppressions accidentelles ou les mises à jour, etc. Comme mentionné précédemment, ces groupes sont assortis d'une limite de sécurité et les utilisateurs qui n'ont pas accès à un groupe de ressources ne peuvent pas utiliser les ressources qu'il contient. Chaque instance de ressource doit faire partie d'un groupe de ressources, sinon elle ne peut pas être déployée.

Ressources et instances de ressource

Les ressources sont créées à partir de types de ressources et sont une instance d'un type de ressource. Une instance peut être unique à l'échelle mondiale ou au niveau d'un groupe de ressources. Le caractère unique est défini à la fois par le nom de la ressource et par son type. Si nous comparons cela avec des structures de programmation orientées objet, les instances de ressources peuvent être vues comme des objets, et les types de ressources, comme des classes. Les services sont consommés par les opérations prises en charge et mis en œuvre par les instances de la ressource. Le type de ressource définit les propriétés et chaque instance doit configurer des propriétés obligatoires lors de la mise en service d'une instance. Certains sont des propriétés obligatoires, tandis que d'autres sont facultatifs. Ils héritent de la sécurité et de la configuration de l'accès du groupe de ressources parent. Ces autorisations et attributions de rôles héritées peuvent être substituées pour chaque ressource. Une ressource peut être verrouillée de sorte que certaines de ses opérations puissent être bloquées et non disponibles aux rôles, utilisateurs et groupes, même si ces derniers y ont accès. Les ressources peuvent être balisées pour faciliter leur découverte et leur gestion.

Fonctions ARM

Voici quelques-unes des principales caractéristiques fournies par ARM :

- **Contrôle d'accès en fonction du rôle (RBAC)** : **Azure Active Directory** (**Azure AD**) authentifie les utilisateurs pour accéder aux abonnements, aux groupes de ressources et aux ressources. ARM implémente OAuth et RBAC au sein de la plateforme, ce qui permet l'autorisation et le contrôle des accès aux ressources, aux groupes de ressources et aux abonnements en fonction des rôles attribués à un utilisateur ou à un groupe. Une autorisation définit l'accès aux opérations sur une ressource. Ces autorisations peuvent autoriser ou refuser l'accès à la ressource. Une définition de rôle est une collection de ces autorisations. Les rôles cartographient les utilisateurs et groupes Azure AD sur les autorisations spécifiques. Les rôles sont ensuite affectés à une portée, qui peut être un individu, une collection de ressources, un groupe de ressources ou un abonnement. Les identités Azure AD (utilisateurs, groupes et principaux de service) ajoutées à un rôle ont accès à la ressource selon les autorisations définies dans le rôle. ARM fournit plusieurs rôles initiaux. Il fournit des rôles de système, tels que **propriétaire**, **contributeur** et **lecteur**. Il fournit également des rôles axés sur les ressources, tels que collaborateur SQL DB, contributeur et contributeur de machine virtuelle. ARM permet également la création de rôles personnalisés.
- **Balises** : les balises sont des paires nom-valeur qui ajoutent des informations et des métadonnées aux ressources. Les ressources et les groupes de ressources peuvent tous deux être étiquetés avec plusieurs balises. Les balises aident à classer les ressources pour faciliter leur détection et leur gestion. Les ressources peuvent être rapidement parcourues et identifiées. Les informations relatives à la facturation et aux coûts peuvent également être récupérées pour les ressources disposant de balises identiques. Bien que cette fonction soit fournie par ARM, un administrateur IT définit son usage et la taxonomie selon les ressources et les groupes de ressources. La taxonomie et les balises, par exemple, peuvent faire référence à des départements, l'usage des ressources, l'emplacement, les projets et tout autre critère jugé pertinent du point de vue des coûts, de l'utilisation, de la facturation et des recherches. Ces balises peuvent ensuite être appliquées à des ressources. Les balises définies au niveau du groupe de ressources ne sont pas héritées par leurs ressources.
- **Stratégies** : les stratégies personnalisées constituent une autre fonction de sécurité fournie par ARM. Des stratégies personnalisées peuvent être créées pour contrôler l'accès aux ressources. Les stratégies sont des conventions et des règles définies qui doivent être respectées lors des interactions avec les ressources et groupes de ressources. La définition d'une stratégie contient la négation explicite d'actions sur les ressources ou l'accès aux ressources. Par défaut, chaque accès est autorisé s'il n'est pas mentionné dans la définition de la stratégie. Ces définitions de stratégies sont assignées à la ressource, au groupe de ressources et à la portée d'abonnements. Il est important de noter que ces stratégies ne remplacent ni ne se substituent au contrôle RBAC. Elles viennent au contraire le compléter.

Les stratégies sont évaluées après l'authentification d'un utilisateur par Azure AD et son autorisation par le service RBAC. ARM fournit le langage de définition de stratégies basé sur JSON relatif à la définition des stratégies. Par exemple, une stratégie peut établir que toutes les ressources mises en services doivent être balisées, ou bien que celles-ci peuvent uniquement être mises en service sur des régions Azure spécifiques.

- **Verrous** : les abonnements, groupes de ressources et les ressources peuvent être verrouillées afin d'éviter toute suppression accidentelle ou mise à jour par un utilisateur authentifié. Les verrous appliqués à des niveaux plus élevés sont transmis aux ressources enfant. Les verrous appliqués au niveau de l'abonnement verrouillent également chaque groupe de ressources ainsi que les ressources en son sein.
- **Multi-région** : Azure fournit plusieurs régions pour la mise en service et l'hébergement des ressources. Avec ARM, les ressources peuvent être mises en service sur des sites différents tout en résidant dans le même groupe de ressources. Un groupe de ressources peut contenir des ressources issues de différentes régions.
- **Idempotence** : cette fonction assure la prévisibilité, la normalisation et la cohérence du déploiement des ressources, en veillant à ce que chaque déploiement maintienne le même état des ressources et la même configuration, quel que soit le nombre de fois où il est exécuté.
- **Extensible** : l'architecture ARM fournit une architecture extensible pour permettre la création et l'intégration de nouveaux fournisseurs de ressources et types de ressources dans la plateforme.

Virtualisation

La virtualisation a constitué une innovation majeure qui a complètement révolutionné notre perception des serveurs physiques. Elle fait référence à l'abstraction d'un objet physique en un objet logique.

La virtualisation des serveurs physiques a conduit à des serveurs virtuels, connus sous le nom de machines virtuelles. Ces machines virtuelles consomment et partagent un processeur physique, une mémoire, un stockage et d'autres matériels avec le serveur physique sur lequel elles sont hébergées. Cela permet une mise en service plus simple et plus rapide des environnements applicatifs sur demande, en plus d'une haute disponibilité et évolutivité à un coût réduit. Un seul serveur physique suffit pour héberger plusieurs machines virtuelles, chacune d'elles contenant son propre système d'exploitation et ses services d'hébergement.

Il n'était plus nécessaire d'acheter des serveurs physiques supplémentaires pour procéder au déploiement de nouvelles applications et services. Les serveurs physiques existants étaient suffisants pour héberger plusieurs machines virtuelles. En outre, dans le cadre de la rationalisation, de nombreux serveurs physiques étaient regroupés ensemble grâce à la virtualisation.

Chaque machine virtuelle contient l'ensemble du système d'exploitation et est complètement isolée des autres machines virtuelles, y compris des ordinateurs hôtes physiques. Même si une machine virtuelle utilise le matériel fourni par le serveur physique de l'hôte, elle dispose d'un contrôle total sur ses ressources assignées et sur son environnement. Ces machines virtuelles peuvent être hébergées sur un réseau, tel qu'un serveur physique doté de sa propre identité.

Azure peut créer des machines virtuelles Linux et Windows en quelques minutes. Microsoft fournit ses propres images, ainsi que des images de ses partenaires et de la communauté ; les utilisateurs peuvent également fournir leurs propres images. Les machines virtuelles sont créées à l'aide de ces images.

Conteneurs

Les conteneurs sont également une technologie de virtualisation ; toutefois, ils ne permettent pas de virtualiser un serveur. Au lieu de cela, un conteneur est une virtualisation au niveau du système d'exploitation. Par conséquent, les conteneurs partagent le noyau du système d'exploitation (fourni par l'hôte), entre eux et avec l'hôte. Plusieurs conteneurs s'exécutant sur un hôte (physique ou virtuel) partagent le noyau du système d'exploitation hôte. Les conteneurs réutilisent le noyau de l'hôte au lieu de disposer chacun d'un noyau dédié.

Les conteneurs sont complètement isolés de leur hôte ou d'autres conteneurs en cours d'exécution sur l'hôte. Les conteneurs Windows utilisent des pilotes de filtre pour le stockage Windows et l'isolation de session afin d'isoler les services du système d'exploitation tels que le système de fichiers, le registre, les processus et les réseaux. Il en va de même pour les conteneurs Linux fonctionnant sur les hôtes Linux. Les conteneurs Linux utilisent l'espace de noms Linux, les groupes de contrôle et le système de fichiers Union pour virtualiser les systèmes d'exploitation hôtes.

Le conteneur se comporte ainsi comme s'il disposait de ressources et d'un système d'exploitation entièrement neufs et intacts. Une telle configuration offre de nombreux avantages, notamment :

- Les conteneurs sont mis en service plus rapidement que les machines virtuelles. La plupart des services du système d'exploitation sont fournis par le système d'exploitation hôte.
- Les conteneurs sont légers et nécessitent moins de ressources informatiques que les machines virtuelles. Aucuns frais induit des ressources du système d'exploitation n'est nécessaire avec les conteneurs.
- Les conteneurs sont beaucoup plus petits que les machines virtuelles.
- Les conteneurs permettent de résoudre les problèmes liés à la gestion de plusieurs dépendances d'applications, le tout de manière automatisée, simple et intuitive.
- Les conteneurs fournissent une infrastructure permettant de définir toutes les dépendances de l'application en un seul et même endroit.

Les conteneurs font partie intégrante de Windows Server 2016 et de Windows 10 ; cependant, ils sont gérés et accessibles à l'aide d'un client Docker et démon Docker. Les conteneurs peuvent être créés sur Azure avec une référence Windows Server 2016 sous forme d'image. Chaque conteneur possède un seul processus principal qui doit être fonctionnel afin de permettre au conteneur d'exister. Un conteneur s'interrompra une fois que ce processus aura pris fin. En outre, un conteneur peut s'exécuter en mode interactif ou en mode distinct, comme un service :

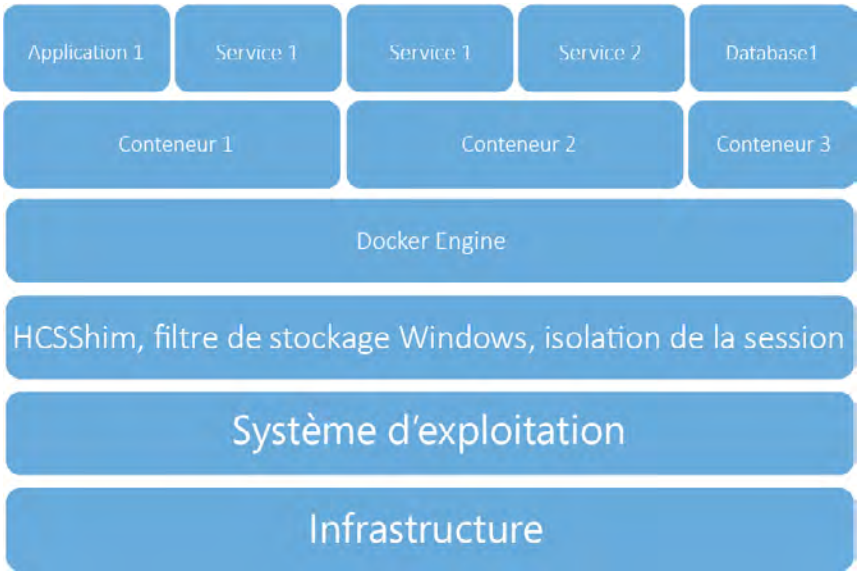


Figure 1.3 : architecture de conteneurs

La Figure 1.3 présente toutes les couches techniques qui activent les conteneurs. La couche la plus basse fournit l'infrastructure de base en matière de réseau, stockage, équilibreurs de charge et cartes réseau. La couche de calcul se trouve au sommet de l'infrastructure, et consiste soit en un serveur physique ou en des serveurs à la fois physiques et virtuels, sur un serveur physique. Cette couche contient le système d'exploitation capable d'héberger les conteneurs. Le système d'exploitation fournit le pilote d'exécution que les couches supérieures utilisent pour appeler le code du noyau et les objets afin d'exécuter des conteneurs. Microsoft a créé **Host Container System Shim** (**HCSShim**) pour gérer et créer des conteneurs, et utilise les pilotes de filtre pour le stockage Windows afin de gérer des images et des fichiers.

L'isolation de l'environnement du conteneur est activée pour la session Windows. Windows Server 2016 et Nano Server fournissent le système d'exploitation, activent les fonctions de conteneur et exécutent le client Docker au niveau de l'utilisateur ainsi que Docker Engine. Docker Engine utilise les services de HCSShim, les pilotes de filtre de stockage et des sessions pour créer plusieurs conteneurs sur le serveur, chacun contenant un service, une application ou une base de données.

Docker

Docker fournit des fonctionnalités de gestion pour les conteneurs Windows. Il comprend les deux exécutables suivants :

- Le démon Docker
- Le client Docker

Le démon Docker constitue la force motrice de la gestion des conteneurs. Il s'agit d'un service Windows capable de gérer toutes les activités relatives aux conteneurs sur l'hôte. Le client Docker interagit avec le démon Docker et est responsable de la capture et de la transmission d'entrées dans le démon Docker. Le démon Docker fournit l'exécution, les bibliothèques, les pilotes graphiques et le moteur capable de créer, gérer et surveiller les conteneurs et les images sur le serveur hôte. Il permet de créer des images personnalisées qui sont utilisées pour développer et livrer des applications sur plusieurs environnements.

Interaction avec le Cloud intelligent

Azure fournit plusieurs moyens de se connecter, d'automatiser et d'interagir avec le Cloud intelligent. Toutes ces méthodes nécessitent que les utilisateurs soient authentifiés à l'aide d'informations d'identification valides avant de pouvoir être utilisées. Les différentes façons de se connecter à Azure sont les suivantes :

- Le portail Azure
- PowerShell
- L'interface de ligne de commande Azure
- L'API REST Azure

Le portail Azure

Le portail Azure est un excellent point de départ. Le portail Azure permet aux utilisateurs de se connecter et de créer et gérer les ressources Azure manuellement. Le portail fournit une interface utilisateur intuitive et conviviale via le navigateur. Le portail Azure fournit un moyen simple d'accéder aux ressources à l'aide de **panneaux**. Les panneaux affichent toutes les propriétés d'une ressource, les fichiers journaux, le coût, sa relation avec les autres ressources, les balises, les options de sécurité, etc. L'intégralité du déploiement du Cloud peut être géré via le portail.

PowerShell

PowerShell est un shell de ligne de commande basé sur les objets et un langage de script utilisé pour l'administration, la configuration et la gestion d'infrastructure et d'environnements. Il repose sur .NET Framework et fournit des fonctions d'automatisation. PowerShell est devenu un véritable élément de première classe pour les administrateurs IT et les développeurs de fonctions d'automatisation qui souhaitent gérer et contrôler l'environnement Windows. Aujourd'hui, presque tous les environnements Windows et Linux peuvent être gérés par PowerShell. En réalité, quasiment tous les aspects d'Azure peuvent également être gérés par PowerShell. Azure fournit une prise en charge de PowerShell enrichie. Il fournit un module PowerShell pour chaque fournisseur de ressources contenant des centaines de cmdlets de commande. Les utilisateurs peuvent utiliser ces cmdlets de commande dans leurs scripts pour automatiser les interactions avec Azure. Le module Azure PowerShell est disponible via le programme d'installation de la plateforme Web et via la **bibliothèque PowerShell**. Windows Server 2016 et Windows 10 assurent la gestion des packages et des modules **`PowerShellGet`** pour fournir des téléchargements simples et rapides et permettre l'installation de modules PowerShellGet dans la bibliothèque de PowerShell. Le module **`PowerShellGet`** fournit la cmdlet de commande **`Install-Module`** pour télécharger et installer des modules sur le système.

L'installation d'un module consiste simplement à copier des fichiers de module dans des emplacements du module bien spécifiques, ce qui peut être effectué de la manière suivante :

```
Import-module PowerShellGet

Install-Module -Name az -verbose
```

La commande **`Import-module`** importe un module et ses fonctions associées au sein de la portée d'exécution actuelle et **`Install-Module`** permet d'installer des modules.

L'interface de ligne de commande Azure

Azure fournit également Azure CLI 2.0, qui peut être déployé sur Linux, Windows, ainsi que sur les systèmes d'exploitation macOS. L'interface Azure CLI 2.0 est un nouvel utilitaire de ligne de commande qui permet de gérer les ressources Azure. Azure CLI 2.0 est optimisé pour gérer et administrer les ressources Azure de la ligne de commande et créer des scripts d'automatisation qui fonctionnent de concours avec ARM. L'interface de ligne de commande peut être utilisée pour exécuter des commandes à l'aide du shell Bash ou de la ligne de commande Windows. L'interface CLI Azure est très répandue parmi les utilisateurs non Windows, car elle permet d'interagir avec Azure sur Linux et macOS. La procédure d'installation d'Azure CLI 2.0 est disponible à l'adresse https://docs.microsoft.com/cli/azure/install-azure-cli?view=azure-cli-latest.

L'API REST Azure

Toutes les ressources Azure sont exposées aux utilisateurs via des points de terminaison REST. Les API REST sont des points de terminaison de service qui implémentent les opérations HTTP (ou méthodes), offrant un accès aux ressources du service pour **créer, récupérer**, **mettre à jour** ou **supprimer** (**CRUD**) ces dernières. Les utilisateurs peuvent consommer ces API afin de créer et de gérer les ressources. En effet, les mécanismes de l'interface CLI et PowerShell utilisent ces API REST en interne pour interagir avec les ressources sur Azure.

Modèles ARM

Dans une section précédente, nous avons observé les fonctions de déploiement telles que les fonctions multiservices, multi-régions, extensibles et idempotentes fournies par ARM. Les modèles ARM sont principalement des moyens de mettre en service des ressources dans ARM. Les modèles ARM assurent la mise en œuvre des fonctions de déploiement d'ARM.

Les modèles ARM fournissent un modèle déclaratif grâce auquel des ressources, leur configuration, les scripts et les extensions sont spécifiés. Les modèles ARM s'appuient sur le format **JSON** (**(JavaScript Object Notation**). Ils utilisent la syntaxe et les conventions JSON pour déclarer et configurer les ressources. Les fichiers JSON sont basés sur des fichiers texte lisibles et conviviaux pour les utilisateurs.

Ils peuvent être stockés dans un référentiel de code source et disposent d'un contrôle de version. Ils permettent également de représenter l'IAC qui peut être utilisé pour mettre en service les ressources dans un groupe de ressource Azure plusieurs fois et ce de manière prévisible et uniforme. Un modèle a besoin d'un groupe de ressources pour le déploiement. Il peut uniquement être déployé sur un groupe de ressources, qui doit exister avant l'exécution d'un déploiement du modèle. Un modèle n'est pas capable de créer un groupe de ressources.

Les modèles fournissent la flexibilité nécessaire afin qu'une solution soit générique et modulaire de par sa conception et sa mise en œuvre. Les modèles permettent d'accepter les paramètres des utilisateurs, de déclarer des variables internes, de définir des dépendances entre les ressources, de relier les ressources au sein de groupes de ressources identiques ou différents et d'exécuter d'autres modèles. Ils fournissent également des expressions de type langage script et des fonctions qui les rendent personnalisables et dynamiques lors de l'exécution.

Déploiements

PowerShell permet les deux modes suivants pour le déploiement de modèles :

- **Progressif** : dans le cadre d'un déploiement progressif, les ressources déclarées sont ajoutées dans le modèle qui n'existe pas dans un groupe de ressources, tandis que les ressources d'un groupe de ressources ne faisant pas partie d'une définition de modèle demeurent inchangées, tout comme celles qui se trouvent dans un groupe de ressources existant à la fois dans le modèle et dans le groupe de ressource, avec un état de configuration identique.
- **Exhaustif** : en revanche, dans le cadre d'un déploiement exhaustif les ressources déclarées dans un modèle sont ajoutées au groupe de ressources, les ressources qui n'existent pas dans le modèle sont supprimées du groupe de ressources, et les ressources qui existent à la fois dans le groupe de ressources et dans le modèle avec le même état de configuration demeurent inchangées.

Résumé

Le Cloud est un paradigme relativement nouveau, qui en est encore à ses débuts. De nombreuses innovations et capacités vont voir le jour au fil du temps. Azure est l'un des plus grands fournisseurs de services Cloud actuellement et fournit des capacités enrichies via des déploiements de solutions IaaS, PaaS, SaaS et hybrides. En effet, Azure Stack, une implémentation de Cloud privé Microsoft, sera disponible prochainement. Cette solution offrira sur un Cloud privé les mêmes fonctions qu'un Cloud public. Ces derniers pourront tous deux être connectés et fonctionner ensemble de manière transparente.

La mise en route avec Azure est très simple, toutefois les développeurs et les architectes peuvent se retrouver coincés si la conception et l'architecture de leurs solutions ne sont pas appropriées. Ce manuel a pour but de fournir des conseils et des directives afin de garantir une architecture des solutions appropriées, grâce aux services et aux ressources adéquats. Chaque service d'Azure est une ressource. Il est important de comprendre comment ces ressources sont organisées et gérées dans Azure. Ce chapitre fournit un contexte autour d'ARM et des groupes, qui constituent les cadres de base qui fournissent des éléments constitutifs pour les ressources. ARM fournit un ensemble de services pour les ressources qui permettent d'assurer l'uniformité, la normalisation et la cohérence de leur gestion. Ces services, tels que RBAC, les balises, les stratégies et les verrous sont à la disposition de chaque ressource et fournisseur de ressources. Azure fournit également des fonctions d'automatisation enrichies afin d'automatiser et d'interagir avec les ressources. Des outils tels que PowerShell, les modèles ARM et l'interface CLI Azure peuvent être intégrés à des pipelines de lancement et dans le cadre d'un déploiement et d'une livraison continus. Les utilisateurs peuvent se connecter à Azure à partir d'environnements hétérogènes à l'aide de ces outils d'automatisation.

Le chapitre suivant aborde certaines points architecturaux qui permettent de résoudre les problèmes courants liés au déploiement basé sur le Cloud et de s'assurer que les applications sont sécurisées, disponibles, évolutives et simples à gérer à long terme.

2

Disponibilité, évolutivité et surveillance de la solution Azure

Les préoccupations architecturales, telles que la haute disponibilité et l'évolutivité, sont quelques-unes des plus grandes priorités de n'importe quel architecte. Cela est fréquent dans de nombreux projets et solutions. Toutefois, cela devient encore plus important lors du déploiement d'applications sur le Cloud, en raison de la complexité intrinsèque. La plupart du temps, la complexité ne provient pas de l'application, mais des choix disponibles en termes de ressources similaires sur le Cloud. L'autre problème complexe qui découle du Cloud est la disponibilité constante de nouvelles fonctions. Ces nouvelles fonctions peuvent quasiment rendre les décisions d'un architecte totalement redondantes en rétrospective.

Dans ce chapitre, nous allons examiner, du point de vue d'un architecte, le déploiement d'applications hautement disponibles et évolutives sur Azure.

Azure est une plateforme mature fournissant de nombreuses options permettant d'implémenter la haute disponibilité et l'évolutivité à plusieurs niveaux. Un architecte doit absolument connaître ces dernières, y compris les différences entre elles et les coûts impliqués, et enfin, il doit choisir une solution appropriée qui répondra aux exigences de solution. Il n'existe pas de solution universelle, toutefois il n'y en a qu'une qui soit adaptée pour chaque projet.

L'une des plus grandes priorités des organisations est de s'assurer que leurs applications et systèmes sont disponibles pour les utilisateurs, chaque fois que ces derniers ont besoin d'y recourir. Ils souhaitent que leurs applications soient opérationnelles, fonctionnelles et qu'elles restent disponibles pour leurs clients même en cas d'imprévu. Cela est le thème de ce chapitre, la haute disponibilité. *Garder la lumière allumée*, voici une métaphore courante pour désigner la haute disponibilité. Le maintien d'une haute disponibilité pour les applications n'est pas une tâche aisée et les organisations doivent consacrer beaucoup de temps, d'énergie, de ressources et d'argent pour y parvenir. En outre, il existe toujours le risque que l'implémentation d'une organisation ne produise pas les résultats souhaités. Azure fournit de nombreuses fonctions de haute disponibilité pour les **machines virtuelles** (**MV**) et les **plateformes en tant que service** (**PaaS**). Dans ce chapitre, nous aborderons les fonctions d'architecture et de conception offertes par Azure afin d'assurer la haute disponibilité de l'exécution des applications et services.

Dans ce chapitre, nous allons aborder les thèmes suivants :

- Haute disponibilité
- Haute disponibilité d'Azure
- Considérations architecturales relatives à la haute disponibilité
- Évolutivité
- Mises à niveau et maintenance

Haute disponibilité

La haute disponibilité représente l'une des principales exigences techniques non fonctionnelles pour tout service digne de ce nom et son déploiement. La haute disponibilité désigne la fonction d'un service ou d'une application qui maintient ce dernier ou cette dernière opérationnel(le) en permanence, atteignant ou dépassant le **contrat de niveau de service** (**SLA**) fixé. Certains accords SLA sont garantis aux utilisateurs selon le type d'un service. Le service doit être disponible à la consommation sur la base de son SLA. Par exemple, un SLA peut stipuler une disponibilité de 99 % pour une application durant toute l'année. Cela signifie que le service doit être disponible à la consommation par les utilisateurs pendant 361,35 jours. Le SLA n'est donc pas respecté si sa disponibilité n'est pas garantie durant cette période. La plupart des applications stratégiques définissent leurs SLA de disponibilité élevée avec 99,999 % par an. Cela signifie que l'application doit être opérationnelle et disponible tout au long de l'année, mais peut être indisponible pendant 5,2 heures uniquement. Si le temps d'arrêt est supérieur à cette limite, vous pouvez bénéficier d'un crédit, lequel sera calculé en fonction du pourcentage de disponibilité total.

Il est important de noter ici que la haute disponibilité est définie en fonction du temps, à savoir par année, mois, semaine ou une combinaison de ces durées.

Un service ou une application se compose de plusieurs composants, et ceux-ci sont déployés sur des niveaux et des couches séparés. En outre, celui-ci ou celle-ci est déployé(e) sur un **système d'exploitation** (**SE**) et hébergé(e) sur une machine physique ou virtuelle. Il ou elle consomme des services réseau et de stockage à des fins diverses. Il ou elle peut même être tributaire de systèmes externes. Afin que ces services ou applications soient hautement disponibles, il est essentiel que les réseaux, le stockage, le système d'exploitation, les machines physiques ou virtuelles et chaque composant de l'application soit conçu en gardant à l'esprit les exigences du SLA en termes de haute disponibilité. Un processus de cycle de vie de l'application doit être défini et utilisé afin de s'assurer que la haute disponibilité est prise en compte dès la phase de planification de l'application jusqu'à son lancement. Cela implique également l'introduction de la redondance. Des ressources redondantes doivent être incluses dans l'architecture globale de l'application et du déploiement, afin de s'assurer qu'en cas d'échec de l'un de ces composants, l'élément restant prend la relève et satisfait aux requêtes du client.

Voici quelques-uns des principaux facteurs qui influent sur la haute disponibilité d'une application :

- Maintenance planifiée
- Maintenance non planifiée
- Architecture de déploiement de l'application

Nous étudierons chacun de ces facteurs dans les sections suivantes. Examinons de plus près la façon dont la haute disponibilité est assurée pour les déploiements dans Azure.

Haute disponibilité d'Azure

Il est très difficile de parvenir à une haute disponibilité conforme aux exigences du SLA auquel elle est assujettie. Azure fournit un grand nombre de fonctions qui favorisent la haute disponibilité, des applications depuis le système d'exploitation hôte et invité aux applications utilisant PaaS. Les architectes peuvent utiliser ces fonctions pour parvenir à la haute disponibilité de leurs applications à l'aide d'une configuration, au lieu de développer ces applications en partant de zéro ou en s'appuyant sur des outils tiers.

Dans cette section, nous allons voir les fonctions et caractéristiques offertes par Azure afin de rendre les applications hautement disponibles. Avant d'entrer dans les détails relatifs à l'architecture et à la configuration, il est important de comprendre les concepts ayant trait à la haute disponibilité Azure.

Concepts

Les concepts fondamentaux fournis par Azure pour atteindre la haute disponibilité sont les suivants :

- Ensembles de disponibilité
- Domaine d'erreur
- Domaine de mise à jour
- Zones de disponibilité

Comme vous le savez, il est très important de tenir compte de la haute disponibilité des solutions dans leur conception. Les charges de travail peuvent être stratégiques et nécessiter une architecture hautement disponible. Étudions de plus près chacun des concepts de haute disponibilité dans Azure. Commençons par les ensembles de disponibilité.

Ensembles de disponibilité

La haute disponibilité dans Azure est principalement assurée par la redondance. La redondance signifie qu'il existe plus d'une instance de la ressource du même type, qui prendra le relais en cas de défaillance de la ressource principale. Toutefois, le fait de disposer de plusieurs ressources similaires ne rend pas une application hautement disponible. Par exemple, plusieurs machines virtuelles peuvent être provisionnées dans un abonnement, mais la simple présence de plusieurs machines virtuelles ne les rend pas hautement disponibles. Azure fournit une ressource connue sous le nom de groupe à haute disponibilité qui, si elle est associée à plusieurs MV, peut être hautement disponible. Deux machines virtuelles minimum doivent être hébergées dans un groupe à haute disponibilité afin de rendre ces dernières hautement disponibles. Toutes les machines virtuelles du groupe à haute disponibilité deviennent hautement disponibles car elles sont placées sur des racks physiques distincts dans le datacenter Azure. Pendant les mises à jour, ces machines virtuelles sont mises à jour une à une, et non pas toutes en même temps. Les groupes à haute disponibilité fournissent un domaine d'erreur et un domaine de mise à jour, ce qui sera abordé dans la section suivante. En bref, les ensembles de disponibilité fournissent une redondance au niveau du datacenter similaire au stockage redondant localement.

Il est important de noter que les ensembles de disponibilité fournissent une haute disponibilité dans un datacenter. Si l'intégralité d'un datacenter est en panne, la disponibilité de l'application sera affectée. Afin de vous assurer que les applications sont toujours disponibles même lorsqu'un datacenter tombe en panne, Azure a introduit une nouvelle fonction connue sous le nom de zones de disponibilité, que nous allons aborder par la suite

Selon la liste des concepts fondamentaux, le concept suivant est le domaine d'erreur. Le domaine d'erreur est souvent noté par l'acronyme FD. La section suivante explique en quoi consiste un FD et sa pertinence lors de la conception de solutions hautement disponibles.

Domaine d'erreur

Les domaines d'erreur (**FD**) sont un groupe de machines virtuelles qui partagent une source d'alimentation et un commutateur réseau communs. Quand une machine virtuelle est mise en service et affectée à un ensemble de disponibilité, elle est hébergée au sein d'un domaine d'erreur. Chaque ensemble de disponibilité dispose de deux ou trois domaines d'erreur par défaut, selon les régions Azure. Certaines en fournissent deux, tandis que d'autres fournissent trois domaines d'erreur dans un ensemble de disponibilité. Les domaines d'erreur ne sont pas configurables par les utilisateurs.

Lorsque plusieurs machines virtuelles sont créées, elles sont placées sur des domaines d'erreur distincts. Si le nombre de machines virtuelles est supérieur à la quantité de domaines d'erreur, les machines virtuelles supplémentaires sont placées sur des domaines d'erreur existants. Par exemple, s'il y a cinq machines virtuelles, il y aura des domaines d'erreur hébergés sur plusieurs machines virtuelles.

Les domaines d'erreur sont reliés à des armoires physiques du datacenter Azure. Les domaines d'erreur offrent une haute disponibilité lors de tout temps d'arrêt imprévu en raison de faille matérielle, d'alimentation ou réseau. Étant donné que chaque machine virtuelle est placée sur un rack différent avec un équipement, une alimentation et un réseau distincts, d'autres machines virtuelles continuent de s'exécuter en cas de défaillance d'un rack.

Le prochain dans la liste est le domaine de mise à jour.

Domaine de mise à jour

Un domaine d'erreur prend en charge les temps d'arrêt imprévus tandis qu'un domaine de mise à jour gère les interruptions de maintenance planifiée. Chaque machine virtuelle est également affectée à un domaine de mise à jour et toutes les machines virtuelles de ce domaine de mise à jour redémarrent ensemble. Il peut y avoir jusqu'à 20 domaines de mise à jour dans un ensemble de disponibilité. Les domaines de mise à jour ne sont pas configurables par les utilisateurs. Lorsque plusieurs machines virtuelles sont créées, elles sont placées sur des domaines de mise à jour distincts. Si plus de 20 machines virtuelles sont configurées sur un ensemble de disponibilité, celles-ci sont placées de manière alternée sur ces domaines de mise à jour. Les domaines de mise à jour prennent en charge la maintenance planifiée. Vous pouvez consulter les détails de la maintenance planifiée et définir les alertes dans **Service Health** dans le portail Azure**.**

La section suivante porte sur les zones de disponibilité.

Zones de disponibilité

Il s'agit d'un concept relativement nouveau, introduit dans Azure et très similaire à la redondance de zone, que nous avons abordée lorsque nous avons discuté des comptes de stockage. Les zones de disponibilité offrent une haute disponibilité au sein d'une région en plaçant des instances de machines virtuelles sur des datacenters distincts au sein d'une même région. Les zones de disponibilité sont applicables à de nombreuses ressources dans Azure, notamment aux machines virtuelles, aux disques gérés, aux groupes de machines virtuelles identiques (VMSS) et aux équilibreurs de charge. La liste complète des ressources prises en charge par les zones de disponibilité est disponible à l'adresse https://docs.microsoft.com/azure/availability-zones/az-overview#services-that-support-availability-zones. L'impossibilité de configurer la disponibilité à travers les zones a été pendant longtemps une lacune dans Azure, qui a été finalement corrigée avec l'introduction des zones de disponibilité.

Chaque région Azure comprend plusieurs datacenters dotés d'une puissance, d'une mise en réseau et d'un refroidissement indépendants. Certaines régions ont plus de datacenters que d'autres. Ces datacenters dans la région sont connus sous le nom de zones. Pour assurer la résilience, il existe un minimum de trois zones distinctes dans toutes les régions activées. Le déploiement des machines virtuelles dans une zone de disponibilité garantit que celles-ci se trouvent dans différents datacenters et sur divers racks et réseaux. Ces datacenters dans une région se rapportent à des réseaux à grande vitesse et il n'y a aucun décalage de communication entre ces machines virtuelles. La *Figure 2.1* illustre la configuration des zones de disponibilité dans une région :

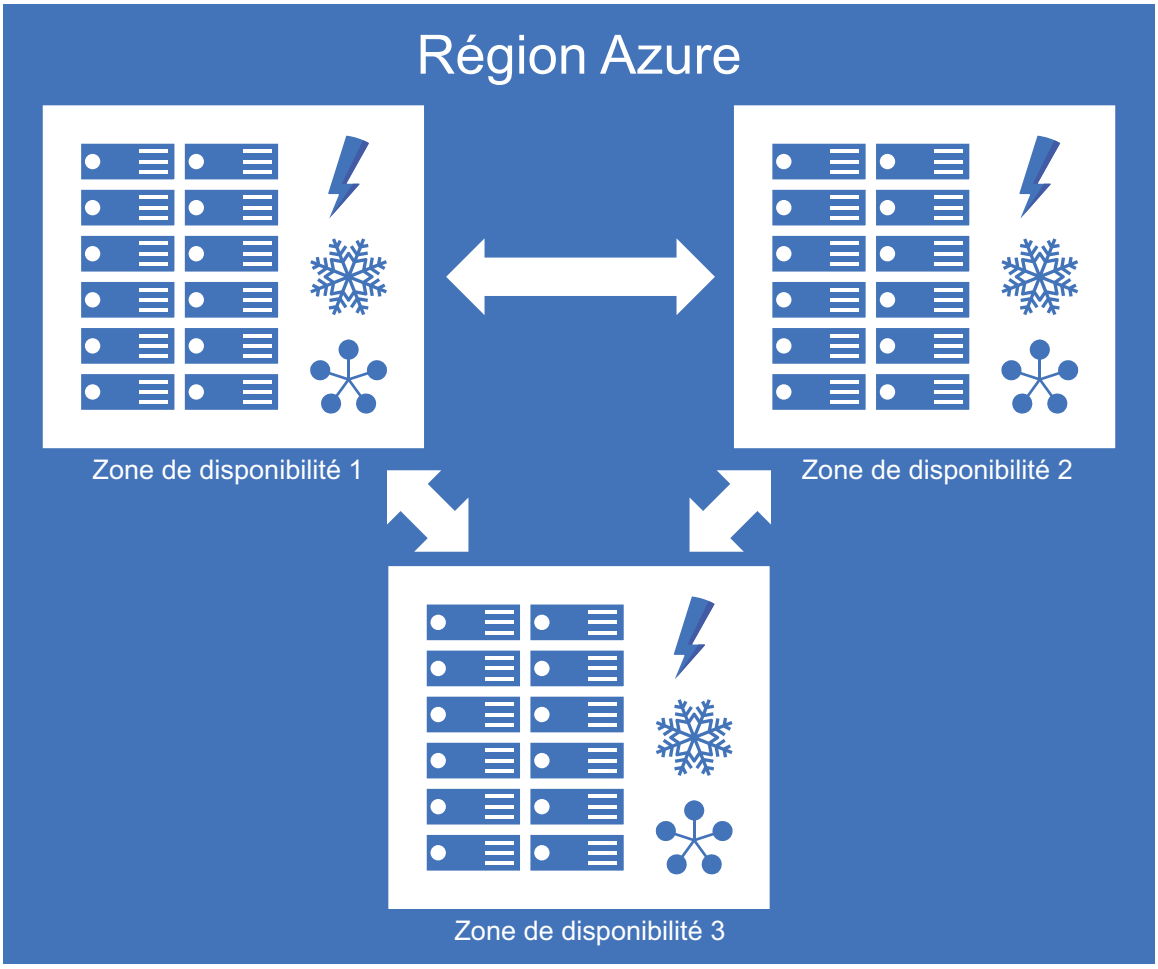


Figure 2.1 : zones de disponibilité dans une région

Pour plus d'informations sur les zones de disponibilité, accédez à l'adresse https://docs.microsoft.com/azure/availability-zones/az-overview.

Les services redondants interzone répliquent vos applications et données dans les zones de disponibilité afin de les protéger contre les points de défaillance uniques.

Si une application nécessite une disponibilité plus élevée et si vous souhaitez vous assurer de sa disponibilité même si une région Azure entière est en panne, le prochain échelon du parcours vers la disponibilité est la fonction Traffic Manager, qui sera abordée plus loin dans ce chapitre. Étudions désormais le concept d'équilibrage de charge pour les machines virtuelles dans Azure.

Équilibrage de charge

L'équilibrage de la charge, comme son nom l'indique, désigne le processus d'équilibrage de la charge entre les machines virtuelles et les applications. Avec une machine virtuelle, il n'est pas nécessaire de mettre en place un équilibreur de charge car la totalité de la charge se trouve sur une seule machine virtuelle, et aucune autre ne partage la charge. Cependant, si plusieurs machines virtuelles contiennent le même service et la même application, il est possible de répartir la charge entre celles-ci, grâce à l'équilibrage de charge. Azure fournit quelques ressources permettant d'activer l'équilibrage de la charge :

- **Équilibreurs de charge** : un équilibreur de charge Azure permet de concevoir des solutions avec la haute disponibilité. Dans la pile **Transmission Control Protocol** (**TCP**), il s'agit d'un équilibreur de charge de niveau de transport couche 4. Il s'agit d'un équilibreur de charge de couche 4 qui répartit le trafic entrant entre les instances intègres des services définies dans un ensemble d'équilibrage de charge. Les équilibreurs de charge de niveau 4 fonctionnent au niveau du transport et disposent des informations de réseau telles que les adresses IP et les ports, afin de décider de la cible pour la requête entrante. Les équilibreurs des charges seront présentés plus en détail plus loin dans ce chapitre.
- **Passerelles Application Gateway** : les passerelles Azure Application Gateway garantissent une haute disponibilité pour vos applications. Il s'agit d'équilibreurs de charge de couche 7 qui répartissent le trafic entrant sur les instances intègres des services. Les équilibreurs de charge de niveau 7 peuvent fonctionner au niveau de l'application et contiennent des informations sur l'application telles que les cookies, HTTP, HTTPS et les sessions pour la requête entrante. Les passerelles Azure Application Gateway sont présentées plus en détail plus loin dans ce chapitre. Les passerelles d'application sont également utilisées lors du déploiement d'Azure Kubernetes Service, spécifiquement pour les scénarios dans lesquels le trafic d'entrée à partir d'Internet doit être acheminé vers les services Kubernetes dans le cluster.
- **Azure Front Door** : Azure Front Door est très similaire aux passerelles d'application, sans toutefois fonctionner au niveau de la région ou du datacenter. Au lieu de cela, il permet d'acheminer les demandes à travers toutes les régions du monde. Il est doté du même ensemble de fonctionnalités que celui des passerelles d'application, tout en fonctionnant au niveau mondial. Il fournit également un pare-feu d'application Web pour le filtrage des demandes et fournit d'autres protections liées à la sécurité. Parmi les fonctions proposées figurent l'affinité de session, la terminaison TLS et le routage basé sur les l'URL.
- **Traffic Manager** : Traffic Manager contribue à l'acheminement des requêtes au niveau mondial dans plusieurs régions en fonction de l'intégrité et de la disponibilité des points de terminaison régionaux. Pour y parvenir, il utilise des entrées de redirection DNS. Cette solution est hautement résiliente et n'affecte pas les services lors des pannes de région.

Nous venons de découvrir les méthodes et les services qui permettent d'équilibrer les charges. Le moment est donc venu d'expliquer comment rendre les machines virtuelles hautement disponibles.

Haute disponibilité des machines virtuelles

Les machines virtuelles fournissent des capacités de calcul. Elles fournissent une puissance de traitement et d'hébergement pour les applications et les services. Si l'application est déployée sur une seule machine virtuelle et que celle-ci tombe en panne, l'application ne sera pas disponible. Si l'application se compose de plusieurs niveaux et que chaque niveau est déployé dans sa propre instance unique d'une machine virtuelle, même un temps d'arrêt d'une seule instance de machine virtuelle peut rendre l'application non disponible. Azure essaie de rendre même les instances de machine virtuelle uniques hautement disponibles 99,9 % du temps, en particulier si ces machines virtuelles à instance unique utilisent un stockage Premium pour leurs disques. Azure fournit un SLA plus élevé pour les machines virtuelles qui sont regroupées dans un ensemble de disponibilité. Il fournit un contrat SLA de 99,95 % pour les machines virtuelles qui font partie d'un ensemble de disponibilité avec deux machines virtuelles ou plus. Le SLA est de 99,99 % si les machines virtuelles sont placées dans des zones de disponibilité. La section suivante porte sur la haute disponibilité des ressources de traitement.

Haute disponibilité des capacités de traitement

Les applications qui exigent une haute disponibilité doivent être déployées sur plusieurs machines virtuelles appartenant au même ensemble de disponibilité. Si les applications sont composées de plusieurs niveaux, chacun d'eux doit disposer d'un groupe de machines virtuelles sur leur ensemble de disponibilité dédié. En bref, si une application comporte trois niveaux, il devrait y avoir trois ensembles de disponibilité et au minimum six machines virtuelles (deux dans chaque ensemble de disponibilité) pour rendre l'ensemble de l'application hautement disponible.

Comment Azure fournit-il donc des SLA ainsi que la haute disponibilité pour les machines virtuelles se trouvant dans un ensemble de disponibilité, chacun de ces derniers regroupant plusieurs machines virtuelles ? C'est une question que vous vous posez peut-être.

C'est à ce moment-là que les concepts que nous avons étudiés précédemment sont utiles : les domaines d'erreur et les domaines de mise à jour. Lorsqu'Azure détecte plusieurs machines virtuelles dans un ensemble de disponibilité, il place ces machines virtuelles sur un domaine d'erreur distinct. En d'autres termes, ces machines virtuelles sont placées sur des armoires physiques distinctes. Cela garantit qu'au moins une machine virtuelle reste disponible, même en cas de panne matérielle, d'armoire ou d'alimentation. Un ensemble de disponibilité compte deux ou trois domaines d'erreur et selon le nombre de machines virtuelles dans un ensemble de disponibilité, celles-ci sont placées sur des domaines d'erreur distincts ou fonctionnent de manière répétée par alternance. Cela garantit que la haute disponibilité n'est pas affectée en cas de panne de l'armoire.

Azure place également ces machines virtuelles sur un domaine de mise à jour séparé. En d'autres termes, Azure marque ces machines virtuelles en interne de telle sorte qu'elles sont corrigées et mises à jour l'une après l'autre ; ainsi tout redémarrage dans un domaine de mise à jour n'a aucune incidence sur la disponibilité de l'application. Cela garantit que la haute disponibilité n'est pas affectée par la maintenance de la machine virtuelle ou de l'hôte. Il est important de noter qu'Azure n'est pas responsable de la maintenance au niveau du système d'exploitation et de l'application.

En plaçant les machines virtuelles dans des domaines d'erreur et de mise à jour distincts, Azure garantit que ces dernières ne seront pas toutes indisponibles en même temps et pourront honorer les requêtes même en cas de maintenance ou d'indisponibilité des composants physiques :

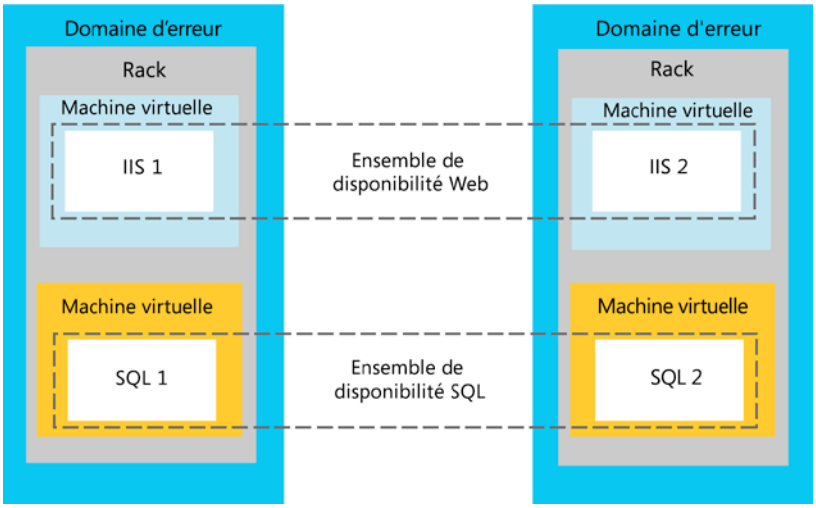


Figure 2.2 : distribution des machines virtuelles dans les domaines de mise à jour et d'erreur

La Figure 2.2 illustre quatre machines virtuelles (deux ont un serveur **Internet Information Services (IIS)** et les deux autres ont des serveurs SQL Server). Les machines virtuelles IIS et SQL font partie d'ensembles de disponibilité. Les machines virtuelles IIS et SQL se trouvent dans un domaine d'erreur distinct et sur différentes armoires dans le datacenter. Elles se trouvent également dans des domaines de mise à jour différents.

La *Figure 2.3* illustre la relation entre les domaines d'erreur et de mise à jour :

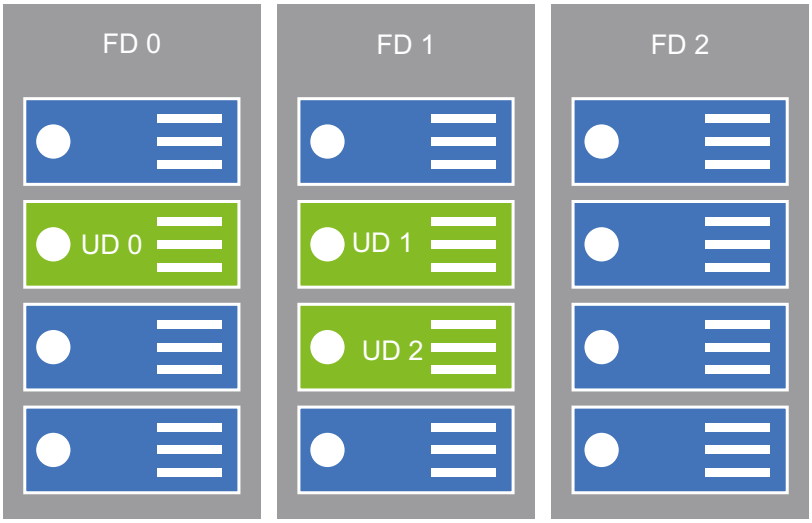


Figure 2.3 : disposition des domaines de mise à jour et d'erreur dans un ensemble de disponibilité

Jusqu'à présent, nous avons discuté de la haute disponibilité pour les ressources de traitement. Dans la section suivante, vous découvrirez comment la haute disponibilité peut être mise en œuvre pour les options PaaS.

Plateformes haute disponibilité

Azure propose un grand nombre de nouvelles fonctions en matière de haute disponibilité pour les PaaS. Certaines d'entre elles sont présentées ici :

- Conteneurs dans les services d'application
- Groupes Azure Container Instances
- Azure Kubernetes Service
- D'autres orchestrateurs de conteneurs, tels que DC/OS et Swarm

L'autre plateforme importante qui garantit une haute disponibilité est **Service Fabric**. Les orchestrateurs de conteneurs et Service Fabric qui incluent Kubernetes garantissent que le nombre d'instances d'application souhaité est toujours opérationnel dans un environnement. Cela signifie que même si l'une des instances est défaillante dans l'environnement, l'orchestrateur en sera informé grâce à une surveillance active et activera une nouvelle instance sur un autre nœud, conservant ainsi le nombre idéal et escompté d'instances. Tout cela est réalisé sans aucune interférence manuelle ou automatisée de l'administrateur.

Bien que Service Fabric permette à tout type d'application de devenir hautement disponible, les orchestrateurs tels que Kubernetes, DC/OS et Swarm sont spécifiques aux conteneurs. En outre, il est important de comprendre que ces plateformes fournissent des fonctions qui aident à déployer des mises à jour propagées plutôt qu'une mise à jour massive qui pourrait affecter la disponibilité de l'application.

Lorsque nous avons discuté de la haute disponibilité des machines virtuelles, nous avons évoqué brièvement l'équilibrage de charge. Examinons ce concept de plus près pour mieux comprendre son fonctionnement dans Azure.

Équilibreurs de charge dans Azure

Azure fournit deux ressources qui fonctionnent comme des équilibreurs de charge. Il fournit un équilibreur de charge de niveau 4 qui fonctionne au niveau de la couche de transport au sein de la pile TCP OSI et un équilibreur de charge de niveau 7 (passerelle d'application) qui fonctionne aux niveaux de l'application et de la session.

Bien que les passerelles d'application et les équilibreurs de charge fournissent des fonctions de base afin d'équilibrer la charge, ils servent des objectifs différents. Dans certains cas, le déploiement des passerelles Application Gateway s'avère plus utile que les équilibreurs de charge.

Une passerelle Application Gateway fournit les fonctions suivantes, qui ne sont pas disponibles dans les équilibreurs de charge Azure :

- **Pare-feu des applications Web** : il s'agit d'un pare-feu supplémentaire qui vient s'ajouter à celui du système d'exploitation et qui a la capacité de lire les messages entrants. Cela permet d'identifier toute attaque Web courante et de s'en prémunir, par exemple l'injection de code SQL, les attaques de script multisite et les détournements de session.
- **L'affinité de session par cookie** : les équilibreurs de charge répartissent le trafic entrant parmi les instances de services qui sont en bonne santé et relativement libres. Une requête peut être desservie par n'importe quelle instance de service. Certaines applications nécessitent toutefois des caractéristiques avancées dans le cadre desquelles toutes les requêtes subséquentes à la première requête doivent être traitées par la même instance de service. Cela s'appelle l'affinité de session basée sur les cookies. Une passerelle Application Gateway fournit une affinité de session basée sur les cookies afin de maintenir une session utilisateur sur la même instance de service à l'aide de cookies.
- **Le déchargement Secure Sockets Layer (SSL)** : le chiffrement et le déchiffrement des données de la requête et de la réponse sont interprétés par SSL et sont généralement une opération coûteuse. Idéalement, les serveurs Web doivent consacrer leurs ressources au traitement et à la diffusion des requêtes plutôt qu'au chiffrement et au déchiffrement du trafic. Le déchargement SSL permet de transférer ce processus de chiffrement depuis le serveur Web à l'équilibreur de charge, ce qui permet de consacrer davantage de ressources aux serveurs Web qui traitent les requêtes des utilisateurs. La requête de l'utilisateur est cryptée, mais est déchiffrée au niveau de la passerelle Application Gateway plutôt que par le serveur Web. La requête de la passerelle Application Gateway vers le serveur Web n'est pas chiffrée.
- **SSL end-to-end** : bien que le déchargement SSL soit une fonction intéressante pour certaines applications, d'autres applications sécurisées critiques doivent procéder à un chiffrement et à un déchiffrement SSL, même si le trafic passe par des équilibreurs de charge. Une passerelle Application Gateway peut être configurée pour procéder également à un chiffrement SSL end-to-end.
- **Routage de contenu par URL** : les passerelles Application Gateway sont également utiles pour rediriger le trafic vers des serveurs différents, selon le contenu de l'URL des requêtes entrantes. Cela permet d'héberger plusieurs services aux côtés d'autres applications.

Équilibreurs de charge Azure

Un équilibreur de charge Azure répartit le trafic entrant en fonction des informations de transport dont il dispose. Il s'appuie sur les fonctions suivantes :

- Une adresse IP d'origine
- Une adresse IP cible
- Un numéro de port d'origine
- Un numéro de port cible
- Un type de protocole : TCP ou HTTP

Un équilibreur de charge Azure peut être un équilibreur de charge privé ou public. Un équilibreur de charge privé peut être utilisé pour distribuer le trafic au sein du réseau interne. Étant donné qu'il s'agit d'opérations internes, aucune adresse IP publique n'est affectée et il est impossible d'y accéder sur Internet. Un équilibreur de charge public dispose d'une adresse IP publique externe qui lui est associée et est accessible via Internet. La *Figure 2.4* illustre l'intégration d'équilibreurs de charge internes (privés) et publics à une solution unique pour gérer le trafic interne et externe, respectivement :

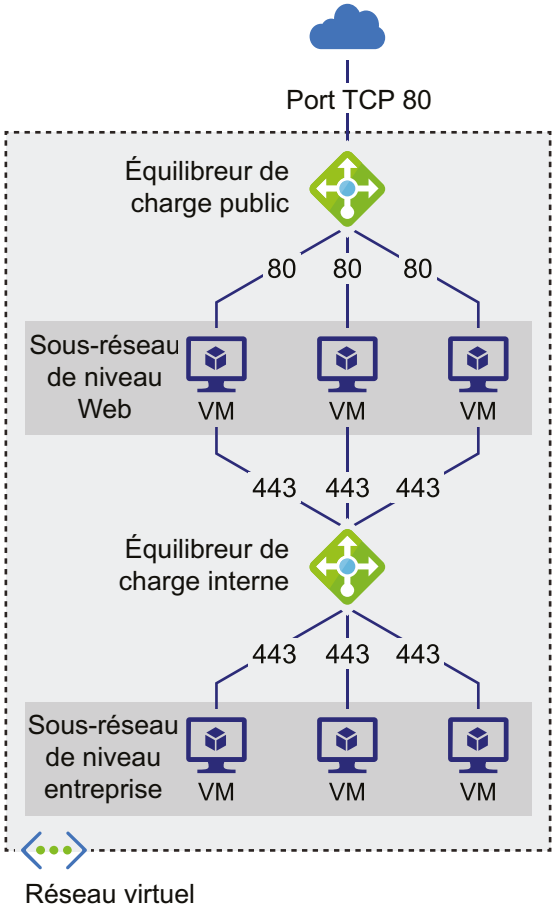


Figure 2.4 : distribution du trafic à l'aide d'équilibreurs de charge Azure

Dans la *Figure 2.4*, les utilisateurs externes accèdent aux machines virtuelles via l'équilibreur de charge public, puis le trafic de la machine virtuelle est réparti sur un autre ensemble de machines virtuelles à l'aide d'un équilibreur de charge interne.

Nous avons comparé la façon dont les équilibreurs de charge Azure diffèrent des passerelles Application Gateway. Nous allons aborder les passerelles Application Gateway en détail dans la section suivante.

Azure Application Gateway

Un équilibreur de charge Azure permet d'activer des solutions au niveau de l'infrastructure. Cependant, des services ou des fonctions avancées sont parfois requis lors de l'utilisation d'un l'équilibreur de charge. Ces services avancés incluent la terminaison SSL, les sessions rémanentes, la sécurité avancée et bien plus encore. Une passerelle d'application Application Gateway Azure fournit ces fonctions supplémentaires ; une passerelle d'application Application Gateway Azure est un équilibreur de charge de niveau 7 qui fonctionne avec l'application et la charge utile de session dans une pile TCP OSI.

Les passerelles Application Gateway disposent de plus d'informations que les équilibreurs de charge Azure afin de prendre des décisions concernant l'acheminement des requêtes et l'équilibrage des charges entre serveurs. Les passerelles Azure Application Gateway sont gérées par Azure et sont hautement disponibles.

Une application Application Gateway se situe entre les utilisateurs et les machines virtuelles, comme illustré dans la *Figure 2.5* :

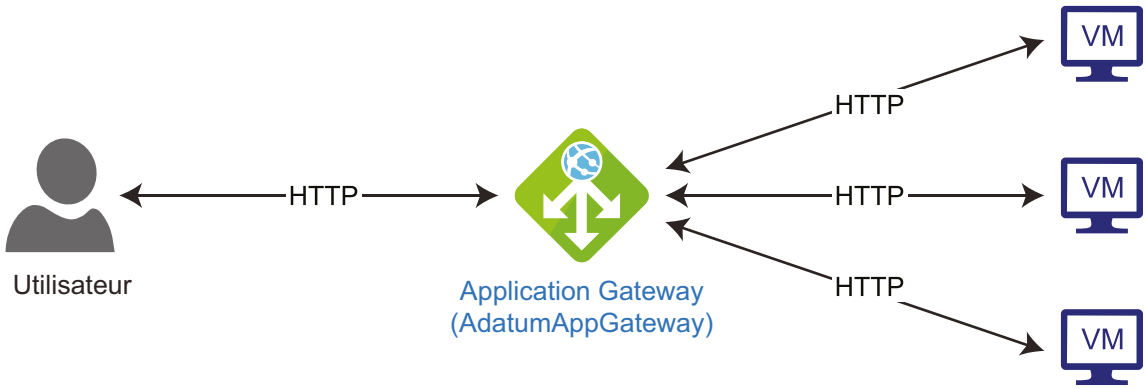


Figure 2.5 : une passerelle Azure Application Gateway

Les passerelles Application Gateway sont un service géré. Elles utilisent le routage **ARR (Application Request Routing)** pour acheminer les demandes vers différents services et points de terminaison. La création d'une passerelle Application Gateway nécessite une adresse IP privée ou publique. La passerelle Application Gateway achemine ensuite le trafic HTTP/HTTPS vers les points de terminaison configurés.

Une passerelle Application Gateway est semblable à l'équilibreur de charge Azure du point de vue de la configuration, mais comporte des structures et des fonctions supplémentaires. Les passerelles Application Gateway peuvent être configurées avec une adresse IP frontend, un certificat, une configuration de port, un pool principal, une affinité de session et des informations de protocole.

L'autre service que nous avons évoqué en matière de haute disponibilité pour les machines virtuelles est Azure Traffic Manager. Découvrons ce service plus en détail dans la section suivante.

Azure Traffic Manager

Maintenant que vous avez bien compris en quoi consistent les équilibreurs de charge Azure et les passerelles Application Gateway, il est temps d'examiner en détail Traffic Manager. Les équilibreurs de charge Azure et les passerelles Application Gateway sont des ressources indispensables pour assurer une haute disponibilité au sein d'un datacenter ou d'une région ; cependant, pour atteindre une haute disponibilité sur l'ensemble des datacenters et des régions, une autre ressource est nécessaire, dénommée Traffic Manager.

Traffic Manager permet de créer des solutions haute disponibilité qui s'étendent sur plusieurs datacenters, régions et zones géographiques. Traffic Manager diffère des équilibreurs de charge. Il utilise le **DNS** (**Domain Name Service**) pour rediriger les requêtes vers un point de terminaison approprié, déterminé selon l'état d'intégrité et la configuration du point de terminaison. Traffic Manager n'est pas un proxy ou une passerelle et il voit pas le trafic transitant entre le client et le service. Il redirige simplement les requêtes selon les points de terminaison les plus appropriés.

Azure Traffic Manager vous permet de contrôler la répartition du trafic sur l'ensemble des points de terminaison de l'application. Un point de terminaison peut être assimilé à un service Internet hébergé dans ou hors d'Azure.

Les points de terminaison sont des URL publiques accessibles via Internet. Les applications sont mises en service dans plusieurs zones géographiques et régions Azure. Les applications déployées dans chaque région disposent d'un point de terminaison unique visé par **DNS CNAME**. Ces points de terminaison sont cartographiés sur le point de terminaison de Traffic Manager. Lorsqu'une instance Traffic Manager est mise en service, elle obtient un point de terminaison par défaut avec une extension URL `.trafficmanager.net`.

Lorsqu'une requête parvient à l'URL Traffic Manager, celui-ci détecte le point de terminaison le plus approprié dans sa liste et y redirige la requête. En bref, Azure Traffic Manager agit comme un DNS global pour identifier la région qui servira la requête.

Cependant, comment Traffic Manager sait-il quels points de terminaison utiliser pour y rediriger les requêtes client ? Il existe deux aspects implémentés par Traffic Manager afin de déterminer le point de terminaison et la région les plus appropriés.

Tout d'abord, Traffic Manager surveille activement l'état d'intégrité de tous les points de terminaison. Il peut surveiller l'état d'intégrité des machines virtuelles, des services Cloud et des services d'application. S'il constate que l'état d'intégrité d'une application déployée sur une région n'est pas adapté pour y rediriger le trafic, il redirige les requêtes vers un point de terminaison intègre.

En outre, Traffic Manager peut être configuré avec les informations de routage. Il existe quatre méthodes de routage de trafic dans Traffic Manager, comme suit :

- **Priorité** : cette méthode doit être utilisée lorsque tout le trafic doit être acheminé vers un point de terminaison par défaut et que des sauvegardes sont disponibles si les points de terminaison principaux ne sont pas disponibles.
- **Pondérée** : cette méthode doit être utilisée pour répartir le trafic sur les points de terminaison de manière uniforme ou selon des pondérations définies.
- **Performance** : cette méthode doit être utilisée pour les points de terminaison situés dans différentes régions et si les utilisateurs doivent être réacheminés vers le point de terminaison le plus proche de leur emplacement. Cela a une conséquence directe sur la latence du réseau.
- **Géographique** : cette méthode doit être utilisée pour rediriger les utilisateurs vers un point de terminaison (Azure, externe ou imbriqué) en fonction de la zone géographique la plus proche. Cette méthode permet de garantir la conformité en matière de protection de données, de localisation et de collecte de trafic en fonction des régions.
- **Sous-réseau** : il s'agit d'une nouvelle méthode de routage qui aide à fournir aux clients des points de terminaison différents en fonction de leurs adresses IP. Dans cette méthode, une plage d'adresses IP est affectée à chaque point de terminaison. Ces plages d'adresses IP sont mappées à l'adresse IP du client pour déterminer un point de terminaison de retour approprié. À l'aide de cette méthode de routage, il est possible de fournir un contenu différent à différentes personnes en fonction de leur adresse IP d'origine.
- **Valeurs multiples** : il s'agit également d'une nouvelle méthode dans Azure. Dans cette méthode, plusieurs points de terminaison sont retournés au client et n'importe lequel d'entre eux peut être utilisé. Cela garantit que si un point de terminaison est défectueux, d'autres points de terminaison peuvent être utilisés à sa place. Cela permet d'augmenter la disponibilité globale de la solution.

Il convient de noter qu'une fois que Traffic Manager détermine qu'un point de terminaison est valide et intègre, les clients se connectent directement à l'application. Découvrons à présent les fonctionnalités d'Azure dédiées au routage des requêtes des utilisateurs à l'échelle mondiale.

La section suivante porte sur un autre service, appelé Azure Front Door. Ce service est similaire à Azure Application Gateway, à une différence près. Découvrons Azure Front Door plus en détail.

Azure Front Door

Azure Front Door est la dernière offre d'Azure qui achemine les requêtes vers des services à l'échelle mondiale et non au niveau d'une région ou d'un datacenter local, comme c'est le cas d'Azure Application Gateway et des équilibreurs de charge. Azure Front Door est similaire à Application Gateway, à une différence près : la portée. Il s'agit d'un équilibreur de charge de couche 7 qui achemine les requêtes vers le point de terminaison de service le plus proche et le plus performant, déployé dans plusieurs régions. Il offre des fonctions telles que la terminaison TLS, l'affinité de session, le routage basé sur l'URL et l'hébergement sur plusieurs sites, ainsi qu'un pare-feu d'application Web. Il est similaire à Traffic Manager car il est par défaut résilient aux pannes affectant toute une région et propose des fonctionnalités de routage. Il sonde également régulièrement l'intégrité des points de terminaison afin de garantir que les requêtes sont acheminées uniquement vers des points de terminaison sains.

Il propose quatre méthodes de routage différentes :

- **Latence** : les requêtes seront acheminées vers des points de terminaison qui présenteront la latence end-to-end la plus faible.
- **Priorité** : les requêtes seront acheminées vers un point de terminaison principal et un point de terminaison secondaire en cas de défaillance du point de terminaison principal.
- **Pondérée** : les requêtes seront acheminées en fonction des pondérations attribuées aux points de terminaison.
- **Affinité de session** : les requêtes d'une session seront acheminées vers le même point de terminaison pour utiliser les données de session des requêtes antérieures. La requête d'origine peut être acheminée vers n'importe quel point de terminaison disponible.

Les déploiements nécessitant une résilience au niveau mondial doivent intégrer Azure Front Door dans leur architecture, parallèlement aux passerelles d'application et aux équilibreurs de charge. La section suivante évoque certaines des considérations architecturales qui doivent être prises en compte lors de la conception de solutions hautement disponibles.

Considérations architecturales relatives à la haute disponibilité

Azure offre une haute disponibilité via différents moyens et à différents niveaux. La haute disponibilité peut être au niveau du datacenter, au niveau de la région ou dans Azure. Dans cette section, nous allons aborder certaines des architectures prévues pour la haute disponibilité.

Haute disponibilité au sein des régions Azure

L'architecture illustrée à la *Figure 2.6* illustre le déploiement de la haute disponibilité dans une région Azure. La haute disponibilité est conçue au niveau de ressources individuelles. Dans cette architecture, il existe plusieurs machines virtuelles à chaque niveau, qui sont connectées par l'intermédiaire d'une passerelle d'application ou d'un équilibreur de charge et qui font partie d'un ensemble de disponibilité. Chaque niveau est associé à un ensemble de disponibilité. Ces machines virtuelles sont placées dans des domaines d'erreur et de mise à jour distincts. Bien que les serveurs Web soient connectés aux passerelles Application Gateway, les autres niveaux, telles que ceux de l'application ou de la base de données, disposent d'équilibreurs de charge internes :

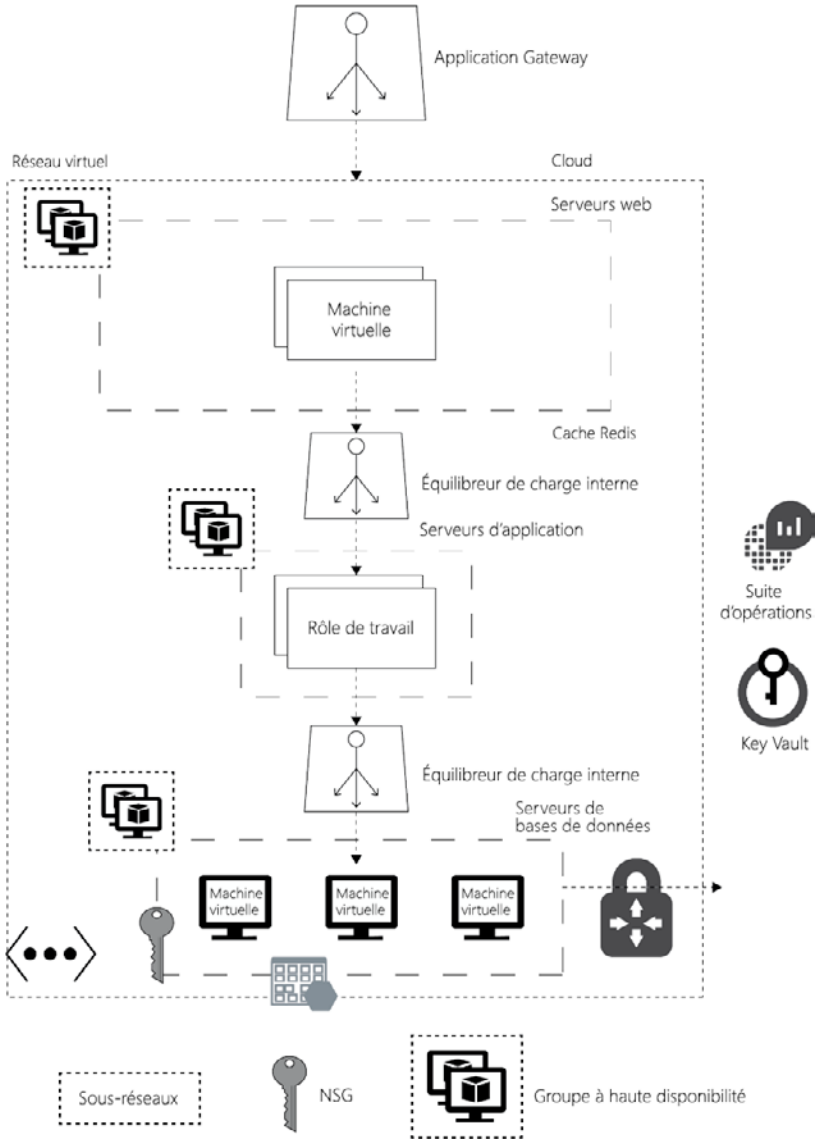


Figure 2.6 : conception de la haute disponibilité au sein d'une région

Maintenant que vous savez comment concevoir des solutions hautement disponibles dans la même région, expliquons comment concevoir une architecture similaire, mais répartie dans différentes régions Azure.

Haute disponibilité sur l'ensemble des régions Azure

L'architecture illustrée ci-après présente des déploiements similaires dans deux régions Azure différentes. Comme illustré dans la *Figure* 2.7, les deux régions disposent des mêmes ressources déployées. La haute disponibilité est conçue au niveau de ressources individuelles au sein de ces régions. Il y a plusieurs machines virtuelles à chaque niveau, connectées par l'intermédiaire d'équilibreurs de charge, et qui font partie d'un ensemble de disponibilité. Ces machines virtuelles sont placées dans des domaines d'erreur et de mise à jour distincts. Bien que les serveurs Web soient connectés à des équilibreurs de charge externes, les autres niveaux, telles que ceux de l'application ou de la base de données, disposent d'équilibreurs de charge internes. Il convient de noter que les équilibreurs de charge des applications auraient pu être utilisés pour les niveaux des serveurs Web et des applications en lieu des équilibreurs de charge Azure si des services avancés sont nécessaires, telles que l'affinité de session, la terminaison SSL, la sécurité avancée via **WAF** (**web application firewall**) et le routage par chemin. Les bases de données des deux régions sont connectées entre elles à l'aide de l'appairage de réseaux virtuels et des passerelles. Ceci est utile lors de la configuration de l'envoi de fichiers journaux, de SQL Server AlwaysOn et pour d'autres techniques de synchronisation de données.

Les points de terminaison des équilibreurs de charge de ces deux régions sont utilisés pour configurer des points de terminaison de Traffic Manager et le trafic est acheminé en fonction de la méthode d'équilibrage de la charge prioritaire. Traffic Manager contribue à réacheminer toutes les requêtes vers la région Est des États-Unis et les bascule vers l'Europe occidentale si la première région n'est pas disponible :

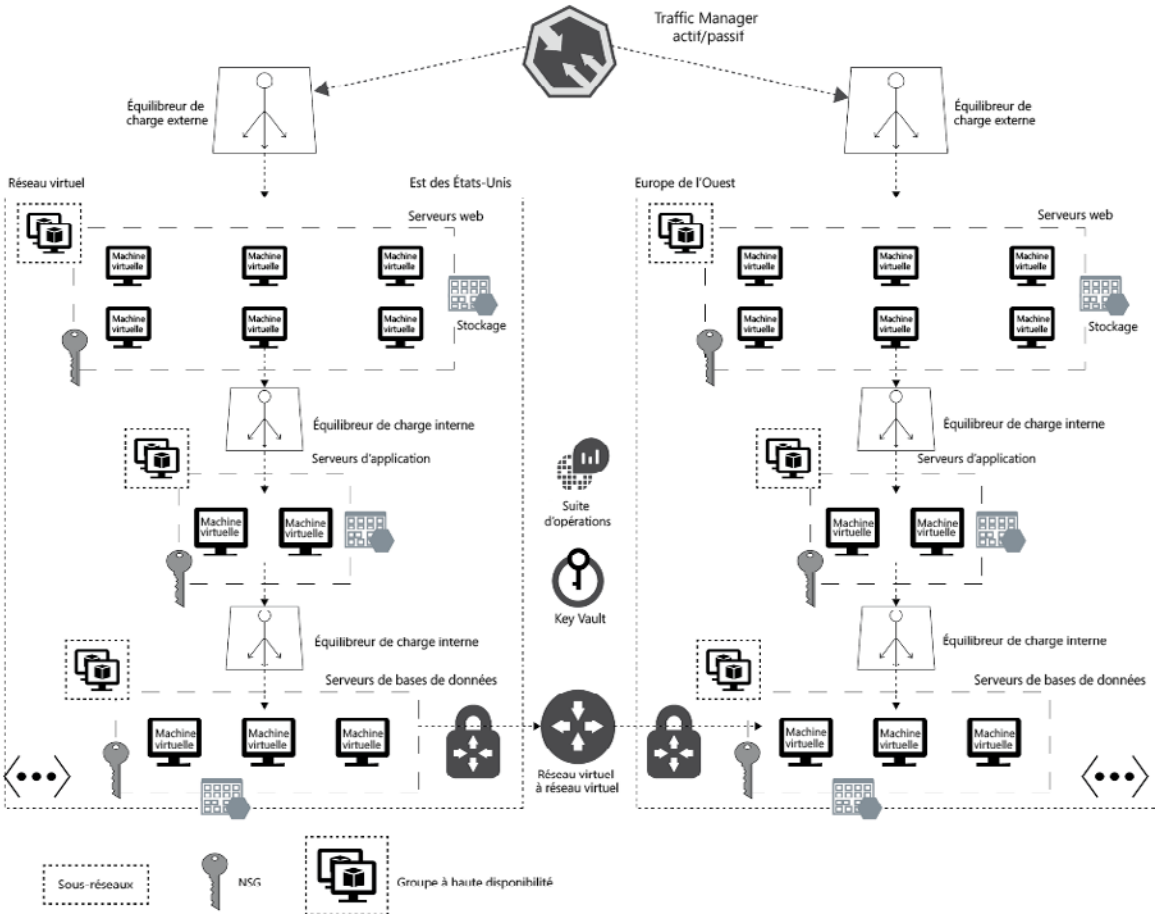


Figure 2.7 : conception de la haute disponibilité sur l'ensemble des régions Azure

La section suivante porte sur l'évolutivité, qui est un autre avantage du Cloud.

Évolutivité

Tous les architectes qui conçoivent une application stratégique se soucient en premier lieu que celle-ci, et les systèmes corrélés, soient disponibles aux utilisateurs. Cependant, il existe un autre composant de l'application tout aussi important, qui constitue l'une des plus grandes priorités pour les architectes, à savoir l'évolutivité de l'application.

Imaginez une situation où une application déployée serait très performante et disponible pour quelques utilisateurs, mais qui le serait de moins en moins à mesure que le nombre de ces derniers augmenterait. Parfois, une application soumise à une charge normale présente des performances satisfaisantes, lesquelles se dégradent à mesure que le nombre d'utilisateurs augmente. Cela se produit surtout en cas de hausse soudaine du nombre d'utilisateurs, si l'environnement n'est pas conçu pour gérer une telle affluence.

Afin de tenir compte de ces pics du nombre d'utilisateurs, vous avez peut-être configuré le matériel et la bande passante en conséquence. Le problème, c'est que cette capacité supplémentaire n'est pas exploitée pendant une grande partie de l'année et par conséquent ne fournit aucun retour sur investissement. Elle est uniquement mise en service pour une utilisation pendant la période des fêtes ou des soldes. J'espère que vous commencez à vous familiariser avec les problèmes que les architectes essaient de résoudre. Tous ces problèmes sont liés au dimensionnement de la capacité et à l'évolutivité d'une application. L'objectif de ce chapitre est de comprendre l'évolutivité en tant que préoccupation architecturale et de découvrir les services Azure qui permettent de la mettre en œuvre.

La planification de la capacité et le dimensionnement font partie des principales priorités pour les architectes lorsqu'ils conçoivent leurs applications et services. Les architectes doivent trouver le juste équilibre au niveau de l'achat et de la mise en service des ressources, pour s'équiper d'une quantité adéquate à leurs besoins. Des ressources insuffisantes peuvent ne pas permettre de servir tous les utilisateurs, qui se tourneront alors vers la concurrence. D'autre part, avoir trop de ressources peut nuire à votre budget et à votre retour sur investissement car la plupart des ressources resteront inutilisées la majeure partie du temps. En outre, le problème est amplifié lorsque la demande varie selon les périodes. Il est quasiment impossible de prédire le nombre d'utilisateurs d'une application tout au long d'une journée et sur toute une année. Toutefois, il est possible de déterminer un nombre approximatif sur la base de données antérieures et grâce à un suivi continu.

L'évolutivité désigne la capacité de gérer un nombre croissant d'utilisateurs et de leur offrir le même niveau de performance que celui d'un trafic moindre en termes d'utilisation de ressources, de déploiement des applications, des processus et de la technologie. L'évolutivité peut désigner le fait de servir davantage de requêtes sans que cela ne nuise aux performances, ou de traiter des tâches plus conséquentes ou plus laborieuses sans entraver en aucun cas les performances.

Les architectes doivent procéder à une planification des capacités et tester le dimensionnement dès la création de leur projet, durant la phase de planification, afin d'assurer l'évolutivité des applications.

Certaines applications disposent de modèles de demande stables, tandis que d'autres sont plus imprévisibles. Ainsi, tandis que les exigences en termes d'évolutivité sont connues pour les applications dont la demande est stable, elles sont plus difficiles à déterminer pour les applications dont l'usage est variable. Le dimensionnement automatique, un concept que nous aborderons dans la section suivante, doit être utilisé pour les applications pour lesquelles il est impossible de prévoir la demande.

Les gens ont souvent tendance à confondre l'évolutivité et les performances. La section suivante compare brièvement ces deux notions.

Évolutivité et performances

Les considérations architecturales du point de vue de l'évolutivité et des performances se confondent souvent, car l'évolutivité consiste à maintenir en tout temps un niveau de performance prédéterminé, quel que soit le nombre d'utilisateurs consommant l'application.

Les performances font référence au fait qu'une application respecte les délais de réponse et le débit prédéfinis. L'évolutivité fait référence à la mise en service de ressources supplémentaires si nécessaire afin d'accueillir plus d'utilisateurs sans que cela ne nuise aux performances.

Il est préférable de comprendre cela en utilisant une analogie : la vitesse d'un train équivaut a directement trait à la performance d'un réseau ferroviaire. Cependant, permettre à plusieurs trains de circuler en parallèle à la même vitesse ou plus rapidement démontre l'évolutivité du réseau ferroviaire.

Maintenant que vous connaissez la différence entre l'évolutivité et les performances, discutons de l'évolutivité offerte par Azure.

Évolutivité Azure

Dans cette section, nous allons voir les fonctions et caractéristiques offertes par Azure afin de rendre les applications hautement disponibles. Avant d'entrer dans les détails relatifs à l'architecture et à la configuration, il est important de comprendre les concepts Azure ayant trait à la haute disponibilité, autrement dit, la mise à l'échelle.

La mise à l'échelle fait référence à l'augmentation ou à la diminution du nombre de ressources utilisées pour traiter les requêtes des utilisateurs. La mise à l'échelle peut être automatique ou manuelle. La mise à l'échelle manuelle nécessite un administrateur qui lancera manuellement le processus de mise à l'échelle, tandis que la mise à l'échelle automatique renvoie à une augmentation ou à une réduction automatique des ressources en fonction des événements survenus dans l'environnement et dans l'écosystème, tels que la mémoire et la disponibilité du CPU. Les ressources peuvent être mises à l'échelle vers le haut ou le bas ou vers l'intérieur et l'extérieur, comme nous le verrons plus loin dans cette section.

Outre les mises à jour propagées, les concepts fondamentaux fournis par Azure pour atteindre la haute disponibilité sont les suivants :

- Dimensionnement
- Augmentation ou réduction des instances
- Mise à l'échelle automatique

Mise à l'échelle (augmentation)

La mise à l'échelle vers un niveau supérieur d'une machine virtuelle ou d'un service revient à ajouter des ressources supplémentaires à des serveurs existants, tels que le processeur, la mémoire et les disques. Cela revient à augmenter la capacité des ressources et du matériel physique existant.

Mise à l'échelle à un niveau inférieur

La mise à l'échelle vers un niveau inférieur d'une machine virtuelle ou d'un service revient à supprimer des ressources en place dans des serveurs existants, tels que le processeur, la mémoire et les disques. Cela revient à réduire la capacité des ressources et du matériel physique et virtuel existant.

Mise à l'échelle (montée en charge)

La montée en charge désigne l'ajout de matériel supplémentaire, en termes de serveurs et de capacités. Cela revient en général à ajouter de nouveaux serveurs, à leur assigner des adresses IP, à y déployer des applications et à les intégrer aux équilibreurs de charge existants de sorte que le trafic y soit acheminé. La montée en charge peut être automatique ou manuelle. Cependant, l'automatisation est recommandée pour atteindre des résultats optimaux :

Figure 2.8 : mise à l'échelle (montée en charge)

Réduction des instances

La réduction des instances désigne le processus de suppression du matériel existant en termes de serveurs et de capacités. Cela implique généralement la suppression de serveurs existants, la libération de leurs adresses IP et leur suppression dans la configuration d'équilibreur de charge existante, de sorte que le trafic ne puisse plus être acheminé vers ces instances. De même que pour la montée en charge, ce processus peut être automatique ou manuel.

Mise à l'échelle automatique

La mise à l'échelle automatique désigne le processus de mise à l'échelle (augmentation, réduction, montée en charge ou suppression d'instance) selon la demande de l'application, et ce de manière automatique. La mise à l'échelle automatique s'avère utile car elle permet de s'assurer en tout temps que le déploiement dispose d'un nombre optimal d'instances de serveur. La mise à l'échelle automatique permet de concevoir des applications tolérantes aux pannes. Elle favorise non seulement l'évolutivité mais également la haute disponibilité des applications. Enfin, elle assure une gestion des coûts optimale. La mise à l'échelle automatique permet de disposer d'une configuration optimale des instances de serveur en fonction de la demande. Cela permet d'éviter une mise en service excessive de serveurs, lesquels sont alors peu utilisés, et de supprimer les serveurs devenus inutiles suite à une montée en charge.

Nous avons discuté de l'évolutivité dans Azure. Azure offre des options d'évolutivité pour la plupart de ses services. Abordons désormais l'évolutivité pour les PaaS dans Azure dans la section suivante.

Évolutivité SaaS

Azure offre la solution App Service dédiée à l'hébergement d'applications gérées. App Service est une offre PaaS d'Azure. Cette solution fournit des services aux plateformes Web et mobiles. Ces plateformes Web et mobile sont gérées en arrière-plan par une infrastructure gérée, elle-même gérée par Azure pour le compte de ses utilisateurs. Les utilisateurs ne voient pas ni ne gèrent cette infrastructure ; toutefois, ils ont la possibilité d'élargir la plateforme et d'y déployer leurs applications. Ainsi, les architectes et les développeurs peuvent se concentrer sur les enjeux propres à leur entreprise au lieu de se préoccuper de la plateforme de base et de la mise en service, de la configuration et du dépannage de l'infrastructure. Les développeurs ont la possibilité de choisir n'importe quel langage, système d'exploitation et structure pour développer leurs applications. App Service fournit plusieurs plans, auxquels sont assortis différents degrés d'évolution. App Service propose les cinq plans suivants :

- **Gratuit** : ce plan utilise une infrastructure partagée. Cela signifie que plusieurs applications seront déployées sur la même infrastructure à partir d'un même ou de plusieurs locataires. Il offre 1 Go de stockage gratuit. Toutefois, aucune installation de mise à l'échelle n'est disponible dans ce plan.
- **Partagé** : ce plan utilise une infrastructure partagée et offre 1 Go de stockage gratuit. En outre, des domaines personnalisés sont également fournis à titre de fonction supplémentaire. Toutefois, aucune installation de mise à l'échelle n'est disponible dans ce plan.
- **Basique** : ce plan dispose de trois références différentes (**SKU**, **Stock Keeping Unit**) : B1, B2 et B3. Chacune dispose d'unités de ressources évolutives en termes de CPU et de mémoire. En bref, elles fournissent une configuration des machines virtuelles plus évoluée, et soutiennent ces services. En outre, elles fournissent un stockage, des domaines personnalisés et un support SSL. Le plan basique fournit des fonctions de base pour une mise à l'échelle manuelle. Ce plan ne propose pas de fonction de mise à l'échelle automatique. Trois instances maximum peuvent être utilisées pour la montée en charge d'une application.
- **Standard** : ce plan propose également trois références SKU différentes, S1, S2 et S3. Chacune dispose d'unités de ressources évolutives en termes de CPU et de mémoire. En bref, elles fournissent une configuration des machines virtuelles plus évoluée, et soutiennent ces services. En outre, ce plan offre un stockage, des domaines personnalisés et un support de SSL similaire au plan basique. Ce plan fournit également une instance Traffic Manager, des emplacements intermédiaires et une sauvegarde quotidienne, en tant que fonction supplémentaire, en plus du plan basique. Le plan standard fournit des fonctions permettant la mise à l'échelle automatique. 10 instances maximum peuvent être utilisées pour la montée en charge de l'application.

- **Premium** : ce plan propose également trois références SKU différentes, P1, P2 et P3. Chacune dispose d'unités de ressources évolutives en termes de CPU et de mémoire. En bref, elles fournissent une configuration des machines virtuelles plus évoluée, et soutiennent ces services. En outre, ce plan offre un stockage, des domaines personnalisés et un support de SSL similaire au plan basique. Ce plan fournit également une instance Traffic Manager, des emplacements intermédiaires et 50 sauvegardes quotidiennes, en tant que fonction supplémentaire, en plus du plan basique. Le plan standard fournit des fonctions permettant la mise à l'échelle automatique. 20 instances maximum peuvent être utilisées pour la montée en charge de l'application.

Nous avons évoqué les niveaux d'évolutivité disponibles pour les services PaaS. Découvrons à présent comment effectuer la mise à l'échelle dans le cas d'un plan App Service.

Dimensionnement PaaS (augmentation et réduction)

Le dimensionnement des services hébergés dans App Service est un processus assez simple. Le menu de mise à l'échelle Azure App Services propose un nouveau volet contenant tous les plans ainsi que leurs références. Le choix d'un plan et d'une référence SKU mettra à l'échelle le service vers le haut ou vers le bas, comme illustré dans la *Figure* 2.9 :

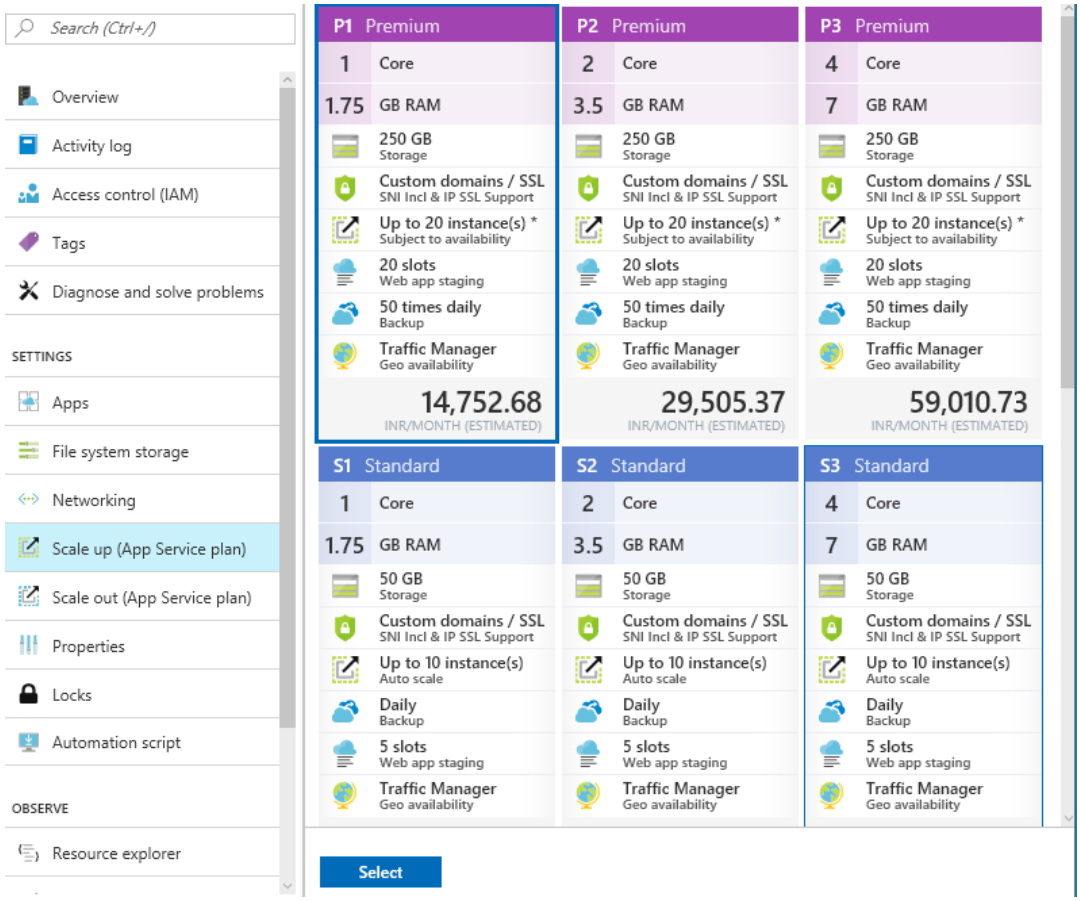


Figure 2.9 : différents plans avec leurs références

Augmentation ou réduction des instances PaaS

Le dimensionnement des services hébergés (montée en charge ou réduction des instances) dans App Service est également un processus assez simple. Le menu de montée en charge d'Azure App Services s'ouvre dans un nouveau volet contenant les options de configuration du dimensionnement.

Par défaut, le dimensionnement automatique est désactivé pour les plans premium et standard. Celui-ci peut être activé via l'option du menu **Dimensionner** et en cliquant sur le bouton **Activer le dimensionnement automatique**, comme indiqué dans la *Figure 2.10* :

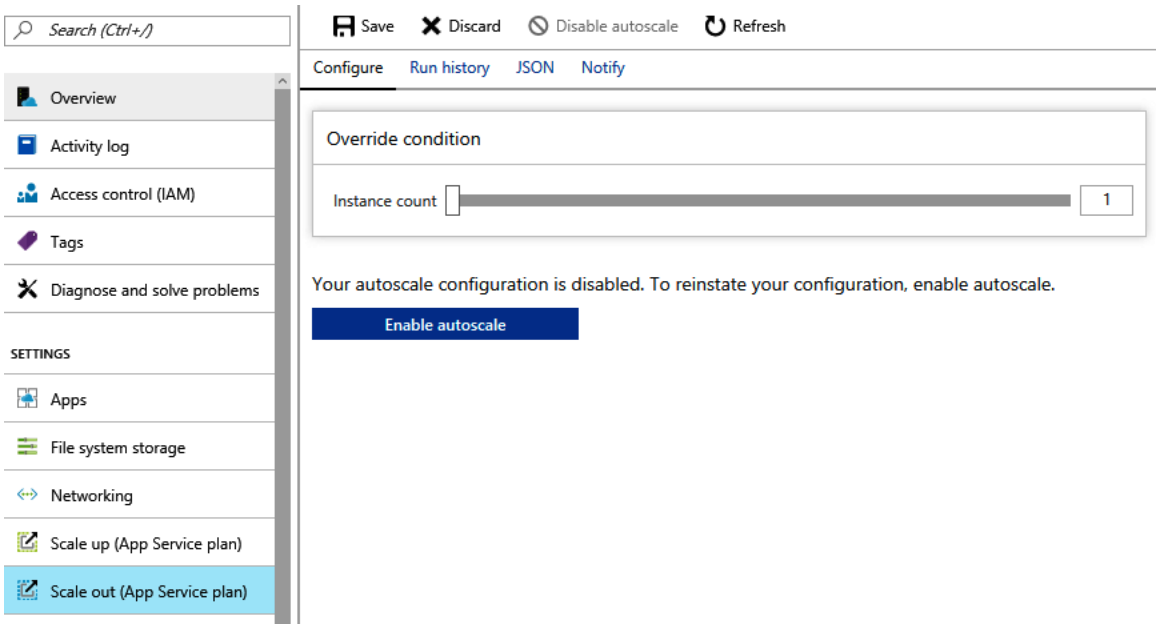


Figure 2.10 : activation de l'option du dimensionnement automatique

Le dimensionnement manuel ne nécessite aucune configuration, toutefois le dimensionnement automatique permet de configurer les propriétés suivantes :

- **Mode de mise à l'échelle** : ceci est basé sur l'une des métriques de performances telles que l'utilisation du CPU ou de la mémoire, ou bien les utilisateurs peuvent simplement spécifier le nombre d'instances pour la mise à l'échelle.
- **Quand procéder au dimensionnement** : plusieurs règles peuvent être définies afin de déterminer quand procéder à un redimensionnement. Chaque règle peut déterminer des critères, tels que la consommation du CPU ou de la mémoire, s'il faut augmenter ou diminuer des instances et le nombre d'instances à augmenter ou à diminuer chaque fois. Au moins une règle de mise à l'échelle (montée en charge et réduction des instances) doit être configurée. La définition d'un seuil permet d'établir des limites minimales et maximales qui déclencheront un dimensionnement automatique, pour augmenter ou réduire le nombre d'instances.
- **Comment dimensionner** : spécifie combien d'instances doivent être créées ou supprimées à chaque opération de redimensionnement :

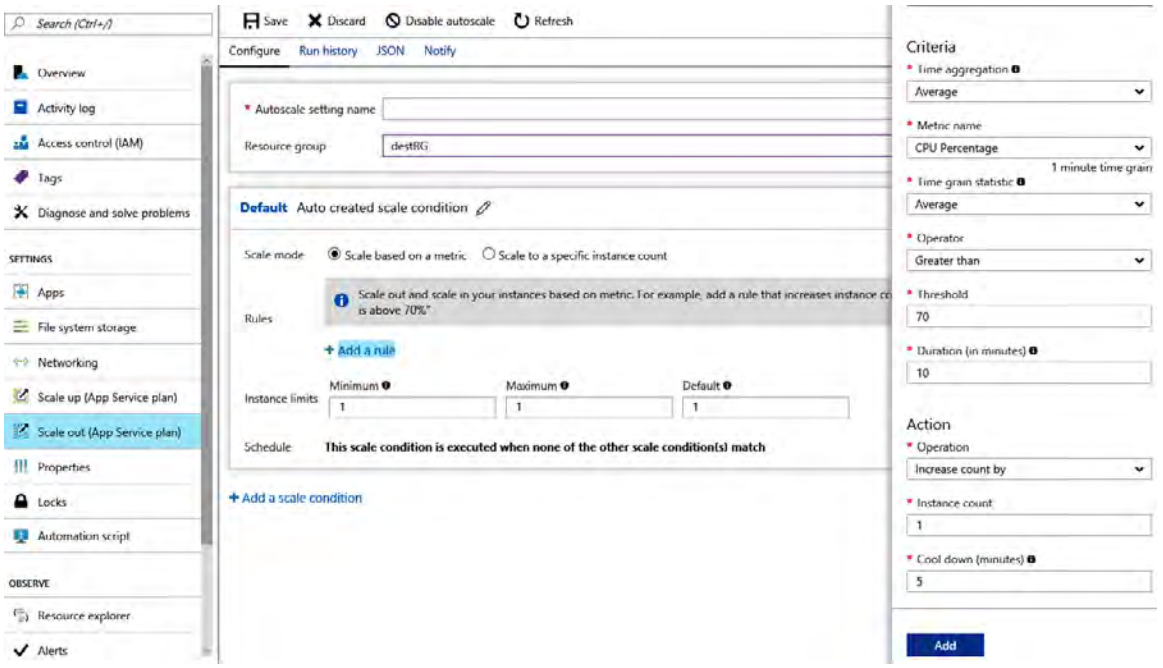


Figure 2.11 : définition des limites d'instance

Il s'agit d'une fonction qui peut être activée dans n'importe quel déploiement. Toutefois, vous devez activer la mise à l'échelle vers le haut ou vers le bas, pour garantir que votre environnement retourne à la capacité normale après la mise à l'échelle.

Nous avons étudié l'évolutivité des PaaS, le moment est venu d'aborder l'évolutivité des IaaS.

Évolutivité IaaS

Certains utilisateurs souhaitent disposer d'un contrôle complet sur leurs infrastructure de base, plateforme et application. Ils préfèrent consommer des solutions IaaS plutôt que PaaS. Ces clients sont responsables du dimensionnement des capacités des machines virtuelles qu'ils créent. Il n'existe aucune configuration initiale pour le dimensionnement manuel ou automatique des machines virtuelles. Ces clients devront écrire leurs propres scripts d'automatisation, déclencheurs et règles afin de mettre en place un dimensionnement automatique. Les machines virtuelles s'assortissent de certaines tâches de maintenance également. Les correctifs, la mise à jour et la mise à niveau des machines virtuelles relèvent de la responsabilité des propriétaires. Les architectes doivent réfléchir à la fois à la maintenance planifiée et non prévue. Ils doivent réfléchir à la manière dont les correctifs doivent être appliqués sur les machines virtuelles, comment celles-ci doivent être organisées et regroupées et tenir compte d'autres facteurs afin d'assurer l'évolutivité et la disponibilité de leur application. Afin de faciliter ces tâches, Azure offre une solution, les **VMSS** (**VM Scale Sets**), que nous aborderons plus en détail par la suite.

Groupes de machines virtuelles identiques (VMSS)

Les VMSS sont des ressources de traitement Azure qui vous permettent d'utiliser, de déployer et de gérer un ensemble de machines virtuelles identiques. Toutes les machines virtuelles étant configurées de la même manière, les VMS peuvent ainsi assurer un dimensionnement automatique efficace, sans nécessiter une mise en service des machines virtuelles préalable. Cela permet de mettre en service plusieurs machines virtuelles identiques reliées entre elles par un réseau virtuel et un sous-réseau.

Un VMSS se compose de plusieurs machines virtuelles, mais celles-ci sont gérées au niveau des VMSS. Toutes les machines virtuelles font partie de cette unité et toute modification sera apportée à l'unité, et ainsi aux machines virtuelles, à l'aide d'un algorithme prédéterminé :

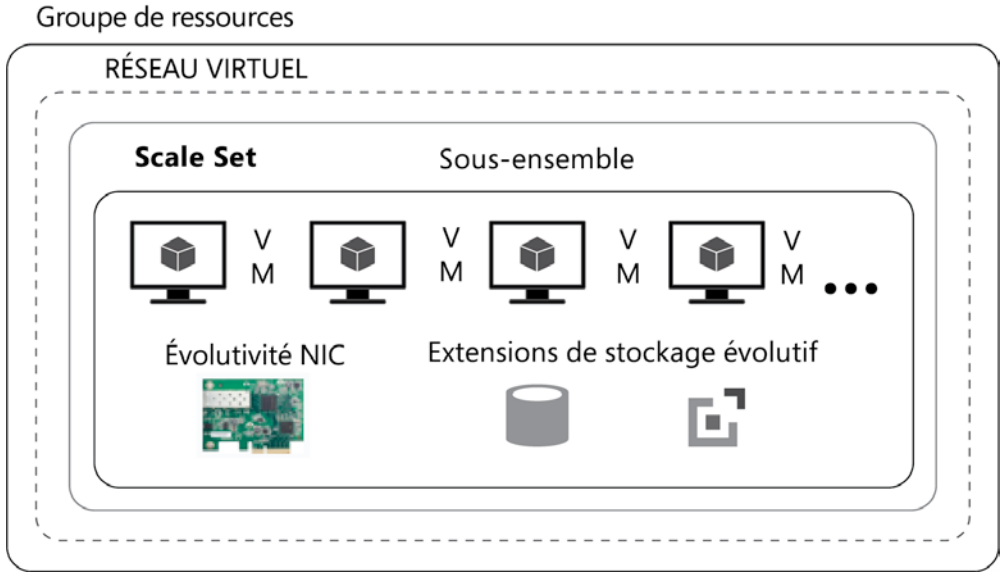


Figure 2.12 : un VMSS

Cela assure une charge équilibrée sur les machines virtuelles, à l'aide d'un équilibreur de charge Azure ou d'une passerelle d'application. Les machines virtuelles peuvent être machines virtuelles Windows ou Linux. Elles peuvent exécuter des scripts automatisés à l'aide d'une extension PowerShell et peuvent être gérées de manière centrale, à l'aide de la configuration souhaitée. Elles peuvent faire l'objet d'un suivi en tant qu'unité et individuellement, à l'aide de Log Analytics également.

Les VMSS peuvent être mis en service depuis le portail Azure, l'interface de ligne de commande Azure (CLI), les modèles Azure Resource Manager, les API REST et les cmdlets de commande PowerShell. Il est possible d'invoquer les API REST et l'interface de ligne de commande Azure depuis n'importe quelle plateforme, environnement, système d'exploitation et n'importe quel langage.

De nombreux services Azure utilisent les VMSS en tant qu'architecture sous-jacente. Ces services sont notamment Azure Batch, Azure Service Fabric et Azure Container Service. Azure Container Service, à son tour, provisionne Kubernetes et DC/OS sur ces VMSS.

Architecture des VMSS

Les VMSS permettent de créer jusqu'à 1 000 machines virtuelles dans un groupe VMSS lors de l'utilisation d'une image de plateforme et 100 machines virtuelles si vous utilisez une image personnalisée. Si le nombre de machines virtuelles est inférieur à 100 dans un groupe VMSS, celles-ci sont placées dans un ensemble de disponibilité unique ; toutefois, si elles sont supérieures à 100, plusieurs ensembles de disponibilité sont créés, lesquels sont dénommés groupes de placement et les machines virtuelles sont réparties entre ces ensembles de disponibilité. Le *chapitre 1*, *Prise en main d'Azure*, nous a appris que les machines virtuelles d'un ensemble de disponibilité sont placées sur des domaines d'erreur et de mise à jour distincts. Les ensembles de disponibilité associés aux VMSS disposent de cinq domaines d'erreur et de mise à jour par défaut. Les VMSS fournissent un modèle qui contient des informations de métadonnées pour tout l'ensemble. Toute modification de ce modèle et son application aura un impact sur toutes les instances de la machine virtuelle. Ces informations incluent le nombre maximal et minimal d'instances de machine virtuelle, la référence SKU et la version du système d'exploitation, le nombre actuel de machines virtuelles, de domaines d'erreur et de mise à jour, etc. Ceci est illustré à la *Figure 2.13* :

Figure 2.13 : machines virtuelles dans un ensemble de disponibilité

Mise à l'échelle VMSS

La mise à l'échelle désigne le fait d'augmenter ou de réduire des ressources de traitement et de stockage. Un VMSS est une ressource riche en fonctionnalités qui rend le processus de mise à l'échelle simple et efficace. Il assure un dimensionnement automatique qui permet de dimensionner le système en fonction d'événements externes et de données, telles que l'utilisation du CPU et de la mémoire. Voici certaines des fonctions de mise à l'échelle des VMSS.

Mise à l'échelle verticale et horizontale

La mise à l'échelle peut être horizontale ou verticale, ou les deux. La mise à l'échelle horizontale est un autre nom pour évoquer la montée en charge et la suppression d'instances, tandis que la mise à l'échelle verticale désigne la mise à l'échelle vers le haut ou vers le bas.

Capacités

Les VMSS disposent d'une propriété de **`capacité`** qui détermine le nombre de machines virtuelles dans un Scale Set. Un VMSS peut être déployé avec une valeur de zéro pour cette propriété. Ce processus ne créera pas une machine virtuelle unique, cependant si un VMSS est mis en service en entrant un nombre pour la propriété de **`capacité`**, cela correspondra au nombre de machines virtuelles créées.

Mise à l'échelle automatique

La mise à l'échelle automatique des machines virtuelles dans un VMSS fait référence à l'ajout ou à la suppression d'instances de machine virtuelle sur la base des environnements configurés pour répondre aux exigences de performances et d'évolutivité d'une application. En général, en l'absence de VMSS, ceci est réalisé à l'aide de procédures opérationnelles et de scripts d'automatisation.

Les VMSS facilitent ce processus d'automatisation à l'aide de la configuration. Au lieu d'écrire des scripts, un VMSS peut être configuré en vue de procéder à un dimensionnement automatique.

Le dimensionnement automatique utilise plusieurs composants intégrés afin d'atteindre son objectif final. Le dimensionnement automatique implique une surveillance des machines virtuelles en continu et recueille des données de télémétrie à leur sujet. Ces données sont stockées, associées, puis évaluées selon un ensemble de règles, afin de déterminer si une mise à l'échelle automatique doit être déclenchée. L'élément déclencheur peut engendrer un dimensionnement vertical ou horizontal.

Le mécanisme du dimensionnement automatique utilise des journaux de diagnostic pour la collecte des données de télémétrie des machines virtuelles. Ces journaux sont stockés dans les comptes de stockage en tant que mesures de diagnostic. Le mécanisme du dimensionnement automatique utilise également le service de surveillance Application Insights qui lit ces mesures, les combine et les stocke dans un compte de stockage.

Les tâches de dimensionnement automatique effectuées en arrière-plan sont exécutées en permanence afin de lire les données de stockage d'Application Insights et de les évaluer selon toutes les règles configurées en vue du dimensionnement automatique, dont le processus est enclenché si l'une de ces règles est satisfaite. Les règles peuvent prendre en compte des mesures issues des machines virtuelles invitées ainsi que du serveur hôte.

Les règles définies à l'aide des descriptions de propriété sont disponibles à l'adresse https://docs.microsoft.com/azure/virtual-machine-scale-sets/virtual-machine-scale-sets-autoscale-overview.

L'architecture de mise à l'échelle automatique VMSS est illustrée à la *Figure 2.14* :

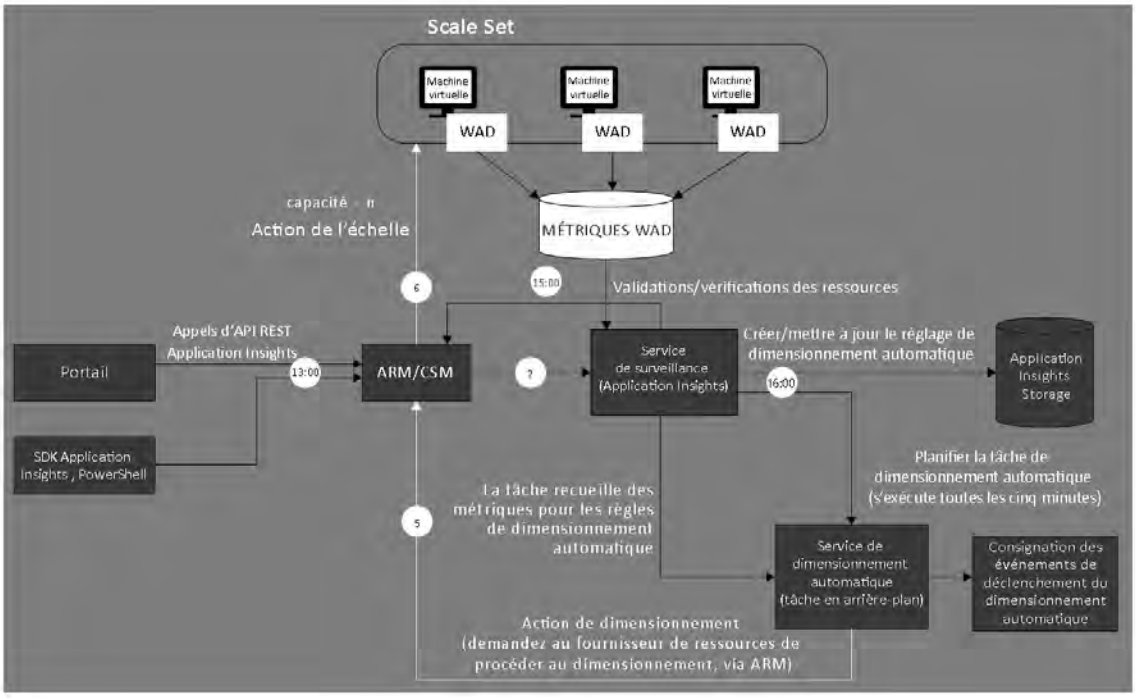


Figure 2.14 : architecture du dimensionnement automatique VMSS

La mise à l'échelle automatique peut être configurée pour des scénarios plus complexes que les mesures générales disponibles dans les environnements. Par exemple, la mise à l'échelle peut se baser sur l'un des événements suivants :

- Un jour spécifique
- Un planning récurrent, par exemple les week-ends
- La semaine ou les week-ends
- Les jours fériés et les événements ponctuels
- Plusieurs mesures de ressource

Celles-ci peuvent être configurées à l'aide d'une propriété de **`planification`** des ressources Application Insights, qui permettra d'enregistrer des règles.

Les architectes veillent à ce qu'au moins deux actions mise à l'échelle, d'augmentation et de réduction, soient configurées ensemble. La configuration de ces deux types de mise à l'échelle ne permettra pas de profiter des avantages issus de la mise à l'échelle exécutée par VMSS.

Nous avons donc abordé les options d'évolutivité dans Azure et les fonctions de mise à l'échelle détaillées pour IaaS et PaaS afin de répondre à vos besoins métier. Selon le modèle de responsabilité partagée, les mises à niveau et la maintenance de la plateforme doivent être réalisées par le fournisseur de Cloud. Dans ce cas, Microsoft prend en charge les mises à niveau et la maintenance associées à la plateforme. Découvrons la procédure associée dans la section suivante.

Mises à niveau et maintenance

Une fois qu'un VMSS et les applications sont déployés, ils doivent être maintenus activement. Des opérations de maintenance planifiée doivent être réalisées régulièrement afin de s'assurer que l'environnement et l'application sont tous deux à jour avec les fonctions les plus récentes du point de vue de la sécurité et de la résilience.

Des mises à niveau peuvent être associées aux applications, à l'instance de machine virtuelle invitée, ou à l'image elle-même. Les mises à niveau peuvent être assez complexes car elles doivent être implémentées sans nuire à la disponibilité, à l'évolutivité et aux performances des environnements et des applications. Afin de s'assurer que les mises à jour peuvent être exécutées sur une instance à un moment donné à l'aide de méthodes d'actualisation par roulement, il est essentiel qu'un VMSS prenne en charge et fournisse des capacités pour ces scénarios avancés.

Un utilitaire est fourni par l'équipe Azure chargée de gérer les mises à jour de VMSS. Il s'agit d'un utilitaire basé sur Python qui peut être téléchargé à l'adresse https://github.com/gbowerman/vmssdashboard. Il effectue des appels de l'API REST à Azure afin de gérer les ensembles de mise à l'échelle. Cet utilitaire peut être utilisé pour démarrer, arrêter, mettre à niveau et réimager des machines virtuelles dans un domaine d'erreur ou un groupe de machines virtuelles, comme illustré à la *Figure 2.15* :

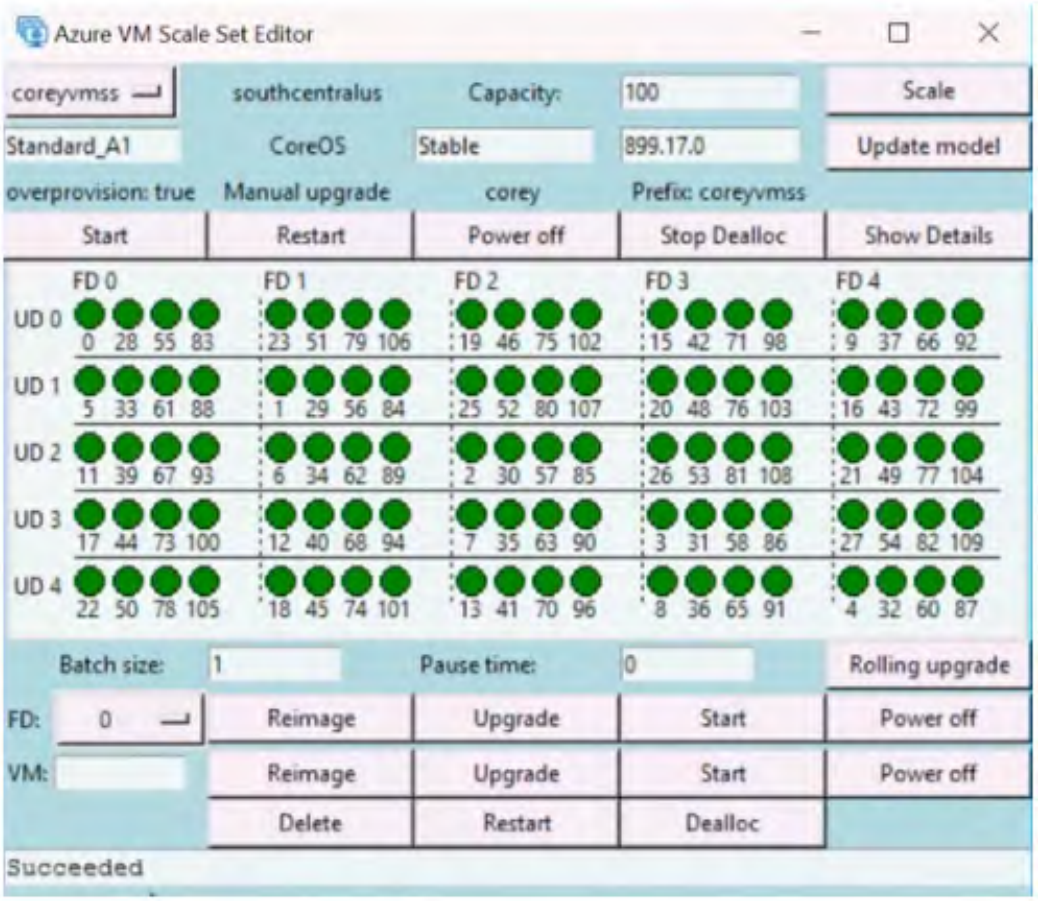


Figure 2.15 : utilitaire dédié à la gestion des mises à jour VMSS

Vous connaissez désormais les principes de base de la mise à niveau et de la maintenance. Le moment est venu d'aborder les mises à jour des applications dans des VMSS.

Mises à jour d'application

Les mises à jour d'application dans des VMSS ne doivent pas être exécutées manuellement. Elles doivent être exécutées dans le cadre de la gestion des versions et de pipeline à l'aide du processus d'automation. En outre, la mise à jour doit être exécutée sur une instance de l'application à la fois, sans affecter la disponibilité et l'évolutivité globales de l'application. Des outils de gestion de la configuration, tels que la **configuration de l'état souhaité** (**DSC**), doivent être déployés afin de gérer les mises à jour de l'application. Un serveur DSC pull peut être configuré avec la dernière version de la configuration d'application et doit être appliqué par roulement sur chaque instance.

La section suivante porte sur la réalisation des mises à jour sur le système d'exploitation invité.

Mises à jour des machines invitées

Les mises à jour d'une machine virtuelle relèvent de la responsabilité de l'administrateur. Azure n'est pas responsable de la correction des machines virtuelles hôtes. Les mises à jour invitées sont en mode de prévisualisation et les utilisateurs doivent gérer les correctifs manuellement ou recourir à des méthodes d'automatisation personnalisée, tels que des scripts et des procédures opérationnelles. Cependant, les mises à niveau de correctifs appliquées par roulement sont en mode de prévisualisation et peuvent être configurées dans le modèle Azure Resource Manager à l'aide d'une stratégie de mise à niveau, comme indiqué ici :

```
"upgradePolicy": {
"mode": "Rolling",
"automaticOSUpgrade": "true" or "false",
  "rollingUpgradePolicy": {
    "batchInstancePercent": 20,
    "maxUnhealthyUpgradedInstanceCount": 0,
    "pauseTimeBetweenBatches": "PT0S"
  }
}
```

Nous savons désormais comment les mises à jour des machines invitées sont gérées dans Azure, voyons comment les mises à jour d'images sont effectuées.

Mises à jour de l'image

Un VMSS peut mettre à jour la version du système d'exploitation sans générer de temps d'arrêt. Des mises à jour du système d'exploitation impliquent la modification de la version ou de la référence de ce dernier, ou encore la modification de l'URI d'une image personnalisée. Une mise à jour sans temps d'arrêt signifie que les machines virtuelles sont mises à jour une à une ou de manière groupée (par exemple, un domaine d'erreur à fois) et non pas toutes en même temps. Ce faisant, les machines virtuelles qui ne sont pas mises à niveau restent opérationnelles.

Jusqu'à présent, nous avons discuté des mises à jour et de la maintenance. Intéressons-nous désormais aux bonnes pratiques en matière de mise à l'échelle des VMSS.

Bonnes pratiques de mise à l'échelle des VMSS

Dans cette section, nous allons passer en revue quelques-unes des bonnes pratiques que les applications doivent implémenter pour tirer parti de la capacité de dimensionnement fournie par les VMSS.

La préférence pour le dimensionnement horizontal

Le dimensionnement horizontal est préférable au dimensionnement vertical. Le dimensionnement vertical permet de redimensionner les instances. Lorsqu'une machine virtuelle est redimensionnée, celle-ci doit généralement être redémarrée, ce qui présente certains inconvénients. Tout d'abord, cela entraîne un temps d'arrêt de la machine. De plus, si des utilisateurs actifs sont connectés à l'application sur cette instance, celle-ci risque d'être indisponible, ce qui peut entraîner des pertes de transactions. Le dimensionnement horizontal n'a pas d'impact sur les machines virtuelles existantes, mais il provisionne les machines plus récentes et les ajoute au groupe.

Nouvelles instances et instances inactives

Le dimensionnement de nouvelles instances peut se faire selon deux approches globales : en créant la nouvelle instance en partant de zéro, ce qui revient à installer les applications, les configurer et les tester, ou en redémarrant les instances inactives lorsque nécessaire, en fonction des exigences d'évolutivité des autres serveurs.

Configurer le nombre maximal et minimal d'instances de manière appropriée

Compte tenu du fait que la valeur correspondant au nombre d'instances minimal et maximal est de deux, et que les instances actuelles sont au nombre de deux, aucune action de dimensionnement ne sera effectuée. Une différence suffisante doit être prévue entre les nombres d'instance maximal et minimal, où les valeurs sont comprises. Le dimensionnement automatique se fait toujours en respectant ces limites.

Accès concurrentiel

Les applications sont conçues de sorte que leur évolutivité soit axée sur l'accès concurrentiel. Les applications doivent utiliser des modèles asynchrones afin de veiller à ce que les requêtes client n'attendent pas l'acquisition des ressources pendant des délais indéfinis si les ressources sont mobilisées par d'autres requêtes. La mise en œuvre de modèles asynchrones dans le code permet de s'assurer que les threads n'attendent pas l'épuisement des ressources et des systèmes de tous les threads disponibles. Les applications doivent implémenter le concept de délais d'expiration si des pannes intermittentes sont à prévoir.

Conception d'applications sans état

Les applications et les services doivent être conçus de sorte à ne présenter aucun état. L'évolutivité peut devenir un véritable défi à relever avec des services présentant un état, alors qu'il est assez simple de dimensionner des services sans état. L'état s'accompagne d'une exigence de composants supplémentaires et d'opérations de mise en œuvre telles que la réplication, le référentiel centralisé ou décentralisé, la maintenance et les sessions rémanentes. Tous ces éléments sont des obstacles au processus d'évolutivité. Imaginez qu'un service maintienne un état actif sur un serveur local. Quel que soit le nombre de requêtes sur l'application dans son ensemble ou sur chaque serveur, les requêtes subséquentes doivent être signifiées par le même serveur. Les requêtes subséquentes ne peuvent être traitées par d'autres serveurs. Cela rend l'évolutivité très difficile.

Mise en cache et CDN (Content Distribution Network)

Les applications et les services doivent tirer parti de la mise en cache. La mise en cache permet d'éliminer les multiples appels ultérieurs aux bases de données ou aux systèmes de fichiers. Cela permet de rendre les ressources disponibles et libres, pour gérer plus de requêtes. Le CDN (Content Distribution Network) est un autre mécanisme de mise en cache de fichiers statiques, tels que les images et les bibliothèques JavaScript. Celui-ci est disponible sur les serveurs à travers le monde. Il rend également les ressources disponibles et gratuites pour les demandes de clients supplémentaires, ce qui rend les applications hautement évolutives.

Conception N+1

La conception N+1 désigne la mise en place d'une redondance au sein d'un déploiement global pour chaque composant. Cela signifie que la redondance est prévue même lorsqu'elle n'est pas requise. Cela peut revenir à ajouter des machines virtuelles, du stockage et des interfaces réseaux.

C'est en appliquant les bonnes pratiques susmentionnées lors de la conception des charges de travail à l'aide de VMSS que vous pourrez améliorer l'évolutivité de vos applications. La section suivante porte sur la surveillance.

Surveillance

La surveillance est un enjeu essentiel en termes d'architecture, qui doit faire partie de toute solution, que celle-ci soit majeure ou mineure, critique ou non, dans le Cloud ou sur site : il s'agit d'un point qui n'est pas à négliger.

La surveillance consiste à suivre les solutions et à saisir diverses informations de télémétrie, à les traiter, à identifier celles qui justifient une alerte en fonction de certaines règles et à les enclencher en conséquence. En règle générale, un agent est déployé au sein de l'environnement afin d'assurer le suivi et d'envoyer les informations de télémétrie à un serveur centralisé, où le reste des opérations de traitement permettant de générer les alertes et ainsi d'informer les parties prenantes seront réalisées.

La surveillance permet de prendre des mesures et d'entreprendre des actions proactives et réactives au niveau d'une solution. Il s'agit également de la première étape vers la vérifiabilité d'une solution. Sans la surveillance des dossiers de fichiers journaux, il est difficile d'auditer un système depuis divers points de vue tels que la sécurité, les performances et la disponibilité.

La surveillance permet d'identifier les problèmes de disponibilité, de performances et d'évolutivité avant même qu'ils ne surviennent. Toute panne matérielle et tout problème de configuration matérielle ou de mise à jour de correctifs peut être détecté(e) bien avant que cela n'affecte les utilisateurs, grâce à la surveillance, pour corriger toute dégradation des performances avant même qu'elle ne survienne.

La surveillance des journaux permet d'identifier, de manière réactive, des zones et des emplacements qui sont à l'origine de problèmes afin d'y remédier plus rapidement et plus efficacement.

Les équipes peuvent identifier des modèles de problèmes récurrents à l'aide des données de télémétrie et ainsi les éliminer en développant de nouvelles solutions et fonctions.

Azure est un environnement de Cloud enrichi qui fournit plusieurs fonctions et ressources efficaces afin d'assurer la surveillance, non seulement au niveau des déploiements dans le Cloud, mais également sur ceux sur site.

Surveillance azure

La première question à se poser est la suivante : « Que doit-on surveiller ? » Cette question est d'autant plus importante pour les solutions qui sont déployées dans le Cloud, car celles-ci ne permettent qu'un contrôle limité.

Certains éléments importants doivent être surveillés. En voici quelques-uns :

- Applications personnalisées
- Ressources Azure
- Systèmes d'exploitation invités (machines virtuelles)
- Systèmes d'exploitation hôtes (serveurs physiques Azure)
- Infrastructure Azure

Azure propose différents services de journalisation et de surveillance pour ces composants. Ceux-ci sont abordés dans les sections suivante.

Journaux d'activités Azure

Auparavant dénommés journaux d'audit et journaux opérationnels, il s'agit d'événements de contrôle dans la plateforme Azure. Ils fournissent des informations et des données de télémétrie au niveau de l'abonnement et non pas au niveau de la ressource individuelle. Ils établissent le suivi des informations concernant toutes les modifications survenues au niveau de l'abonnement, telles que la création, la suppression et la mise à jour des ressources à l'aide d'**Azure Resource Manager** (**ARM**). Les journaux d'activités permettent de déterminer l'identité (principal de service, utilisateurs, groupes) et d'entreprendre des actions (écriture ou mise à jour) sur des ressources (par exemple, stockage, machines virtuelles ou bases de données SQL) à un moment donné dans le temps. Ils fournissent des informations sur les ressources qui ont été modifiées au niveau de leur configuration, mais pas sur leur fonctionnement et sur leur exécution internes. Vous pouvez par exemple obtenir les journaux pour démarrer, redimensionner ou arrêter une machine virtuelle.

Le prochain sujet que nous allons aborder concerne les journaux de diagnostic.

Journaux de diagnostic Azure

Les informations provenant du fonctionnement interne des ressources Azure sont capturées et sont désignées sous le terme **journaux de diagnostic**. Ceux-ci fournissent des données de télémétrie sur l'exploitation des ressources qui sont inhérentes celles-ci. Les ressources ne fournissent pas toutes les journaux de diagnostic, et le contenu de ces derniers est écrit dans un langage qui lui est propre, lequel différera des autres ressources. Les journaux de diagnostic sont configurés individuellement pour chaque ressource. L'enregistrement d'un fichier dans le conteneur d'un service d'objet blob d'un compte de stockage est un exemple de journal de diagnostic.

Les journaux d'application sont le prochain type de journal que nous allons évoquer.

Journaux des applications Azure

Les journaux d'application peuvent être saisis par ressources Application Insights et peuvent être gérés de manière centralisée. Ils permettent d'obtenir des informations sur le fonctionnement interne des applications personnalisées, telles que les mesures de performance, la disponibilité et bien plus encore, qui pourront être exploitées afin d'optimiser la gestion.

Enfin, nous allons discuter des journaux des systèmes d'exploitation invité et hôte. Découvrons en quoi ils consistent.

Journaux des systèmes d'exploitation invité et hôte

Les journaux des systèmes d'exploitation invité et hôte sont mis à la disposition des utilisateurs à l'aide d'Azure Monitor. Ils fournissent des informations sur l'état des systèmes d'exploitation hôte et invité :

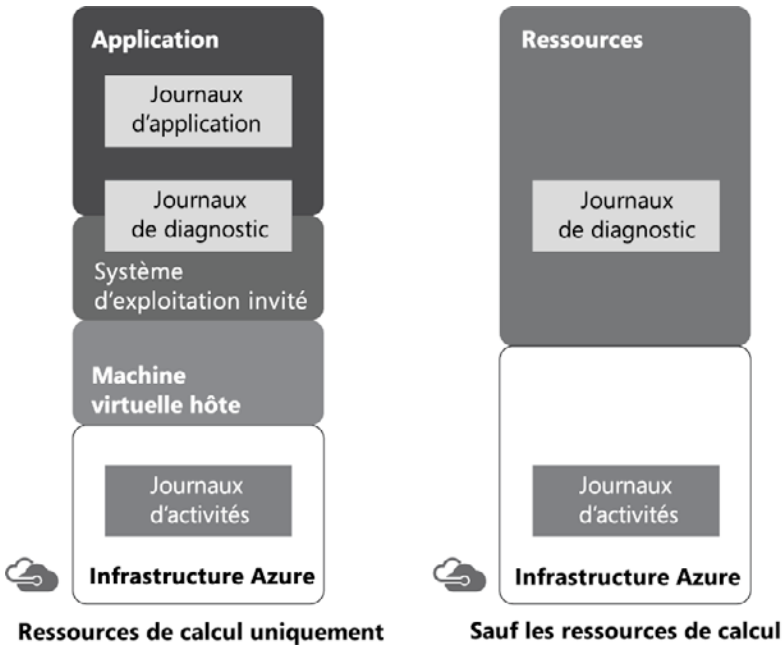


Figure 2.16 : Connexion à Azure

Les ressources Azure importantes associées à la surveillance sont Azure Monitor, Azure Application Insights et log Analytics, précédemment connu sous le nom de **Operational Insights**.

Il existe d'autres outils tels que **System Center Operations Manager** (**SCOM**), qui ne font pas partie de la fonction Cloud, mais qui peuvent être déployés sur des machines virtuelles basées sur IaaS pour surveiller les charges de travail présentes sur Azure ou sur le datacenter sur site. Les trois ressources de surveillance sont abordées dans la section suivante.

Azure Monitor

Azure Monitor est un outil central et une ressource qui fournit toutes les fonctions de gestion nécessaires à la mise en place de la surveillance sur un abonnement Azure. Il fournit des fonctions de gestion des journaux d'activité, des journaux de diagnostic, des mesures, Application Insights et Log Analytics. Il doit être envisagé comme une ressource de tableau de bord et de gestion, pour toutes les autres capacités de surveillance.

Intéressons-nous désormais à Azure Application Insights.

Azure Application Insights

Azure Application Insights permet de mettre en place des capacités centralisées, à l'échelle d'Azure, destinées à la surveillance, à la journalisation et à la mesure des applications personnalisées. Les applications personnalisées peuvent envoyer des mesures, des fichiers journaux et d'autres données de télémétrie à Azure Application Insights. Cette solution fournit également des fonctions de reporting, de création de tableaux de bord et d'analytique enrichies permettant d'exploiter les données entrantes et d'agir en conséquence.

Maintenant que nous avons évoqué Application Insights, examinons un autre service similaire appelé Azure log Analytics.

Azure Log Analytics

Azure Log Analytics assure le traitement centralisé des journaux et la génération de données et d'alertes en conséquence. Les journaux d'activité, de diagnostic, d'application, d'événement et même les journaux personnalisés peuvent envoyer des informations à Log Analytics qui fournira ensuite des capacités enrichies en termes de reporting, de création de tableaux de bord et d'analytique, afin d'interpréter les données entrantes et d'agir en conséquence.

Maintenant que nous connaissons l'objectif de log Analytics, discutons du stockage des journaux dans un espace de travail Log Analytics et de la façon dont ils peuvent être interrogés.

Fichiers journaux

L'espace de travail Log Analytics fournit des capacités de recherche, afin de trouver des entrées du journal spécifiques, d'exporter toutes les données de télémétrie vers Excel et/ou vers Power BI et pour rechercher un langage de requête similaire à SQL, dénommé **langage de requête Kusto** (**KQL**).

L'écran **Recherche de fichiers journaux** est illustré ci-après :

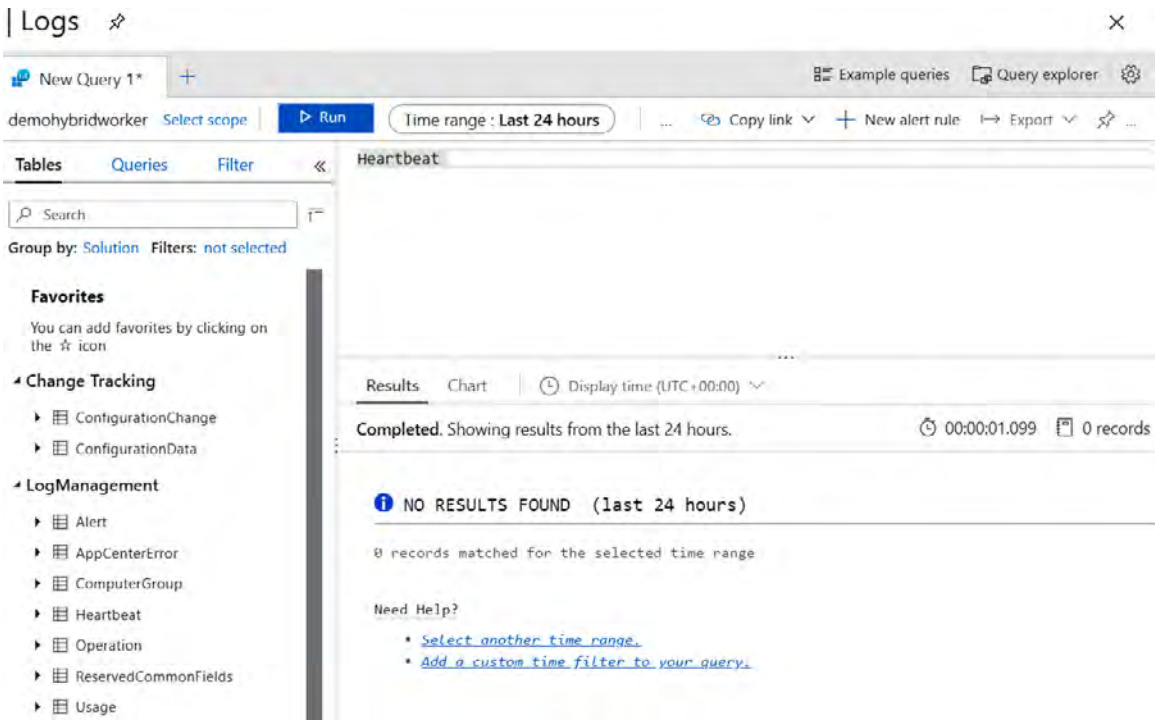


Figure 2.17 : recherche de fichiers journaux dans un espace de travail Log Analytics

La section suivante couvre les solutions log Analytics, qui ressemblent à des fonctionnalités supplémentaires dans un espace de travail log Analytics.

Solutions

Les solutions dans Log Analytics sont des fonctions supplémentaires qui peuvent être ajoutées à un espace de travail en saisissant des données de télémétrie supplémentaires, qui ne sont pas capturées par défaut. Une fois ces solutions ajoutées à un espace de travail, des paquets de gestion appropriés sont envoyés à tous les agents connectés à l'espace de travail afin de les configurer de sorte qu'ils saisissent des données spécifiques aux solutions issues des machines virtuelles et des conteneurs, puis qu'ils les envoient à l'espace de travail Log Analytics. Les solutions de surveillance de Microsoft et des partenaires sont disponibles sur Azure Marketplace.

Azure fournit de nombreuses solutions Log Analytics destinées à suivre et à surveiller différents aspects des environnements et des applications. Au minimum, un ensemble de solutions génériques et applicables à quasiment n'importe quel environnement doit être ajouté à l'espace de travail.

- Capacité et performance
- Intégrité de l'agent
- Suivi des modifications
- Conteneurs
- Sécurité et audit
- Gestion des mises à jour
- Surveillance des performances réseau

Les alertes constituent un autre aspect clé de la surveillance. Les alertes permettent d'avertir les personnes compétentes lors d'événements surveillés. Nous allons étudier les alertes dans les prochaine section.

Alertes

Log Analytics permet de générer des alertes en fonction des données ingérées. Pour ce faire, il exécute une requête prédéfinie, composée de conditions, sur les données entrantes. Si des données ou un groupe de dossiers correspondent aux conditions de ladite requête, le système génère une alerte. Log Analytics fournit un environnement hautement configurable qui permet de déterminer les conditions déclenchant une alerte, la période de temps durant laquelle la requête doit retourner des dossiers, la période de temps durant laquelle la requête doit être exécutée et l'action à entreprendre lorsque la requête retourne des résultats sous la forme d'une alerte :

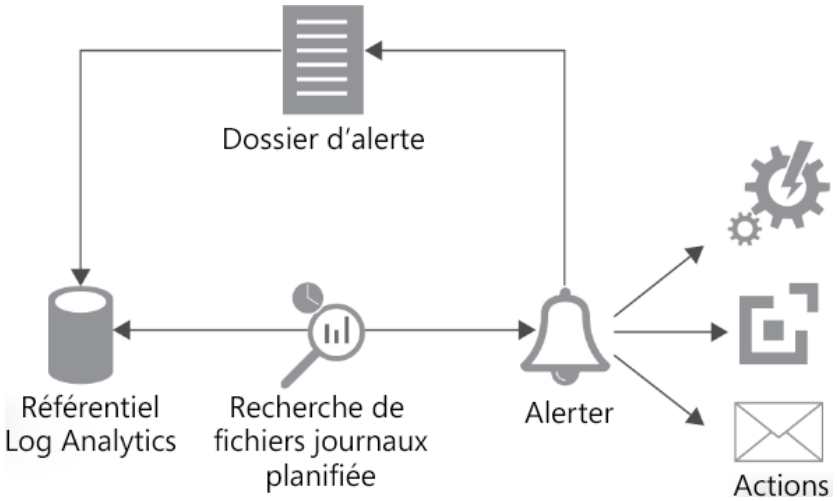


Figure 2.18 : configuration des alertes à l'aide de Log Analytics

Passons en revue les étapes de configuration des alertes via Log Analytics :

1. La première procédure de configuration d'une alerte consiste à ajouter une nouvelle règle d'alerte dans le portail Azure ou d'automatisation dans le menu alerte de la ressource log Analytics.
2. Dans le panneau résultant, sélectionnez la portée de la règle d'alerte. La portée détermine la ressource qui doit être surveillée pour les alertes, il peut s'agir d'une instance de ressource, comme un compte de stockage Azure, un type de ressource, comme une machine virtuelle Azure, un groupe de ressources ou un abonnement :

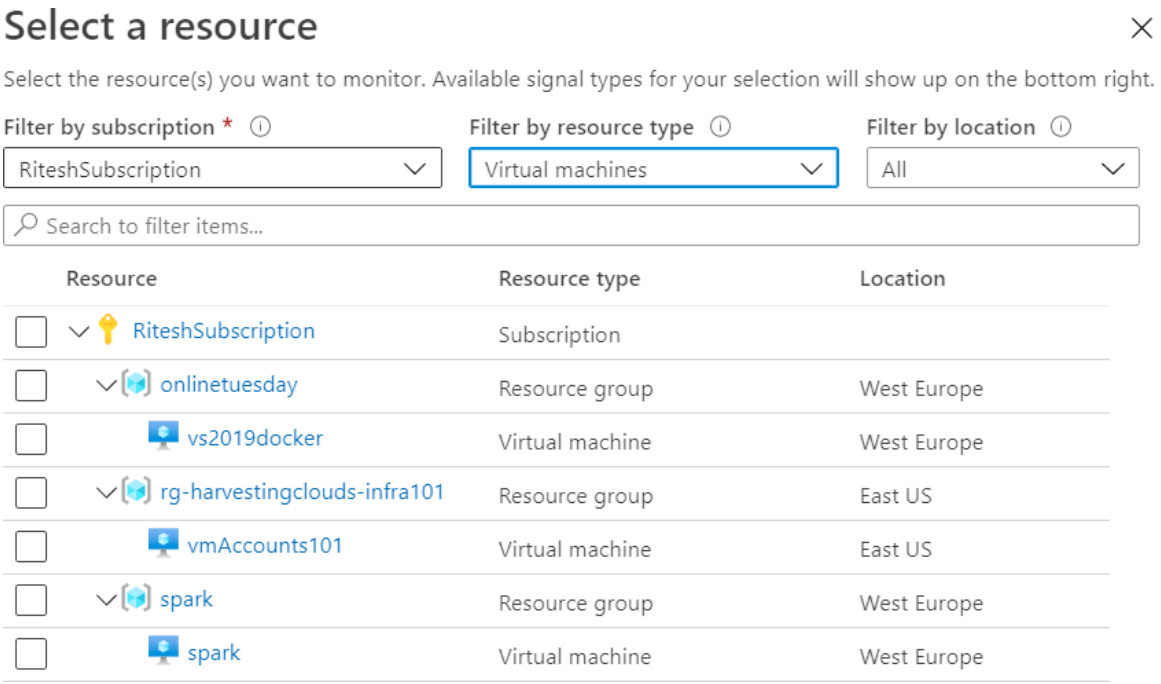


Figure 2.19 : sélection d'une ressource pour l'alerte

3. Une fois la ressource sélectionnée, des conditions doivent être définies pour l'alerte. La condition détermine la règle qui est évaluée par rapport aux journaux et aux mesures de la ressource sélectionnée, et ce n'est qu'une fois que la condition devient vraie qu'une alerte est générée. De nombreux journaux et mesures sont disponibles pour générer des conditions. Dans l'exemple suivant, une alerte est créée avec une valeur de seuil statique de 80 % pour le **pourcentage processeur** (**Moy.**) et les données doivent être collectées toutes les cinq minutes et évaluées chaque minute :

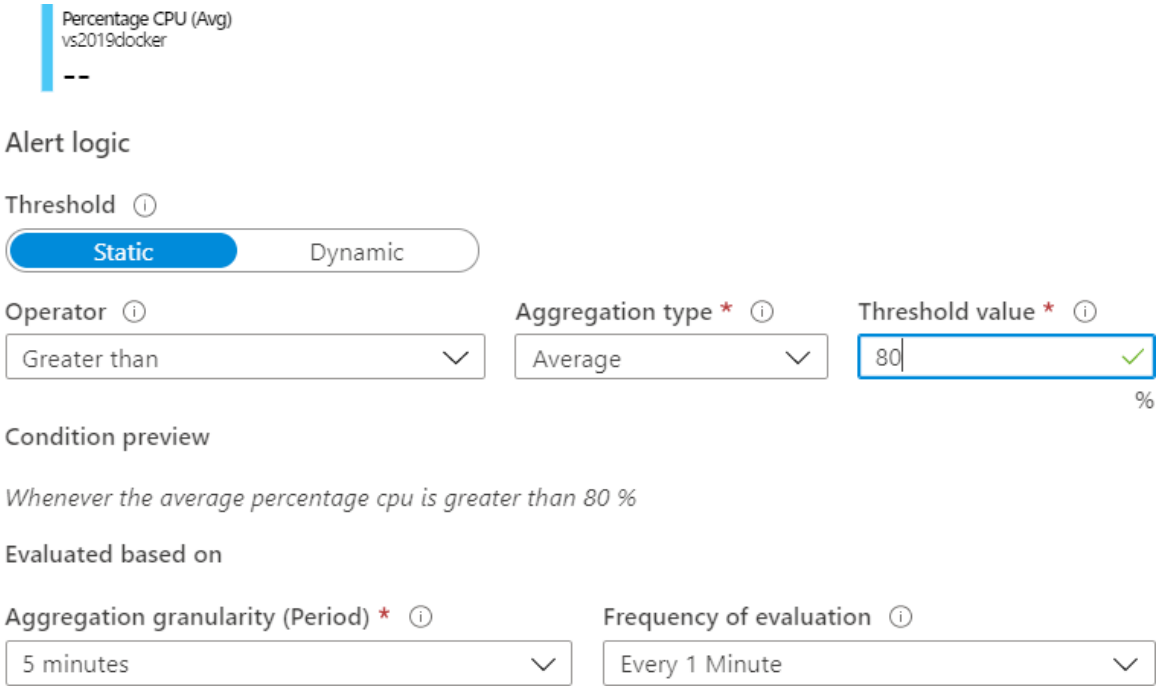


Figure 2.20 : création d'une alerte pour le pourcentage processeur (Moy.)

Les alertes prennent également en charge les seuils dynamiques, qui utilisent le Machine Learning pour apprendre le comportement historique des mesures et détecter les irrégularités susceptibles d'indiquer des problèmes de service.

4. Enfin, créez un groupe d'actions ou réutilisez un groupe existant afin de déterminer les notifications concernant les alertes aux parties prenantes. La section **Action Groups** (Groupes d'action) vous permet de configurer les éléments qui doivent suivre une alerte. En règle générale, une action corrective et/ou une action de notification est nécessaire. Log Analytics fournit huit manières différentes de créer une action. Celles-ci peuvent être assemblées comme bon vous semble. Une alerte exécutera toutes les actions configurées suivantes :
    - **Notification par e-mail/SMS/vocale** : les destinataires configurés reçoivent des notifications vocales ou par e-mail/SMS/push.
    - **Webhook** : un webhook permet d'exécuter tout processus externe arbitraire à l'aide d'un mécanisme HTTP POST. Par exemple, une API REST peut être exécutée, ou les API du gestionnaire de service/ServiceNow peuvent être invoquées pour créer un ticket.
    - **Azure Functions** : ce service exécute une fonction Azure, en transmettant la charge utile nécessaire et en exécutant la logique contenue dans la charge utile.
    - **Logic Apps** : ce service exécute une charge de travail Logic Apps personnalisée.
    - **Envoyer un e-mail au gestionnaire des ressources** : cette option permet d'envoyer des e-mails à une personne assumant le rôle de gestionnaire de ressources Azure, tel qu'un propriétaire, un collaborateur ou un lecteur.
    - **Secure Webhook** : un webhook exécute un processus externe arbitraire à l'aide d'un mécanisme HTTP POST. Les webhooks sont protégés à l'aide d'un fournisseur d'identité, tel qu'Azure Active Directory.
    - **Procédures opérationnelles Automation** : cette action exécute des procédures opérationnelles d'Azure Automation.
    - **ITSM** : les solutions ITSM doivent être mises en service avant d'utiliser cette option. Cela permet de connecter et d'envoyer des informations aux systèmes ITSM.
5. Une fois cette configuration effectuée, vous devez renseigner les champs **Name** (Nom), **Description** et **Severity** (Gravité) relatifs à la règle d'alerte pour générer celle-ci.

Comme nous l'avons mentionné au début de cette section, les alertes jouent un rôle essentiel dans la surveillance et aident le personnel autorisé à prendre les mesures nécessaires en fonction de l'alerte déclenchée.

Résumé

La haute disponibilité et l'évolutivité sont des préoccupations architecturales importantes et cruciales. Quasiment toutes les applications et chaque architecte tente de mettre en œuvre la haute disponibilité. Azure est une plateforme mature qui comprend la nécessité de ces préoccupations architecturales dans les applications et fournit des ressources pour les implémenter à plusieurs niveaux. Ces préoccupations architecturales ne sont pas des éléments secondaires et elles doivent être intégrées au développement du cycle de vie de l'application, et ce dès la phase de planification.

La surveillance est un aspect essentiel de l'architecture de n'importe quelle solution. Il s'agit également de la première étape permettant d'auditer correctement une application. Celle-ci permet aux opérations de gérer la solution, de manière réactive et proactive. Elle fournit les dossiers nécessaires pour le dépannage et la résolution des problèmes qui peuvent survenir sur les plateformes et les applications. Il existe de nombreuses ressources dans Azure qui ont été spécifiquement conçues en vue de la mise en place de la surveillance sur Azure, dans d'autres Clouds et sur les datacenters sur site. Application Insights et Log Analytics sont deux des ressources les plus importantes de ce point de vue. Inutile de dire que la surveillance est un impératif pour optimiser vos solutions et produits, et ce en innovant grâce à des informations pertinentes tirées de la surveillance des données.

Ce chapitre portait uniquement sur la disponibilité, l'évolutivité et la surveillance des solutions, tandis que le chapitre suivant traite des modèles de conception liés aux réseaux virtuels, aux comptes de stockage, aux régions, aux zones de disponibilité et aux ensembles de disponibilité. Ces principes doivent être pris en compte lors de la conception de solutions dans le Cloud pour développer des solutions rentables dotées d'une productivité et d'une disponibilité accrues.

3

Modèle de conception : réseaux, stockage, messagerie et événements

Le chapitre précédent présentait le Cloud Azure ainsi que certains de ses principaux concepts. Ce chapitre aborde les modèles d'application Azure concernant les réseaux virtuels, les comptes de stockage, les régions, les zones de disponibilité et les ensembles de disponibilité. Il s'agit d'éléments cruciaux qui influent sur l'architecture finale délivrée aux clients en termes de coûts, d'efficacité et de productivité globale. Ce chapitre aborde aussi rapidement les modèles de Cloud qui permettent de mettre en place l'évolutivité et les performances au sein d'une architecture.

Dans ce chapitre, nous allons aborder les sujets suivants :

- Conception du réseau virtuel Azure
- Conception du stockage Azure
- Les zones, régions et ensemble de disponibilité Azure
- Modèles de conception Azure relatifs à la messagerie, aux performances et à l'évolutivité

Les zones, régions et ensemble de disponibilité Azure

Azure s'appuie sur de vastes datacenters interconnectés dans un seul grand réseau. Les datacenters sont regroupés en fonction de leur proximité physique dans des régions Azure. Par exemple, des datacenters en Europe occidentale sont disponibles pour les utilisateurs d'Azure dans la région d'Europe occidentale. Les utilisateurs ne peuvent pas choisir leur datacenter préféré. Ils peuvent sélectionner leur région Azure et Azure leur attribuera le datacenter approprié.

La sélection d'une région appropriée est capitale du point de vue de l'architecture de la solution, car cela aura une incidence sur les éléments suivants :

- Disponibilité des ressources
- Conformité des données et de la confidentialité
- Performances de l'application
- Coût de l'exécution des applications

Examinons chacun de ces points plus en détail.

Disponibilité des ressources

Les ressources ne sont pas disponibles dans toutes les régions Azure. Si l'architecture de votre application exige une ressource qui n'est pas disponible dans une région, il est inutile d'opter pour cette région. Au lieu de cela, une région doit être choisie en fonction de la disponibilité des ressources requises par l'application. Il se peut que la ressource ne soit pas disponible durant le développement de l'architecture de l'application et que la feuille de route Azure la rende disponible par la suite.

Par exemple, Log Analytics n'est pas disponible dans toutes les régions. Si vos sources de données se trouvent dans la région A et si l'espace de travail Log Analytics se trouve dans la région B, vous devez payer la bande passante, c'est-à-dire les frais de sortie des données de la région A à la région B. De même, certains services peuvent fonctionner avec des ressources qui se trouvent dans la même région. Par exemple, si vous souhaitez chiffrer les disques de votre machine virtuelle déployée dans la région A, Azure Key Vault doit être déployé dans la région A pour stocker les clés de chiffrement. Avant de déployer des services, vous devez vérifier si vos services de dépendance sont disponibles dans cette région. Cette page de produit permet de vérifier la disponibilité des produits Azure dans différentes régions : https://azure.microsoft.com/global-infrastructure/services.

Conformité des données et de la confidentialité

Chaque pays dispose de ses propres règles concernant la conformité des données et de la confidentialité. Certains pays sont très précis sur le stockage des données de leurs citoyens sur leurs propres territoires. Par conséquent, ces obligations légales doivent être prises en considération lors de l'élaboration de l'architecture de chaque application.

Performances des applications

La performance d'une application dépend du cheminement du réseau emprunté par les requêtes et les réponses pour obtenir leurs destinations et vice-versa. L'emplacement géographique le plus proche de vous n'est peut-être pas toujours la région qui présente la latence est la plus faible. Nous calculons la distance en kilomètres ou en milles, mais la latence est basée sur l'itinéraire emprunté par le paquet. Par exemple, une application déployée en Europe occidentale pour les utilisateurs situés en Asie du Sud-Est ne sera pas aussi performante qu'une application déployée dans la région Asie de l'est pour les utilisateurs situés dans cette même région. Il est donc très important que vous conceviez vos solutions dans la région la plus proche pour obtenir la latence la plus faible et donc les meilleures performances.

Coût de l'exécution des applications

Le coût des services Azure diffère d'une région à l'autre. Il convient d'opter pour une région dont les coûts globaux sont abordables. Un chapitre entier de ce manuel aborde la gestion des coûts (*chapitre 6, Gestion des coûts des solutions Azure*), et fournit davantage de détails du point de vue des coûts.

Jusqu'à présent, nous avons expliqué comment choisir la région appropriée pour concevoir notre solution. Maintenant que nous avons déterminé la région appropriée pour notre solution, étudions comment concevoir nos réseaux virtuels dans Azure.

Réseaux virtuels

Les réseaux virtuels doivent être considérés comme une configuration de réseau LAN physique dans votre bureau ou votre domicile. D'un point de vue conceptuel, ils sont tous deux identiques, bien que le **réseau virtuel Azure** (**VNet**) soit implémenté comme un réseau Software-defined soutenu par une infrastructure réseau physique gigantesque.

Un réseau virtuel est nécessaire pour l'hébergement d'une machine virtuelle. Il fournit un mécanisme de communication sécurisée entre les ressources Azure, afin qu'elles se connectent entre elles. Les réseaux virtuels fournissent des adresses IP internes aux ressources, facilitent l'accès et la connectivité à d'autres ressources, y compris les machines virtuelles sur le même réseau virtuel, acheminent les requêtes et garantissent la connectivité à d'autres réseaux.

Un réseau virtuel se trouve dans un groupe de ressources et est hébergé au sein d'une région, par exemple, Europe occidentale. Il ne peut pas s'étendre sur plusieurs régions, mais peut couvrir tous les datacenters au sein d'une région. Nous pouvons donc étendre les réseaux virtuels sur plusieurs zones de disponibilité dans une région. Concernant la connectivité sur l'ensemble des régions, les réseaux virtuels peuvent être connectés en utilisant la connectivité VNet-à-VNet.

Les réseaux virtuels garantissent également la connectivité des datacenters sur site, activant ainsi les clouds hybrides. Il existe plusieurs types de technologies VPN permettant d'étendre vos datacenters sur site vers le Cloud, telles que le VPN de site à site et le VPN de point-à-site. Il existe également une connectivité dédiée entre le réseau virtuel Azure et les réseaux sur site à l'aide d'ExpressRoute.

Les réseaux virtuels sont gratuits. Chaque abonnement peut créer jusqu'à 50 réseaux virtuels dans toutes les régions. Toutefois, ce nombre peut être augmenté en contactant le support Azure. Vous ne serez pas facturé si les données ne quittent pas la région de déploiement. Au moment de la rédaction de ce chapitre, les transferts de données entrants et sortants entre les zones de disponibilité de la même région ne sont pas facturés. Ils seront toutefois facturés à partir du 1er juillet 2020.

Des informations sur les limites de mise en réseau sont indiquées dans la documentation Microsoft à l'adresse https://docs.microsoft.com/azure/azure-resource-manager/management/azure-subscription-service-limits.

Considérations architecturales concernant les réseaux virtuels

Les réseaux virtuels, comme toute autre ressource, peuvent être configurés à l'aide de modèles ARM, des API REST, de PowerShell et de l'interface CLI. Il est très important de planifier la topologie du réseau dès que possible, afin d'éviter tout désagrément ultérieur durant le cycle de vie du développement. En effet, une fois qu'un réseau est mis en service et que les ressources sont utilisées, il est difficile d'y apporter des modifications sans imposer un temps d'arrêt. Par exemple, pour déplacer une machine virtuelle d'un réseau à un autre, il est nécessaire de mettre cette machine virtuelle hors tension.

Examinons quelques-unes des principales considérations architecturales lors de la conception d'un réseau virtuel.

Régions

Le réseau virtuel est une ressource Azure et est mis en service au sein d'une région, par exemple l'Europe occidentale. Les applications s'étendant sur plusieurs régions nécessiteront des réseaux virtuels distincts, un par région, et auront également besoin d'être connectées à l'aide de la connectivité VNet-à-VNet. La connectivité VNet-à-VNet implique des coûts pour le trafic entrant et sortant. Aucuns frais ne sont facturés pour les données entrantes, contrairement aux données sortantes.

DNS dédié

Le réseau virtuel par défaut utilise le DNS Azure pour résoudre les noms au sein d'un réseau virtuel ainsi que pour la résolution de noms sur Internet. Si une application souhaite utiliser un service de résolution de nom dédié ou se connecter à des datacenters sur site, elle doit mettre en service son propre serveur DNS, lequel doit être configuré dans le réseau virtuel en vue d'une résolution de nom réussie. En outre, vous pouvez héberger votre domaine public dans Azure et gérer complètement les enregistrements à partir du portail Azure, sans devoir gérer d'autres serveurs DNS.

Nombre de réseaux virtuels

Le nombre de réseaux virtuels dépend du nombre de régions, de la consommation de bande passante par les services, de la connectivité entre régions et de la sécurité. Les frais de gestion associés aux réseaux virtuels moins nombreux mais plus importants sont inférieurs à ceux liés à plusieurs réseaux virtuels plus petits.

Nombre de sous-réseaux dans chaque réseau virtuel

Les sous-réseaux fournissent une isolation au sein d'un réseau virtuel. Ils garantissent également une limite de sécurité. **Les groupes de sécurité réseau** (**NSG**) peuvent être associés à des sous-réseaux, ce qui limite ou autorise un accès spécifique aux adresses IP et aux ports. Les composants de l'application associés à des exigences de sécurité et d'accessibilité distinctes doivent être placés dans des sous-réseaux distincts.

Plages IP pour les réseaux et les sous-réseaux

Chaque sous-réseau présente une plage d'adresses IP. La plage d'adresses IP ne doit pas être trop large afin que les IP ne soient pas sous-utilisées, et inversement, ne doit pas être trop petite et encombrer les sous-réseaux en raison du manque d'adresses IP. Il s'agit d'un point à prendre en compte après avoir déterminé les exigences futures du déploiement en termes d'adresses IP.

Une planification doit être effectuée pour les adresses IP et les plages des réseaux Azure, les sous-réseaux et les datacenters sur site. Il convient d'éviter tout chevauchement pour garantir une connectivité et une accessibilité sans effort.

Surveillance

Suivi La surveillance est un aspect architectural essentiel, à inclure dans le déploiement global. Azure Network Watcher offre des capacités de journalisation et de diagnostic qui fournissent des données sur les performances et sur l'intégrité du réseau. Voici quelques-unes des fonctionnalités d'Azure Network Watcher :

- Diagnostic des problèmes de filtrage du trafic réseau vers ou à partir d'une machine virtuelle
- Détermination du prochain saut des itinéraires définis par l'utilisateur
- Affichage des ressources d'un réseau virtuel et de leurs relations
- Surveillance des communications entre une machine virtuelle et un point de terminaison
- Capture du trafic à partir d'une machine virtuelle
- Journaux de flux de NSG, qui consignent les informations relatives au trafic circulant via un NSG. Ces données seront stockées dans le stockage Azure à des fins d'analyses supplémentaires.

Il fournit également des journaux de diagnostic pour toutes les ressources réseau d'un groupe de ressources.

Les performances du réseau peuvent être surveillées via Log Analytics. La solution de gestion Network Performance Monitor fournit des capacités de surveillance du réseau. Elle permet de surveiller l'intégrité, la disponibilité et l'accessibilité des réseaux. Elle permet également de contrôler la connectivité entre le Cloud public et local et entre les sous-réseaux hébergeant les différentes couches des applications multicouches.

Considérations en matière de sécurité

Les réseaux virtuels sont parmi les premiers éléments auxquels accède n'importe quelle ressource sur Azure. La sécurité joue un rôle essentiel pour autoriser ou refuser l'accès à une ressource. Les groupes de sécurité de réseau (NSGs) constituent les principaux moyens de garantir la sécurité des réseaux virtuels. Ils peuvent être associés à des sous-réseaux de réseau virtuel, et chaque flux entrant et sortant est contraint, filtré et autorisé en fonction.

Le **routage défini par l'utilisateur** (**UDR**) et le transfert d'adresse IP permettent également de filtrer et d'acheminer les requêtes de ressources sur Azure. Pour en savoir plus sur les UDR et le tunneling forcé, consultez la page https://docs.microsoft.com/azure/virtual-network/virtual-networks-udr-overview.

Le pare-feu Azure est une offre de pare-feu entièrement géré en tant que service par Azure. Il permet de protéger les ressources de votre réseau virtuel. Le pare-feu Azure peut notamment être utilisé pour le filtrage des paquets au niveau du trafic entrant et sortant. En outre, la fonction de renseignements sur les menaces du pare-feu Azure peut être utilisée pour alerter et refuser le trafic depuis ou vers des adresses IP ou des domaines malveillants. La source de données des adresses IP et des domaines correspond au flux de renseignements sur les menaces de Microsoft.

Les ressources peuvent également être obtenues et protégées via le déploiement d'appareils réseau (https://azure.microsoft.com/solutions/network-appliances) comme Barracuda, F5 et d'autres composants tiers.

Déploiement

Les réseaux virtuels doivent être déployés dans leurs propres groupes de ressources dédiés. Les administrateurs réseau doivent obtenir l'autorisation du propriétaire pour utiliser ce groupe de ressources, tandis que les développeurs ou les membres de l'équipe doivent disposer d'autorisations de collaborateur pour pouvoir créer d'autres ressources Azure dans les autres groupes de ressources consommant les services du réseau virtuel.

Il est également conseillé de déployer des ressources avec des adresses IP statiques dans un sous-réseau dédié, tandis que les ressources liées à des adresses IP dynamiques peuvent se trouver sur un autre sous-réseau.

Des stratégies doivent être créées afin que seuls les administrateurs réseau puissent supprimer le réseau virtuel, mais également être marquées à des fins de facturation.

Connectivité

Les ressources d'une région sur un réseau virtuel peuvent communiquer en toute transparence. Même des ressources se trouvant sur d'autres sous-réseaux au sein d'un réseau virtuel peuvent communiquer entre elles, sans aucune configuration explicite. Des ressources issues de régions différentes ne peuvent pas utiliser le même réseau virtuel. La limite d'un réseau virtuel se trouve au sein d'une région. Pour garantir la communication des ressources entre les régions, des passerelles dédiées doivent se trouver aux deux extrémités pour faciliter la conversation.

Cela dit, si vous souhaitez initier une connexion privée entre deux réseaux dans différentes régions, vous pouvez utiliser un appairage de réseaux virtuels mondial. Dans le cadre d'un appairage de réseaux virtuels mondial, la communication est exécutée via le réseau principal de Microsoft. Ainsi, la communication ne requiert aucun Internet public, aucune passerelle ou aucun chiffrement. Si vos réseaux virtuels se trouvent dans la même région et présentent des espaces d'adressage différents, les ressources d'un réseau ne seront pas en mesure de communiquer entre elles. Étant donné qu'elles se trouvent dans la même région, nous pouvons utiliser l'appairage de réseau virtuel, qui est similaire à l'appairage de réseaux virtuels mondial. la seule différence est que les réseaux virtuels source et de destination sont déployés dans la même région.

De nombreuses organisations disposent d'un Cloud hybride, par conséquent, les ressources Azure doivent parfois communiquer ou se connecter à des datacenters sur site, ou vice-versa. Les réseaux virtuels Azure peuvent se connecter aux datacenters sur site à l'aide de la technologie VPN et ExpressRoute. En effet, un réseau virtuel est capable de se connecter à plusieurs datacenters sur site et à d'autres régions Azure en parallèle. À titre de bonne pratique, chacune de ces connexions doit se trouver dans son sous-réseau dédié, au sein d'un réseau virtuel.

Maintenant que nous avons exploré plusieurs aspects de la mise en réseau virtuelle, découvrons les avantages des réseaux virtuels.

Avantages des réseaux virtuels

Les réseaux virtuels constituent un composant essentiel pour le déploiement de toute solution IaaS significative. Les machines virtuelles ne peuvent pas être mises en service sans réseau virtuel. Outre le fait qu'il s'agisse d'un composant impératif dans les solutions IaaS, ils offrent des avantages conséquents en termes d'architecture, notamment :

- **Isolation** : la plupart des composants d'application sont assortis d'exigences distinctes en matière de sécurité et de bande passante et suivent des cycles de vie différents. Les réseaux virtuels permettent de créer des poches isolées pour ces composants, qui peuvent être gérées indépendamment des autres composants à l'aide des réseaux virtuels et des sous-réseaux.
- **Sécurité** : le filtrage et le suivi des utilisateurs accédant aux ressources constituent des aspects essentiels, assurés par les réseaux virtuels. Ces derniers peuvent bloquer l'accès d'adresses IP et de ports malveillants.
- **Extensibilité** : les réseaux virtuels agissent comme un LAN privé sur le Cloud. Ils peuvent également être étendus à une **Connectivité de réseau étendu** (**WAN**) en connectant d'autres réseaux virtuels à travers le monde et en tant qu'extension des datacenters sur site.

Nous avons découvert les avantages des réseaux virtuels. Intéressons-nous désormais à la façon de tirer parti de ces avantages et de concevoir un réseau virtuel pour héberger notre solution. La section suivante porte sur la conception des réseaux virtuels.

Conception d'un réseau virtuel

Dans cette section, nous allons examiner certains des éléments conceptuels et des cas d'utilisation les plus courants des réseaux virtuels.

Les réseaux virtuels peuvent être utilisés de diverses manières. Une passerelle peut être déployée à chaque point de terminaison de réseau virtuel pour assurer la sécurité et transmettre les paquets en assurant l'intégrité et la confidentialité. Une passerelle est un élément essentiel pour connecter les réseaux sur site, cependant, il s'agit d'un élément facultatif pour l'appairage de réseau virtuel Azure. En outre, vous pouvez utiliser la fonctionnalité Gateway Transit pour simplifier le processus d'extension de votre datacenter sur site sans déployer plusieurs passerelles. Gateway Transit vous permet de partager une passerelle ExpressRoute ou VPN avec tous les réseaux virtuels appariés. Vous pourrez ainsi faciliter la gestion et réduire le coût de déploiement de plusieurs passerelles.

Dans la section précédente, nous avons évoqué l'appairage et mentionné que nous n'utilisons pas les passerelles ou l'Internet public pour établir la communication entre les réseaux appariés. Découvrons à présent certains aspects de conception de l'appairage, en déterminant l'appairage qui doit être utilisé dans des scénarios particuliers.

Connexion aux ressources au sein de la même région et du même abonnement

Plusieurs réseaux virtuels au sein d'une même région et d'un même abonnement peuvent être reliés entre eux. Grâce à l'appairage des réseaux virtuels, les deux réseaux peuvent être connectés et utiliser le réseau privé dorsal d'Azure pour la transmission de paquets entre eux. Les machines virtuelles et les services présents sur ces réseaux peuvent communiquer entre eux, sous réserve des contraintes liées au trafic réseau. Dans le diagramme suivant, VNet1 et VNet2 sont tous deux déployés dans la région ouest des États-Unis. Toutefois, l'espace d'adressage de VNet1 est 172.16.0.0/16, et 10.0.0.0/16 pour VNet2. Par défaut, les ressources de VNet1 ne seront pas en mesure de communiquer avec les ressources de VNet2. Étant donné que nous avons établi l'appairage des deux réseaux virtuels, les ressources seront en mesure de communiquer les unes avec les autres via le réseau de base de Microsoft :

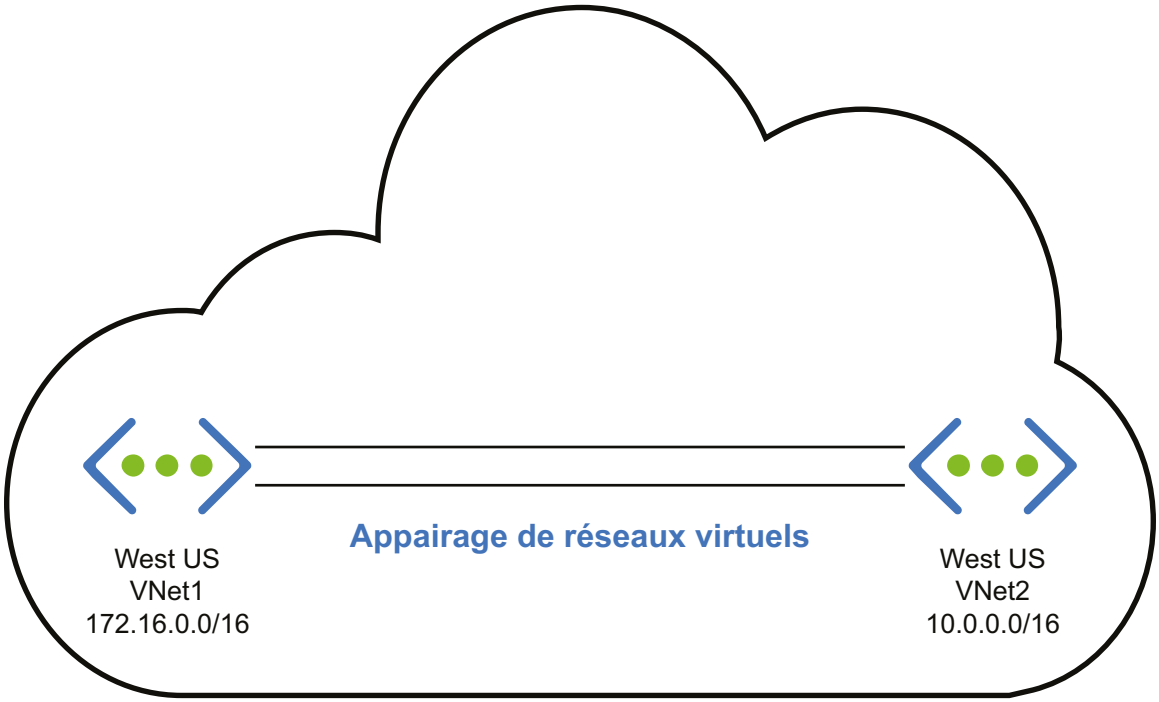


Figure 3.1 : appairage de réseaux virtuels pour les ressources appartenant au même abonnement

Connexion aux ressources au sein de la même région, dans un autre abonnement

Ce scénario est très similaire au précédent, sauf que les réseaux virtuels sont hébergés dans deux abonnements différents. Les abonnements peuvent appartenir au même locataire ou à plusieurs locataires. Si les deux ressources font partie du même abonnement et de la même région, le scénario précédent s'applique. Ce scénario peut être mis en œuvre de deux façons : à l'aide de passerelles ou à l'aide de l'appairage de réseaux virtuels.

Si nous utilisons des passerelles dans ce scénario, nous devons déployer une passerelle aux deux extrémités pour faciliter la communication. Voici la représentation architecturale de l'utilisation des passerelles pour relier deux ressources appartenant à des abonnements différents :

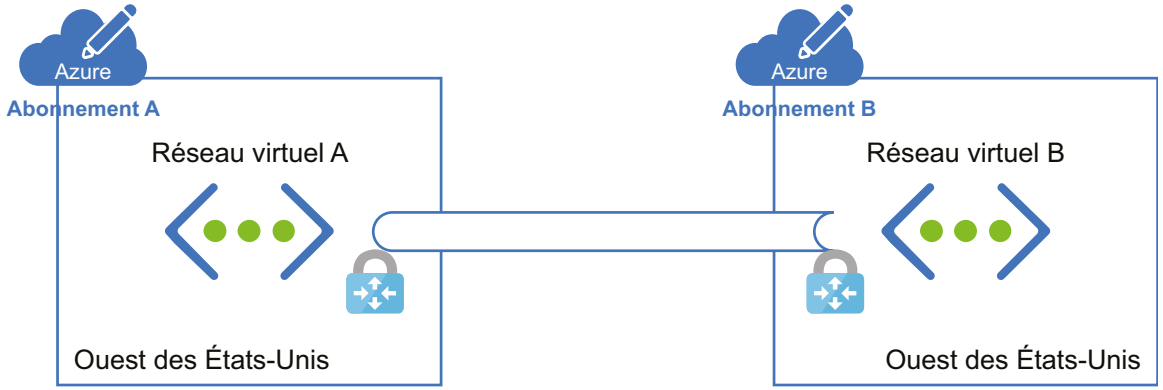


Figure 3.2 : appairage de réseaux virtuels pour des ressources appartenant à des abonnements différents à l'aide de passerelles

Toutefois, le déploiement des passerelles engendre des frais. Nous allons discuter de l'appairage de réseaux virtuels avant de comparer ces deux mises en œuvre pour déterminer ce qui convient le mieux à notre solution.

Aucune passerelle n'est déployée lors de l'appairage. La *Figure* 3.3 illustre la façon dont l'appairage est effectué :

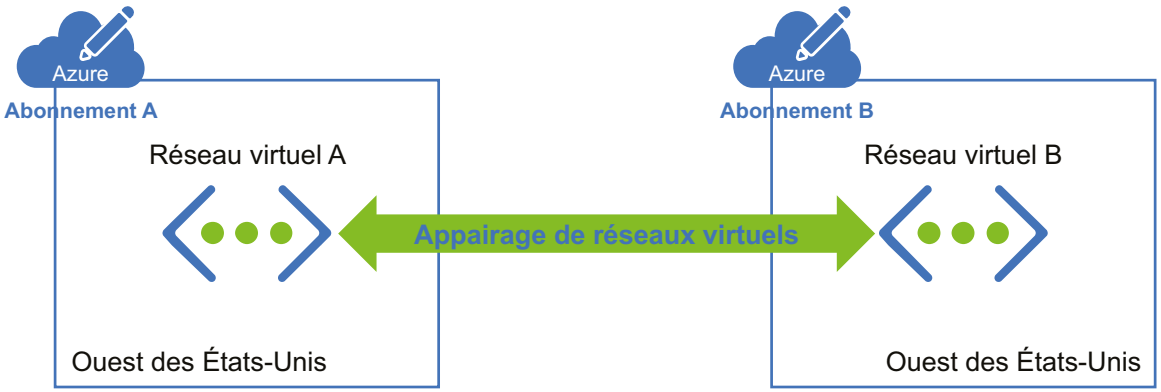


Figure 3.3 : appairage de réseaux virtuels entre les abonnements

L'appairage de réseaux virtuels fournit une connexion à faible latence et à bande passante élevée, et, comme illustré dans le diagramme, aucune passerelle n'est déployée pour garantir la communication. Cette méthode est utile dans les scénarios de réplication de données ou de basculement. Comme nous l'avons mentionné précédemment, l'appairage utilise le réseau de base de Microsoft, en évitant de recourir à l'Internet public.

Les passerelles sont utilisées dans des scénarios où le chiffrement est nécessaire et où la bande passante n'est pas une préoccupation, car il s'agit d'une connexion à bande passante limitée. Cela ne signifie pas que la bande passante est soumise à une contrainte. En outre, cette approche est utilisée lorsque les clients ne sont pas sensibles à la latence.

Jusqu'à présent, nous avons examiné les ressources appartenant à la même région et à des abonnements différents. La section suivante explique comment établir une connexion entre les réseaux virtuels appartenant à deux régions différentes.

Connecter les ressources dans différentes régions et dans un autre abonnement

Dans ce scénario, nous avons à nouveau deux mises en œuvre. L'une utilise une passerelle et l'autre utilise l'appairage de réseaux virtuels mondial.

Le trafic passera par le réseau public, et des passerelles seront déployées aux deux extrémités pour faciliter la connexion chiffrée. La *figure* 3.4 explique comment y parvenir :

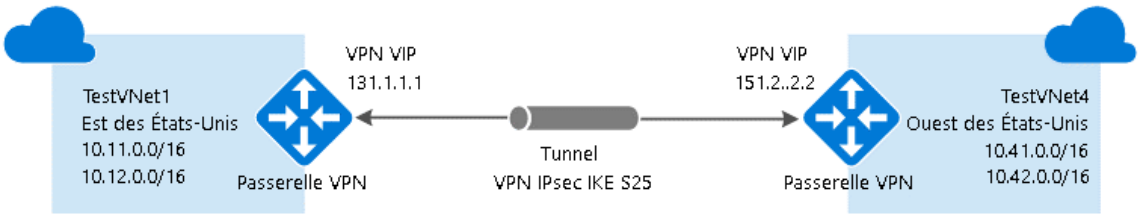


Figure 3.4 : connecter les ressources présentant différents abonnements dans diverses régions

Nous adopterons une approche similaire à l'aide de l'appairage de réseaux virtuels mondial. La *Figure* 3.5 illustre l'appairage de réseaux virtuels mondial :

Figure 3.5 : connexion des ressources appartenant à différentes régions à l'aide d'un appairage de réseaux virtuels mondial

Les considérations relatives au choix des passerelles ou de l'appairage ont déjà été abordées. Ces considérations sont également applicables à ce scénario. Jusqu'à présent, nous avons connecté des réseaux virtuels entre les régions et les abonnements. Nous n'avons pas encore évoqué la connexion d'un datacenter sur site au Cloud. La section suivante explique comment y parvenir.

Connexion à des datacenters sur site

Les réseaux virtuels peuvent être connectés aux datacenters sur site, de sorte qu'Azure et des datacenters sur site deviennent un seul et même réseau étendu (WAN). Des réseaux sur site doivent être déployés sur des passerelles et des VPN aux deux extrémités du réseau. Il existe trois technologies différentes disponibles à cet effet.

VPN de site à site

Une telle méthode doit être utilisée lorsque le réseau Azure et le datacenter sur site sont connectés pour former un réseau étendu, sur lequel n'importe quelle ressource des deux réseaux peut accéder à une autre ressource sur l'un d'eux, qu'ils soient déployés ou non sur un datacenter Azure ou sur site. Les passerelles VPN doivent être disponibles des deux côtés des réseaux, pour des raisons de sécurité. En outre, les passerelles Azure doivent être déployées sur leurs propres sous-réseaux sur les réseaux virtuels connectés aux datacenters sur site. Les adresses IP publiques doivent être affectées aux passerelles sur site afin qu'Azure puisse s'y connecter via le réseau public :

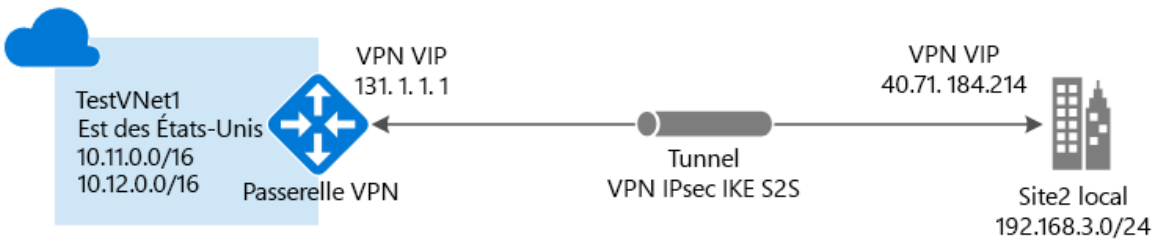


Figure 3.6 : Architecture VPN de site à site

VPN de point à site

Cette approche est similaire à la connectivité VPN de site à site, cependant, un seul serveur ou ordinateur est relié au datacenter sur site. Celui-ci doit être utilisé lorsque très peu d'utilisateurs ou de clients se connectent à Azure de manière sécurisée depuis des sites distants. En outre, il est inutile de disposer d'adresses IP publiques et de passerelles du côté sur site dans ce cas :

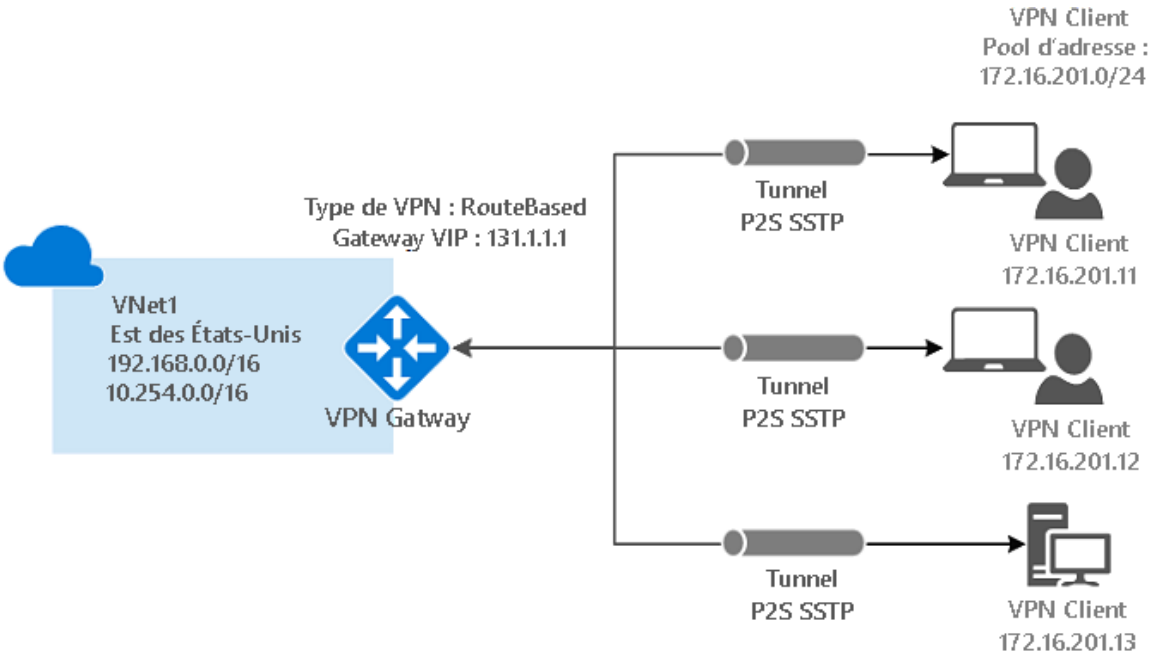


Figure 3.7 : architecture VPN de point à site

ExpressRoute

Les deux VPN site à site et point à site fonctionnent à l'aide de l'Internet public. Ces derniers chiffrent le trafic sur les réseaux, via le VPN et la technologie de certificats. Toutefois, certaines applications doivent être déployées à l'aide de technologies hybrides (certains composants sur Azure et d'autres sur un datacenter sur site), sans utiliser l'Internet public pour se connecter à des datacenters Azure et sur site. Azure ExpressRoute est la meilleure solution pour ces ressources, bien qu'il s'agisse d'une option coûteuse par rapport aux deux types de connexion. Il s'agit également du fournisseur le plus sécurisé et le plus fiable, qui offre une vitesse supérieure et une latence réduite étant donné que le trafic n'atteint jamais l'Internet public. Azure ExpressRoute permet d'étendre les réseaux locaux dans Azure sur une connexion privée dédiée, assurée par un fournisseur de connectivité. Si votre solution est gourmande en réseau, par exemple, une application d'entreprise transactionnelle telle que SAP, l'utilisation d'ExpressRoute est fortement recommandée.

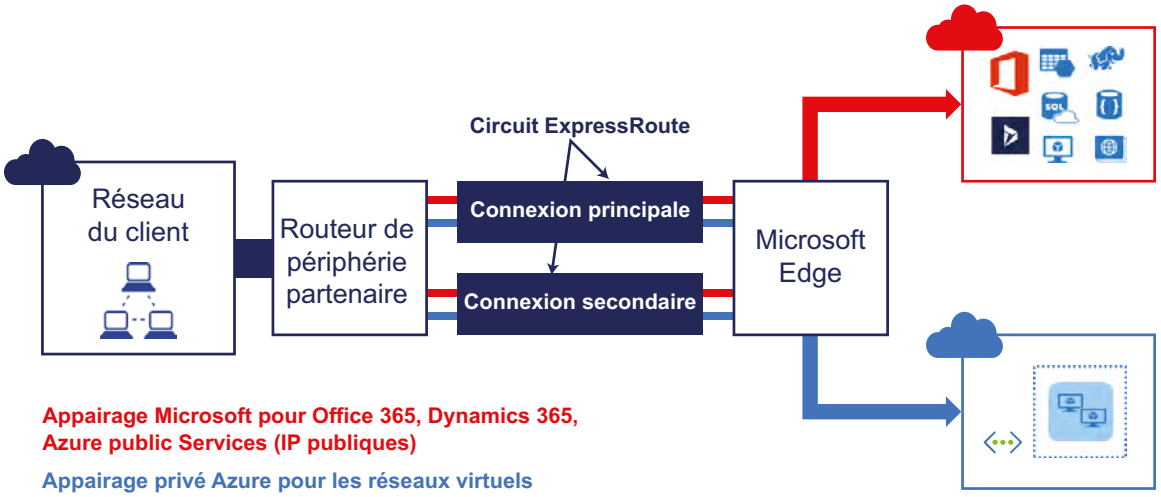


Figure 3.8 : architecture de réseau ExpressRoute

La Figure 3.9 illustre les trois types de réseaux hybrides :

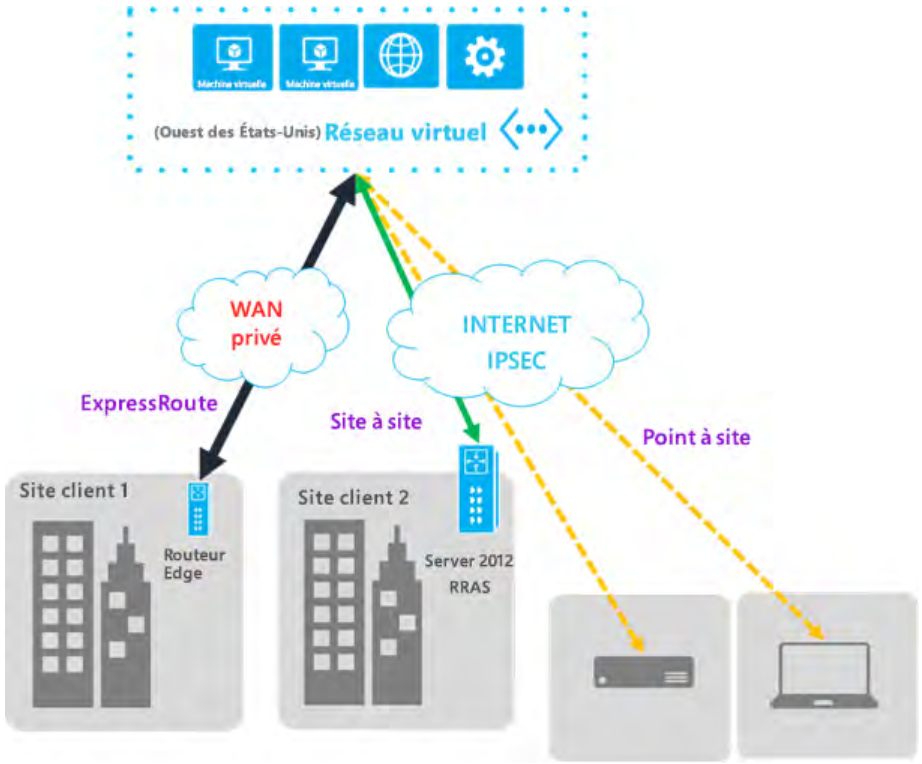


Figure 3.9 : différents types de réseaux hybrides

À titre de bonne pratique, les réseaux virtuels possèdent des sous-réseaux distincts pour chacun des composants logiques assortis de déploiements distincts du point de vue de la sécurité et de l'isolation.

Toutes les ressources que nous déployons dans Azure nécessitent une mise en réseau d'une manière ou d'une autre. Ainsi, il est nécessaire de bien comprendre la mise en réseau pour concevoir des solutions dans Azure. Le stockage constitue un autre élément clé. Vous en apprendrez davantage sur le stockage dans la section suivante.

Stockage

Azure fournit une solution de stockage évolutive, hautement disponible et durable via des services de stockage.

Le stockage est utilisé pour conserver les données à long terme. Le stockage Azure est disponible via Internet dans quasiment tous les langages de programmation.

Catégories de stockage

Le stockage se décline en deux catégories de comptes de stockage :

- Un niveau de performance de stockage standard qui vous permet d'enregistrer des tables, des files d'attente, des fichiers, des blobs et des disques de machine virtuelle Azure.
- Un niveau de performance de stockage premium prenant en charge les disques de machine virtuelle Azure au moment de l'écriture. Le stockage premium offre de meilleures performances et un montant maximal d'opérations d'entrée/sortie par seconde par rapport au stockage général standard. Le stockage premium est actuellement disponible sous forme de disques de données pour les machines virtuelles sauvegardés au format SSD.

Selon le type de données stockées, le stockage est classé en différentes catégories. Examinons les types de stockage de plus près.

Types de stockage

Azure propose quatre types de services de stockage généraux :

- **Azure Blob storage :** ce type de stockage convient particulièrement aux données non structurées, telles que les documents, les images et d'autres types de fichiers. Le stockage blob s'applique au niveau Archive, Chaud ou Froid. Le niveau de stockage à chaud est conçu pour stocker les données qui doivent être consultées très fréquemment. Le niveau de stockage à froid est destiné aux données qui sont consultées moins fréquemment que celles qui sont stockées dans le niveau de stockage à chaud et qui sont stockées pendant 30 jours. Enfin, le niveau Archive est utilisé à des fins d'archivage où la fréquence d'accès est très faible.
- **Stockage de table Azure :** il s'agit d'une banque de données de l'attribut clé de NoSQL. Il doit être utilisé pour les données structurées. Les données sont stockées en tant qu'entités.
- **Stockage File d'attente Azure :** assure un stockage de messages fiable pour un grand nombre de messages. Ces messages sont accessibles depuis n'importe où via des appels HTTP ou HTTPs. Un message de file d'attente peut présenter une taille maximale de 64 Ko.

- **Azure Files :** il s'agit d'un stockage partagé, basé sur le protocole SMB. Il est généralement utilisé pour stocker et partager des fichiers. Il stocke également les données non structurées, mais se distingue principalement du fait qu'il peut être partagé sur le protocole SMB.
- **Disques Azure :** il s'agit du stockage au niveau du bloc pour les machines virtuelles Azure.

Ces cinq types de stockage répondent à différentes exigences architecturales et couvrent quasiment tous les types d'installations de stockage de données.

Fonctionnalités de stockage

Le stockage d'Azure est élastique. Cela signifie que vous pouvez stocker des quantités variables, depuis quelques mégaoctets à plusieurs pétaoctets de données. Vous n'avez pas besoin de bloquer au préalable une certaine capacité, et celle-ci évoluera automatiquement. Les consommateurs ne paient que pour leur utilisation effective du stockage. Voici quelques-uns des principaux avantages du stockage Azure :

- Le stockage Azure est sécurisé. Il est accessible uniquement via le protocole SSL. En outre, l'accès doit être authentifié.
- Le stockage Azure offre la possibilité de générer un jeton **Secure Access Signature** (**SAS**) au niveau du compte qui peut être utilisé par les clients de stockage pour s'authentifier. Il est également possible de générer des jetons SAS pour un niveau de service individuel pour les blobs, les files d'attente, les tables et les fichiers.
- Les données stockées dans le stockage Azure peuvent être chiffrées. Cette fonctionnalité est désignée par le terme sécurisation de données stockées.
- Le chiffrement de disque Azure est utilisé pour chiffrer le système d'exploitation et les données des disques dans des machines virtuelles IaaS. **Le chiffrement côté client** (**CSE**) et **le chiffrement du service de stockage** (**SSE**) sont tous deux utilisés pour chiffrer les données dans le stockage Azure. SSE est un paramètre de stockage Azure qui permet de s'assurer que les données sont chiffrées lors de leur écriture dans le stockage et qu'elles sont déchiffrées lors de la lecture par le moteur de stockage. De cette manière, aucune modification des applications n'est requise pour activer la fonction SSE. Dans CSE, les applications clientes peuvent utiliser le kit de développement logiciel de stockage pour chiffrer les données avant de les envoyer et de les enregistrer dans le stockage Azure. L'application cliente pourra ensuite les déchiffrer lors de la lecture. Ceci assure la sécurité des données en transit et de celles au repos. CSE est tributaire de secrets d'Azure Key Vault.
- Le stockage Azure est hautement disponible et durable. Cela signifie qu'Azure conserve toujours plusieurs copies des comptes Azure. L'emplacement et le nombre de copies dépendent de la configuration de la réplication.

Azure fournit les options de redondance des données et les paramètres de réplication suivants :

- **Stockage localement redondant (LRS)** : au sein d'un emplacement physique unique dans la région principale, il y aura trois réplicas de vos données de façon synchrone. Du point de vue de la facturation, il s'agit de l'option la moins chère. Il n'est toutefois pas recommandé pour les solutions qui nécessitent une haute disponibilité. Le LRS fournit un niveau de durabilité de 99,999999999 % pour les objets au cours d'une année donnée.

- **Stockage redondant interzone (ZRS)** : dans le cas du LRS, les réplicas ont été stockés dans le même emplacement physique. Dans le cas du ZRS, les données seront répliquées de manière synchrone dans les zones de disponibilité de la région principale. Étant donné que chacune de ces zones de disponibilité est un emplacement physique distinct dans la région principale, le ZRS offre une meilleure durabilité et une plus grande disponibilité que le LRS.

- **Stockage géoredondant (GRS)** : le GRS accroît la haute disponibilité via la réplication synchrone de trois copies de données au sein d'une seule région principale à l'aide du LRS. Il copie également les données dans un emplacement physique unique dans la région secondaire.

- **Stockage redondant géozone (GZRS)** : ce stockage est très similaire à GRS, mais au lieu de répliquer des données au sein d'un emplacement physique unique dans la région principale, GZRS les réplique de manière synchrone dans trois zones de disponibilité. Comme nous l'avons mentionné dans le cas du ZRS, étant donné que les zones de disponibilité sont des emplacements physiques isolés au sein de la région principale, le GZRS offre une meilleure durabilité et peut être inclus dans les conceptions hautement disponibles.

- **Stockage géoredondant en accès en lecture (RA-GRS) et stockage géo-redondant interzone avec accès en lecture** : les données répliquées dans la région secondaire par le GZRS ou le GRS ne sont pas disponibles pour la lecture ou l'écriture. Ces données seront utilisées par la région secondaire en cas de basculement du datacenter principal. Le RA-GRS et le RA-GZRS suivent le même modèle de réplication que le GRS et le GZRS respectivement, à la seule différence près que les données répliquées dans la région secondaire via RA-GRS ou RA-GZRS peuvent être lues.

Maintenant que nous avons compris les différentes options de stockage et de connexion disponibles sur Azure, intéressons-nous à l'architecture sous-jacente de la technologie.

Considérations architecturales pour les comptes de stockage

Les comptes de stockage doivent être mis en service dans la même région que celle des autres composants de l'application. Cela permet d'utiliser le même réseau principal de datacenter sans générer des frais de réseau.

Les services de stockage Azure dispose de cibles d'évolutivité en termes de capacités, de taux de transaction et de bande passante. Un compte de stockage général permet de stocker 500 To de données. Si vous devez stocker plus de 500 To de données, vous devez créer plusieurs comptes de stockage ou utiliser un stockage premium.

Les performances maximales d'un stockage standard sont de 20 000 IOPS ou 60 Mo de données par seconde. Toute requête supérieure en termes d'opérations d'entrée/sortie par seconde ou de données par seconde sera bloquée. Si cela n'est pas suffisant pour vos applications du point de vue des performances, un stockage premium ou plusieurs comptes de stockage doivent être utilisés. Pour un compte, la limite d'évolutivité appliquée à l'accès aux tables s'élève à 20 000 (1 Ko chacune) entrées. Le nombre d'entités insérées, mises à jour, supprimées ou numérisées contribuera à atteindre la cible. Une file d'attente unique peut traiter environ 2 000 messages (1 Ko chacun) par seconde, et chaque **`AddMessage`**, **`GetMessage`** et **`DeleteMessage`** sera traité comme un message. Si ces valeurs ne sont pas suffisantes pour votre application, vous devez répartir les messages sur plusieurs files d'attente.

La taille d'une machine virtuelle détermine la taille et la capacité des disques de données disponibles. Bien que les machines virtuelles plus importantes disposent de disques de données dotés d'une plus grande capacité d'opérations d'entrée/sortie par seconde, la capacité maximale reste limitée à 20 000 opérations d'entrée/sortie par seconde et 60 Mo par seconde. Il convient de noter qu'il s'agit là des valeurs maximales, et qu'en général des niveaux moindres doivent être pris en considération lors de la finalisation de l'architecture de stockage.

Au moment de la rédaction de cet ouvrage, les comptes GRS offrent une cible de bande passante de 10 Gbit/s aux États-Unis pour les entrées et de 20 Gbit/s si le RA-GRS/GRS est activé. Concernant les comptes LRS, les limites sont supérieures à celles du GRS. Pour les comptes LRS, les entrées sont limitées à 20 Gbit/s et les sorties à 30 Gbit/s. Les valeurs sont inférieures en dehors des États-Unis : la cible de bande passante s'élève à 10 Gbit/s et à 5 Gbit/s pour les sorties. Si vous avez besoin d'une bande passante supérieure, vous pouvez contacter le support Azure afin qu'il vous propose différentes options.

Les comptes de stockage doivent être activés en vue de l'authentification à l'aide de jetons SAS. Ils ne doivent pas être autorisés pour un accès anonyme. En outre, concernant le stockage blob, différents conteneurs doivent être créés avec des jetons SAS distincts, lesquels seront générés en fonction des différents types et catégories de clients ayant accès à ces conteneurs. Ces jetons SAS doivent être régénérés périodiquement afin de s'assurer que les clés ne peuvent être devinées ou déchiffrées. Les jetons SAS et d'autres options de sécurité sont abordés plus en détail dans le *chapitre 8, Architecture d'applications sécurisées sur Azure*.

En général, les blobs récupérés pour les comptes de stockage blob doivent être mis en cache. Nous pouvons déterminer si le cache est caduque en comparant sa dernière propriété modifiée pour extraire à nouveau le dernier blob.

Les comptes de stockage Azure fournissent des fonctions d'accès concurrentiel afin de veiller à ce qu'un même fichier et des données ne soient pas modifiés simultanément par plusieurs utilisateurs. En voici quelques-unes :

- **Accès concurrentiel optimiste** : il permet à plusieurs utilisateurs de modifier des données simultanément, mais lors de l'écriture, il vérifie si les données ou un fichier a été modifié. Si tel est le cas, il informe les utilisateurs afin d'extraire à nouveau les données et effectue une nouvelle mise à jour. Il s'agit de la concurrence d'accès par défaut pour les tables.
- **Accès concurrentiel pessimiste** : lorsqu'une application tente de mettre à jour un fichier, elle place un verrou qui rejette explicitement les mises à jour des autres utilisateurs sur ce fichier. Il s'agit de la concurrence d'accès par défaut pour les fichiers lorsque ceux-ci sont accédés à l'aide du protocole SMB.
- **Last-write-wins** : dans ce contexte, les mises à jour ne sont pas limitées et l'utilisateur le plus récent est celui qui met à jour le fichier, quel que soit le contenu initial. Il s'agit de la concurrence d'accès par défaut pour les files d'attente, les blobs et les fichier (lorsque ceux-ci sont accédés à l'aide du protocole REST).

À ce stade, vous devez connaître les différents services de stockage et comment ils peuvent être utilisés dans vos solutions. La section suivante porte sur les modèles de conception et explique le rôle qu'ils jouent au niveau des conceptions architecturales.

Modèles de conception dans le Cloud.

Les modèles de conception sont des solutions éprouvées à des problèmes connus en termes de conception. Ils s'agit de solutions réutilisables qui peuvent être appliquées aux problèmes. Ils ne s'agit pas de code ou de modèles réutilisables qui peuvent être incorporés tel quel au sein d'une solution. Il s'agit de descriptions et de directives documentées qui permettent de résoudre un problème. Un problème peut se manifester dans différents contextes et les modèles de conception peuvent permettre de le résoudre. Azure fournit de nombreux services et chacun d'eux offre des fonctions et des capacités spécifiques. Ces services sont très simples à utiliser, mais la création de solutions exploitant plusieurs services à la fois peut s'avérer laborieuse. En outre, il est difficile d'assurer à la fois la haute disponibilité, la super-évolutivité, la fiabilité, les performances et la sécurité d'une solution.

Les modèles de conception d'Azure fournissent des solutions prêtes à l'emploi qui peuvent être adaptées à des problèmes spécifiques. Cela facilite la création de solutions hautement disponibles, fiables, sécurisées et centrées sur les performances dans Azure. Bien qu'il existe de nombreux modèles et que certains d'entre eux soient abordés en détail dans les chapitres suivants, certains des modèles de messagerie, de performances et d'évolutivité sont mentionnés dans le présent chapitre. En outre, des liens sont fournis pour obtenir une description détaillée de ces modèles. Ces modèles de conception doivent faire l'objet d'un manuel complet à part entière. Ils sont mentionnés dans le présent document afin de vous informer de leur existence et de vous fournir des références offrant des informations plus détaillées.

Modèles de messagerie

Les modèles de messagerie permettent de connecter des services de manière libre. Cela signifie que ces services ne communiquent jamais directement entre eux. Au lieu de cela, un service génère et envoie un message à un courtier (généralement une file d'attente) et tout autre service intéressé par ce message peut le récupérer et le traiter. Il n'y a aucune communication directe entre le service de l'expéditeur et celui du destinataire. Cette dissociation rend non seulement les services et les applications plus fiables dans l'ensemble, mais également plus robustes et tolérants aux pannes. Les récepteurs peuvent recevoir et lire des messages à leur propre rythme.

La messagerie permet de créer des modèles asynchrones. La messagerie consiste à envoyer des messages d'une entité à une autre. Ces messages sont créés et transmis par un expéditeur, stockés dans un stockage durable, puis consommés par les destinataires.

Les principaux défis en termes d'architecture que la messagerie permet de résoudre sont les suivants :

- **Durabilité** : les messages sont stockés dans un stockage durable et les applications peuvent les lire plus tard, une fois qu'ils sont reçus dans le cas d'un basculement.
- **Fiabilité** : les messages permettent de mettre en œuvre la fiabilité car ils sont persistants sur le disque et ne sont jamais perdus.
- **Disponibilité des messages** : les messages sont disponibles en vue d'une consommation par les applications après la restauration de la connectivité et avant les temps d'arrêt.

Azure fournit des files d'attente Service Bus et des thèmes pour mettre en œuvre les modèles de messagerie au sein des applications. Le Stockage File d'attente Azure peut également servir aux mêmes fins.

Choisir entre les files d'attente Azure Service Bus et le Stockage en file d'attente revient à décider de la durée de stockage des messages, leur taille, la latence et les coûts. Azure Service Bus prend en charge les messages d'une taille de 256 Ko, tandis que le Stockage en file d'attente prend en charge des messages d'une taille de 64 Ko. Azure Service Bus peut stocker des messages pour une durée illimitée, tandis que le Stockage en file d'attente peut stocker des messages pendant sept jours. Le coût et les temps de latence sont plus élevés pour les files d'attente Service Bus.

Selon les besoins et les exigences de votre application, les facteurs préalables doivent être pris en compte afin de décider quelle file d'attente est la plus appropriée. La section suivante porte sur les différents types de modèles de messagerie.

Le patron de consommateurs concurrents

Un seul consommateur de messages fonctionne de manière synchrone, à moins que l' application elle-même implémente la logique de lecture de messages asynchrone. Le modèle de consommateurs concurrents permet de mettre en œuvre une solution dans le cadre de laquelle plusieurs consommateurs sont prêts à traiter les messages entrants et sont en concurrence pour chaque message. Cela permet de créer des solutions dotées d'une architecture évolutive à haute disponibilité. Ce modèle est évolutif car le fait de disposer de plusieurs consommateurs permet de traiter un plus grand nombre de messages sur une plus courte période. Il s'agit d'une solution hautement disponible, car au moins un consommateur peut traiter les messages, même si certains consommateurs sont bloqués.

Ce modèle doit être utilisé lorsque chaque message est indépendant des autres. Les messages eux-mêmes contiennent toutes les informations nécessaires pour permettre à un consommateur d'accomplir une tâche. Ce modèle ne doit pas être utilisé s'il n'y a une dépendance parmi les messages. Les consommateurs doivent être en mesure d'effectuer les tâches de manière isolée. En outre, ce modèle est pertinent s'il existe une demande variable pour les services. Les consommateurs supplémentaires peuvent être ajoutés ou supprimés selon la demande.

Une file d'attente des messages est requise pour mettre en œuvre les modèles de consommateurs concurrents. Les modèles provenant de sources multiples traversent une seule file d'attente qui est connectée à plusieurs consommateurs à l'autre extrémité. Ces consommateurs doivent supprimer chaque message après l'avoir lu afin d'éviter qu'il ne soit traité à nouveau :

Figure 3.10 : le patron de consommateurs concurrents

Consultez la documentation de Microsoft disponible à l'adresse https://docs.microsoft com/azure/architecture/patterns/competing-consumers pour en savoir plus sur ce patron.

Le patron de file d'attente prioritaire

Il est souvent nécessaire de prioriser certains messages par rapport aux autres. Ce modèle est important pour les applications qui fournissent différents **contrats de niveau de service** (**SLA**) aux consommateurs, lesquels offrent des services basés sur différents plans et abonnements.

Modèles de file d'attente premier entré, premier sorti. Les messages sont traités selon un ordre séquentiel. Cependant, grâce au modèle de file d'attente prioritaire, il est possible d'accélérer le traitement de certains messages compte tenu de leur plus grande priorité. Il existe plusieurs manières de le mettre en œuvre. Si la file d'attente offre la possibilité d'attribuer une priorité et de réorganiser les messages en fonction de celle-ci, même une seule file d'attente suffit pour implémenter ce modèle :

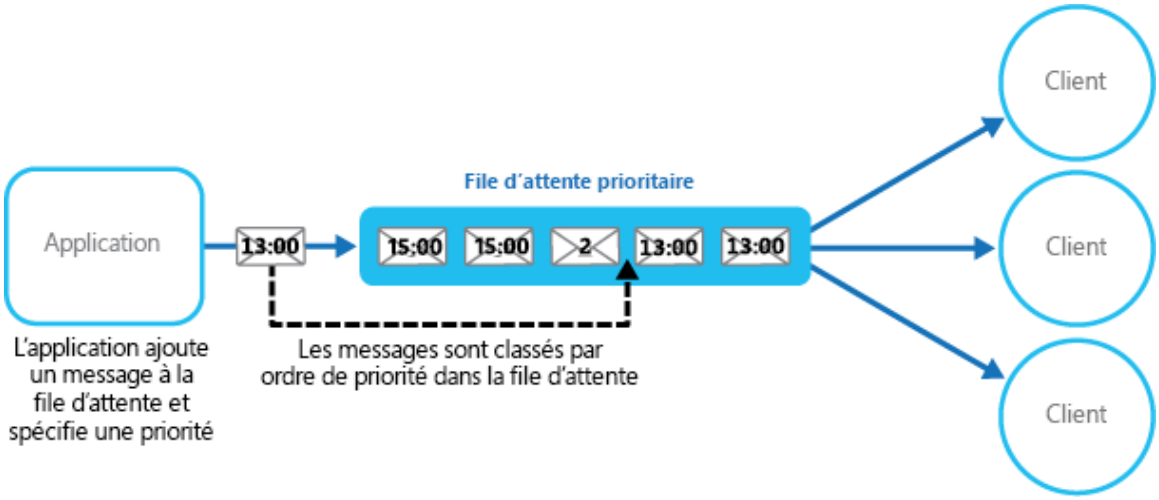


Figure 3.11 : modèle de file d'attente prioritaire unique

Toutefois, si la file d'attente ne permet pas de réorganiser les messages, des files d'attente distinctes peuvent être créées pour établir des priorités différentes et chaque file d'attente peut être associée à des consommateurs distincts :

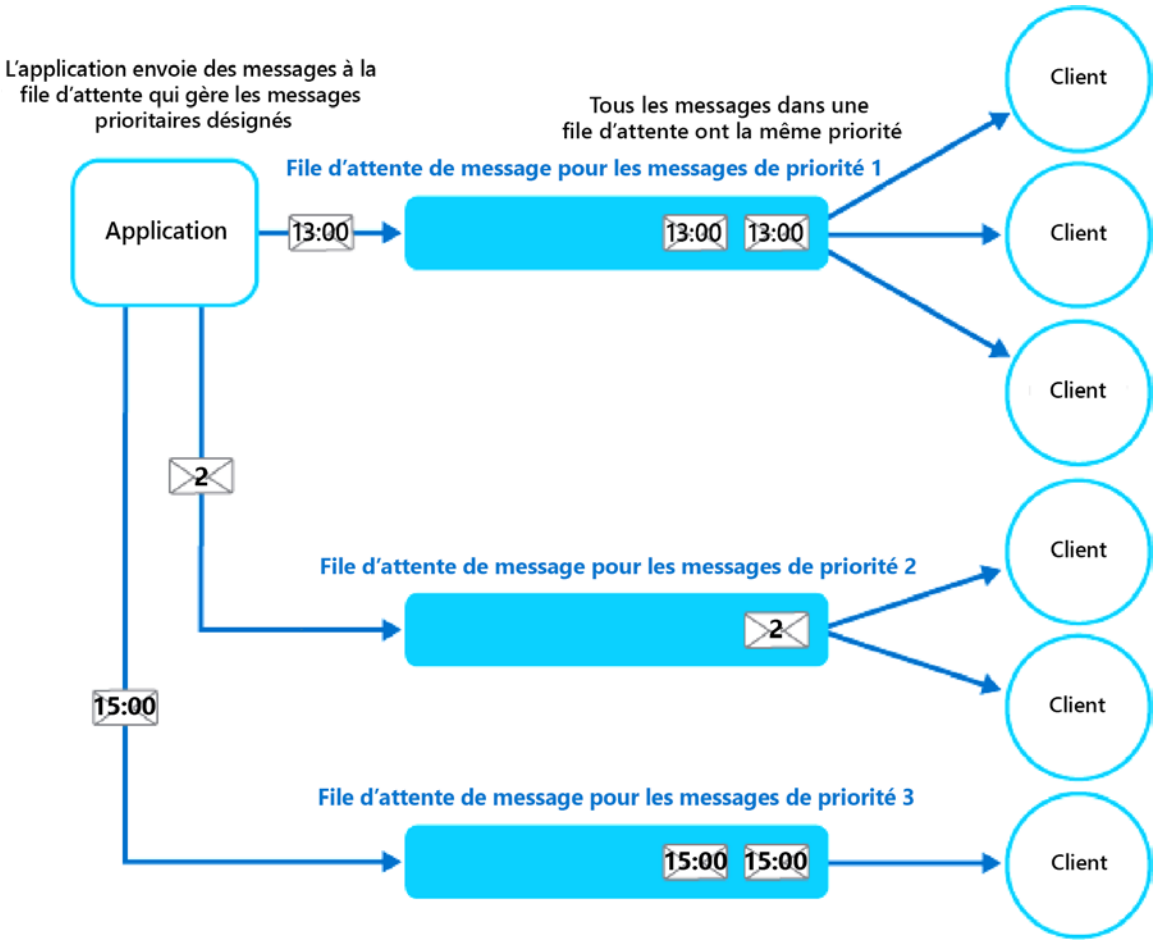


Figure 3.12 : utilisation de files d'attente de messages distinctes pour différentes priorités

En effet, ce modèle peut réutiliser le patron de consommateurs concurrents afin d'accélérer le traitement des messages de chaque file d'attente à l'aide de plusieurs consommateurs. Consultez la documentation Microsoft disponible à l'adresse https://docs.microsoft.com/azure/architecture/patterns/priority-queue pour en savoir plus sur le modèle de file d'attente prioritaire.

Le patron Queue-Based Load Leveling (Nivellement de charge basé sur la file d'attente)

Le patron de nivellement de charge basé sur la file d'attente réduit l'impact des pics de demande sur la disponibilité et la vigilance des tâches et des services. Une file d'attente fait office de tampon entre une tâche et un service. Elle peut être appelée pour gérer les charges lourdes inattendues qui peuvent provoquer des interruptions de service ou des délais d'attente. Ce patron permet de traiter les problèmes de performances et de fiabilité. Pour éviter que le service ne soit surchargé, nous allons introduire une file d'attente qui stockera un message jusqu'à ce qu'il soit récupéré par le service. Les messages seront extraits de la file d'attente par le service de manière cohérente et seront traités.

La *Figure* 3.13 illustre le fonctionnement du patron de nivellement de charge basé sur la file d'attente :

Figure 3.13 : le patron de nivellement de charge basé sur la file d'attente

Même si ce patron permet de gérer des pics de demande inattendue, il ne s'agit pas de la solution la plus adaptée à la conception de services présentant une latence minimale. Par ailleurs, la latence, qui est une mesure des performances, sera évoquée dans la section suivante, avec les modèles de performance et d'évolutivité.

Modèles de performances et évolutivité

Performance et évolutivité vont de pair. Les performances mesurent la rapidité d'un système à exécuter une action dans un intervalle de temps donné de manière positive. D'autre part, l'évolutivité est la capacité d'un système à gérer une charge inattendue sans compromettre ses performances, ni la vitesse à laquelle il peut être élargi avec les ressources disponibles. Cette section présente quelques modèles de conception relatifs à la performance et à l'évolutivité.

Patron CQRS (Command and Query Responsibility Segregation, Séparation des responsabilités commande/requête**)**

CQRS n'est pas un modèle spécifique à Azure, mais un patron général qui peut être appliqué à n'importe quelle application. Il augmente les performances globales et la réactivité d'une application.

CQRS est un modèle qui sépare les opérations qui lisent les données (requêtes) et les opérations qui mettent à jour les données (commandes) à l'aide d'interfaces distinctes. Cela signifie que les modèles de données utilisés pour effectuer les requêtes et les mises à jour sont différents. Les modèles peuvent ensuite être isolés, comme le montre la *Figure 3.14*, mais ce n'est pas une nécessité absolue.

Ce patron doit être utilisé lorsque des règles métier volumineuses et complexes sont exécutées lors de la mise à jour et de la récupération des données. Par ailleurs, ce modèle présente un excellent cas d'utilisation dans lequel une équipe de développeurs peut se concentrer sur le modèle de domaine complexe qui fait partie du patron d'écriture, alors qu'une autre équipe peut se concentrer sur le patron de lecture et les interfaces utilisateur. Il est également judicieux d'utiliser ce patron lorsque le rapport entre la lecture et l'écriture est biaisé. Les performances des lectures de données doivent être affinées séparément des performances des écritures de données.

Le CQRS permet non seulement d'améliorer les performances d'une application, mais facilite également la conception et la mise en œuvre entre plusieurs équipes. En raison de sa capacité à utiliser des modèles distincts, CQRS n'est pas adapté si vous utilisez des outils de génération de modèles et d'échafaudages :

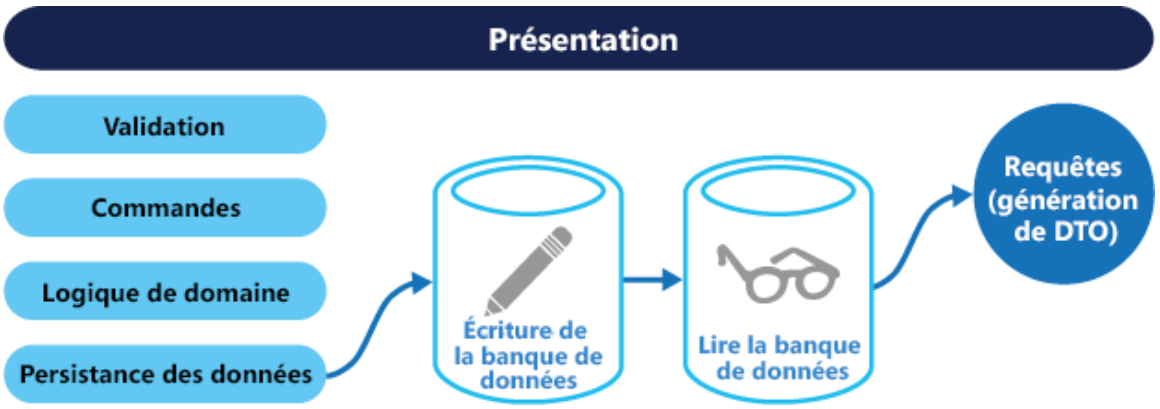


Figure 3.14 : le patron CQRS

Consultez la documentation de Microsoft disponible à l'adresse https://docs.microsoft.com/azure/architecture/patterns/cqrs pour en savoir plus sur ce patron.

Le patron Event Sourcing (Matérialisation d'événements)

Comme la plupart des applications fonctionnent avec des données et que les utilisateurs travaillent avec celles-ci, l'approche classique de l'application consiste à maintenir et à mettre à jour l'état actuel des données. La lecture des données depuis la source, les modifier et les mettre à jour vers l'état actuel avec la valeur modifiée constituent l'approche de traitement habituelle des données. Toutefois, il peut y avoir des limites :

- Étant donné que les opérations de mise à jour sont directement effectuées dans la banque de données, les performances et la réactivité globales sont donc affectées.
- Si plusieurs utilisateurs travaillent sur les données et les mettent à jour, des conflits peuvent survenir et certaines des mises à jour appropriées risquent d'échouer.

La solution consiste à mettre en œuvre le patron de matérialisation d'événements, où les modifications seront enregistrées dans un magasin ajout uniquement. Une série d'événements sera poussée par le code de l'application vers le magasin d'événements, où elle sera conservée. Les événements sont conservés comme données persistantes dans un magasin d'événements qui agit comme système d'enregistrement concernant l'état actuel des données. Les consommateurs seront avertis et pourront traiter les événements si nécessaire, une fois qu'ils sont publiés.

Le patron de matérialisation d'événements est indiqué à la *Figure* 3.15 :

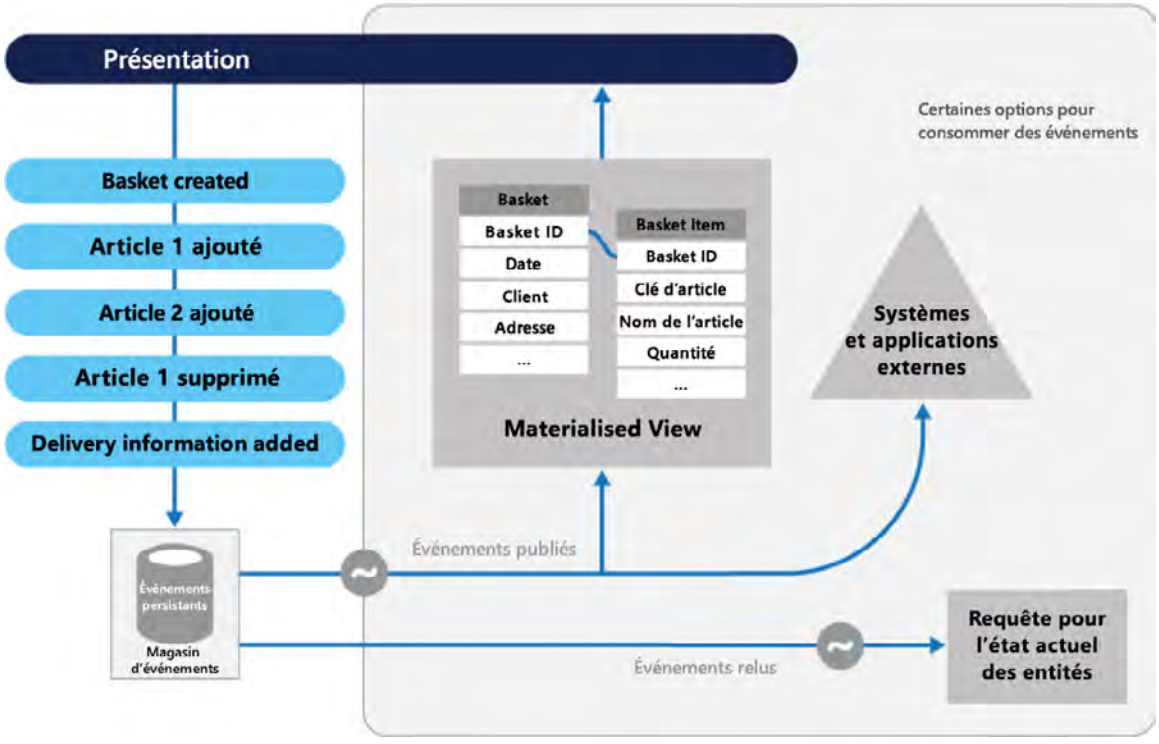


Figure 3.15 : le patron de matérialisation d'événements

Pour en savoir plus sur ce patron, accédez à l'adresse https://docs.microsoft.com/azure/architecture/patterns/event-sourcing.

Le patron Throttling (Limitation)

Parfois, certaines applications sont soumises à des exigences SLA très strictes du point de vue des performances et de l'évolutivité, indépendamment du nombre d'utilisateurs consommant le service. Dans ces circonstances, il est important d'implémenter des modèles de restriction afin de limiter le nombre de requêtes autorisées à être exécutées. Il est impossible de prévoir chaque fois avec précision la charge sur les applications. En cas de pics de charge sur l'application, une restriction permet de réduire la pression sur les serveurs et les services en contrôlant la consommation de ressources. L'infrastructure Azure est un très bon exemple de ce modèle.

Ce modèle doit être utilisé dans des cas où le respect d'un accord SLA constitue une priorité pour les applications, afin d'empêcher certains utilisateurs de consommer plus de ressources que ce qui leur est alloué, d'augmenter les pics et explosions de la demande et d'optimiser la consommation des ressources du point de vue des coûts. Il s'agit de scénarios pertinents pour les applications conçues afin d'être déployées sur le Cloud.

Plusieurs stratégies peuvent être utilisées pour gérer la limitation d'une application. La stratégie de restriction peut rejeter des nouvelles requêtes une fois le seuil dépassé, ou bien informer l'utilisateur qu'une requête se trouve dans la file d'attente et sera exécutée une fois que le nombre de requêtes aura diminué.

La *Figure 3.16* illustre la mise en œuvre du modèle de limitation dans un système multi-locataires, où une limite d'utilisation de ressources fixe est attribuée à chaque locataire. Une fois cette limite franchie, toute demande supplémentaire de ressources est limitée, ce qui permet ainsi de maintenir une quantité des ressources suffisante pour les autres locataires :

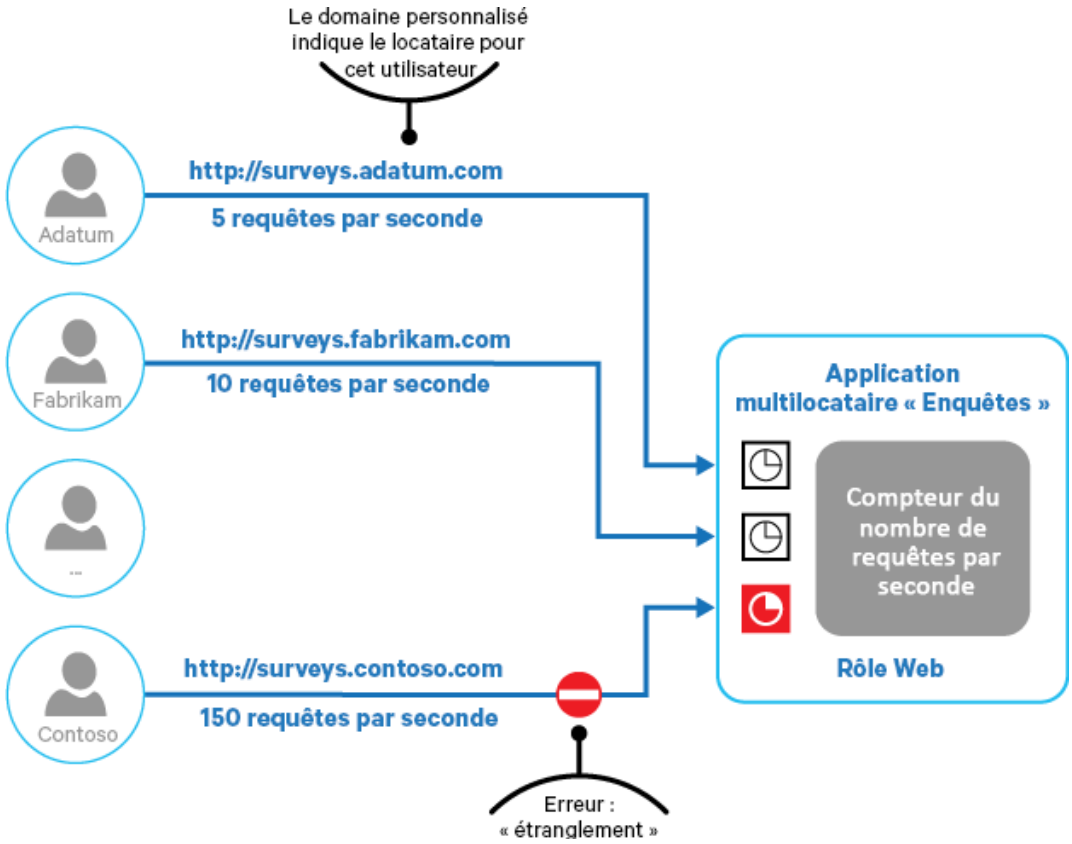


Figure 3.16 : le patron de limitation

Pour en savoir plus sur ce modèle, accédez à l'adresse https://docs.microsoft.com/azure/architecture/patterns/throttling.

Patron Retry (Nouvelle tentative)

Le modèle de nouvelle tentative est un modèle extrêmement important pour rendre des applications et des services plus résilients aux défaillances transitoires. Imaginez que vous essayez de vous connecter et d'utiliser un service et que celui-ci n'est pas disponible, quelle qu'en soit la raison. Si le service sera de nouveau disponible bientôt, il est logique de répéter les tentatives jusqu'à établir une connexion réussie. Cela rendra l'application plus robuste, tolérante aux pannes et stable. Dans Azure, la plupart des composants s'exécutent sur Internet, et cette connexion à Internet peut produire des failles transitoires par intermittence. Étant donné que ces défaillances peuvent être rectifiées en quelques secondes, une application ne devrait pas tomber en panne. L'application doit être conçue de sorte à pouvoir retenter d'utiliser le service de manière répétée en cas de panne. Elle doit également interrompre toute nouvelle tentative lorsqu'elle réussit ou quand elle détecte un problème qui sera long à résoudre.

Ce modèle devrait être appliqué lorsqu'une application est susceptible de faire face à des pannes transitoires car elle interagit avec un service distant ou accède à une ressource distante. Ces failles sont censées être de courte durée et le fait de répéter une requête ayant déjà échoué peut résulter en une réussite lors d'une tentative subséquente.

Le modèle de nouvelle tentative peut adopter différentes stratégies de nouvelle tentative, selon la nature des erreurs et de l'application :

- **Retenter un certain nombre de fois** : cela indique que l'application tentera de communiquer avec le service un nombre de fois bien précis avant de conclure à un échec et de lever une exception. L'application effectuera par exemple trois tentatives de connexion à un autre service. Si le système parvient à se connecter durant ces trois tentatives, l'intégralité de l'opération est réussie, sinon une exception sera levée.
- **Nouvelle tentative selon une échéance** : cela indique que l'application tentera de communiquer avec le service à plusieurs reprises pendant un nombre de secondes ou de minutes bien déterminé et attendra un certain délai avant de lancer une nouvelle tentative. Par exemple, l'application tentera de se connecter au service toutes les trois secondes pendant 60 secondes. Si la connexion est établie au cours de cette période, l'intégralité de l'opération sera réussie. Sinon, elle lèvera une exception.
- **Transfert et report de la nouvelle tentative** : cela indique que l'application tentera de communiquer avec le service à plusieurs reprises selon une certaine échéance, et ajoutera un délai supplémentaire à chaque nouvelle tentative. Par exemple, de nouvelles tentatives seront initiées pendant 60 secondes au total, durant lesquelles la première tentative sera effectuée au bout d'une seconde, la deuxième au bout de deux secondes à compter de la tentative précédente, la troisième au bout de quatre secondes à compter de la tentative précédente, etc. Le nombre total de nouvelles tentatives est ainsi réduit.

La Figure 3.17 illustre le patron de nouvelle tentative. La première requête obtient une réponse HTTP 500, la seconde tentative obtient également HTTP 500 comme réponse, et enfin, la requête aboutit et obtient la réponse HTTP 200 :

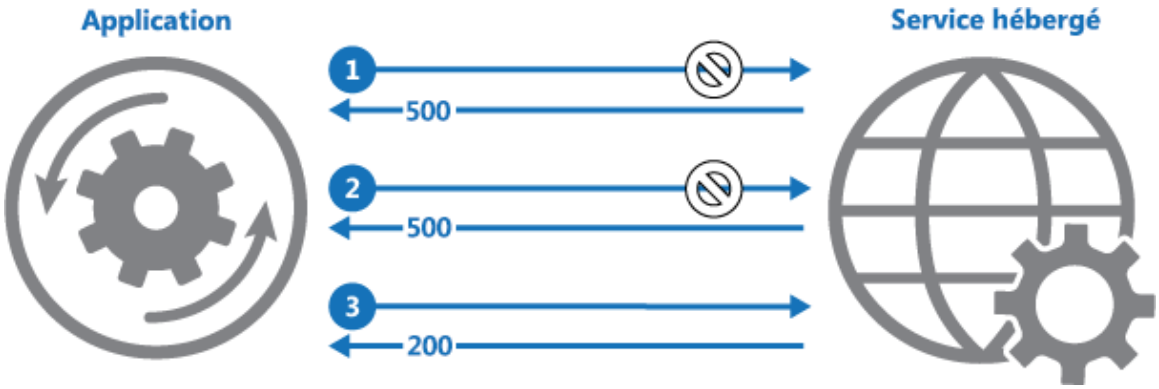


Figure 3.17 : le patron de nouvelle tentative

Consultez cette documentation de Microsoft disponible à l'adresse https://docs.microsoft.com/azure/architecture/patterns/retry pour en savoir plus sur ce patron.

Le patron Circuit Breaker (Disjoncteur)

Il s'agit d'un modèle extrêmement utile. Imaginez que vous essayez de vous connecter et d'utiliser un service et que celui-ci n'est pas disponible, quelle qu'en soit la raison. Si le service n'est pas opérationnel en une courte durée, il est inutile de répéter les tentatives jusqu'à établir une connexion. De plus, ces nouvelles tentatives accaparent des ressources qui pourraient être utilisées à d'autres fins.

Le modèle de disjoncteur permet d'éviter ce gaspillage de ressources. Il empêche les applications d'essayer à plusieurs reprises de se connecter et d'utiliser un service qui n'est pas disponible. Il permet également aux applications de détecter si un service est à nouveau opérationnel, et de s'y connecter.

Pour implémenter le modèle de disjoncteur, toutes les requêtes envoyées au service sont transférées à un service qui agit comme un proxy du service original. Le but de ce service de proxy est de maintenir un certain état du système et d'agir comme passerelle vers le service initial. Il maintient trois états. Davantage d'états peuvent être inclus selon les besoins de l'application.

Les états minimaux nécessaires pour implémenter ce modèle sont les suivants :

- **Ouvert** : cet état indique que le service est indisponible, et le système envoie une exception à l'application immédiatement au lieu de l'autoriser à effectuer une nouvelle tentative ou à patienter jusqu'à l'expiration du délai. Lorsque le service est de nouveau disponible, l'état passe à demi-ouvert.
- **Fermé** : cet état indique que le service est en bonne santé et que l'application peut s'y connecter. Généralement, un compteur indique le nombre d'échecs avant toute transition à l'état Ouvert.
- **Demi-ouvert** : à un moment donné, tandis que le service est en cours d'exécution, cet état permet à un nombre limité de requêtes d'être envoyées. Cet état est un test décisif qui vérifie si les requêtes transmises aboutissent ou non. Si les requêtes aboutissent, l'état passe de Demi-ouvert à Fermé. Cet état peut également implémenter un compteur, afin d'autoriser un certain nombre de requêtes à aboutir avant de pouvoir passer à l'état Fermé.

Les trois états et leurs transitions sont illustrés à la *Figure 3.18* :

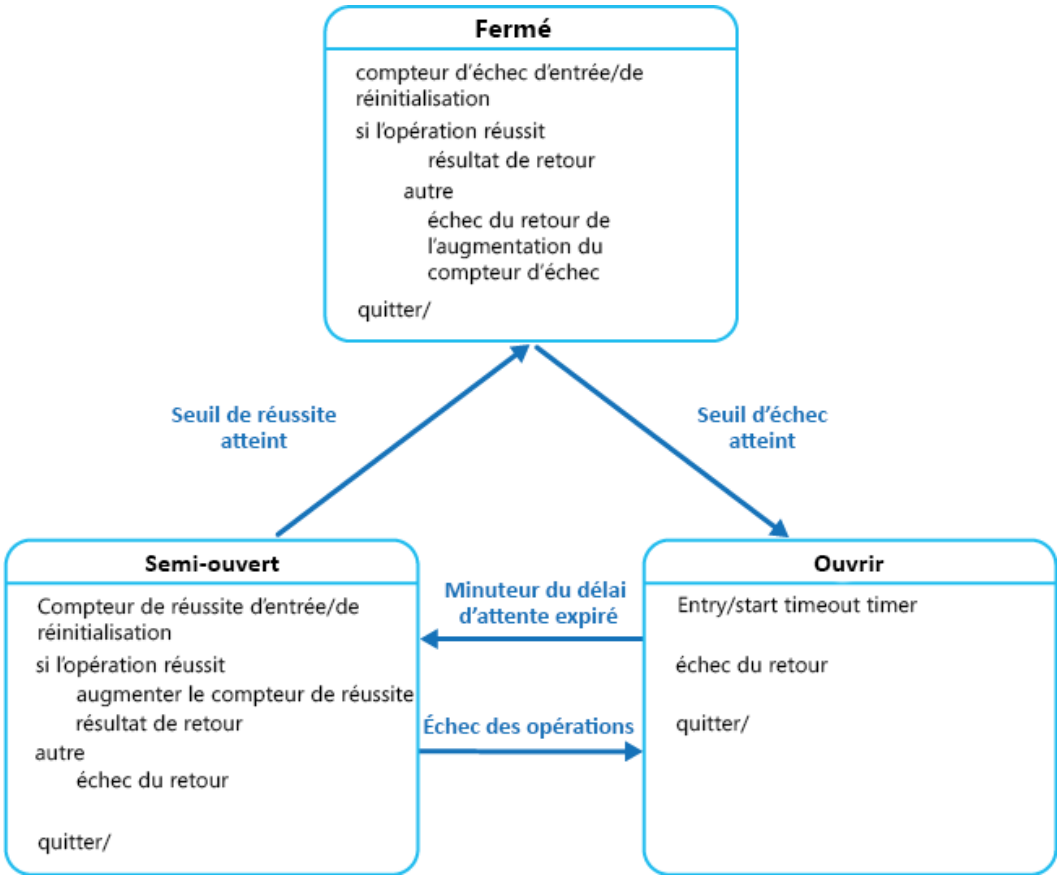


Figure 3.18 : le patron de disjoncteur

Pour en savoir plus sur ce patron, consultez la documentation de Microsoft à l'adresse https://docs.microsoft.com/azure/architecture/patterns/circuit-breaker.

Dans cette section, nous avons discuté des modèles de conception qui peuvent être utilisés pour concevoir des applications fiables, évolutives et sécurisées dans le Cloud. D'autres modèles existent et ceux-ci sont évoqués plus en détail à l'adresse https://docs.microsoft.com/azure/architecture/patterns.

Résumé

Plusieurs services sont disponibles dans Azure et la plupart d'entre eux peuvent être combinés pour créer des solutions concrètes. Ce chapitre a présenté les trois principaux services d'Azure : régions, stockage et réseaux. Ces derniers forment le système dorsal de la plupart des solutions déployées dans n'importe quel Cloud. Ce chapitre fournit des détails sur ces services, et explique en quoi leur configuration et leur mise en service peuvent influencer les décisions de conception.

Les points essentiels à prendre en compte du point de vue du stockage et des réseaux ont été présentés en détail dans ce chapitre. Les réseaux et le stockage offrent tous deux de nombreux choix et il est important d'opter pour une configuration adaptée à vos exigences.

Enfin, certains des modèles de conception majeurs liés à la messagerie, comme les consommateurs concurrents, les files d'attente prioritaires, et le nivellement de charge ont été décrits. Les modèles associés à CQRS et aux limitations ont été illustrés et d'autres modèles tels que le modèle de nouvelles tentatives et de disjoncteur ont été abordés. Nous conserverons ces modèles comme base de référence lorsque nous déploierons nos solutions.

Le chapitre suivant porte sur l'automatisation des solutions que nous allons concevoir. À mesure que nous avançons dans le monde de l'automatisation, chaque organisation souhaite éliminer les frais généraux associés à la création de ressources, un par un et cette tâche est fastidieuse. L'automatisation permet d'y parvenir et celle-ci sera évoquée plus en détail dans le chapitre suivant.

4

Automatiser l'architecture sur Azure

Chaque organisation souhaite réduire les efforts et les erreurs manuels, et l'automatisation joue un rôle important dans la mise en place de la prévisibilité, de la normalisation et de la cohérence dans la conception des produits et dans les opérations. L'automatisation a été la principale priorité de presque tous les **directeurs des systèmes d'information** (**DSI**) et des responsables numériques, lesquels souhaitent garantir la haute disponibilité, l'évolutivité et la fiabilité de leurs systèmes, tout en répondant aux besoins de leurs clients.

L'automatisation a pris de l'importance avec l'avènement du Cloud, car de nouvelles ressources peuvent être mises en service à la volée sans nécessiter l'acquisition de ressources matérielles. Par conséquent, les entreprises Cloud souhaitent automatiser la plupart de leurs activités afin de réduire l'utilisation abusive, les erreurs, la gouvernance, la maintenance et l'administration.

Dans ce chapitre, nous allons évaluer Azure Automation, un service majeur qui fournit des fonctionnalités d'automatisation, ainsi que ses capacités de différenciation par rapport à d'autres services apparemment similaires. Ce chapitre couvre les sujets suivants :

- Le paysage Azure Automation
- Le service Azure Automation
- Les ressources dédiées aux services Azure Automation
- Écriture des procédures opérationnelles Azure Automation
- Webhooks
- Travailleurs hybrides

Commençons par découvrir Azure Automation, un service Cloud dédié à l'automatisation des processus.

Automatisation

L'automatisation est nécessaire pour la mise en service, l'exploitation, la gestion et la désinstallation des ressources IT au sein d'une organisation. La *Figure 4.1* illustre plus en détail chacun de ces cas d'utilisation :

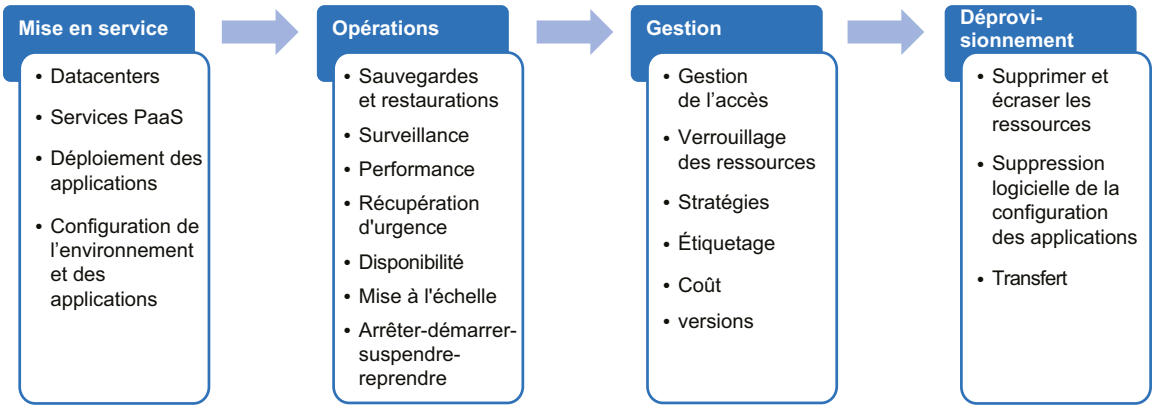


Figure 4.1 : cas d'utilisation de l'automatisation

Avant l'avènement du Cloud, les ressources IT étaient principalement hébergées sur site, et celles-ci exigeaient généralement des processus manuels. Toutefois, suite à l'essor du Cloud, l'automatisation a fait l'objet d'un intérêt accru. Cet engouement est dû à l'agilité et à la flexibilité de la technologie Cloud, qui permettent de mettre en service, de désinstaller et de gérer ces ressources à la volée, beaucoup plus rapidement qu'auparavant. Parallèlement à cette flexibilité et à cette agilité, les exigences sont plus prévisibles et cohérentes avec le Cloud. En effet, les entreprises peuvent désormais créer des ressources en toute facilité.

Microsoft dispose d'un excellent outil dédié à l'automatisation IT, connu sous le nom de System Center Orchestrator. Il s'agit d'un excellent outil d'automatisation pour les environnements locaux et Cloud, qui est toutefois un produit et non un service. Il doit être concédé sous licence et déployé sur des serveurs. Des procédures opérationnelles peuvent ensuite être exécutées pour effectuer des modifications sur les environnements Cloud et sur site.

Microsoft s'est rendu compte qu'une solution d'automatisation devait être proposée aux clients en tant que service pour remplacer ce produit qui devait être acheté et déployé. C'est alors qu'Azure Automation voit le jour.

Azure Automation

Azure propose un service appelé **Azure Automation**, qui est un service essentiel pour l'automatisation des processus, des activités et des tâches, non seulement sur Azure, mais également sur site. À l'aide d'Azure Automation, les entreprises peuvent automatiser leurs processus et tâches liés au traitement, à la suppression, à l'exploitation et à la gestion de leurs ressources dans le Cloud, les environnements informatiques, les plateformes et les langages. La *Figure 4.2* illustre certaines fonctions d'Azure Automation :

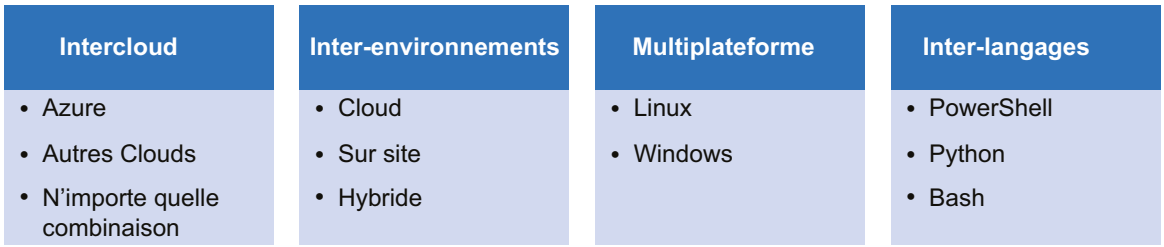


Figure 4.2 : fonctions d'Azure Automation

Architecture d'Azure Automation

Azure Automation comprend plusieurs composants, et chacun d'entre eux est complètement découplé des autres. La majeure partie de l'intégration se produit au niveau de la banque de données, et les composants ne communiquent pas entre eux directement.

Lorsqu'un compte Automation est créé sur Azure, il est géré par un service de gestion. Le service de gestion est un point de contact unique pour toutes les activités au sein d'Azure Automation. Toutes les requêtes issues du portail, y compris l'enregistrement, la publication et la création de procédures opérationnelles, ou l'exécution, l'arrêt, la suspension, le démarrage et les tests sont envoyés au service de gestion de l'automatisation et celui-ci écrit les données de requête dans sa banque de données. Il crée également un enregistrement de travail dans la banque de données et, en fonction de l'état des travailleurs de la procédure opérationnelle, il l'attribue à un travailleur.

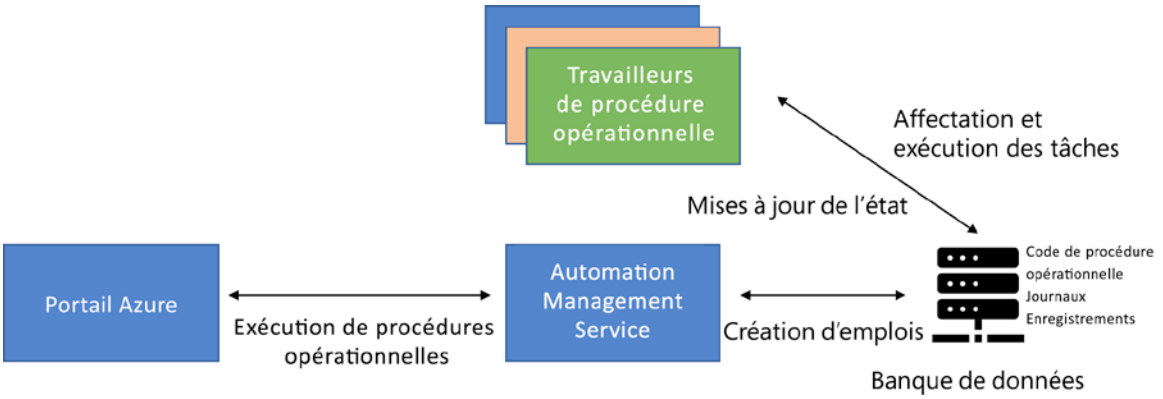


Figure 4.3 : architecture d'Azure Automation

Le travailleur continue d'interroger la base de données pour tous les nouveaux travaux qui lui sont attribués. Une fois qu'il trouve une tâche qui lui est attribuée, il récupère les informations de ce travail et commence à exécuter la tâche à l'aide de son moteur d'exécution. Les résultats sont réécrits dans la base de données, lus par le service de gestion et affichés de nouveau sur le portail Azure.

Les travailleurs hybrides que nous évoquerons plus loin dans ce chapitre sont également les travailleurs de procédure opérationnelle, bien qu'ils ne soient pas présentés dans la *Figure* 4.3.

La première étape à effectuer pour utiliser Azure Automation consiste à créer un compte. Une fois le compte créé, tous les autres artefacts sont créés dans le compte.

Le compte agit comme la principale ressource de premier niveau qui peut être gérée à l'aide de groupes de ressources Azure et de son propre plan de contrôle.

Le compte doit être créé au sein d'une région, et toute l'automatisation de ce compte est exécutée sur les serveurs de cette région.

Il est important de choisir judicieusement la région, de préférence à proximité d'autres ressources Azure que le compte Automation intègre ou gère, afin de réduire le trafic réseau et la latence entre les régions.

Le compte Automation prend également en charge quelques comptes **d'identification**, qui peuvent être créés à partir du compte Automation. Étant donné que ces comptes d'identification sont similaires à un compte de service, nous les créons principalement pour exécuter des actions. Même si nous parlons généralement de comptes d'identification, ceux-ci se divisent en deux catégories : les comptes d'identification classiques Azure et les comptes d'identification. Ces deux types de compte sont utilisés pour se connecter aux abonnements Azure. Le compte d'identification classique Azure permet de se connecter à Azure à l'aide de l'API **Azure Service Management**, et le compte d'identification est utilisé pour se connecter à Azure grâce à l'API **Azure Resource Management** (**ARM**).

Ces deux comptes utilisent des certificats pour s'authentifier avec Azure. Ces comptes peuvent être créés lors de la création du compte Automation. Vous pouvez également les créer à un stade ultérieur dans le portail Azure.

Il est recommandé de créer ces comptes d'identification après la création d'un compte Automation. En effet, s'ils sont créés lors de la configuration du compte Automation, Automation génère des certificats et des principaux de service en coulisses avec la configuration par défaut. Si vous avez besoin de renforcer le contrôle et de personnaliser la configuration de ces comptes d'identification (par exemple en utilisant un certificat ou un principal de service existant), les comptes d'identification doivent être créés après le compte Automation.

Une fois le compte Automation créé, il fournit un tableau de bord qui permet d'activer plusieurs scénarios d'automatisation.

Certains des scénarios importants qui peuvent être activés à l'aide d'un compte Automation sont liés aux éléments suivants :

- Automatisation des processus
- Gestion de la configuration
- Gestion des mises à jour

L'automatisation consiste à écrire des scripts réutilisables et génériques, afin qu'ils puissent être réutilisés dans plusieurs scénarios. Par exemple, un script d'automatisation doit être assez générique pour démarrer et arrêter n'importe quelle machine virtuelle dans tous les groupes de ressources quels que soient l'abonnement et le groupe de gestion. Les informations de serveur de machine virtuelle relatives au codage en dur, ainsi que les noms de groupes de ressources, d'abonnement et de groupe de gestion, aboutiront à la création de plusieurs scripts similaires, et toute modification apportée à l'un d'entre eux changera tous les scripts. Il est préférable de créer un script unique à cet effet à l'aide de paramètres et de variables de script. Veillez également à ce que les valeurs soient saisies par l'exécuteur de ces artefacts.

Étudions de plus près chacun des scénarios susmentionnés.

Automatisation des processus

L'automatisation des processus fait référence au développement de scripts qui reflètent des processus réels. L'automatisation des processus comprend plusieurs activités, et chacune d'entre elles effectue une tâche discrète. Ensemble, ces activités forment un processus complet. Les activités peuvent être exécutées en fonction de si l'activité précédente a été exécutée avec succès ou non.

L'infrastructure qui exécute l'automatisation des processus doit satisfaire à certaines exigences. Voici certaines d'entre elles :

- La capacité à créer des charges de travail
- Une capacité d'exécution de longue durée
- La possibilité d'enregistrer l'état d'exécution lorsque la charge de travail n'est pas terminée, ce qui est également connu sous le nom de point de contrôle et d'hydratation
- La capacité à reprendre à partir du dernier état enregistré au lieu de commencer dès le début

Le prochain scénario que nous allons découvrir est la gestion de la configuration.

Gestion de la configuration

La gestion de la configuration fait référence au processus de gestion de la configuration du système tout au long de son cycle de vie. Azure Automation State configuration est le service de gestion de configuration Azure qui permet aux utilisateurs d'écrire, de gérer et de compiler la configuration PowerShell DSC pour les nœuds Cloud et les datacenters sur site.

Azure Automation State Configuration nous permet de gérer les machines virtuelles Azure, les machines virtuelles classiques Azure et les machines physiques ou virtuelles (Windows/Linux) sur site, tout en prenant en charge les machines virtuelles dans d'autres fournisseurs de Cloud.

L'un des plus grands avantages d'Azure Automation State Configuration est l'évolutivité. Nous pouvons gérer des milliers de machines à partir d'une seule interface de gestion centralisée. Nous pouvons attribuer des configurations à des machines en toute simplicité et vérifier si elles sont conformes à la configuration souhaitée.

Azure Automation présente un autre avantage : il peut être utilisé en tant que référentiel pour stocker vos **Configurations d'état souhaité** (**DSC**) afin de les utiliser en temps opportun.

La section suivante porte sur la gestion des mises à jour.

Gestion des mises à jour

Comme vous le savez déjà, la gestion des mises à jour incombe au client. Celui-ci doit en effet gérer les mises à jour et les correctifs des services IaaS. La fonction Update Management d'Azure Automation peut être utilisée pour automatiser ou gérer les mises à jour et les correctifs de vos machines virtuelles Azure. Plusieurs méthodes permettent d'activer Update Management sur votre machine virtuelle Azure :

- Depuis votre compte Automation
- En accédant au portail Azure
- À l'aide d'une procédure opérationnelle
- À partir d'une machine virtuelle Azure

Les machines virtuelles Azure constituent la méthode la plus facile pour activer cette fonction. Toutefois, si vous avez un grand nombre de machines virtuelles et si vous devez activer Update Management, optez pour une solution évolutive telle qu'une procédure opérationnelle ou un compte Automation.

Maintenant que vous connaissez les scénarios, découvrons les concepts liés à Azure Automation.

Concepts liés à Azure Automation

Vous savez maintenant qu'Azure Automation nécessite un compte, appelé compte Azure Automation. Intéressons-nous désormais aux concepts liés à Azure Automation. Il est très important de comprendre le sens de chacun de ces termes, car nous allons les utiliser tout au long de ce chapitre. Commençons par les procédures opérationnelles.

Procédures opérationnelles

Une procédure opérationnelle Azure Automation est une collection d'instructions de script qui représente une étape unique dans l'automatisation de processus ou une automatisation complète des processus. Il est possible d'invoquer d'autres procédures opérationnelles à partir d'une procédure opérationnelle parent, et ces procédures opérationnelles peuvent être créés dans plusieurs langages de script. Les langages qui prennent en charge la création de procédures opérationnelles sont les suivants :

- PowerShell
- Python 2 (au moment de la rédaction du présent document)
- Charges de travail PowerShell
- PowerShell graphique
- Charges de travail graphiques PowerShell

La création d'un compte Automation est très simple et peut être effectuée à l'aide du portail Azure. Dans le panneau **All Services** (Tous les services), recherchez **Automation Account** (Compte Automation) ou recherchez-le dans le portail Azure. Comme mentionné précédemment, lorsque vous créez un compte, vous pourrez également créer un compte d'identification. La *Figure* 4.4 montre les entrées requises pour créer un compte Automation :

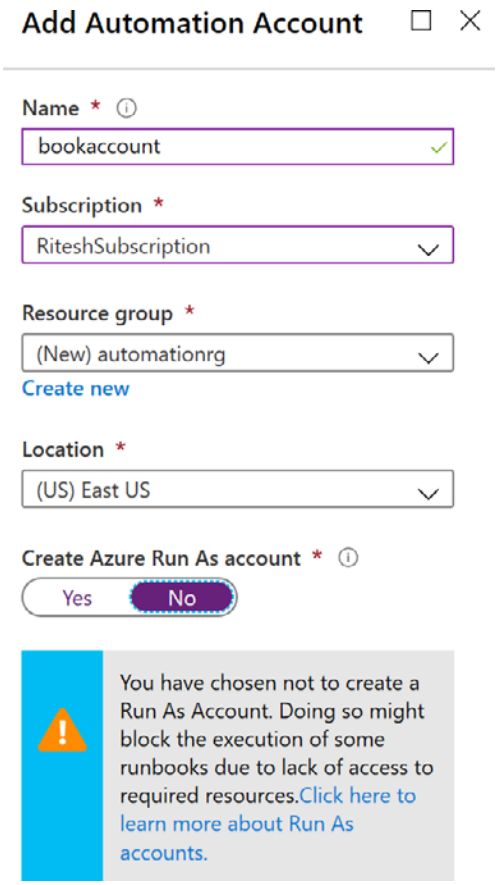


Figure 4.4 : création d'un compte Automation

Comptes d'identification

Par défaut, les comptes Azure Automation n'ont pas accès aux ressources incluses dans les abonnements Azure, y compris l'abonnement dans lequel ils sont hébergés. Un compte doit pouvoir accéder à un abonnement Azure et à ses ressources afin de les gérer. Un compte d'identification permet de fournir un accès aux abonnements et aux ressources en leur sein.

Il s'agit d'un exercice facultatif. Il peut y avoir au maximum un compte d'identification pour chaque abonnement classique et basé sur le gestionnaire de ressources. Toutefois, un compte Automation peut avoir besoin de se connecter à de nombreux abonnements. Si c'est le cas, il est conseillé de créer des ressources partagées pour chacun des abonnements et de les utiliser dans des procédures opérationnelles.

Après avoir créé le compte Automation, accédez à la vue **Run as accounts** (Comptes d'identification) sur le portail. Deux types de compte peuvent être créés. La *Figure 4.5* illustre les options de création d'un **Azure Run As Account** (Compte d'identification Azure) et d'un **Azure CIaasic Run As Account** (Compte d'identification classique Azure) disponibles dans le panneau **Run as accounts** (Comptes d'identification) :

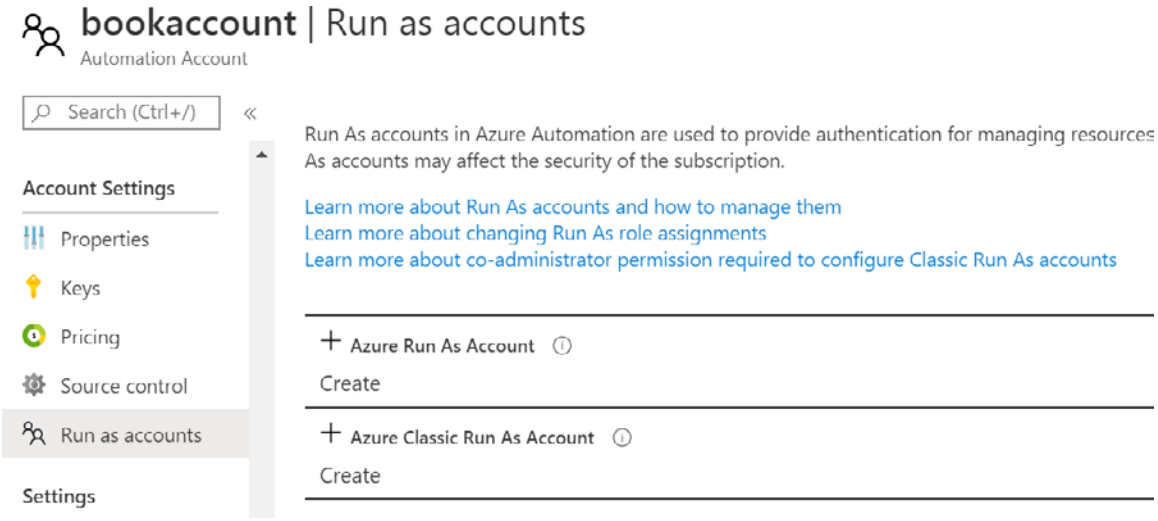


Figure 4.5 : options de comptes d'identification Azure

Ces comptes d'identification peuvent être créés via le portail Azure, PowerShell, l'interface de ligne de commande. Pour en savoir plus sur la création de ces comptes à l'aide de PowerShell, accédez à l'adresse suivante : https://docs.microsoft.com/azure/automation/manage-runas-account.

Dans le cas du compte d'identification ARM, ce script crée un principal de service Azure AD et un certificat, en accordant des autorisations RBAC aux contributeurs au principal de service nouvellement créé sur l'abonnement.

Tâches

La soumission d'une demande de tâche n'est pas directement liée à l'exécution de la demande de tâche en raison de l'architecture découplée d'Azure Automation. Elles sont liées indirectement à l'aide d'une banque de données. Lorsqu'une demande d'exécution d'une procédure opérationnelle est reçue par Automation, un enregistrement est créé dans sa base de données avec toutes les informations appropriées. Un autre service est exécuté sur plusieurs serveurs dans Azure, appelé Hybrid Runbook Worker. Ce service recherche toutes les nouvelles entrées ajoutées à la base de données pour l'exécution d'une procédure opérationnelle. Une fois qu'un nouvel enregistrement est détecté, celui-ci est verrouillé afin qu'aucun autre service ne puisse le lire, avant d'exécuter la procédure opérationnelle.

Ressources

Les ressources Azure Automation se réfèrent à des artefacts partagés qui peuvent être utilisés dans les procédures opérationnelles. Elles sont illustrées à la *Figure* 4.6 :

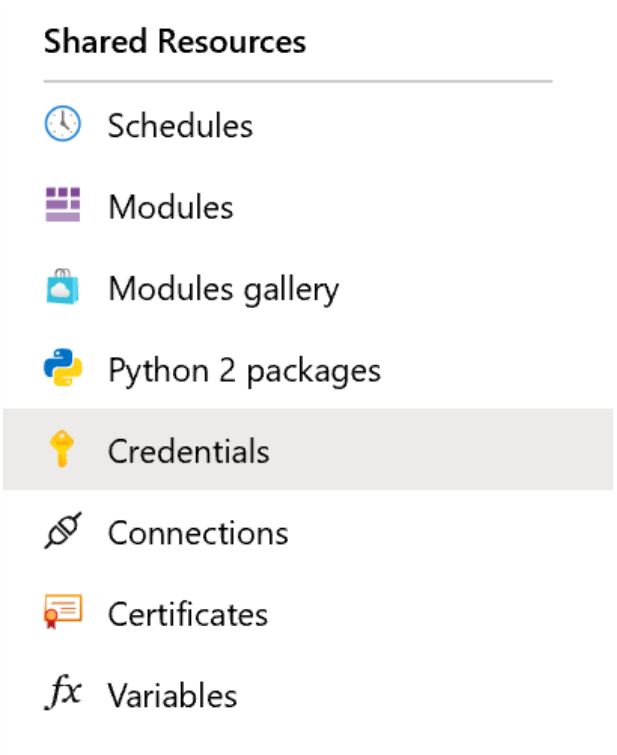


Figure 4.6 : artefacts partagés dans Azure Automation

Informations d'identification

Les informations d'identification font référence aux secrets, tels que la combinaison d'un nom d'utilisateur et d'un mot de passe, qui peuvent être utilisés pour se connecter à d'autres services d'intégration nécessitant une authentification. Ces informations d'identification peuvent être utilisées dans des procédures opérationnelles à l'aide de la cmdlet de commande PowerShell **`Get-AutomationPSCredential`** ainsi que du nom associé :

```
$myCredential = Get-AutomationPSCredential -Name 'MyCredential'
```

La syntaxe Python requiert que nous importions le module **`automationassets`** et que nous utilisions la fonction **`get_automation_credential`**, ainsi que l'authentifiant associé :

```
import automationassets
cred = automationassets.get_automation_credential("credtest")
```

Certificats

Les certificats font référence au certificat X.509 qui peut être acheté auprès des autorités de certification ou être auto-signé. Les certificats sont utilisés à des fins d'identification dans Azure Automation. Chaque certificat comporte une paire de clés connues sous le format clés privées/publiques. La clé privée est utilisée pour créer une ressource de certificat dans Azure Automation, tandis que la clé publique doit être disponible dans le service cible. À l'aide de la clé privée, le compte Automation peut créer une signature numérique et l'ajouter à la requête avant de l'envoyer au service cible. Le service cible peut extraire les détails (le hachage) de la signature numérique à l'aide de la clé publique déjà disponible et déterminer l'identité de l'expéditeur de la requête.

Les ressources de certificat stockent des informations de certificat et des clés dans Azure Automation. Ces certificats peuvent être utilisés directement dans des procédures opérationnelles. Ils sont également utilisés par les actifs de la connexion. La section suivante illustre la consommation des certificats dans une ressource de connexion. La ressource de connexion du principal de service Azure utilise une empreinte de certificat pour identifier le certificat qui doit être utilisé, tandis que d'autres types de connexion utilisent le nom de la ressource de certificat pour accéder au certificat.

Une ressource de certificat peut être créée en fournissant un nom et en téléchargeant un certificat. Il est possible de télécharger des certificats publics (fichiers **`.cer`**), ainsi que des certificats privés (fichiers **`.pfx`**). La partie privée du certificat comporte également un mot de passe qui doit être utilisé pour accéder au certificat.

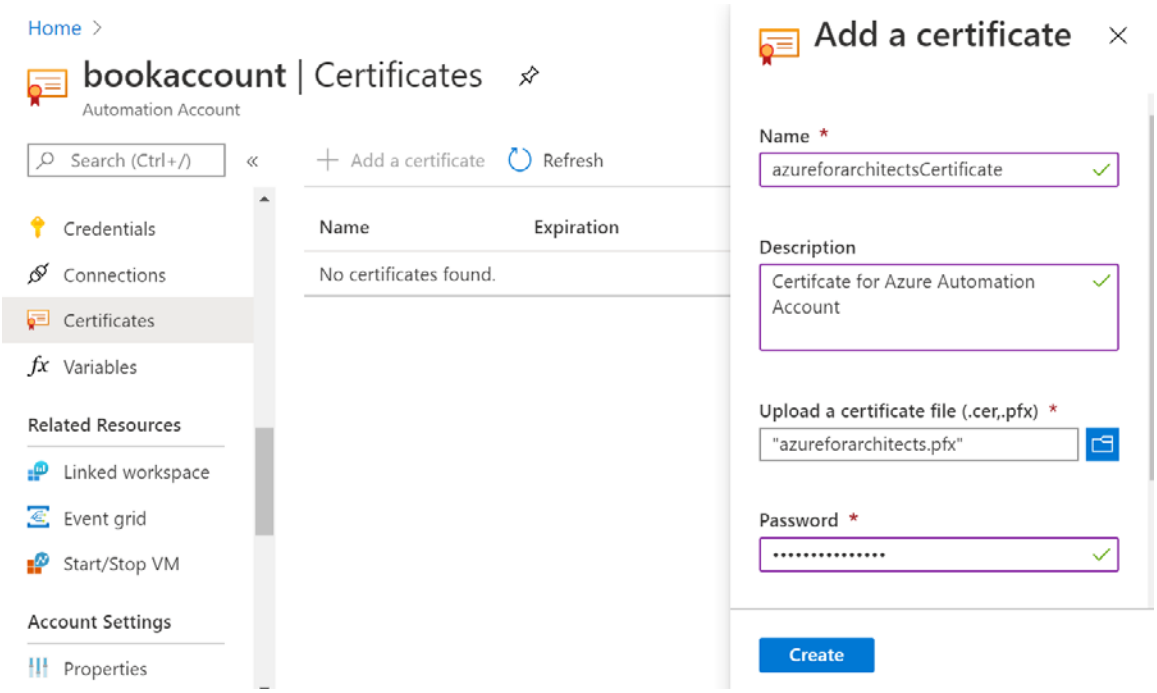


Figure 4.7 : ajout d'un certificat à Azure Automation

La création d'un certificat implique la saisie d'un nom et d'une description, le téléchargement du certificat et la saisie d'un mot de passe (dans le cas des fichiers `.pfx`). De plus, l'utilisateur doit être informé si le certificat est exportable ou non.

Un certificat doit être disponible avant que cette ressource de certificat ne puisse être créée. Les certificats peuvent être achetés auprès d'autorités de certification ou peuvent être générés. Les certificats générés sont appelés certificats auto-signés. Il est toujours recommandé d'utiliser les certificats émis par des autorités de certification pour des environnements importants tels que les environnements de production. Les certificats auto-signés suffisent pour des environnements de développement.

Pour générer un certificat auto-signé à l'aide de PowerShell, utilisez cette commande :

```
$cert = New-SelfSignedCertificate -CertStoreLocation "Cert:\CurrentUser\my"
-KeySpec KeyExchange -Subject "cn=azureforarchitects"
```

Un certificat sera créé dans le magasin de certificats de l'utilisateur actuel dans votre dossier personnel. Étant donné que ce certificat doit également être téléchargé vers la ressource de certificat Azure Automation, il doit être exporté vers le système de fichiers local, comme illustré à la *Figure 4.8* :

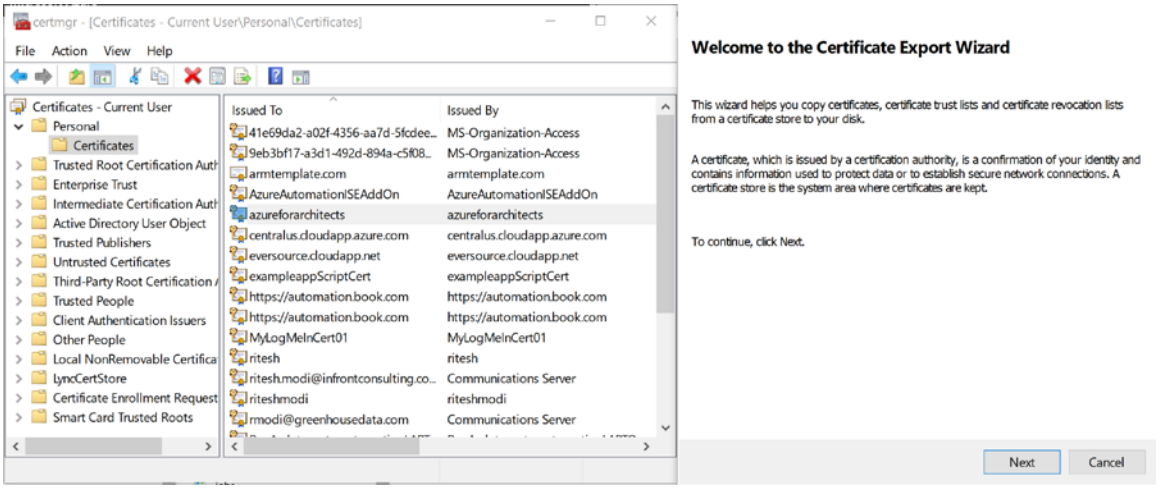


Figure 4.8 : exportation du certificat

Lors de l'exportation du certificat, la clé privée doit également être exportée. L'option **Yes, export the private key** (Oui, exporter la clé privée) doit donc être sélectionnée.

Sélectionnez l'option **Personal Information Exchange** (Échange d'informations personnelles). Les autres champs doivent conserver les valeurs par défaut.

Saisissez un mot de passe et le nom de fichier **C:\azureforarchitects.pfx** afin de procéder à l'exportation.

La connexion à Azure peut être effectuée de plusieurs façons. Toutefois, les certificats constituent la méthode la plus sécurisée. Un principal de service est créé sur Azure à l'aide du certificat. Le principal de service peut être authentifié à l'aide du certificat. La clé privée du certificat est détenue par l'utilisateur et la partie publique est conservée sur Azure. La section suivante porte sur la création d'un principal de service à l'aide du certificat créé dans cette section.

Création d'un principal de service à l'aide d'informations d'identification de certificat

Un principal de service peut être créé via le portail Azure, l'interface de ligne de commande Azure ou Azure PowerShell. Le script de création d'un principal de service à l'aide d'Azure PowerShell est disponible dans cette section.

Une fois que vous vous êtes connecté à Azure, le certificat créé dans la section précédente est converti en codage base64. Un principal de service, **azureforarchitects** est créé, et les informations d'identification de certificat sont associées au principal de service nouvellement créé. Enfin, le nouveau principal de service se voit accorder des autorisations de contrôle d'accès en fonction du rôle de contributeur sur l'abonnement :

```
Login-AzAccount

$certKey = [system.Convert]::ToBase64String($cert.GetRawCertData())

$sp =  New-AzADServicePrincipal -DisplayName "azureforarchitects"

New-AzADSpCredential -ObjectId $sp.Id -CertValue $certKey -StartDate
$cert.NotBefore -EndDate $cert.NotAfter

New-AzRoleAssignment -RoleDefinitionName contributor -ServicePrincipalName
$sp.ApplicationId

Get-AzADServicePrincipal -ObjectId $sp.Id

$cert.Thumbprint

Get-AzSubscription
```

Pour créer une ressource de connexion, l'ID d'application peut être obtenu à l'aide de la cmdlet de commande **`Get-AzADServicePrincipal`**. Le résultat est illustré à la *Figure* 4.9 :

```
ServicePrincipalNames : {http://azureforarchitects, ef52538d-9eb6-45e0-bf67-7f484b84cd25}
ApplicationId         : ef52538d-9eb6-45e0-bf67-7f484b84cd25
ObjectType            : ServicePrincipal
DisplayName           : azureforarchitects
Id                    : 15ee7335-9b96-4ae7-957f-62e4997e7b4d
Type                  :
```

Figure 4.9 : vérification du principal de service

L'empreinte de certificat peut être obtenue à l'aide de la référence de certificat ainsi que de **`SubscriptionId`**, qui peut être obtenue à l'aide de la cmdlet de commande **`Get-AzSubscription`**.

Connexions

Les ressources de connexion sont utilisées pour créer des informations de connexion sur des services externes. À cet égard, même Azure est considéré comme un service externe. Les ressources de connexion contiennent toutes les informations nécessaires pour effectuer la connexion à un service. Trois types de connexion prêts à l'emploi sont proposés par Azure Automation :

- Azure
- Certificat classique Azure
- Principal de service Azure

Il est recommandé d'utiliser un principal de service Azure pour vous connecter aux ressources Azure Resource Manager et d'utiliser le certificat classique Azure pour les ressources classiques Azure. Il est important de noter qu'Azure Automation ne fournit aucun type de connexion pour se connecter à Azure à l'aide d'informations d'identification telles qu'un nom d'utilisateur et un mot de passe.

Les certificats Azure et classiques Azure sont de nature similaire. Ils permettent de se connecter aux ressources basées sur les API de gestion Azure Service. En effet, Azure Automation crée une connexion de certificat classique Azure tout en créant un compte d'identification classique.

Le principal de service Azure est utilisé en interne par des comptes d'identification pour se connecter aux ressources basées sur Azure Resource Manager.

Une nouvelle ressource de connexion de type **AzureServicePrincipal** est illustrée à la *Figure 4.10*. Elle a besoin des éléments suivants :

- Le nom de la connexion. Un nom doit obligatoirement être saisi.
- Une description de la connexion. Cette valeur est facultative.
- Sélectionnez un **Type** approprié. Il est obligatoire de sélectionner une option. **`AzureServicePrincipal`** est sélectionné pour créer une ressource de connexion à toutes les fins de ce chapitre.
- **ApplicationId** , également connu sous le nom de **`clientid`**, est l'ID d'application généré lors de la création d'un principal de service. La section suivante illustre le processus de création d'un principal de service à l'aide d'Azure PowerShell. Il est obligatoire de fournir un ID d'application.
- **TenantId** est l'identificateur unique du locataire. Ces informations sont disponibles dans le portail Azure ou à l'aide de la cmdlet de commande **`Get-AzSubscription`**. Un identificateur de locataire doit obligatoirement être fourni.
- **CertificateThumbprint** est l'identificateur de certificat. Ce certificat doit déjà être chargé dans Azure Automation à l'aide de la ressource de certificat. Une empreinte de certificat doit obligatoirement être fournie.
- **SubscriptionId** est l'identificateur de l'abonnement. Il est obligatoire de fournir un ID d'abonnement.

Vous pouvez ajouter une nouvelle connexion à l'aide du panneau **Connexions** dans le compte Automation, comme illustré à la *Figure 4.10* :

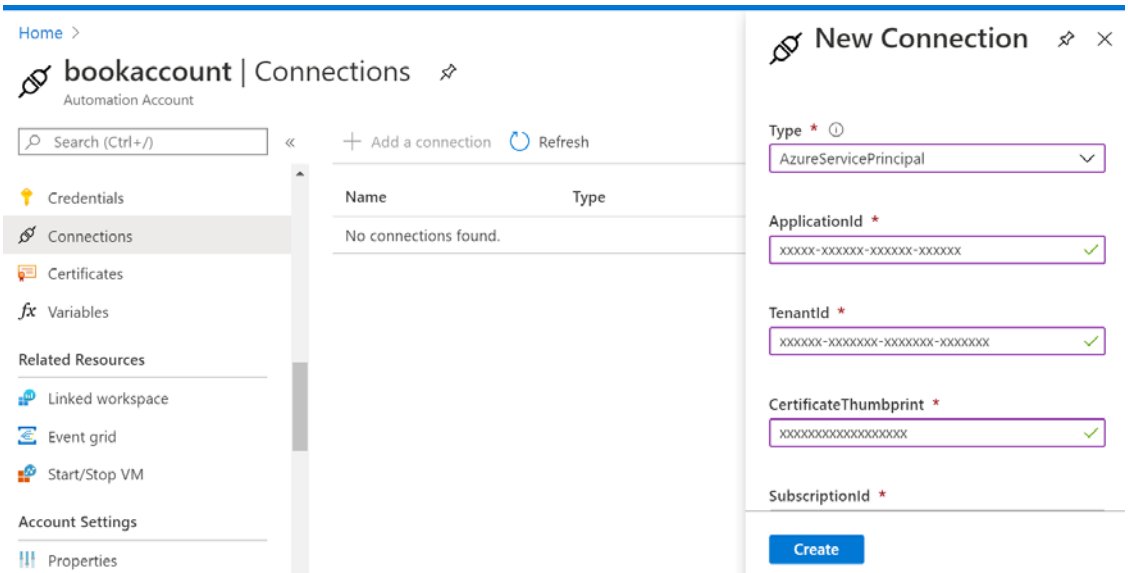


Figure 4.10 : ajout d'une nouvelle connexion au compte Automation

Création et exécution de procédures opérationnelles

Azure Automation permet la création de scripts d'automatisation connus sous le nom de procédures opérationnelles. Plusieurs procédures opérationnelles peuvent être créées à l'aide du portail Azure ou de PowerShell ISE. Elles peuvent également être importées à partir de la **Galerie de Runbooks**. Des fonctionnalités spécifiques peuvent être recherchées dans la galerie et l'intégralité du code est affichée dans la procédure opérationnelle.

Une procédure opérationnelle peut accepter des valeurs de paramètres, comme tout script PowerShell normal. L'exemple suivant utilise un seul paramètre nommé **`connectionName`** de type **`string`**. Il est obligatoire de fournir une valeur pour ce paramètre lors de l'exécution de cette procédure opérationnelle :

```
param(
    [parameter(mandatory=$true)]
    [string] $connectionName
)

$connection = Get-AutomationConnection  -name $connectionName
$subscriptionid = $connection.subscriptionid
$tenantid = $connection.tenantid
$applicationid = $connection.applicationid
$cretThumbprint = $connection.CertificateThumbprint

Login-AzureRMAccount -CertificateThumbprint $cretThumbprint
-ApplicationId $applicationid -ServicePrincipal -Tenant $tenantid

Get-AzureRMVM
```

La procédure opérationnelle utilise la cmdlet de commande **`Get-AutomationConnection`** pour référencer la ressource de connexion partagée. Le nom de la ressource est contenu dans la valeur du paramètre. Une fois que la référence à la ressource de connexion a été effectuée, les valeurs de la référence de connexion sont renseignées dans la variable **`$connection`**, et par la suite, elles sont affectées à plusieurs autres variables.

La cmdlet de commande **`Login-AzureRMAccount`** s'authentifie avec Azure et fournit les valeurs obtenues à partir de l'objet de connexion. Elle utilise le principal de service créé précédemment dans ce chapitre pour l'authentification.

Enfin, la procédure opérationnelle appelle la cmdlet de commande **`Get-AzureRMVm`** pour répertorier toutes les machines virtuelles de l'abonnement.

Par défaut, Azure Automation fournit toujours des modules **`AzureRM`** pour travailler avec Azure. Les modules **`Az`** ne sont pas installés par défaut. Nous allons ensuite installer un module **`Az`** manuellement dans le compte Azure Automation et utiliser des cmdlets de commande dans les procédures opérationnelles.

Procédures opérationnelles parent et enfant

Les procédures opérationnelles ont un cycle de vie, qui va de leur création à leur exécution. Ces cycles de vie peuvent être divisés en états de création et en états d'exécution.

Le cycle de vie de création est illustré à la *Figure 4.11*.

Lorsqu'une nouvelle procédure opérationnelle est créée, est présente l'état **New** (Nouveau). Elle est ensuite modifiée et enregistrée plusieurs fois et présente alors l'état **In edit** (En édition) et enfin, lorsqu'elle est publiée, son état est modifié à **Published** (Publié). Il est également possible de modifier une procédure opérationnelle publiée, et dans ce cas, celle-ci présente l'état **In edit** (En édition) à nouveau.

Figure 4.11 : cycle de vie de création

Le cycle de vie de l'exécution est décrit ci-après.

Le cycle de vie commence par le début d'une requête d'exécution de procédure opérationnelle. Une procédure opérationnelle peut être exécuté de plusieurs façons :

- Manuellement à l'aide du portail Azure
- Grâce à une procédure opérationnelle parent en tant que procédure opérationnelle enfant
- À l'aide d'un webhook

Peu importe la façon dont une procédure opérationnelle est initiée, le cycle de vie reste le même. Une requête d'exécution de procédure opérationnelle est reçue par le moteur Automation. Le moteur Automation crée une tâche et l'attribue à un travailleur de procédure opérationnelle. Actuellement, la procédure opérationnelle présente l'état **Queued** (Mis en file d'attente).

Il existe plusieurs travailleurs de procédure opérationnelle, et celui qui est choisi reprend la demande de tâche et modifie l'état à **Starting** (Démarrage). Si des problèmes de script et d'analyse surviennent dans le script à ce stade, l'état est modifié et passe à **Failed** (En échec) et l'exécution est interrompue.

Une fois que l'exécution de la procédure opérationnelle est lancée par le travailleur, l'état est modifié à **Running** (En cours d'exécution). La procédure opérationnelle peut présenter plusieurs états différents une fois qu'elles est en cours d'exécution.

L'état de la procédure opérationnelle sera modifié à **Completed** (Terminé) si l'exécution se produit sans aucune exception non gérée ou de fin.

La procédure opérationnelle peut être arrêtée manuellement par l'utilisateur, et présentera alors l'état **Stopped** (Arrêté).

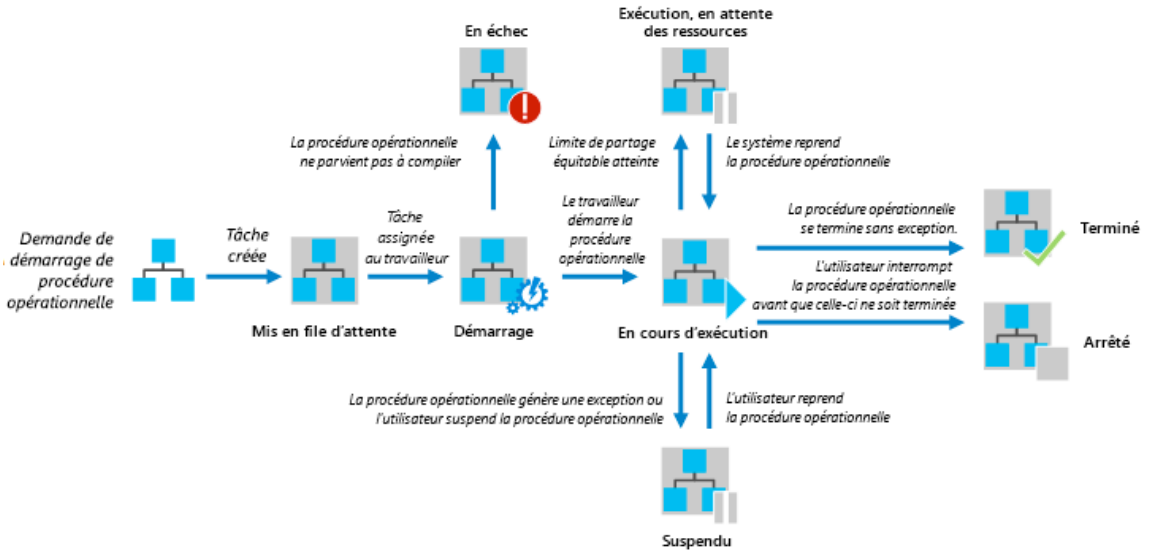


Figure 4.12 : cycle de vie de l'exécution des procédures opérationnelles

L'utilisateur peut également suspendre et reprendre l'exécution de la procédure opérationnelle.

Création d'une procédure opérationnelle

Une procédure opérationnelle peut être créée à l'aide du portail Azure, en accédant au menu **Runbook** (Procédure opérationnelle) dans le volet de navigation de gauche. Une procédure opérationnelle est associée à un nom et un type. Le type détermine le langage de script utilisé pour créer la procédure opérationnelle. Nous avons déjà discuté des langages possibles et dans ce chapitre, PowerShell sera principalement utilisé pour tous les exemples.

La création d'une procédure opérationnelle PowerShell est exactement la même que celle d'un script PowerShell. Elle peut déclarer et accepter plusieurs paramètres : les paramètres peuvent avoir des attributs tels que des types de données, qui sont obligatoires (comme tous les attributs de paramètre PowerShell). Elle peut appeler des cmdlets de commande PowerShell dont les modules sont disponibles et déjà chargés et déclarés, et peut appeler des fonctions et renvoyer des sorties.

Une procédure opérationnelle peut également invoquer une autre procédure opérationnelle. Elle peut invoquer une procédure opérationnelle enfant en ligne dans le processus et le contexte d'origine, ou dans un processus et un contexte distincts.

L'appel d'une procédure opérationnelle en ligne est similaire à l'appel d'un script PowerShell. L'exemple suivant illustre l'appel d'une procédure opérationnelle enfant à l'aide de l'approche en ligne :

```
.\ConnectAzure.ps1 -connectionName "azureforarchitectsconnection"

Get-AzSqlServer
```

Dans le code précédent, nous avons vu comment la procédure opérationnelle **ConnectAzure** accepte un paramètre nommé **connectionName** et une valeur appropriée lui est fournie. Cette procédure opérationnelle établit une connexion vers Azure après s'est authentifiée auprès d'un principal de service. Consultez la syntaxe permettant d'appeler une procédure opérationnelle enfant. Elle est très similaire à l'appel d'un script PowerShell général avec des paramètres.

La ligne de code suivante, **Get-AzVm**, récupère les informations appropriées dans Azure et répertorie les détails de la machine virtuelle. Vous remarquerez que, bien que l'authentification se produise au sein d'une procédure opérationnelle enfant, la cmdlet de commande **Get-AzVm** réussit et répertorie toutes les machines virtuelles dans l'abonnement, car la procédure opérationnelle enfant s'exécute dans la même tâche que celle de la procédure opérationnelle parent et partage le contexte.

Une procédure opérationnelle enfant peut également être appelée à l'aide de la cmdlet de commande **Start-AzurermAutomationRunbook** fournie par Azure Automation. Cette cmdlet de commande accepte le nom du compte Automation, le nom du groupe de ressources et le nom de la procédure opérationnelle, ainsi que des paramètres, comme indiqué ici :

```
$params = @{"connectionName"="azureforarchitectsconnection"}
$job = Start-AzurermAutomationRunbook '
    -AutomationAccountName 'bookaccount' '
    -Name 'ConnectAzure' '
    -ResourceGroupName 'automationrg' -parameters $params
if($job -ne $null) {
    Start-Sleep -s 100
    $job = Get-AzureAutomationJob -Id $job.Id -AutomationAccountName
'bookaccount'
    if ($job.Status -match "Completed") {
        $jobout = Get-AzureAutomationJobOutput '
                                    -Id $job.Id '
                                    -AutomationAccountName 'bookaccount' '
                                    -Stream Output
                    if ($jobout) {Write-Output $jobout.Text}
    }
}
```

L'utilisation de cette approche crée une tâche différente de la tâche parent et celles-ci s'exécutent dans des contextes différents.

Utilisation des modules Az

Jusqu'à présent, tous les exemples ont utilisé des modules **`AzureRM`**. Les procédures opérationnelles précédemment affichées seront réécrites pour utiliser les cmdlets de commande du module **`Az`**.

Comme mentionné précédemment, les modules **`Az`** ne sont pas installés par défaut. Ils peuvent être installés à l'aide du menu **Modules gallery** (Galerie des modules) dans Azure Automation.

Recherchez **`Az`** dans la galerie. Plusieurs modules associés seront affichés dans les résultats. Si le module **`Az`** est sélectionné pour être importé et installé, il génère une erreur indiquant que ses modules dépendants ne sont pas installés et qu'ils doivent être installés avant d'installer le module actuel. Le module se trouve dans le panneau **Modules gallery** (Galerie des modules) en effectuant une recherche du terme **`Az`**, comme illustré à la *Figure 4.13* :

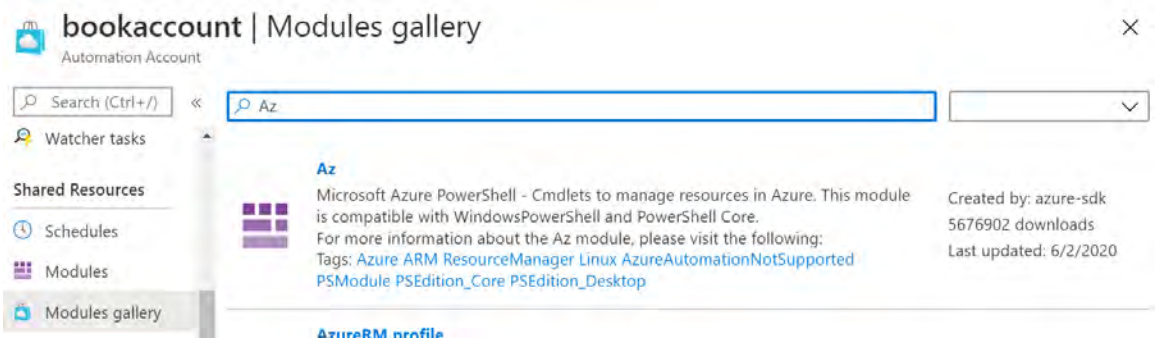


Figure 4.13 : recherche du module Az dans le panneau Galerie des modules

Au lieu de sélectionner le module **`Az`**, sélectionnez **Az.Accounts** et importez le module en suivant l'assistant, comme illustré à la *Figure 4.14* :

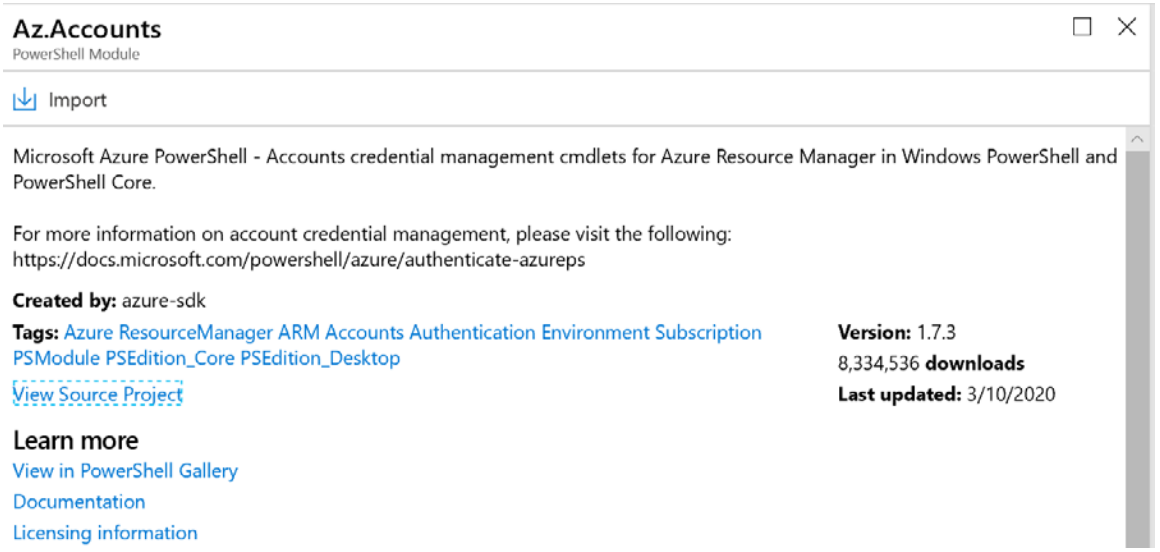


Figure 4.14 : importation du module Az.Accounts

Une fois **Az.Accounts** installé, le module **Az.Resources** peut être importé. Les cmdlets de commande relatives aux machines virtuelles Azure sont disponibles dans le module **Az.Compute** et peuvent également être importées à l'aide de la méthode utilisée pour importer les **Az.Accounts**

Une fois ces modules importés, les procédures opérationnelles peuvent utiliser les cmdlets de commande fournies par ces modules. La procédure opérationnelle **ConnectAzure** précédemment affichée a été modifié pour utiliser le module **Az** :

```
param(
    [parameter(mandatory=$true)]
    [string] $connectionName
)

$connection = Get-AutomationConnection  -name $connectionName
$subscriptionid = $connection.subscriptionid
$tenantid = $connection.tenantid
$applicationid = $connection.applicationid
$cretThumbprint = $connection.CertificateThumbprint

Login-AzAccount -CertificateThumbprint $cretThumbprint
-ApplicationId $applicationid -ServicePrincipal
-Tenant $tenantid  -SubscriptionId  $subscriptionid

Get-AzVm
```

Les deux dernières lignes du code sont importantes. Elles utilisent les cmdlets de commande **Az** au lieu des cmdlets de commande **AzureRM**.

L'exécution de cette procédure opérationnelle générera des résultats similaires à ceux-ci :

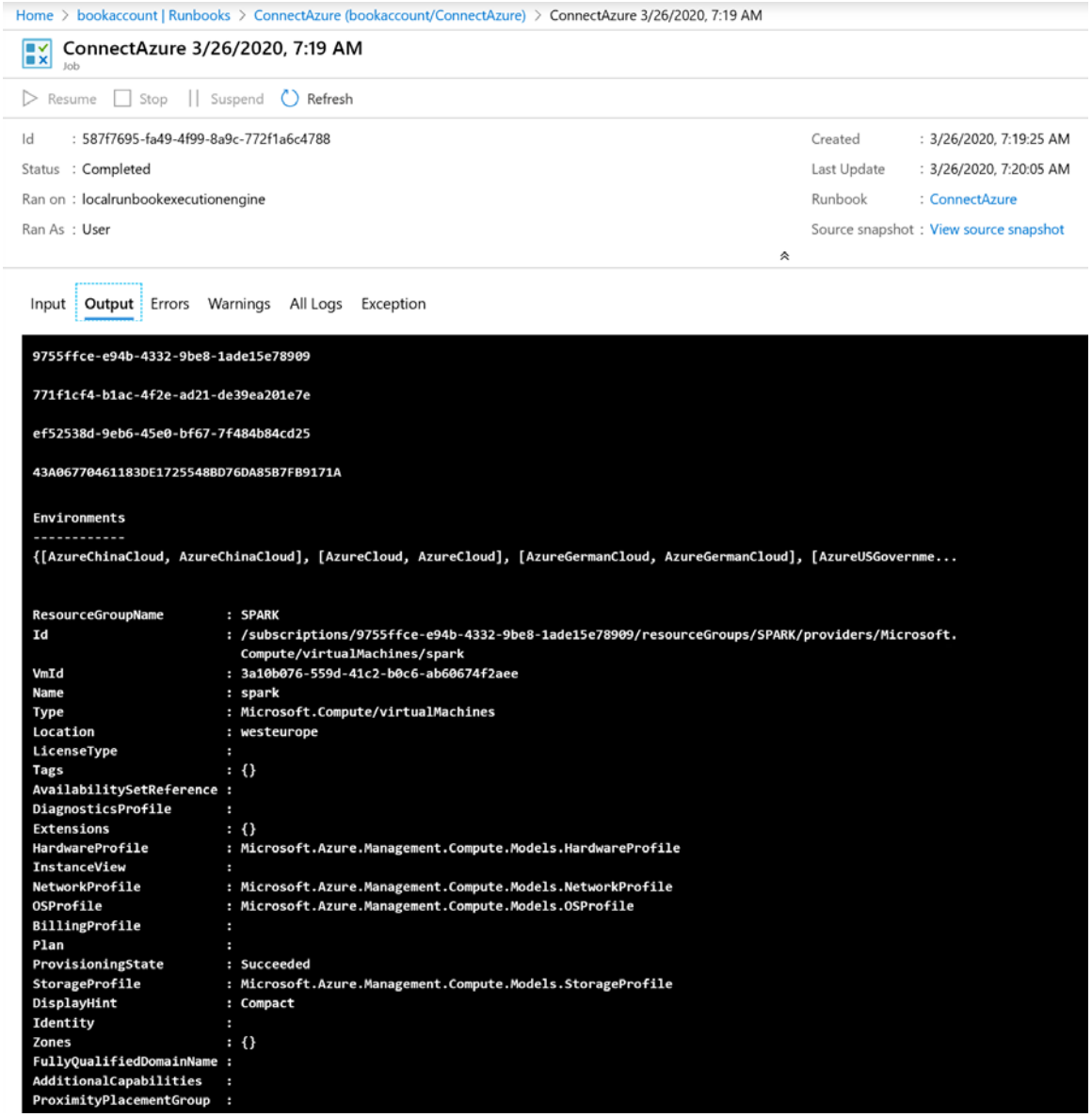


Figure 4.15 : le module Az.Accounts a été importé avec succès

Dans la section suivante, nous allons travailler avec webhooks.

Webhooks

Les webhooks sont devenus célèbres après l'avènement des points de terminaison REST et des charges utiles de données JSON. Les webhooks sont un concept important et une décision architecturale dans l'extensibilité de n'importe quelle application. Les webhooks sont des espaces réservés qui sont laissés dans des zones spéciales d'une application afin que l'utilisateur de l'application puisse remplir ces espaces réservés avec des URL de point de terminaison contenant une logique personnalisée. L'application appellera l'URL du point de terminaison, en transmettant automatiquement les paramètres nécessaires, puis exécutera la connexion qui y est disponible.

Les procédures opérationnelles Azure Automation peuvent être appelées manuellement à l'aide du portail Azure. Elles peuvent également être appelées à l'aide des cmdlets de commande PowerShell et d'Azure CLI. Des SDK sont disponibles dans plusieurs langages capables d'invoquer des procédures opérationnelles.

Les webhooks sont l'une des façons les plus puissantes d'invoquer une procédure opérationnelle. Il est important de noter que les procédures opérationnelles contenant la logique principale ne doivent jamais être exposées directement en tant que webhook. Elles doivent être appelées à l'aide d'une procédure opérationnelle parent, et c'est cette dernière qui doit être exposée en tant que webhook. La procédure opérationnelle parent doit s'assurer que les contrôles appropriés sont effectués avant d'appeler la principale procédure opérationnelle enfant.

La première phase de création d'un webhook consiste à créer une procédure opérationnelle normalement, comme nous l'avons fait précédemment. Une fois qu'une procédure opérationnelle a été créé, elle sera exposée en tant que webhook.

Une nouvelle procédure opérationnelle basée sur PowerShell nommée **`exposedrunbook`** est créée. Cette procédure opérationnelle utilise un seul paramètre, **`$WebhookData`**, du type d'objet. Celui-ci doit être nommé **`verbatim`**. Cet objet est créé par le runtime Azure Automation et est fourni à la procédure opérationnelle. Le runtime Azure Automation construit cet objet après avoir obtenu les valeurs d'en-tête de requête HTTP et le contenu du corps. Il renseigne ensuite les propriétés **`RequestHeader`** et **`RequestBody`** de cet objet :

```
param(
    [parameter(mandatory=$true)]
    [object] $WebhookData


)

$webhookname = $WebhookData.WebhookName
$headers = $WebhookData.RequestHeader
$body = $WebhookData.RequestBody

Write-output "webhook header data"
```

```
Write-Output $webhookname
Write-output $headers.message
Write-output $headers.subject

 $connectionname = (ConvertFrom-Json -InputObject $body)

./connectAzure.ps1 -connectionName  $connectionname[0].name
```

Les trois propriétés importantes de cet objet sont **`WebhookName`** , **`RequestHeader`** et **`RequestBody`**. Les valeurs sont extraites de ces propriétés et envoyées au flux de sortie par la procédure opérationnelle.

Le contenu de l'en-tête et du corps peut être tout ce que l'utilisateur fournit lors de l'appel du webhook. Ces valeurs sont remplies dans les propriétés respectives et deviennent disponibles dans la procédure opérationnelle. Dans l'exemple précédent, deux en-têtes sont définis par l'appelant, à savoir le **`message`**et l'en-tête **`état`**. L'appelant fournira également le nom de la connexion partagée à utiliser dans le cadre du contenu du corps.

Une fois la procédure opérationnelle créée, elle doit être publiée avant qu'un webhook puisse être créé. Après avoir publié la procédure opérationnelle, cliquez sur le menu **Webhook** en haut pour démarrer le processus de création d'un webhook pour la procédure opérationnelle, comme illustré à la *Figure 4.16* :

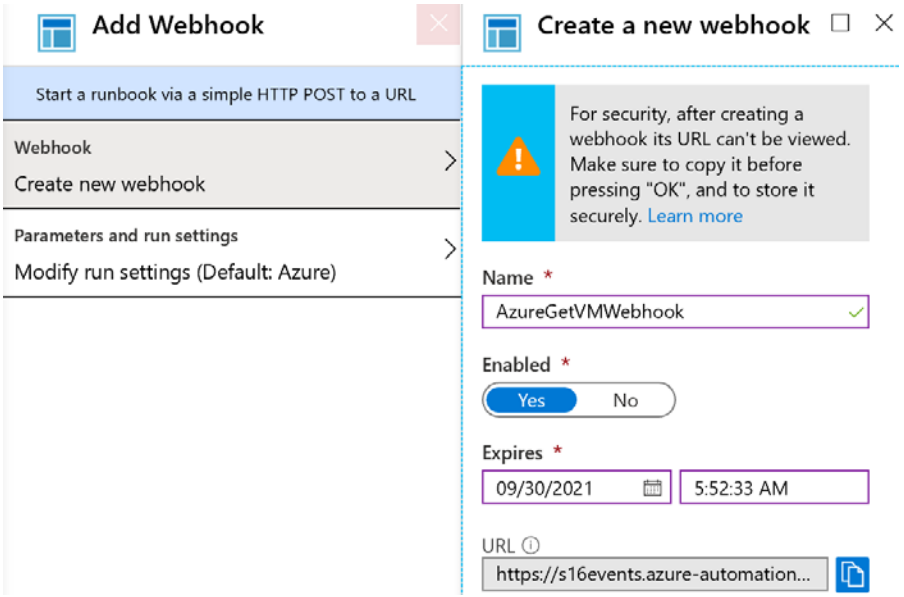


Figure 4.16 : création d'un webhook

Un nom doit être indiqué pour le webhook. Cette valeur est disponible dans la procédure opérationnelle à l'aide du paramètre **`WebhookData`** avec le nom de propriété **`WebhookName`**.

Le webhook peut présenter l'état **activé** ou **désactivé** , et il peut expirer à une date et une heure donnée. Il génère également une URL unique pour ce webhook et cette procédure opérationnelle. Cette URL doit être fournie à tous ceux qui souhaitent invoquer le webhook.

Appel d'un webhook

Les webhooks sont appelés en tant que requêtes HTTP à l'aide de la méthode **POST**. Lorsqu'un webhook est appelé, la requête HTTP aboutit dans Azure Automation pour démarrer une procédure opérationnelle. Il crée l'objet **WebHookData**, le remplit avec l'en-tête HTTP entrant et les données du corps, puis crée une tâche qui doit être effectuée par un travailleur de procédure opérationnelle. Cet appel utilise l'URL du webhook générée lors de l'opération précédente.

Le webhook peut être appelé à l'aide de Postman, par n'importe quel code capable d'appeler un point de terminaison **REST** à l'aide de la méthode **POST**. Dans l'exemple suivant, PowerShell sera utilisé pour appeler le webhook :

```
$uri = "https://s16events.azure-automation.net/
webhooks?token=rp0w93L60fAPYZQ4vryx1%2baN%2bS1Hz4F3qVdUaKUDzgM%3d"


$connection  = @(
            @{  name="azureforarchitectsconnection"}


        )
$body = ConvertTo-Json -InputObject $ connection
$header = @{ subject="VMS specific to Ritesh";message="Get all virtual
machine details"}



$response = Invoke-WebRequest -Method Post -Uri $uri -Body $body -Headers
$header
$jobid = (ConvertFrom-Json ($response.Content)).jobids[0]
```

Le code PowerShell déclare l'URL du webhook et exprime le corps au format JSON. Le champ **name** (nom) est défini sur **azureforarchitectsconnection** et l'en-tête est défini avec deux paires nom-valeur d'en-tête : **subject** (sujet) et **message**. Les données d'en-tête et de corps peuvent être récupérées dans la procédure opérationnelle à l'aide du paramètre **WebhookData**.

La cmdlet de commande **invoke-webrequest** déclenche la requête sur le point de terminaison mentionné précédemment à l'aide de la méthode **POST**, en fournissant à la fois l'en-tête et le corps.

La requête est asynchrone, et au lieu de la sortie de la procédure opérationnelle actuelle, l'identificateur de tâche est retourné en tant que réponse HTTP. Elle est également disponible dans le contenu de la réponse. La tâche est illustré à la *Figure 4.17* :

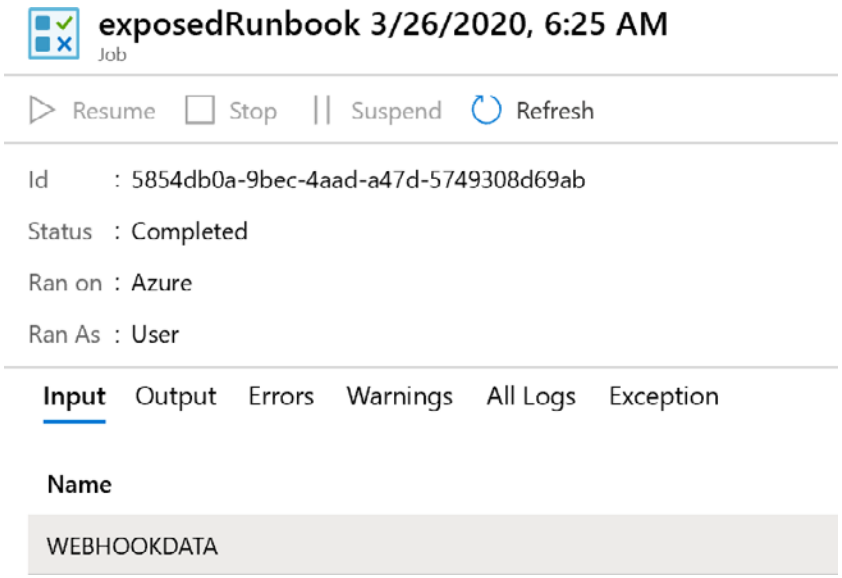


Figure 4.17 : consultation de la tâche

Les valeurs reçues dans le service d'automatisation de la procédure opérationnelle dans la requête HTTP s'affichent si vous cliquez sur **`WEBHOOKDATA`** :

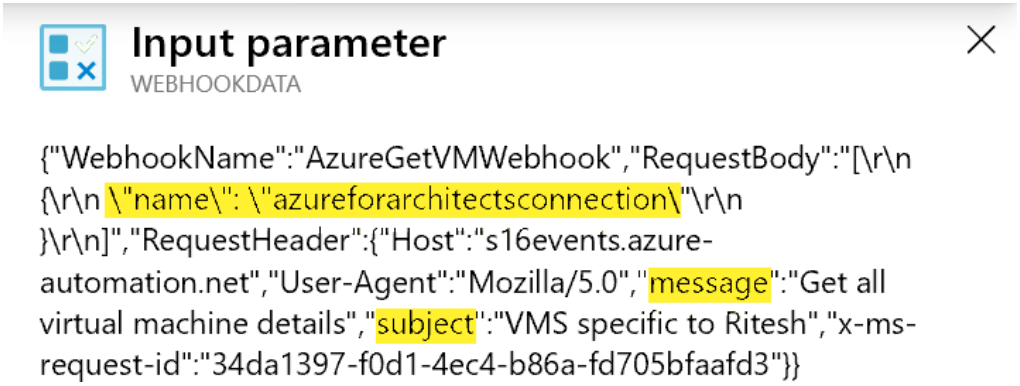


Figure 4.18 : vérification de la sortie

Cliquez sur le menu de sortie pour afficher la liste des machines virtuelles et SQL Server dans l'abonnement.

Les prochains concepts importants appliqués à Azure Automation sont Azure Monitor et les travailleurs hybrides, lesquels sont couverts en détail au cours des sections suivantes.

Appel d'une procédure opérationnelle à l'aide d'Azure Monitor

Les procédures opérationnelles Azure Automation peuvent être invoquées en tant que réponse aux alertes générées dans Azure. Azure Monitor est le service central qui gère les journaux et les mesures entre les ressources et les groupes de ressources dans un abonnement. Vous pouvez utiliser Azure Monitor pour créer des règles d'alerte et des définitions qui, lorsqu'elles sont déclenchées, exécutent les procédures opérationnelles Azure Automation. Ils peuvent invoquer une procédure opérationnelle Azure Automation dans sa forme par défaut ou un webhook qui, à son tour, peut exécuter la procédure opérationnelle associée. Cette intégration entre Azure Monitor et la possibilité d'invoquer des procédures opérationnelles ouvre de nombreuses opportunités d'automatisation pour corriger automatiquement l'environnement, dimensionner les ressources de calcul, ou prendre des mesures correctives sans intervention manuelle.

Les alertes Azure peuvent être créées et configurées dans les ressources individuelles et les niveaux de ressources. Toutefois, il est toujours judicieux de centraliser les définitions d'alertes pour faciliter et optimiser la maintenance et l'administration.

Passons au processus d'association d'une procédure opérationnelle à une alerte et à l'appel d'une procédure opérationnelle dans le cadre de l'alerte déclenchée.

La première étape consiste à créer une alerte, comme illustré à la *Figure* 4.19 :

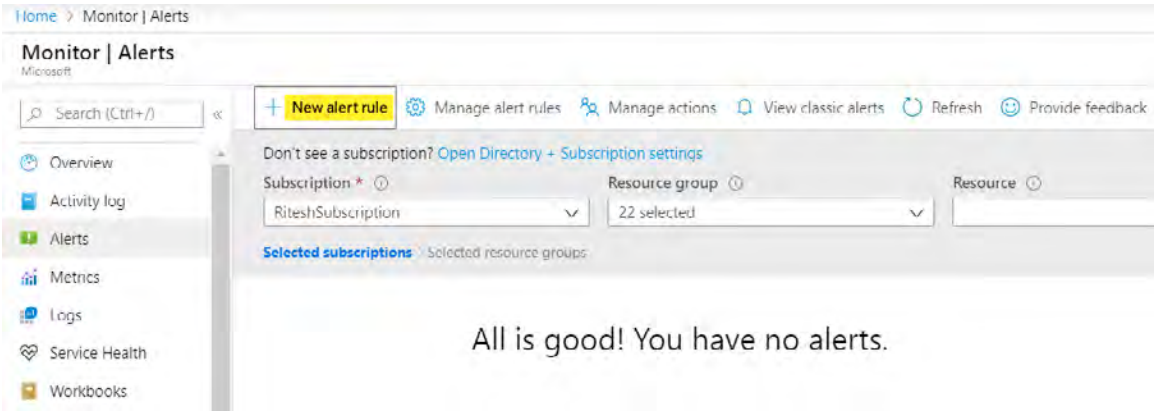


Figure 4.19 : création d'une règle d'alerte

Sélectionnez une ressource qui doit être surveillée et évaluée pour la génération d'alertes. Un groupe de ressources a été sélectionné dans la liste. Celui-ci active automatiquement toutes les ressources au sein du groupe de ressources. Il est possible de supprimer les sélections de ressources du groupe de ressources :

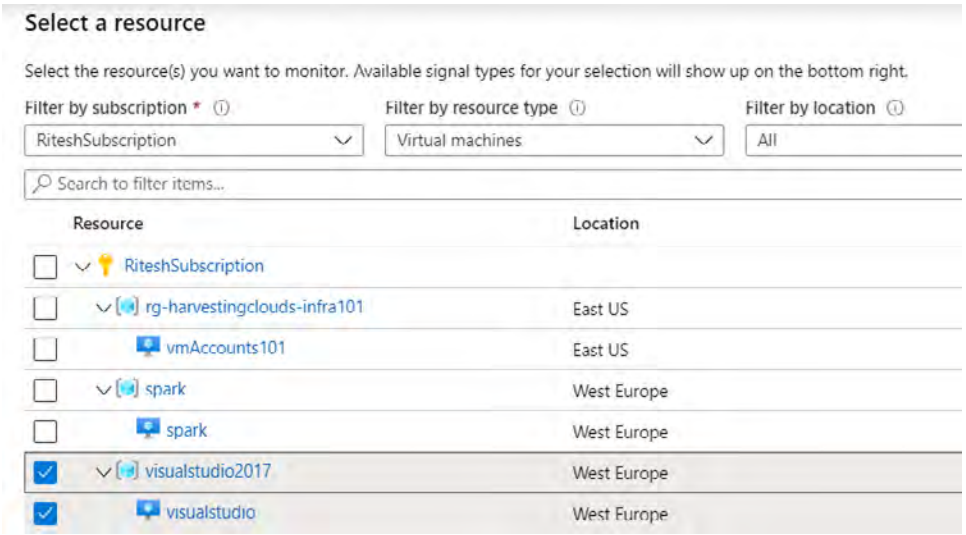


Figure 4.20 : sélection de la portée de l'alerte

Configurez la condition et les règles qui doivent être évaluées. Sélectionnez le nom de signal **Power Off Virtual Machine** après avoir sélectionné le **Signal type** (Type de signal) **Activity Log** (Journal d'activité) :

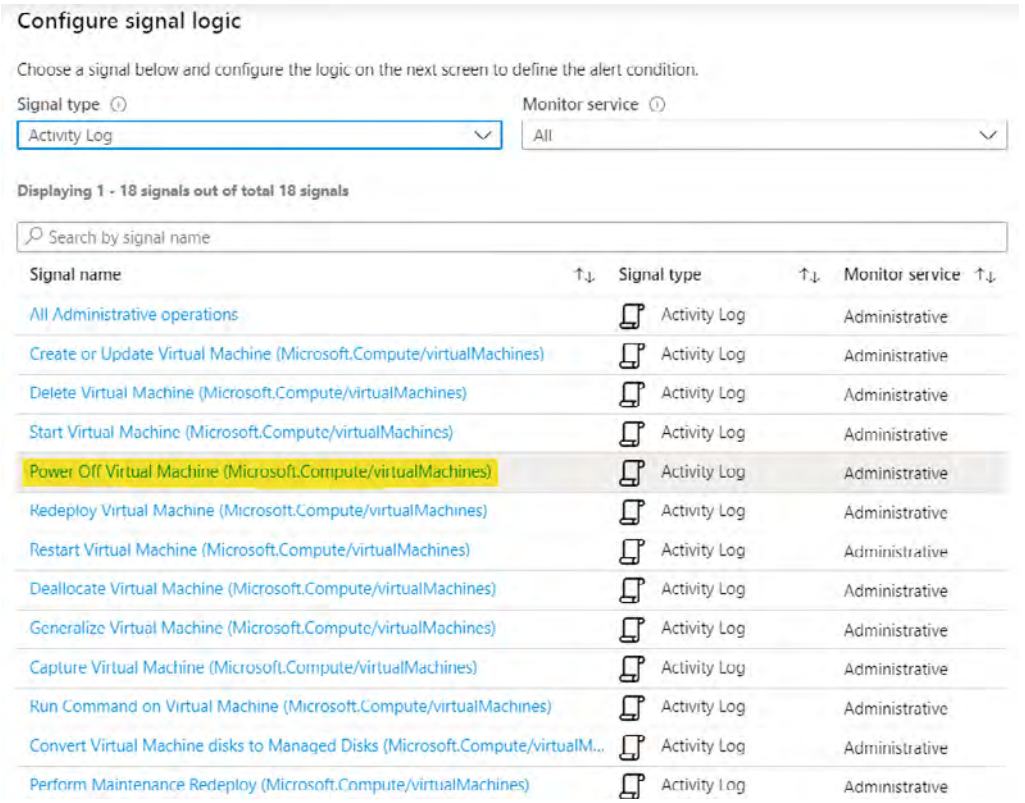


Figure 4.21 : sélection du type de signal

La fenêtre résultante vous permet de configurer la **logique/la condition d'alerte**. Sélectionnez `critical` (critique) pour **Event Level** (Niveau d'événement) et définissez **Status** (État) à `Succeeded` (Réussite) :

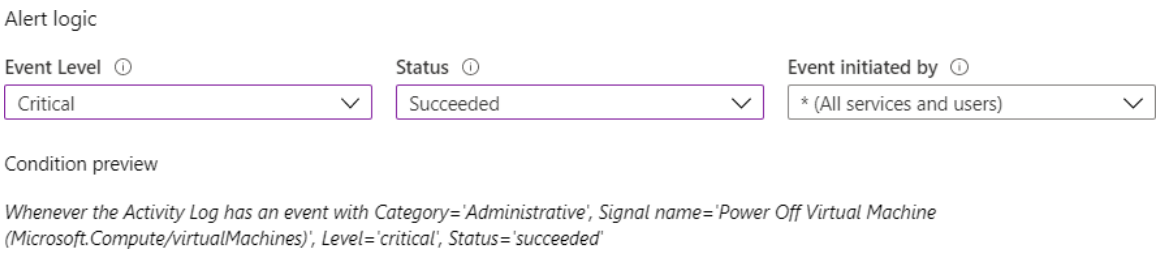


Figure 4,22 : définition de la logique d'alerte

Après avoir déterminé la condition d'alerte, la configuration la plus importante est celle qui configure la réponse à l'alerte en appelant une procédure opérationnelle. Nous pouvons utiliser des **groupes d'action** pour configurer la réponse à une alerte. Ils offrent de nombreuses options pour appeler une fonction Azure, un webhook ou une procédure opérationnelle Azure Automation, ainsi que pour envoyer des e-mails et des SMS.

Créez un groupe d'action en indiquant un nom, un nom court, un abonnement d'hébergement, un groupe de ressources et un **nom d'action**. Correspondant au **Action name** (Nom d'action), sélectionnez l'option `Automation Runbook` (Procédure opérationnelle Automation) en tant que **Action Type** (Type d'action) :

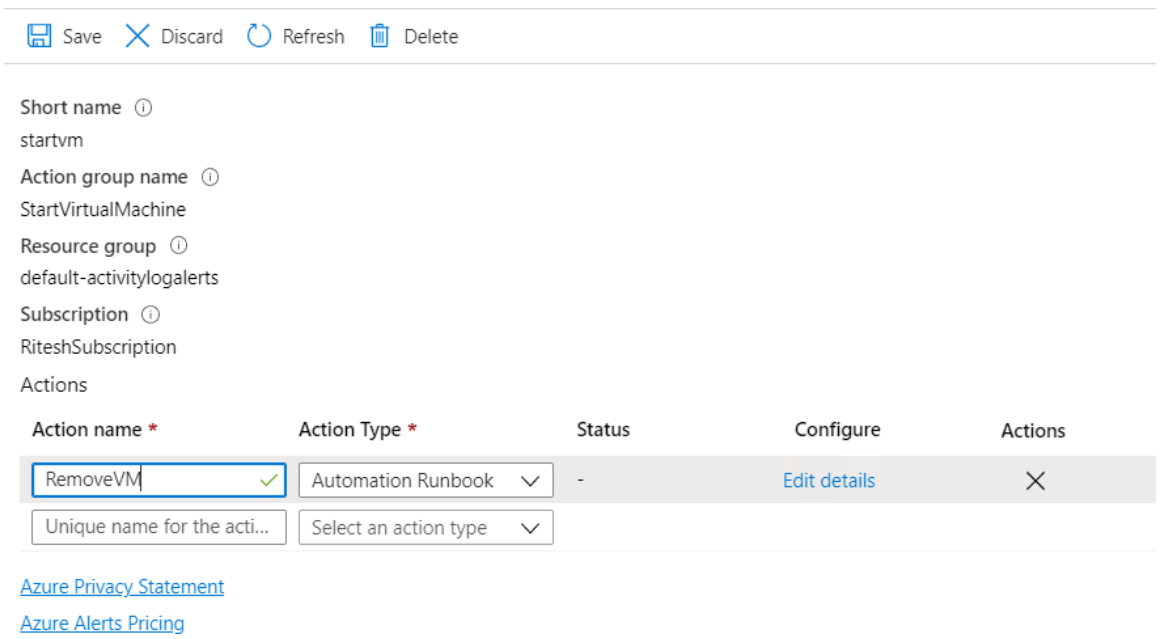


Figure 4,23 : configuration du groupe d'action

La sélection d'une procédure opérationnelle d'automatisation ouvrira un autre panneau permettant de sélectionner un compte et une procédure opérationnelle Azure Automation appropriés. Plusieurs procédures opérationnelles sont prêtes à l'emploi et l'une d'entre elles a été utilisée ici :

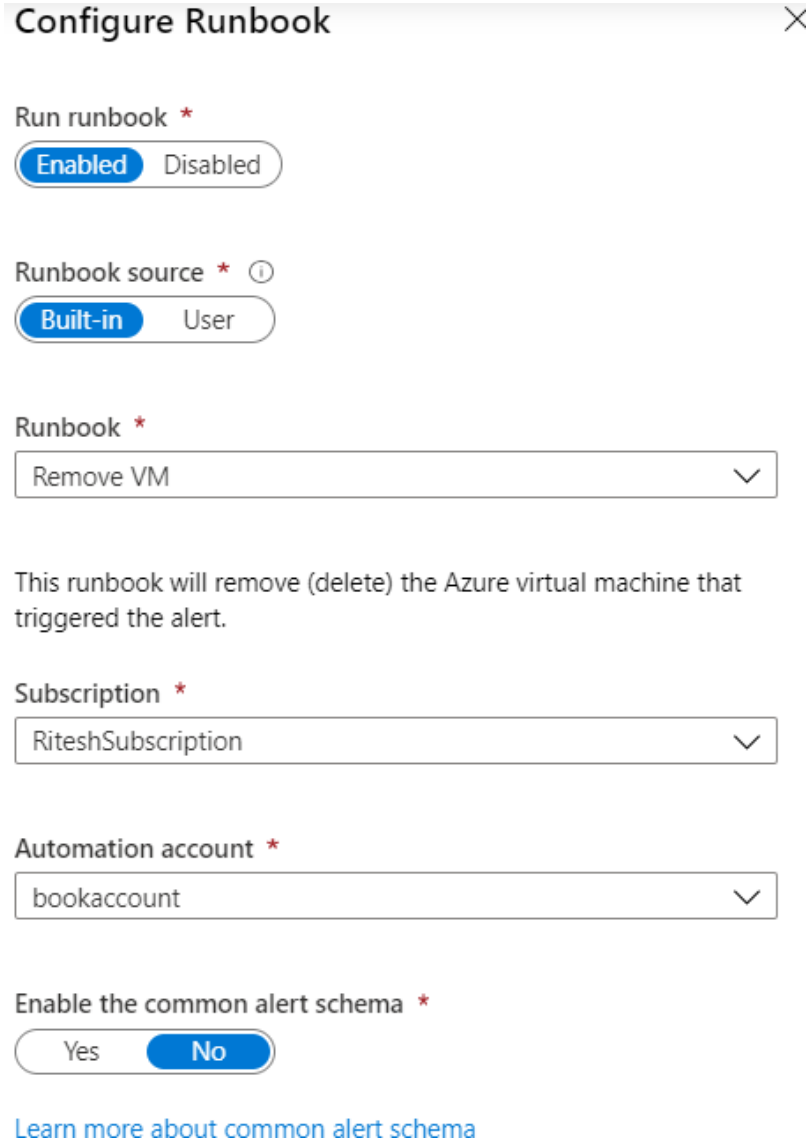


Figure 4.24 : création de la procédure opérationnelle

Enfin, indiquez un nom et un groupe de ressources d'hébergement pour créer une alerte.

Si la machine virtuelle est désallouée manuellement, la condition d'alerte est satisfaite et elle déclenchera une alerte :

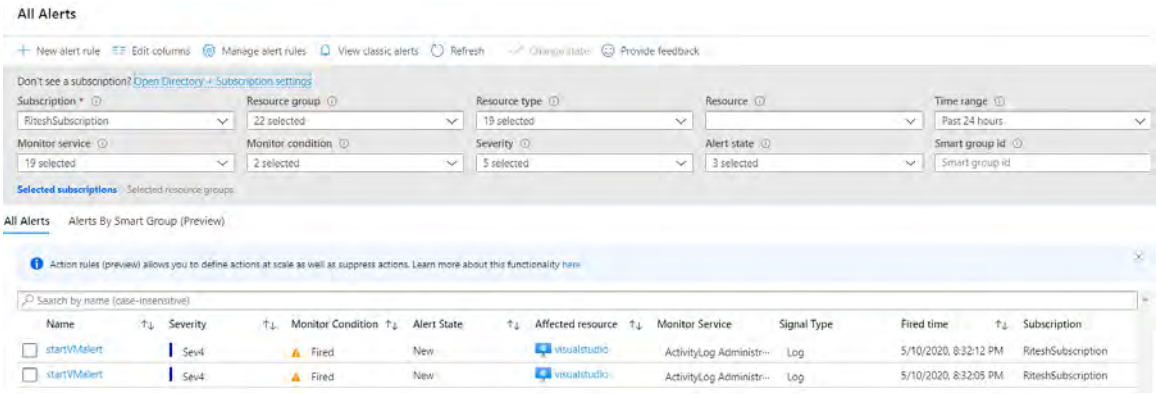


Figure 4.25 : alertes de test

Si vous vérifiez les détails de la machine virtuelle après quelques secondes, vous constaterez que la machine virtuelle est supprimée :

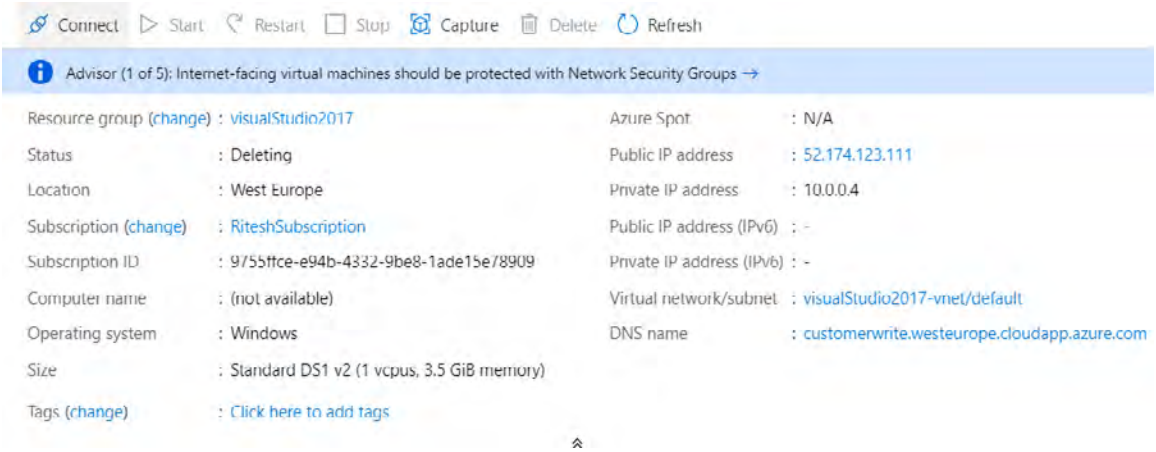


Figure 4.26 : vérification des résultats

Travailleurs hybrides

Jusqu'à présent, toute l'exécution des procédures opérationnelles a été effectué principalement sur l'infrastructure fournie par Azure. Les travailleurs de procédure opérationnelle sont des ressources de calcul Azure qui sont mises en service par Azure avec des modules et des actifs appropriés déployés sur ces dernières. Toute l'exécution des procédures opérationnelles se produit sur ce calcul. Toutefois, les utilisateurs peuvent apporter leur propre calcul et exécuter la procédure opérationnelle sur ce calcul fourni par l'utilisateur plutôt que sur le calcul Azure par défaut.

Cela présente plusieurs avantages. Tout d'abord, l'intégralité de l'exécution et de ses journaux appartient à l'utilisateur, Azure n'ayant aucune visibilité sur ceux-ci. Deuxièmement, le calcul fourni par l'utilisateur peut se trouver sur n'importe quel Cloud, ainsi que sur site.

L'ajout d'un travailleur hybride implique plusieurs étapes

- Tout d'abord, un agent doit être installé sur le calcul fourni par l'utilisateur. Microsoft fournit un script capable de télécharger et de configurer automatiquement l'agent. Ce script est disponible à l'adresse https://www.powershellgallery.com/packages/New-OnPremiseHybridWorker/1.6.

  Le script peut également être exécuté à partir de PowerShell ISE en tant qu'administrateur au sein du serveur qui doit faire partie du travailleur hybride à l'aide de la commande suivante :

  ```
  Install-Script -Name New-OnPremiseHybridWorker -verbose
  ```

- Une fois le script installé, il peut être exécuté en même temps que les paramètres relatifs aux détails du compte Azure Automation. Un nom est également fourni pour le travailleur hybride. Si le nom n'existe pas déjà, il sera créé ; s'il existe, le serveur sera ajouté au travailleur hybride existant. Plusieurs serveurs peuvent être associés à un seul travailleur hybride, et il est également possible d'avoir plusieurs travailleurs hybrides :

  ```
  New-OnPremiseHybridWorker.ps1 -AutomationAccountName bookaccount
  -AAResourceGroupName automationrg '
  -HybridGroupName "localrunbookexecutionengine" '
  -SubscriptionID xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
  ```

- Une fois l'exécution terminée, la navigation vers le portail affichera une entrée pour un travailleur hybride, comme illustré à la *Figure* 4.27 :

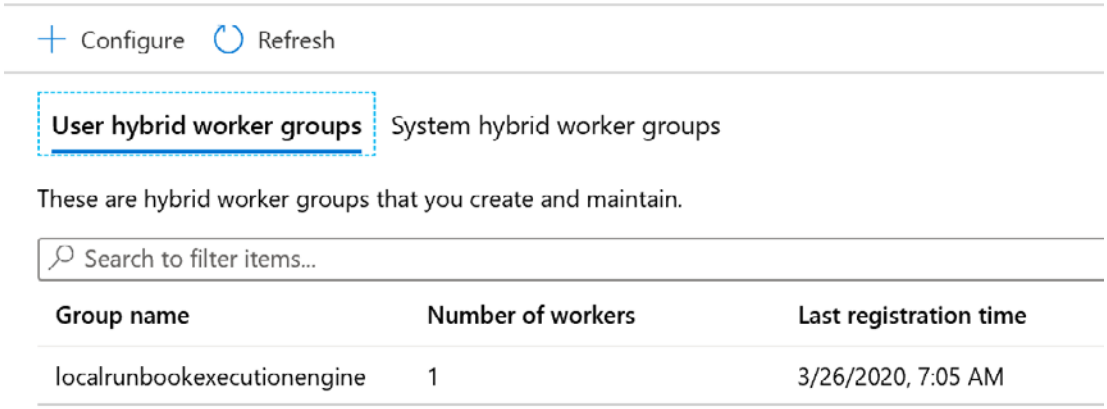


Figure 4.27 : vérification des groupes de travailleurs hybrides de l'utilisateur

- Si, à ce moment-là, une procédure opérationnelle Azure est exécutée et présente une dépendance sur le module **`Az`** et un certificat personnalisé chargé sur la ressource de certificat, elle échouera car le module **`Az`** et le certificat seront introuvables :

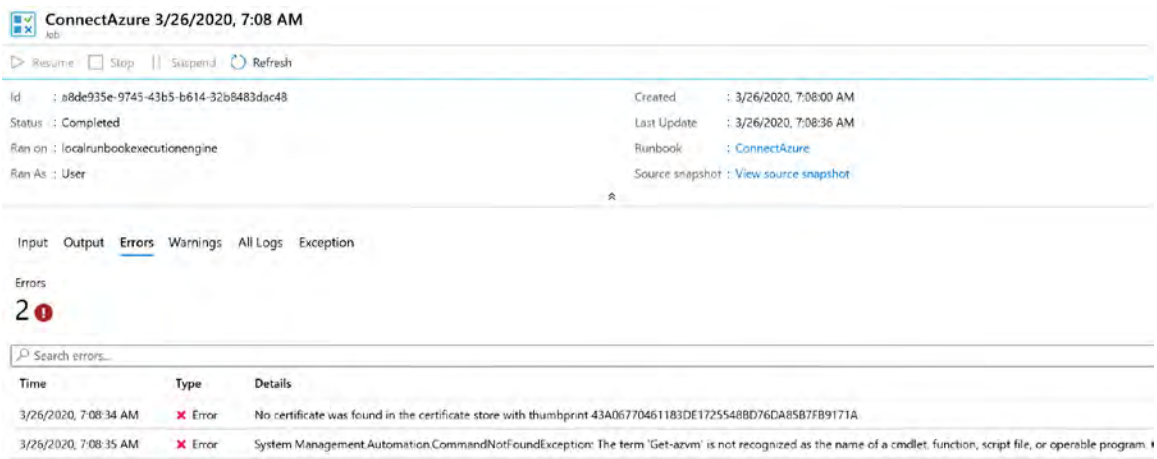


Figure 4.28 : vérification des erreurs

- Installez le module **`Az`** à l'aide de la commande suivante sur le serveur :

```
Install-module -name Az -AllowClobber -verbose
```

Il est également important que le certificat **`.pfx`** soit disponible sur ce serveur. Le certificat précédemment exporté doit être copié sur le serveur et installé manuellement.

- Après l'installation du module `Az` et du certificat, la réexécution de la procédure opérationnelle sur le travailleur hybride est illustrée à la *Figure* 4.29 . Elle doit afficher la liste des machines virtuelles de l'abonnement :

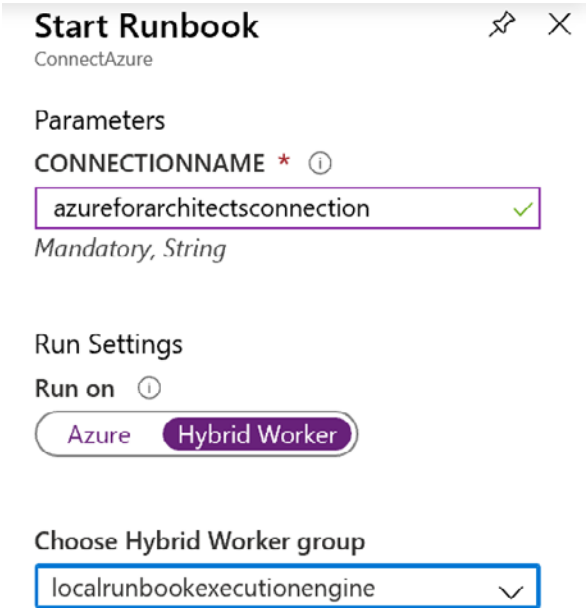


Figure 4.29 : configuration d'une procédure opérationnelle à exécuter sur un travailleur hybride

Lorsque nous avons abordé les différents scénarios, nous avons parlé de la gestion de la configuration. Dans la section suivante, nous allons aborder plus en détail la gestion de la configuration avec Azure Automation.

Configuration d'état Azure Automation

Azure Automation fournit un serveur pull **Desired State Configuration** (**DSC**) avec chaque compte Azure Automation. Le serveur pull peut contenir des scripts de configuration qui peuvent être tirés par des serveurs dans les clouds et sur site. Cela signifie qu'Azure Automation peut être utilisé pour configurer n'importe quel serveur, où qu'il soit hébergé dans le monde.

Le DSC a besoin d'un agent local sur ces serveurs, également appelé **gestionnaire de configuration locale** (**LCM**). Il doit être configuré avec le serveur pull Azure Automation DSC afin de pouvoir télécharger la configuration requise et configurer automatiquement le serveur.

La configuration automatique peut être planifiée de façon à être périodique (toutes les demi-heures par défaut), et si l'agent détecte une déviation dans la configuration du serveur par rapport à celle qui est indiquée dans le script DSC, il l'a corrige automatiquement et ramène le serveur à l'état souhaité et attendu.

Dans cette section, nous allons configurer un serveur hébergé sur Azure. Le processus restera le même, quel que soit l'emplacement du serveur : sur un Cloud ou sur site.

La première étape consiste à créer une configuration DSC. Un exemple de configuration est illustré ici, et des configurations complexes peuvent être créées de la même manière :

```
configuration ensureiis {
import-dscresource -modulename psdesiredstateconfiguration


node localhost {
        WindowsFeature iis {
            Name = "web-server"
            Ensure = "Present"


        }
    }
}
```

La configuration est assez simple à effectuer. Elle importe le module DSC de base **PSDesiredStateConfiguration** et déclare une configuration à nœud unique. Cette configuration n'est associée à aucun nœud spécifique et peut être utilisée pour configurer n'importe quel serveur. La configuration doit configurer un serveur Web IIS et s'assurer qu'il est présent sur n'importe quel serveur auquel elle est appliquée.

Cette configuration n'est pas encore disponible sur le serveur pull Azure Automation DSC. La première étape consiste donc à importer la configuration dans le serveur pull. Cette opération peut être effectuée à l'aide de la cmdlet de commande **Import-AzAutomationDscConfiguration** du compte Automation, comme illustré ci-après :

```
Import-AzAutomationDscConfiguration -SourcePath "C:\Ritesh\ensureiis.ps1"
-AutomationAccountName bookaccount -ResourceGroupName automationrg -Force
-Published
```

Voici quelques éléments importants à prendre en compte. Le nom de la configuration doit correspondre au nom du fichier et ne doit contenir que des caractères alphanumériques et des soulignements. Une bonne convention de nommage consiste à utiliser des combinaisons verbe/nom. Le chemin du fichier de configuration et les détails du compte Azure Automation sont nécessaires aux cmdlets de commande pour importer le script de configuration.

À ce stade, la configuration est visible dans le portail :

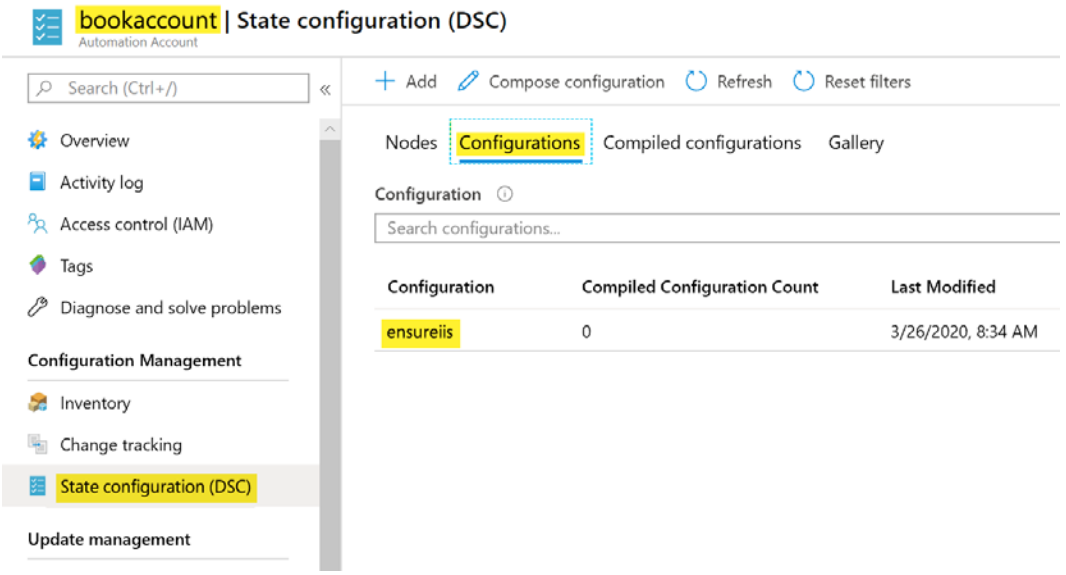


Figure 4.30 : ajout d'une configuration

Une fois que le script de configuration est importé, il est compilé et stocké dans le serveur pull DSC à l'aide de la cmdlet de commande **`Start-AzAutomationDscCompilationJob`**, comme illustré ci-après :

```
Start-AzAutomationDscCompilationJob -ConfigurationName 'ensureiis'
-ResourceGroupName 'automationrg' -AutomationAccountName 'bookaccount'
```

Le nom de la configuration doit correspondre à celui qui a été récemment téléchargé, et la configuration compilée doit être disponible dans l'onglet **Compiled configurations** (Configurations compilées), comme illustré à la *Figure 4.31* :

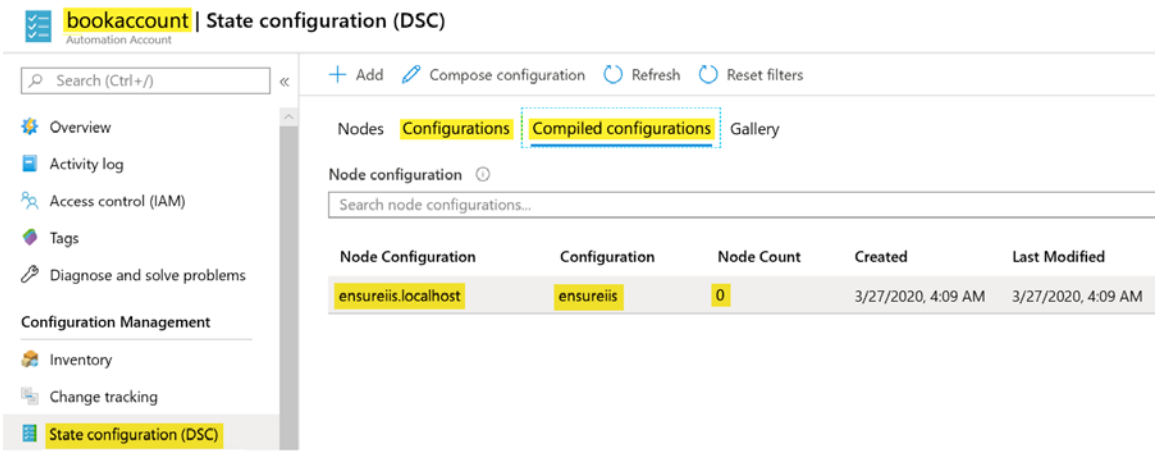


Figure 4.31 : liste des configurations compilées

Il est important de noter que le **nombre de nœuds** affiché dans la *Figure 4.31* est **`0`**. Cela signifie qu'une configuration de nœud nommée **`ensureiss.localhost`** existe, mais qu'elle n'est affectée à aucun nœud. L'étape suivante consiste à attribuer la configuration au nœud.

Une configuration DSC compilée est désormais disponible sur le serveur pull DSC, sans qu'il n'y ait de nœuds à gérer. L'étape suivante consiste à intégrer les machines virtuelles et à les associer au serveur pull DSC. Cette opération est effectuée à l'aide de la cmdlet de commande **`Register-AzAutomationDscNode`** :

```
Register-AzAutomationDscNode -ResourceGroupName 'automationrg'
-AutomationAccountName 'bookaccount' -AzureVMLocation "west
Europe" -AzureVMResourceGroup 'spark' -AzureVMName 'spark'
-ConfigurationModeFrequencyMins 30 -ConfigurationMode 'ApplyAndAutoCorrect'
```

Cette cmdlet de commande utilise le nom du groupe de ressources pour la machine virtuelle et le compte Azure Automation. Elle configure également le mode de configuration et la propriété **`configurationModeFrequencyMins`** du gestionnaire de configuration locale de la machine virtuelle. Cette configuration vérifie et corrige automatiquement tout écart par rapport à la configuration qui lui est appliquée toutes les 30 minutes.

Si **`VMresourcegroup`** n'est pas spécifié, la cmdlet de commande tente de trouver la machine virtuelle dans le même groupe de ressources que le compte Azure Automation, et si aucune valeur d'emplacement n'est indiquée pour la machine virtuelle, elle tente de trouver la machine virtuelle dans la région Azure Automation. Il est toujours préférable de renseigner ces valeurs. Notez que cette commande ne peut être utilisée que pour les machines virtuelles Azure, car elle demande **`AzureVMname`** explicitement. Pour les serveurs situés sur d'autres Clouds et sur site, utilisez la cmdlet de commande **`Get-AzAutomationDscOnboardingMetaconfig`**.

Une nouvelle entrée de configuration de nœud est désormais affichée dans le portail, comme illustré à la *Figure 4.32* :

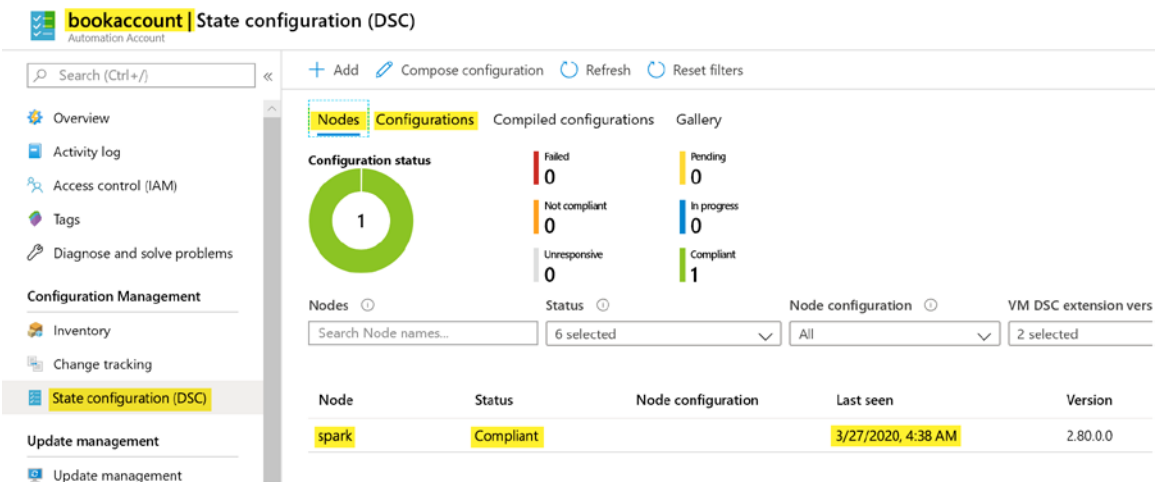


Figure 4.32 : vérification de l'état du nœud

Les informations relatives au nœud peuvent être obtenues comme suit :

```
$node = Get-AzAutomationDscNode -ResourceGroupName 'automationrg'
-AutomationAccountName 'bookaccount' -Name 'spark'
```

Une configuration peut être attribuée au nœud :

```
Set-AzAutomationDscNode -ResourceGroupName 'automationrg'
-AutomationAccountName 'bookaccount' -NodeConfigurationName 'ensureiis.
localhost' -NodeId $node.Id
```

Une fois la compilation terminée, elle peut être attribuée aux nœuds. L'état initial est **Pending** (En attente), comme illustré à la *Figure* 4.33 :

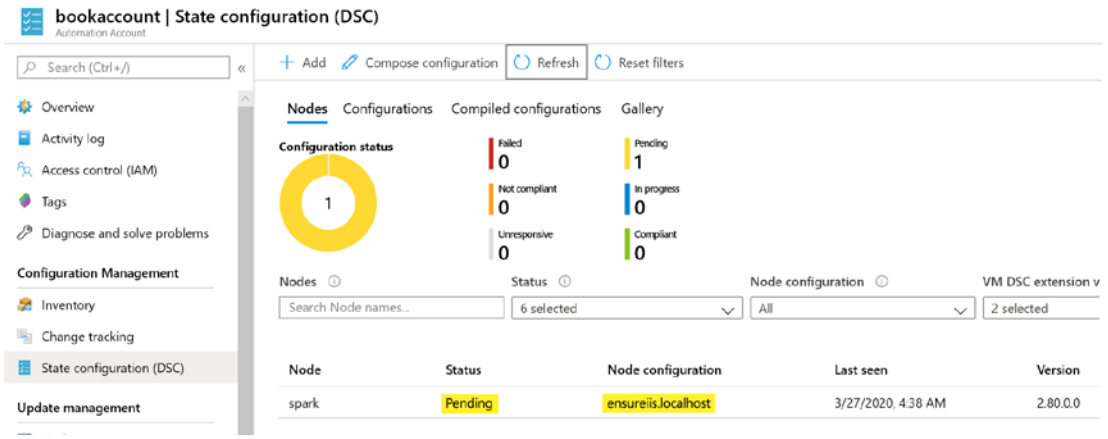


Figure 4.33 : vérification de l'état du nœud

Après quelques minutes, la configuration est appliquée au nœud, le nœud devient **conforme** et présente l'état **Completed** (Terminé) :

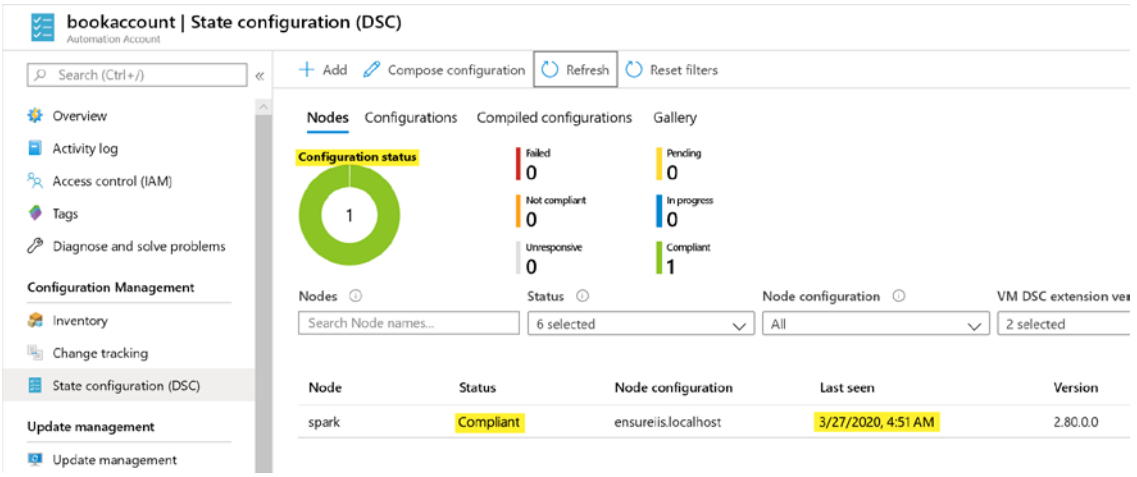


Figure 4.34 : vérification de la conformité du nœud

Plus tard, la connexion au serveur et la vérification de l'installation du serveur Web (IIS) confirment son installation, comme illustré à la *Figure* 4.35 :

```
PS C:\Users\citynextadmin> Get-WindowsFeature -Name web-server

Display Name                    Name                        Install State
------------                    ----                        -------------
[X] Web Server (IIS)            Web-Server                      Installed
```

Figure 4.35 : vérification si l'état souhaité a été atteint

La section suivante porte sur la tarification d'Azure Automation.

Tarification d'Azure Automation

Azure Automation n'engendre aucun coût si aucune procédure opérationnelle n'est exécutée sur celui-ci. Le coût d'Azure Automation est facturé par minute pour l'exécution des tâches de procédure opérationnelle. Si le nombre total de minutes d'exécution de procédure opérationnelle s'élève à 10 000, le coût d'Azure Automation sera de 0,002 USD par minute multiplié par 9 500, car les 500 premières minutes sont gratuites.

D'autres coûts associés à Azure Automation sont appliqués selon les fonctions consommées. Par exemple, un serveur pull DSC ne coûte rien au sein d'Azure Automation, tout comme l'intégration des machines virtuelles Azure sur le serveur pull. Toutefois, si des serveurs n'appartenant pas à Azure sont intégrés, généralement à partir d'autres Clouds ou sur site, les cinq premiers serveurs sont gratuits, mais toutes les fonctions supplémentaires coûtent 6 USD par serveur et par mois dans la région ouest des États-Unis.

La tarification peut varier d'une région à l'autre. Il est judicieux de vérifier la tarification sur la page de tarification officielle : https://azure.microsoft.com/pricing/details/automation.

Vous vous demandez peut-être pourquoi créer un compte Automation alors que nous pouvons déployer des applications sans serveur via Azure Functions. Dans la section suivante, nous découvrirons les principales différences entre Azure Automation et l'automatisation sans serveur.

Comparaison avec l'automatisation sans serveur

Azure Automation et les technologies sans serveur Azure, en particulier Azure Functions, sont très similaires et présentent les mêmes fonctionnalités. Il s'agit toutefois de services distincts présentant des capacités et des tarifs différents.

Il est important de comprendre qu'Azure Automation est une suite complète dédiée à l'automatisation des processus et à la gestion de la configuration, tandis qu'Azure Functions est conçu pour la mise en œuvre de fonctionnalités métier.

Azure Automation est utilisé pour automatiser les processus de mise en service, de désinstallation, de gestion et d'exploitation de l'infrastructure et de la gestion de la configuration par la suite. D'autre part, Azure Functions est conçu pour la création de services, en mettant en œuvre des fonctionnalités qui peuvent faire partie de microservices et d'autres API.

Azure Automation n'est pas conçu pour une mise à l'échelle illimitée, et la charge doit être modérée, tandis qu'Azure Functions peut gérer le trafic illimité et la mise à l'échelle automatique.

De nombreuses ressources partagées, telles que les connexions, les variables et les modules, peuvent être réutilisées dans les procédures opérationnelles au sein d'Azure Automation. Toutefois, il n'existe aucun concept partagé prêt à l'emploi dans Azure Functions.

Azure Automation peut gérer l'état intermédiaire grâce à des points de contrôle et continuer à partir du dernier état enregistré, tandis que les fonctions Azure sont généralement sans état et ne maintiennent aucun état.

Résumé

Azure Automation est un service Azure important. Il s'agit du seul service dédié à l'automatisation des processus et à la gestion de la configuration. Ce chapitre aborde de nombreux concepts importants concernant Azure Automation et l'automatisation des processus Azure, notamment les ressources partagées telles que la connexion, les certificats et les modules.

Nous avons découvert comment créer des procédures opérationnelles, y compris l'appel de procédures opérationnelles de différentes manières, telles que les relations parent-enfant, les webhooks et l'utilisation du portail. Ce chapitre abordait également l'architecture et le cycle de vie des procédures opérationnelles.

Nous avons également examiné l'utilisation des travailleurs hybrides et, à la fin du chapitre, nous avons exploré la gestion de la configuration à l'aide d'un serveur pull DSC et d'un gestionnaire de configuration locale. Enfin, nous avons comparé d'autres technologies, telles qu'Azure Functions.

Dans le chapitre suivant, nous allons explorer la conception des stratégies, des verrous et des balises pour les déploiements Azure.

5

Conception des stratégies, des verrous et des balises pour les déploiements Azure

Azure est une plateforme Cloud polyvalente. Les clients peuvent non seulement créer et déployer leur application, mais également gérer et régir activement leurs environnements. Les Clouds font généralement l'objet d'un modèle de paiement à l'utilisation, dans le cadre duquel un client souscrit et peut déployer quasiment toutes sortes d'éléments dans le Cloud. Il peut s'agir d'une seule machine virtuelle basique ou de milliers d'entre elles, avec des **références** (**SKU**) plus élevées. Azure n'empêchera aucun client de provisionner les ressources qu'il souhaite approvisionner. Un grand nombre d'individus peut avoir accès à l'abonnement Azure au sein d'une organisation. Il est nécessaire de mettre en place un modèle de gouvernance, de sorte que seules les ressources soient mises en service par des individus autorisés à les créer. Azure fournit des fonctions de gestion des ressources, telles que le **contrôle RBAC** (**Role based Access Control**), la stratégie Azure, les groupes de gestion, les plans, et les verrous de ressources destinés à gérer et à assurer la gouvernance de ces ressources.

La gestion des coûts, de l'usage et des informations représente un aspect essentiel de la gouvernance. L'équipe de direction d'une organisation exige d'être tenue informée en tout temps de la consommation et du coût du Cloud. Elle souhaite connaître quel est le pourcentage des coûts totaux attribuable à chaque équipe, département et service. En bref, elle souhaite disposer de rapports tenant compte des différents aspects de la consommation et des coûts. Azure fournit une fonction de marquage qui permet de fournir ce genre d'informations à la volée.

Dans ce chapitre, nous allons aborder les thèmes suivants :

- Groupes de gestion Azure
- Balises Azure
- Stratégie Azure
- Verrous Azure
- Azure RBAC
- Plans Azure
- Implémentation des fonctions de gouvernance Azure

Groupes de gestion Azure

Commençons par découvrir les groupes de gestion Azure car ceux-ci seront mentionnés dans la plupart des sections à venir. Les groupes de gestion désignent un niveau de portée vous permettant d'attribuer ou de gérer efficacement les rôles et les stratégies. Les groupes de gestion sont très utiles si vous disposez de plusieurs abonnements.

Les groupes de gestion agissent comme un espace réservé dédié à l'organisation des abonnements. Les groupes de gestion peuvent également être imbriqués. Si vous appliquez une stratégie ou un accès au niveau du groupe de gestion, ceux-ci seront hérités par les groupes de gestion et les abonnements sous-jacents. Au niveau de l'abonnement, cette stratégie ou cet accès sera hérité(e) par les groupes de ressources, puis enfin par les ressources.

La hiérarchie des groupes de gestion est illustrée ici :

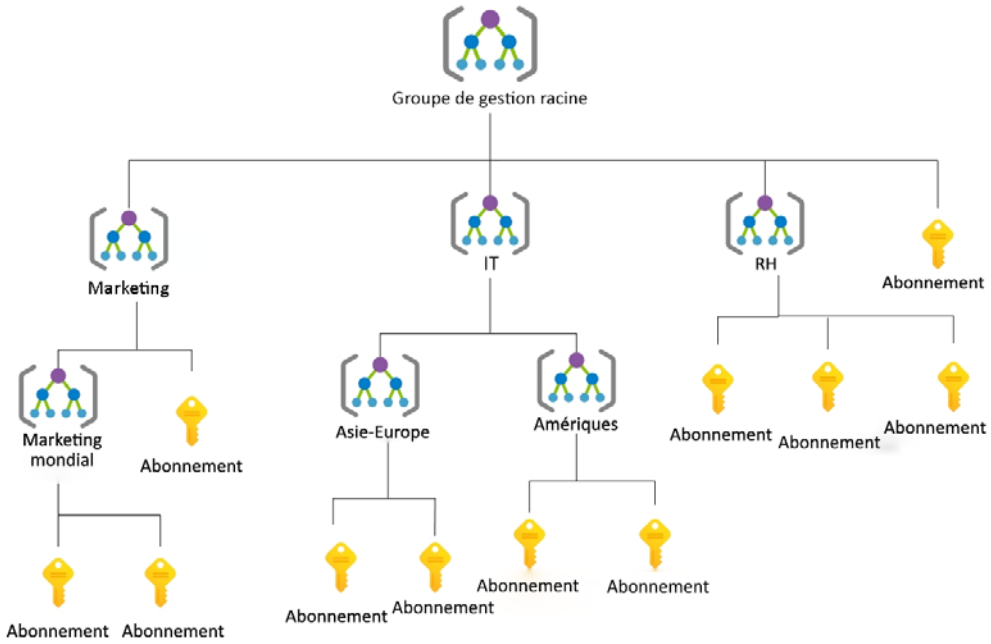


Figure 5.1 : hiérarchie des groupes de gestion Azure

La Figure 5.1 indique que nous utilisons des groupes de gestion pour séparer les opérations des différents services, tels que le marketing, l'IT et les ressources humaines. Des groupes de gestion et des abonnements imbriqués se trouvent au sein de chacun de ces services, afin de faciliter l'organisation des ressources selon une hiérarchie définie pour la gestion des stratégies et des accès. Nous étudierons par la suite l'utilisation des groupes de gestion en tant que portée pour la gouvernance, la gestion des stratégies et la gestion des accès.

La section suivante porte sur les balises Azure, qui jouent un autre rôle essentiel dans le regroupement logique des ressources.

Balises Azure

Azure permet de marquer des groupes de ressources et ressources avec des paires nom-valeur. Ce marquage facilite une organisation logique et la classification des ressources. Azure permet également de marquer 50 paires nom-valeur pour les groupes de ressources et leurs ressources. Bien que le groupe de ressources soit un conteneur ou un espace réservé pour les ressources, son marquage ne revient pas à marquer également les ressources en son sein. Les groupes de ressources et les ressources doivent être marqués selon leur utilisation, ce qui est expliqué plus loin dans cette section. Les balises sont liées à un abonnement, un groupe de ressources ou une ressource. Azure accepte n'importe quelle paire nom-valeur, et il est donc important qu'une organisation définisse les noms et leurs valeurs possibles.

Mais pourquoi le marquage est-il important ? En d'autres termes, quels problèmes peuvent être résolus grâce au marquage ? Le marquage comporte les avantages suivants :

- **Catégorisation des ressources** : un abonnement Azure est utilisé par plusieurs départements et rôles au sein d'une organisation. Il est essentiel que la direction identifie les propriétaires de ces ressources. Le marquage permet d'attribuer des identifiants à ces ressources, qui peuvent représenter des départements ou des rôles.
- **Gestion de l'information pour les ressources Azure** : les ressources Azure d'un abonnement peuvent être mises en service par quiconque y a accès. Les organisations souhaitent classer les ressources de manière appropriée, conformément à des stratégies de gestion des informations. Ces stratégies peuvent reposer sur la gestion du cycle de vie des applications, par exemple la gestion des environnements de développement, de test et de production. Ces politiques peuvent également se fonder sur l'utilisation, ou sur d'autres priorités. Chaque organisation définit d'une manière qui lui est propre ses catégories d'information et Azure utilise pour cela les balises.
- **Gestion des coûts** : le marquage dans Azure permet d'identifier les ressources selon leur catégorisation. Les requêtes peuvent être exécutées sur Azure pour identifier le coût par catégorie, par exemple. Par exemple, le coût des ressources dans Azure pour la création d'un environnement destiné aux services des finances et marketing peut être aisément établi. En outre, Azure fournit également des informations sur la facturation, sur la base de ces balises. Cela aide à déterminer les taux de consommation des équipes, départements ou groupes.

Les balises sont assorties de certaines limites dans Azure, toutefois :

- Azure autorise 50 paires nom-valeur maximum à associer à des groupes de ressources.
- Les balises ne peuvent être héritées. Les balises appliquées à un groupe de ressources ne s'appliquent pas aux ressources individuelles contenues dans ce dernier. Toutefois, il est très facile d'oublier de marquer des ressources durant leur mise en service. La stratégie Azure fournit le mécanisme garantissant que les balises sont étiquetées avec la valeur appropriée durant la mise en service. Nous examinerons ces stratégies en détail plus loin dans ce chapitre.

Les balises peuvent être affectées aux ressources et aux groupes de ressources à l'aide de PowerShell, d'Azure CLI 2.0, de modèles Azure Resource Manager, du portail Azure et des API REST d'Azure Resource Manager.

Un exemple de catégorisation en vue de la gestion des informations à l'aide des balises Azure est illustré ci-après :

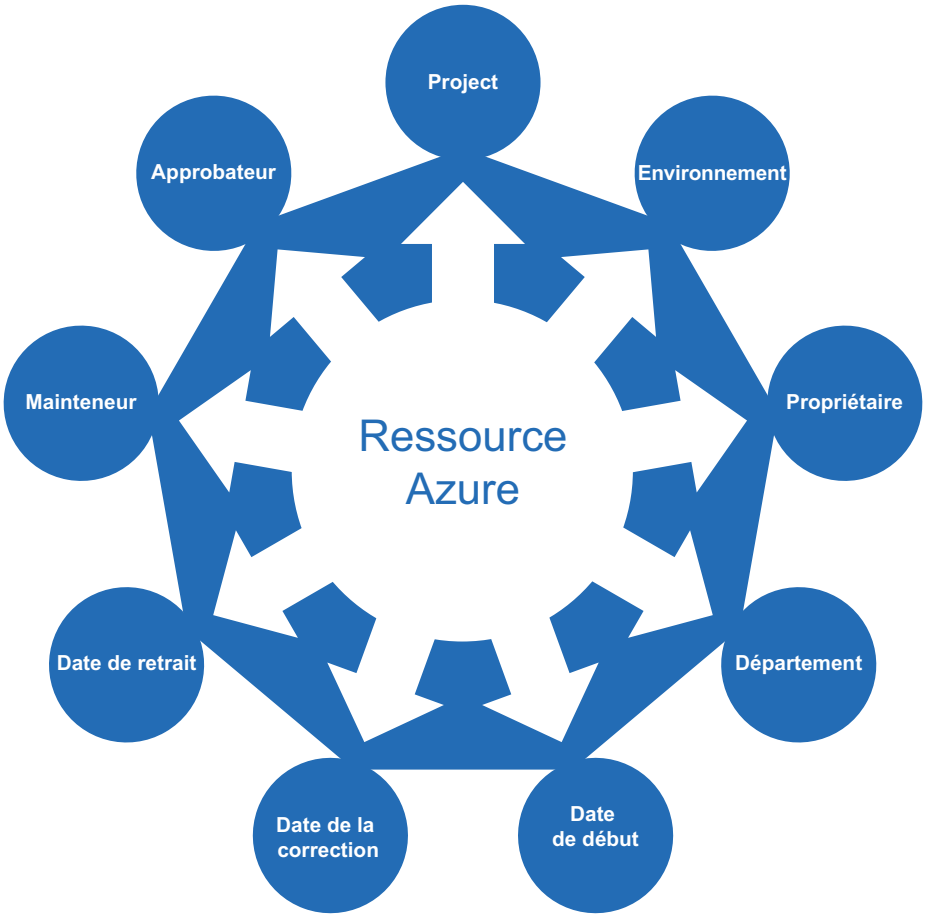


Figure 5.2 : catégorisation de la gestion des informations à l'aide de balises Azure

Dans cet exemple, les paires nom-valeur **Department**, **Project**, **Environment**, **Owner**, **Approver**, **Maintainer**, **Start Date**, **Retire Date** et **Patched Date** sont utilisées pour marquer les ressources. Il est extrêmement simple de trouver toutes les ressources associées à une balise particulière ou à une combinaison de balises à l'aide de PowerShell, d'Azure CLI ou de l'API REST. La section suivante porte sur l'utilisation de PowerShell pour attribuer des balises aux ressources.

Balises avec PowerShell

Les balises peuvent être gérées à l'aide de PowerShell, de modèles Azure Resource Manager, du portail Azure et des API REST. Dans cette section, PowerShell servira à créer et à appliquer des balises. PowerShell fournit une cmdlet de commande pour la récupération et l'étiquetage des ressources et des groupes de ressources :

- Pour récupérer les balises associées à une ressource à l'aide de PowerShell, la cmdlet de commande **`Get-AzResource`** peut être utilisée :

```
(Get-AzResource -Tag @{ "Environment"="Production"}).Name
```

- Pour récupérer les balises associées à un groupe de ressources à l'aide de PowerShell, la commande suivante peut être utilisée :

```
Get-AzResourceGroup -Tag @{"Environment"="Production"}
```

- Pour définir des balises sur un groupe de ressources, la cmdlet de commande **`Update-AzTag`** peut être utilisée :

```
$tags = @{"Dept"="IT"; "Environment"="Production"}
$resourceGroup = Get-AzResourceGroup -Name demoGroup
New-AzTag -ResourceId $resourceGroup.ResourceId -Tag $tags
```

- Pour définir des balises sur une ressource, la même cmdlet de commande **`Update-AzTag`** peut être utilisée :

```
$tags = @{"Dept"="Finance"; "Status"="Normal"}
$resource = Get-AzResource -Name demoStorage -ResourceGroup demoGroup
New-AzTag -ResourceId $resource.id -Tag $tags
```

- Vous pouvez mettre à jour les balises existantes à l'aide de la commande **`Update-AzTag`**. Toutefois, vous devez spécifier l'opération en tant que **`Merge`** ou **`Replace`**. L'option **`Merge`** ajoutera les nouvelles balises que vous transmettez aux balises existantes, tandis que l'opération **`Replace`** remplacera toutes les anciennes balises par les nouvelles. Voici un exemple de mise à jour de balises dans un groupe de ressources sans remplacer les balises existantes :

```
$tags = @{"Dept"="IT"; "Environment"="Production"}
$resourceGroup = Get-AzResourceGroup -Name demoGroup
Update-AzTag -ResourceId $resourceGroup.ResourceId -Tag $tags -Operation Merge
```

Intéressons-nous désormais aux balises avec modèles Azure Resource Manager.

Balises avec modèles Azure Resource Manager

Le modèle de gestionnaire des ressources facilite également la définition de balises pour chaque ressource. Celui-ci peut être utilisé pour assigner plusieurs balises à chaque ressource, comme suit :

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/
deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": [
    {
      "apiVersion": "2019-06-01",
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[concat('storage', uniqueString(resourceGroup().id))]",
      "location": "[resourceGroup().location]",
      "tags": {
        "Dept": "Finance",
        "Environment": "Production"
      },
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": { }
    }
    ]
}
```

Dans l'exemple précédent, deux balises, **`Dept`** et **`Environment`**, sont ajoutées à une ressource du compte de stockage à l'aide des modèles Azure Resource Manager.

Marquage des groupes de ressources et des ressources

Il est absolument essentiel que les architectes décident de l'architecture de taxonomie et d'information pour les ressources et les groupes de ressources Azure. Ils doivent définir les catégories selon lesquelles les ressources seront classées, en fonction des exigences des requêtes. Toutefois, ils doivent également déterminer si des balises doivent être associées aux ressources individuelles ou aux groupes de ressources.

Si toutes les ressources au sein d'un groupe de ressources ont besoin de la même balise, il est alors préférable de marquer le groupe de ressources, même si les balises n'héritent pas des ressources dans le groupe de ressources. Si votre organisation souhaite que les balises soient transmises à toutes les ressources sous-jacentes, vous pouvez écrire un script PowerShell pour récupérer les balises du groupe de ressources et les mettre à jour pour les ressources du groupe de ressources. Il est important de prendre en considération les requêtes pour déterminer si les balises doivent être appliquées au niveau de la ressource ou du groupe de ressource. Si les requêtes se rapportent à des types de ressources individuels sur un abonnement et entre des groupes de ressources, l'assignation des balises à des ressources individuelles est plus logique. Toutefois, si l'identification des groupes de ressources est suffisante pour que vos requêtes soient efficaces, les balises doivent être appliquées uniquement aux groupes de ressources. Si vous déplacez des ressources entre des groupes de ressources, les balises appliquées au niveau du groupe de ressources seront perdues. Si vous déplacez des ressources, pensez à ajouter des balises à nouveau.

Stratégie Azure

La section précédente a abordé l'application de balises en vue de déploiements Azure. Les balises sont excellentes pour l'organisation des ressources ; cependant nous n'avons pas encore abordé le point suivant : comment les organisations veillent-elles à ce que les balises soient appliquées pour chaque déploiement ? Il est nécessaire d'assurer la mise en application automatisée des balises Azure sur les ressources et les groupes de ressources. Aucune vérification dans Azure ne permet de s'assurer que les balises appropriées sont appliquées aux ressources et aux groupes de ressources. Ceci n'est pas seulement spécifique aux balises : cela s'applique à la configuration de n'importe quelle ressource sur Azure. Par exemple, vous voudrez peut-être restreindre l'endroit où vos ressources peuvent être provisionnées géographiquement (uniquement dans la région Est des États-Unis, par exemple).

Vous avez peut-être déjà compris que cette section s'attache à la formulation d'un modèle de gouvernance sur Azure. La gouvernance est un élément essentiel dans Azure, car elle garantit que tous les individus ayant accès à l'environnement Azure ont connaissance des priorités et des processus de l'organisation. Elle permet également de maîtriser les coûts. Elle permet de définir des conventions d'organisation facilitant la gestion des ressources.

Chaque stratégie peut être établie à l'aide de diverses règles et plusieurs stratégies peuvent être appliquées aux abonnements et aux groupes de ressources. Si les règles sont satisfaites, les stratégies peuvent exécuter diverses actions. Cette action peut consister à refuser la transaction en cours ou à l'auditer, c'est-à-dire à la consigner dans des fichiers journaux et à autoriser son exécution, et à ajouter des métadonnées à la transaction si cela est nécessaire.

Une convention de nommage des ressources, le marquage des ressources, les types de ressources à mettre en service, l'emplacement des ressources, ou toute combinaison de ces règles constituent tout autant d'exemples de stratégies.

Azure fournit de nombreuses stratégies prédéfinies, tout en vous permettant de créer des stratégies personnalisées. Un langage de stratégie basé sur JSON peut être utilisé pour définir des stratégies personnalisées.

Maintenant que vous connaissez le but et le cas d'utilisation de la stratégie Azure, intéressons-nous désormais aux stratégies intégrées, au langage de stratégie et aux stratégies personnalisées.

Stratégies intégrées

Azure fournit un service permettant de créer des stratégies personnalisées, cependant, des stratégies préconfigurées sont également disponibles, lesquelles sont fréquemment utilisées, notamment à des fins de gouvernance. Ces stratégies ont trait à l'emplacement, aux types de ressources et aux balises autorisées. Davantage d'informations au sujet de ces stratégies intégrées sont disponibles à l'adresse https://docs.microsoft.com/azure/azure-resource-manager/resource-manager-policy.

Langage de la stratégie

Les stratégies Azure utilisent un langage JSON pour définir et décrire les stratégies. L'adoption d'une stratégie comporte deux étapes. Celle-ci doit être définie, puis être appliquée et attribuée. Les stratégies ont une certaine portée et peuvent être appliquées au niveau du groupe de gestion, de l'abonnement ou du groupe de ressources.

Les stratégies sont définies à l'aide de blocs **`si... alors`**, de même que dans de nombreux langages de programmation fréquemment utilisés. Le bloc **`si`** est exécuté afin d'évaluer les conditions et selon le résultat de ces mêmes conditions, le bloc **`alors`** est exécuté :

```
{
  "if": {
    <condition> | <logical operator>
  },
  "then": {
    "effect": "deny | audit | append"
  }
}
```

Les stratégies fournissent non seulement des conditions **`si`** simples, mais il est également possible d'assembler plusieurs conditions **`si`** de manière logique afin de créer des règles complexes. Ces conditions peuvent être assemblées à l'aide d'opérateurs **`ET`**, **`OU`** et **`NON`** :

- La syntaxe **`ET`** exige que toutes les conditions soient vraies.
- La syntaxe **`OU`** requiert que l'une des conditions soit vraie.
- La syntaxe **`NON`** inverse le résultat de la condition.

La syntaxe **ET** est illustrée ci-après. Elle est représentée par le mot clé **allOf** :

```
"if": {
  "allOf": [
    {
       "field": "tags",
        "containsKey": "application"
    },
    {
      "field": "type",
      "equals": "Microsoft.Storage/storageAccounts"
    }
  ]
},
```

La syntaxe **OU** est illustrée ci-après. Elle est représentée par le mot clé **anyOf** :

```
"if": {
  "anyOf": [
    {
       "field": "tags",
        "containsKey": "application"
    },
    {
      "field": "type",
      "equals": "Microsoft.Storage/storageAccounts"
    }
  ]
},
```

La syntaxe **NON** est illustrée ci-après. Elle est représentée par le mot clé **not** :

```
"if": {
  "not": [
    {
       "field": "tags",
```

```
          "containsKey": "application"
      },
      {
        "field": "type",
        "equals": "Microsoft.Storage/storageAccounts"
      }
    ]
  },
```

En effet, ces opérateurs logiques peuvent être combinés comme suit :

```
  "if": {
    "allOf": [
      {
        "not": {
          "field": "tags",
          "containsKey": "application"
        }
      },
      {
        "field": "type",
        "equals": "Microsoft.Storage/storageAccounts"
      }
    ]
  },
```

Ceci est très similaire aux conditions **`Si`** dans les langages de programmation populaires tels que C# et Node.js :

```
  If ("type" == "Microsoft.Storage/storageAccounts") {
        Deny
  }
```

Il convient de noter qu'il n'existe aucune action **`Autoriser`** mais qu'il y a bien une action **`Refuser`**. Cela signifie que les règles de stratégies devraient être écrites en gardant à l'esprit la possibilité d'un déni. Les règles doivent évaluer les conditions et **`refuser`** l'action si les conditions sont satisfaites.

Champs autorisés

Les champs qui sont autorisés dans les conditions tout en définissant les stratégies sont les suivants :

- **`Nom`** : nom de la ressource qui fait l'objet de la stratégie. Ce champ est très spécifique et applicable à une ressource par son utilisation.
- **`Type`** : le type de ressource, tel que **`Microsoft. Compute/VirtualMachines`**. Ce champ appliquerait la stratégie à toutes les instances de machines virtuelles, par exemple.
- **`Emplacement`** : l'emplacement (autrement dit, la région Azure) d'une ressource.
- **`Balises`** : les balises associées à une ressource.
- **`Alias de propriété`** : propriétés spécifiques à une ressource. Ces propriétés sont différentes selon les ressources.

La section suivante porte sur la protection des ressources dans les environnements de production.

Verrous Azure

Les verrous sont un mécanisme permettant d'interrompre certaines activités exécutées sur les ressources. Le contrôle RBAC confère des droits aux utilisateurs/groupes/application, dans une certaine mesure. Il existe des rôles RBAC prédéfinis, comme les rôles propriétaire, contributeur et lecteur. Avec le rôle de contributeur, il est possible de supprimer ou de modifier une ressource. Comment empêcher de telles activités même si l'utilisateur a un rôle de contributeur ? C'est alors que les verrous Azure s'avèrent utiles.

Les verrous Azure ont deux utilités :

- Ils peuvent verrouiller les ressources de sorte qu'elles ne puissent être supprimées, même avec un accès propriétaire.
- Ils peuvent verrouiller la ressource de telle sorte qu'elle ne puisse être supprimée, et que sa configuration ne soit pas modifiable.

Cela est en général très utile pour les ressources d'un environnement de production qui ne doivent pas être modifiées ou supprimées de manière accidentelle.

Les verrous peuvent être appliqués aux niveaux de l'abonnement, du groupe de ressources, du groupe de gestion et des ressources individuelles. Les verrous sont héritées entre abonnements, groupes de ressources et ressources. En appliquant un verrou au niveau parent, celui-ci sera hérité par les ressources au niveau enfant. Les ressources ajoutées ultérieurement dans la sous-portée héritent également de la configuration de verrou par défaut. L'application d'un verrou au niveau des ressources empêchera la suppression du groupe de ressources contenant la ressource.

Les verrous sont appliqués uniquement aux opérations qui permettent de gérer une ressource et non pas aux opérations internes à celle-ci. Les utilisateurs auront besoin d'autorisations RBAC **`Microsoft.Authorization/*`** ou **`Microsoft.Authorization/locks/*`** pour créer et modifier des verrous.

Les verrous peuvent être créés et appliqués à l'aide du portail Azure, d'Azure PowerShell, d'Azure CLI, des modèles Azure Resource Manager et de l'API REST.

La création d'un verrou à l'aide d'un modèle Azure Resource Manager est effectuée comme suit :

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "lockedResource": {
      "type": "string"
    }
  },
  "resources": [
    {
      "name": "[concat(parameters('lockedResource'), '/Microsoft.
Authorization/myLock')]",
      "type": "Microsoft.Storage/storageAccounts/providers/locks",
      "apiVersion": "2019-06-01",
      "properties": {
        "level": "CannotDelete"
      }
    }
  ]
}
```

La section **`ressources`** du code de modèle Azure Resource Manager se compose d'une liste de toutes les ressources à provisionner ou à mettre à jour dans Azure. Il existe une ressource de compte de stockage, et le compte de stockage présente une ressource de verrou. Le nom du verrou est fourni à l'aide de la concaténation de chaîne dynamique et le verrou appliqué est du type **`CannotDelete`**, ce qui signifie que le compte de stockage est verrouillé pour la suppression. Le compte de stockage ne peut être supprimé qu'après la suppression du verrou.

La création et l'application d'un verrou sur une ressource à l'aide de PowerShell se présentent comme suit :

```
New-AzResourceLock -LockLevel CanNotDelete -LockName LockSite '
  -ResourceName examplesite -ResourceType Microsoft.Web/sites '
  -ResourceGroupName exampleresourcegroup
```

La création et l'application d'un verrou sur un groupe de ressources à l'aide de PowerShell se présentent comme suit :

```
New-AzResourceLock -LockName LockGroup -LockLevel CanNotDelete '
  -ResourceGroupName exampleresourcegroup
```

La création et l'application d'un verrou sur une ressource à l'aide d'Azure CLI se présentent comme suit :

```
az lock create --name LockSite --lock-type CanNotDelete \
  --resource-group exampleresourcegroup --resource-name examplesite \
  --resource-type Microsoft.Web/sites
```

La création et l'application d'un verrou sur un groupe de ressources à l'aide d'Azure CLI se présentent comme suit :

```
az lock create --name LockGroup --lock-type CanNotDelete \ --resource-group
exampleresourcegroup
```

Pour créer ou supprimer des verrous de ressources, l'utilisateur doit avoir accès aux actions **`Microsoft.Authorization/*`** ou **`Microsoft.Authorization/locks/*`**. Vous pouvez également attribuer des autorisations granulaires. Les propriétaires et les administrateurs d'accès utilisateur auront accès à la création ou à la suppression des verrous par défaut.

Si vous vous demandez en quoi consistent les mots-clés **`Microsoft.Authorization/*`** et **`Microsoft.Authorization/locks/*`**, vous en saurez plus à ce sujet dans la section suivante.

Intéressons-nous désormais à Azure RBAC.

Azure RBAC

Azure fournit une authentification à l'aide d'Azure Active Directory pour ses ressources. Une fois qu'une identité a été authentifiée, il convient de décider des ressources que l'identité sera autorisée à accéder. C'est ce que l'on appelle une autorisation. L'autorisation évalue les autorisations qui ont été accordées à une identité. Toute personne ayant accès à un abonnement Azure doit disposer de suffisamment d'autorisations pour effectuer son travail uniquement.

L'autorisation est aussi connue sous le nom RBAC. Le contrôle RBAC dans Azure se réfère à l'attribution d'autorisations aux identités selon une certaine portée. Cette portée peut être un groupe de gestion, un abonnement, un groupe de ressources ou des ressources individuelles.

Le contrôle RBAC contribue à la création et à l'attribution de différentes autorisations, pour différentes identités. Cela permet de séparer les fonctions au sein des équipes au lieu d'attribuer toutes les autorisations à tous ses membres. Le RBAC responsabilise les individus vis-à-vis de leurs tâches uniquement, car les autres n'y auront pas accès et ne pourront donc pas les effectuer à leur place. Il convient de noter que lorsqu'un accès à plus vaste portée est accordé automatiquement, les ressources enfant héritent de ces autorisations. Par exemple, la fourniture d'une identité avec un accès en lecture pour un groupe de ressources signifie que l'identité aura également un accès en lecture à toutes les ressources de ce groupe.

Azure fournit trois rôles intégrés et généraux. Les voici :

- Le rôle de propriétaire, qui dispose d'un accès complet à toutes les ressources
- Le rôle de contributeur, qui a accès à des ressources en lecture/écriture
- Le rôle de lecteur qui a simplement un accès en lecture aux ressources

Azure comporte plus de rôles, mais ils sont spécifiques aux ressources, tels que les rôles contributeur réseau et gestionnaire de sécurité.

Pour afficher tous les rôles fournis par Azure pour toutes les ressources, exécutez la commande **`Get-AzRoleDefinition`** dans la console PowerShell.

Chaque définition de rôle s'assortit d'actions autorisées et non autorisées. Par exemple, dans le rôle de propriétaire, toutes les actions sont autorisées et aucune d'elles n'est interdite :

```
PS C:\Users\riskaria> Get-AzRoleDefinition -Name "Owner"

Nom              : Propriétaire
Id               : 8e3af657-a8ff-443c-a75c-2fe8c4bcb635
IsCustom         : Faux
Description      : vous permet de tout gérer, y compris l'accès aux
ressources.
Actions          : {*}
NotActions       : {}
DataActions      : {}
NotDataActions   : {}
AssignableScopes : {/}
```

Chaque rôle est composé de plusieurs autorisations. Chaque ressource fournit une liste d'opérations. Les opérations prises en charge par une ressource peuvent être obtenues en utilisant la cmdlet de commande **`Get-AzProviderOperation`**. Cette cmdlet de commande utilise le nom du fournisseur et de la ressource afin de récupérer les opérations :

```
PS C:\Users\riskaria> Get-AzProviderOperation -OperationSearchString
"Microsoft.Insights/*" | select Operation
```

Cela donnera le résultat suivant :

```
PS C:\Users\riskaria> Get-AzProviderOperation -OperationSearchString
"Microsoft.Insights/*" | select Operation


Operation
---------
Microsoft.Insights/Metrics/Action
Microsoft.Insights/Register/Action
Microsoft.Insights/Unregister/Action
Microsoft.Insights/ListMigrationDate/Action
Microsoft.Insights/MigrateToNewpricingModel/Action
Microsoft.Insights/RollbackToLegacyPricingModel/Action


.
.
.
.
.
.
.
.
Microsoft.Insights/PrivateLinkScopes/PrivateEndpointConnectionProxies/Read
Microsoft.Insights/PrivateLinkScopes/PrivateEndpointConnectionProxies/Write
Microsoft.Insights/PrivateLinkScopes/PrivateEndpointConnectionProxies/Delete
Microsoft.Insights/PrivateLinkScopeOperationStatuses/Read
Microsoft.Insights/DiagnosticSettingsCategories/Read
```

La sortie affichée ici fournit toutes les actions disponibles au sein du fournisseur de ressources **`Microsoft.Insights`** sur les ressources associées. Les ressources incluent des **`métriques`**, des **`registres`** et d'autres éléments, tandis que les actions comprennent la **`lecture`**, l'**`écriture`** et d'autres éléments.

Examinons désormais les rôles personnalisés.

Rôles personnalisés

Azure fournit de nombreux rôles génériques prêts à l'emploi, tels que le propriétaire, le contributeur et le lecteur, ainsi que des rôles spécialisés spécifiques aux ressources, comme le contributeur de machine virtuelle. Si le rôle de lecteur est assigné à un utilisateur/groupe ou à un principal de service, les autorisations de lecteur sont attribuées à la portée. La portée peut être une ressource, un groupe de ressources ou un abonnement. De même, un contributeur peut lire et modifier la portée affectée. Un contributeur de machine virtuelle peut modifier les paramètres de la machine virtuelle et pas les autres paramètres de ressources. Toutefois, dans certains scénarios, les rôles existants peuvent ne pas répondre à nos exigences. Si c'est le cas, Azure permet également la création de rôles personnalisés. Ils peuvent être attribués aux utilisateurs, aux groupes et aux principaux de service et sont applicables aux ressources, aux groupes de ressources, ainsi qu'aux abonnements.

Les rôles personnalisés sont créés en combinant plusieurs autorisations. Par exemple, un rôle personnalisé peut se composer d'opérations provenant de multiples ressources. Dans le bloc de code suivant, une nouvelle définition de rôle est en cours de création, mais au lieu de définir toutes les propriétés manuellement, l'un des rôles **« Virtual Machine Contributor »** existants est récupéré car il correspond presque à la configuration du nouveau rôle personnalisé. Évitez d'utiliser le même nom que ceux des rôles prédéfinis, car cela engendrerait un conflit. Ensuite, la propriété ID est annulée et un nouveau nom, ainsi qu'une nouvelle description sont fournis. Le code efface également toutes les actions, ajoute des actions, ajoute une nouvelle portée après avoir effacé la portée existante et crée enfin un rôle personnalisé :

```
$role = Get-AzRoleDefinition "Virtual Machine Contributor"
$role.Id = $null
$role.Name = "Virtual Machine Operator"
$role.Description = "Can monitor and restart virtual machines."
$role.Actions.Clear()
$role.Actions.Add("Microsoft.Storage/*/read")
$role.Actions.Add("Microsoft.Network/*/read")
$role.Actions.Add("Microsoft.Compute/*/read")
$role.Actions.Add("Microsoft.Compute/virtualMachines/start/action")
$role.Actions.Add("Microsoft.Compute/virtualMachines/restart/action")
$role.Actions.Add("Microsoft.Authorization/*/read")
$role.Actions.Add("Microsoft.Resources/subscriptions/resourceGroups/read")
$role.Actions.Add("Microsoft.Insights/alertRules/*")
$role.Actions.Add("Microsoft.Support/*")
$role.AssignableScopes.Clear()
$role.AssignableScopes.Add("/subscriptions/548f7d26-b5b1-468e-ad45-
```

```
6ee12accf7e7")

New-AzRoleDefinition -Role $role
```

Le portail Azure propose une fonction d'aperçu qui vous permet de créer des rôles RBAC personnalisés à partir du portail Azure lui-même. Vous avez la possibilité de créer des rôles à partir de zéro, de cloner un rôle existant ou de commencer à écrire le manifeste JSON. La *Figure* 5.3 montre le panneau **Create a custom role** (Créer un rôle personnalisé), qui est disponible dans la section **IAM > +Add** :

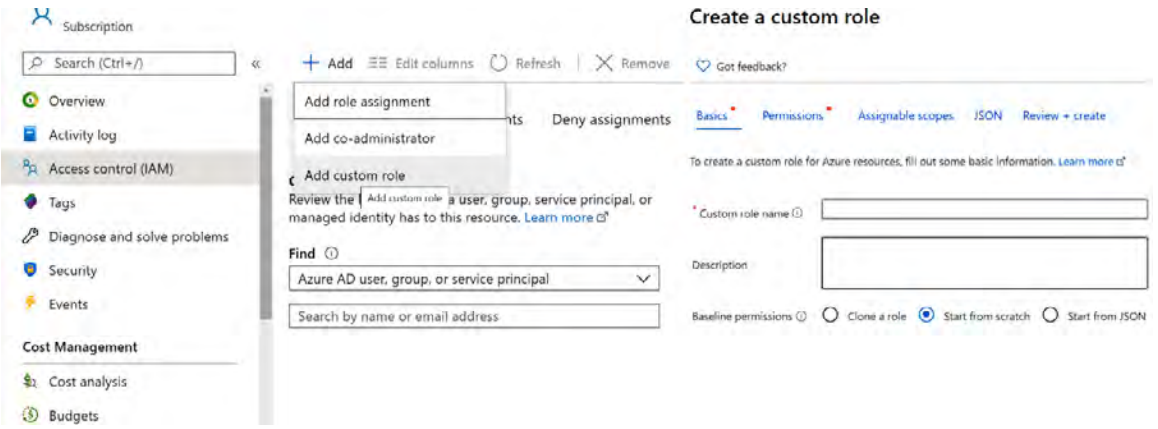


Figure 5.3 : création de rôles personnalisés dans le portail Azure

Le processus de création de rôle personnalisé peut ainsi être réalisé sans engagement.

En quoi les verrous diffèrent-ils de RBAC ?

Les verrous et le contrôle RBAC sont deux choses différentes. RBAC aide à autoriser ou à refuser des autorisations pour les ressources. Ces autorisations ont trait à l'exécution d'opérations telles que la lecture, l'écriture et la mise à jour des ressources. Les verrous, en revanche, permettent d'annuler des autorisations de configuration ou de suppression de la ressource.

Dans la section suivante, nous allons discuter des plans Azure, qui nous aident à réaliser l'orchestration d'artefacts, tels que les attributions de rôles, les affectations de stratégie et bien plus encore, dont nous avons discuté jusqu'à présent.

Plans Azure

Vous serez familiarisé avec le mot plan, qui fait référence au plan ou au dessin utilisé par un architecte pour concevoir une solution. De même, dans Azure, les architectes du Cloud utilisent les plans Azure pour définir un ensemble de ressources Azure reproductibles qui respectent les normes, les processus et les modèles d'application d'une organisation.

Les plans nous permettent d'orchestrer le déploiement de diverses ressources et d'autres artefacts, tels que :

- Attributions de rôles
- Affectation de stratégie
- Modèles Azure Resource Manager
- Groupes de ressources

Les objets de plan Azure sont répliqués dans plusieurs régions et pris en charge par Azure Cosmos DB. La réplication contribue à fournir un accès cohérent aux ressources et à maintenir les normes de l'entreprise, quelle que soit la région dans laquelle vous effectuez le déploiement.

Les plans Azure comprennent divers artefacts, et la liste des artefacts pris en charge est disponible ici :https://docs.microsoft.com/azure/governance/blueprints/overview#blueprint-definition.

Les plans peuvent être créés à l'aide du portail Azure, d'Azure PowerShell, d'Azure CLI, des API REST ou des modèles ARM.

La section suivante porte sur un exemple de mise en œuvre des fonctions de gouvernance Azure. Les services et fonctions tels que RBAC, la stratégie Azure Policy et les verrous de ressources Azure seront utilisés dans l'exemple.

Exemple de mise en œuvre des fonctions de gouvernance Azure

Dans cette section, nous allons examiner un exemple d'implémentation d'architecture pour une entreprise fictive qui souhaite mettre en œuvre des fonctions de gouvernance et de gestion des coûts Azure.

Contexte

Company Inc est une entreprise mondiale qui met en œuvre une solution de réseaux sociaux sur une plateforme IaaS Azure. Elle utilise des serveurs Web et des serveurs d'application déployés sur des machines virtuelles et des réseaux Azure. Le serveur SQL Azure sert de base de données back-end.

RBAC pour Company Inc

La première tâche consiste à veiller à ce que les équipes et les propriétaires d'application appropriés puissent accéder à leurs ressources. Il est reconnu que chaque équipe a des exigences différentes. Dans un souci de clarté, SQL Azure est déployé dans un groupe de ressources distinct des éléments Azure IaaS.

L'administrateur attribue les rôles suivants pour l'abonnement :

Rôle	Assigné à	Description
Propriétaire	Administrateur	Gère tous les groupes de ressources et l'abonnement
Responsable de la sécurité	Administrateur de sécurité	Ce rôle permet aux utilisateurs de consulter le centre Azure Security Center et le statut des ressources.
Contributeur	Gestion des infrastructures	Gestion des machines virtuelles et d'autres ressources.
Lecteur	Développeurs	Peut voir les ressources, mais ne peut pas les modifier. Les développeurs doivent travailler sur leurs environnements de développement/test.

Tableau 5.1 : différents rôles avec des détails d'accès

Stratégie Azure

L'entreprise doit implémenter la stratégie Azure pour garantir que ses utilisateurs provisionnent toujours les ressources en respectant les directives de l'entreprise.

Les stratégies Azure régissent divers aspects liés au déploiement des ressources. Les stratégies régiront également les mises à jour après le déploiement initial. Certaines des stratégies qui devraient être mises en œuvre sont fournies dans la section suivante.

Déploiements à certains emplacements

Les ressources Azure et les déploiements ne peuvent être exécutés que pour certains emplacements choisis. Il ne serait pas possible de déployer des ressources dans des régions en dehors de la stratégie. Par exemple, les régions autorisées sont l'Europe de l'Ouest et l'Est des États-Unis. Il ne devrait pas être possible de déployer des ressources dans une autre région.

Balises des ressources et des groupes de ressources

Chaque ressource dans Azure, y compris les groupes de ressources, aura obligatoirement des balises affectées. Les balises comprendront, au minimum, des détails sur le département, l'environnement, les données de création et le nom du projet.

Journaux de diagnostic et Application Insights pour toutes les ressources

Chaque ressource déployée sur Azure doit avoir des journaux de diagnostic et des journaux d'application activés dans la mesure du possible.

Verrous Azure

Une entreprise doit mettre en œuvre des verrous Azure pour éviter la suppression accidentelle des ressources cruciales. Chaque ressource cruciale pour le fonctionnement d'une solution doit être verrouillée. Cela signifie que même les administrateurs des services exécutés sur Azure n'auront pas la possibilité de supprimer ces ressources. Pour supprimer une ressource, il faut impérativement enlever le verrou.

Voici quelques points à prendre en compte :

Tous les environnements de production et de pré-production, à l'exception des environnements de développement et de test, seraient verrouillés pour suppression.

Tous les environnements de développement et de test qui ont des instances uniques seraient également verrouillés pour suppression.

Toutes les ressources liées à l'application Web seraient verrouillées pour la suppression de tous les environnements de production.

Toutes les ressources partagées seraient verrouillées pour suppression, quel que soit l'environnement.

Résumé

Dans ce chapitre, vous avez appris que la gouvernance et la gestion des coûts font partie des principales priorités des entreprises adoptant le Cloud. Un abonnement Azure assorti d'un modèle de paiement à l'utilisation peut nuire au budget de l'entreprise, car quiconque ayant accès à cet abonnement peut mettre en service autant de ressources qu'il le souhaite. Certaines ressources sont gratuites, mais d'autres sont très coûteuses.

Vous avez également appris qu'il est important que les organisations parviennent à maîtriser les coûts associés au Cloud. Les balises contribuent à générer des rapports de facturation. Ces rapports peuvent se baser sur les départements, les projets, les propriétaires, ou sur tout autre critère. Le coût est un facteur important, mais la gouvernance l'est tout autant. Azure fournit des verrous, des stratégies et un contrôle RBAC pour mettre en place la gouvernance appropriée. Les stratégies permettent de s'assurer que les opérations effectuées sur les ressources peuvent être rejetées ou auditées, les verrous veillent à ce que ces ressources ne puissent être modifiées ou supprimées et le contrôle RBAC permet de s'assurer que les collaborateurs disposent des autorisations optimales pour réaliser les tâches qui leur incombent. Ces fonctions permettent aux entreprises de mettre en place un système de gouvernance et de maîtrise des coûts sur leurs déploiements Azure.

Dans le prochain chapitre, nous évoquerons la gestion des coûts dans Azure. Nous étudierons différentes méthodes d'optimisation, la gestion des coûts et les API de facturation.

6

Gestion des coûts des solutions Azure

Dans le chapitre précédent, nous avons discuté des balises, des stratégies et des verrous et de la façon dont ils permettent de garantir la conformité. Les balises nous permettent d'ajouter des métadonnées à nos ressources et d'assurer la gestion logique des ressources. Dans le portail Azure, nous pouvons filtrer les ressources en fonction des balises. Si nous supposons qu'il existe un grand nombre de ressources, un cas de figure courant dans les entreprises, le filtrage nous permettra de faciliter la gestion de nos ressources. Un autre avantage des balises réside dans le fait qu'elles peuvent être utilisées pour filtrer nos rapports de facturation ou d'utilisation en termes de balises. Ce chapitre porte sur la gestion des coûts des solutions Azure.

Les économies en termes de coûts constituent la principale raison qui motive la transition vers le Cloud. Un abonnement Azure ne requiert aucun coût initial. Azure propose un abonnement basé sur la consommation. Azure mesure l'utilisation des ressources et fournit des factures mensuelles. La consommation d'Azure n'est pas plafonnée. Comme nous sommes sur un Cloud public, Azure (comme n'importe quel autre fournisseur de services) applique des limites strictes et souples sur le nombre de ressources qui peuvent être déployées. Les limites souples peuvent être augmentées en contactant le support Azure. Certaines ressources sous soumises à une limite stricte. Les limites de service sont indiquées à l'adresse https://docs.microsoft.com/azure/azure-resource-manager/management/azure-subscription-service-limits, et la limite par défaut varie en fonction du type d'abonnement.

Il est essentiel que les entreprises surveillent rigoureusement leur consommation et leur utilisation des services Azure. Bien qu'il soit possible de créer des stratégies afin de mettre en place des normes et conventions organisationnelles, il est également nécessaire de disposer de toute information relative à la facturation et à la consommation des données. En outre, les entreprises doivent adopter de bonnes pratiques de consommation des ressources Azure de sorte à maximiser les retours. Pour ce faire, les architectes doivent être familiarisés avec les ressources et les fonctionnalités Azure, les coûts qui leur sont associés et réaliser des analyses coûts-bénéfices des fonctions et des solutions.

Dans ce chapitre, nous allons aborder les thèmes suivants :

- Offre Azure en détail
- Facturation
- Facturation
- Utilisation et quotas
- Utilisation et API de facturation
- Calculatrice de tarifs Azure
- Bonnes pratiques en matière d'optimisation des coûts

Abordons de plus près chacun de ces points.

Offre Azure en détail

Azure propose différentes offres à l'achat. Jusqu'à présent, nous avons discuté des offres basées sur la consommation, mais il y en a d'autres, telles que les **Contrats d'entreprise** (**EA**), le sponsoring Azure et Azure dans le CSP. Nous traiterons chacun de ces points car ils sont très importants pour la facturation :

- **Paiement basé sur la consommation :** il s'agit d'une offre courante, où les clients paient en fonction de leur consommation. Les tarifs sont indiqués dans la documentation publique d'Azure. Les clients reçoivent une facture chaque mois pour l'utilisation de Microsoft et peuvent payer via une carte de crédit ou un mode de paiement de facture.
- **Contrats d'entreprise :** les contrats d'entreprise impliquent un engagement monétaire auprès de Microsoft. Les organisations signent un accord avec Microsoft et promettent d'utiliser x quantité de ressources Azure. Si l'utilisation dépasse la quantité convenue, le client recevra une facture pour la quantité excédentaire. Les clients peuvent créer plusieurs comptes dans un contrat d'entreprise et souscrire à plusieurs abonnements au sein de ces comptes. Il existe deux types de contrats d'entreprise : les contrats d'entreprise directs et indirects. Les clients ayant souscrit des contrats d'entreprise directs entretiennent une relation de facturation directe avec Microsoft, tandis que la facturation est gérée par un partenaire dans le cadre d'un contrat d'entreprise indirect. Les clients ayant souscrit des contrats d'entreprise obtiendront de meilleures offres et des remises grâce à leur engagement auprès de Microsoft. Le contrat d'entreprise est géré via un portail appelé le portail EA (https://ea.azure.com) et vous devez disposer de privilèges d'inscription pour accéder à ce portail.

- **Azure dans le CSP :** Azure dans le CSP est l'endroit où un client contacte un partenaire **fournisseur de solutions de cloud computing** (**CSP**), et ce partenaire pourvoit à un abonnement pour le client. La facturation sera entièrement gérée par le partenaire. Le client n'entretiendra pas de relation de facturation directe avec Microsoft. Microsoft facture le partenaire et le partenaire facture le client, en ajoutant sa marge.
- **Sponsoring Azure :** Microsoft parraine les start-ups, les ONG et d'autres organisations à but non lucratif qui utilisent Azure. Le parrainage est défini pour une durée déterminée et un montant fixe de crédit. Si la durée expire ou si le crédit est épuisé, l'abonnement sera converti en abonnement basé sur la consommation. Si les organisations souhaitent renouveler leur droit de parrainage, elles doivent travailler avec Microsoft.

Nous avons décrit quelques-unes des offres Azure. La liste complète est disponible à l'adresse https://azure.microsoft.com/support/legal/offer-details, qui inclut d'autres offres, notamment Azure pour les étudiants, Pass Azure et les abonnements développement/test.

Abordons à présent la facturation dans Azure.

Comprendre la facturation

Azure est un utilitaire de services qui propose les fonctions suivantes :

- Aucun coût initial
- Aucuns frais d'annulation
- Facturation par seconde, par minute ou par heure selon le type de ressource
- Paiement à l'utilisation

Dans de tels cas, il est difficile d'estimer les coûts initiaux relatifs à la consommation des ressources Azure. Chaque ressource dans Azure possède son propre modèle de coût et de facturation en fonction du stockage, de l'utilisation et de périodes de temps. Il est très important pour la direction et pour le service financier d'une entreprise d'établir un suivi de l'utilisation et des coûts. Azure fournit des options de création des rapports qui permettent à la direction et aux administrateurs de générer des rapports d'utilisation et de facturation selon plusieurs critères.

Le portail Azure fournit des informations détaillées sur la facturation et l'utilisation des services via la fonction **Gestion des coûts + facturation**, laquelle est accessible depuis le panneau de navigation principal, comme illustré à la *Figure 6.1* :

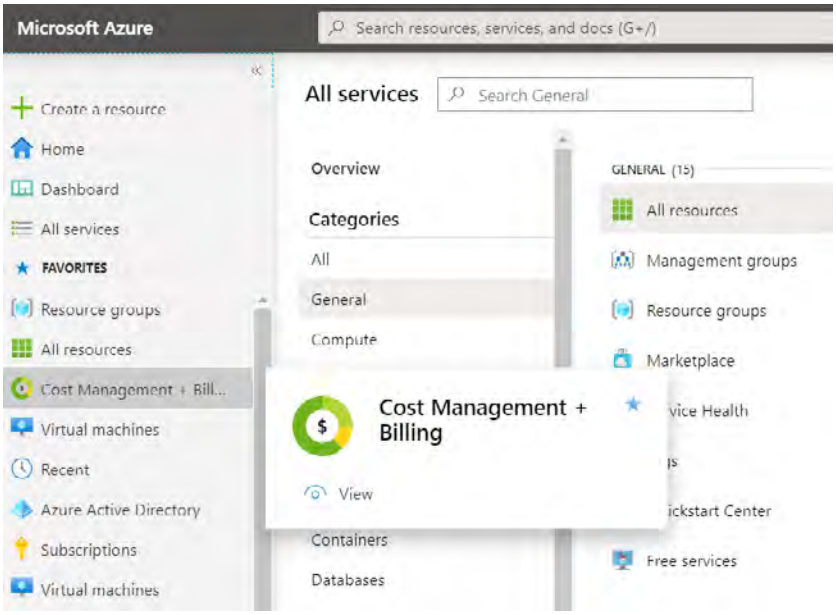


Figure 6.1 : Service Gestion des coûts + facturation dans le portail Azure

Veuillez noter que si votre facturation est gérée par un CSP, vous n'aurez pas accès à cette fonctionnalité. Les clients du CSP peuvent afficher leurs coûts dans le cadre du paiement basé sur la consommation s'ils font migrer leurs abonnements hérités du CSP vers le plan Azure. Nous allons aborder le plan Azure et la plateforme Modern Commerce plus loin dans ce chapitre.

La fonction **Gestion des coûts + facturation** affiche tous les abonnements et les portées de facturation auxquelles vous avez accès, comme illustré à la *Figure* 6.2 :

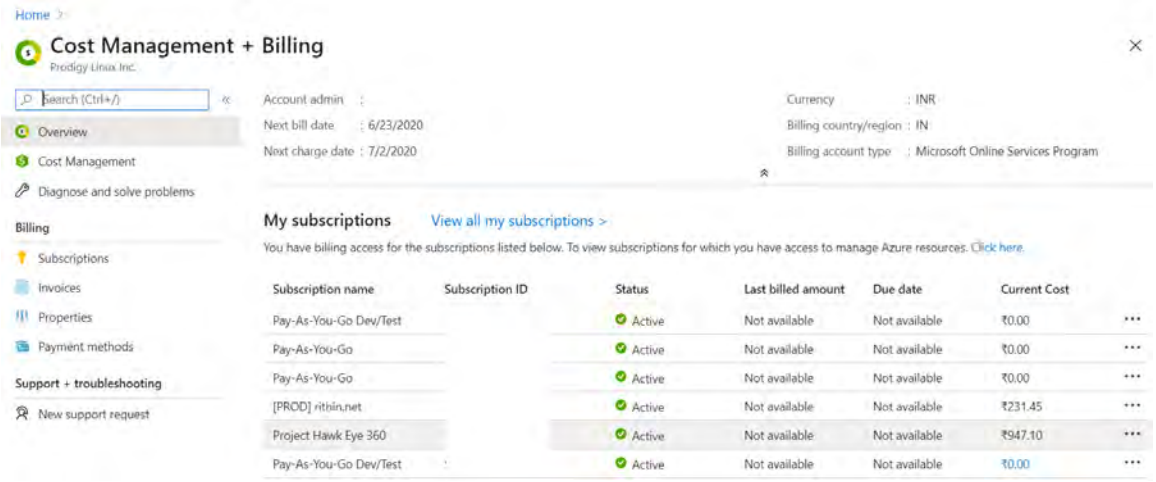


Figure 6.2 : présentation de la facturation des abonnements des utilisateurs

La section **Cost Management** (Gestion des coûts) comporte plusieurs panneaux, tels que :

- **Cost analysis** (Analyse des coûts) pour analyser l'utilisation d'une portée.
- **Budgets** pour définir les budgets.
- **Cost alerts** (Alertes de coûts) pour notifier les administrateurs lorsque l'utilisation dépasse un certain seuil.
- **Advisor recommendations** (Recommandations Advisor) pour obtenir des conseils en vue de réaliser des économies potentielles. Azure Advisor sera présenté en détail dans la dernière section de ce chapitre.
- **Exports** (Exportations) pour automatiser les exportations d'utilisation vers le stockage Azure.
- **Cloudyn** est un outil utilisé par les partenaires CSP pour analyser les coûts, car ils n'ont pas accès à Cost Management.
- **Connecteurs pour AWS** pour connecter vos données de consommation AWS à Azure Cost Management.

Les différentes options disponibles dans Azure Cost Management sont illustrées à la *Figure 6.3* :

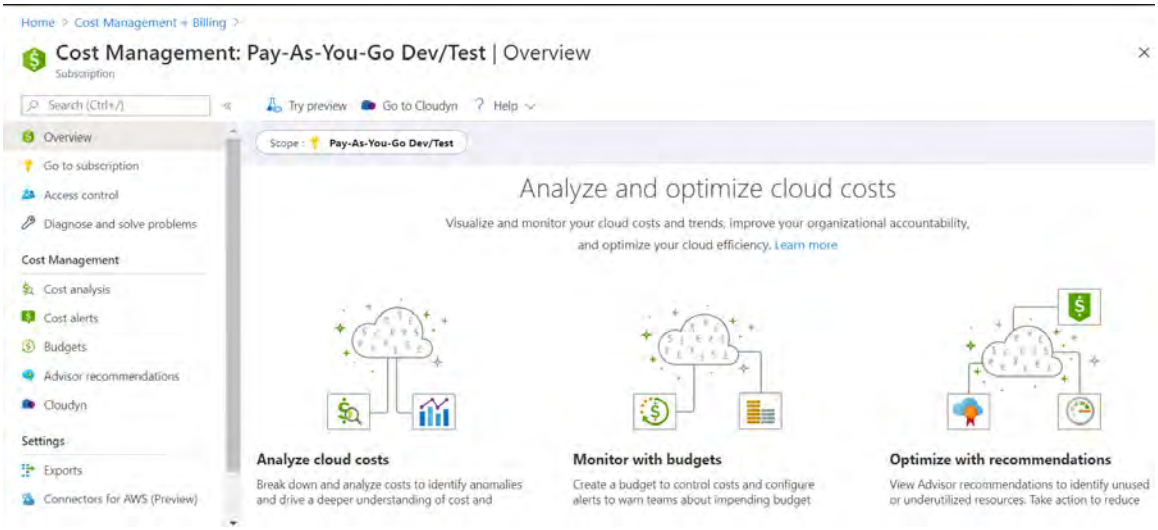


Figure 6.3 : présentation de Cost Management

Cliquez sur le menu **Cost analysis** (Analyse des coûts) de ce panneau pour obtenir un tableau de bord interactif complet, qui permet d'analyser les coûts à l'aide de différentes dimensions et mesures :

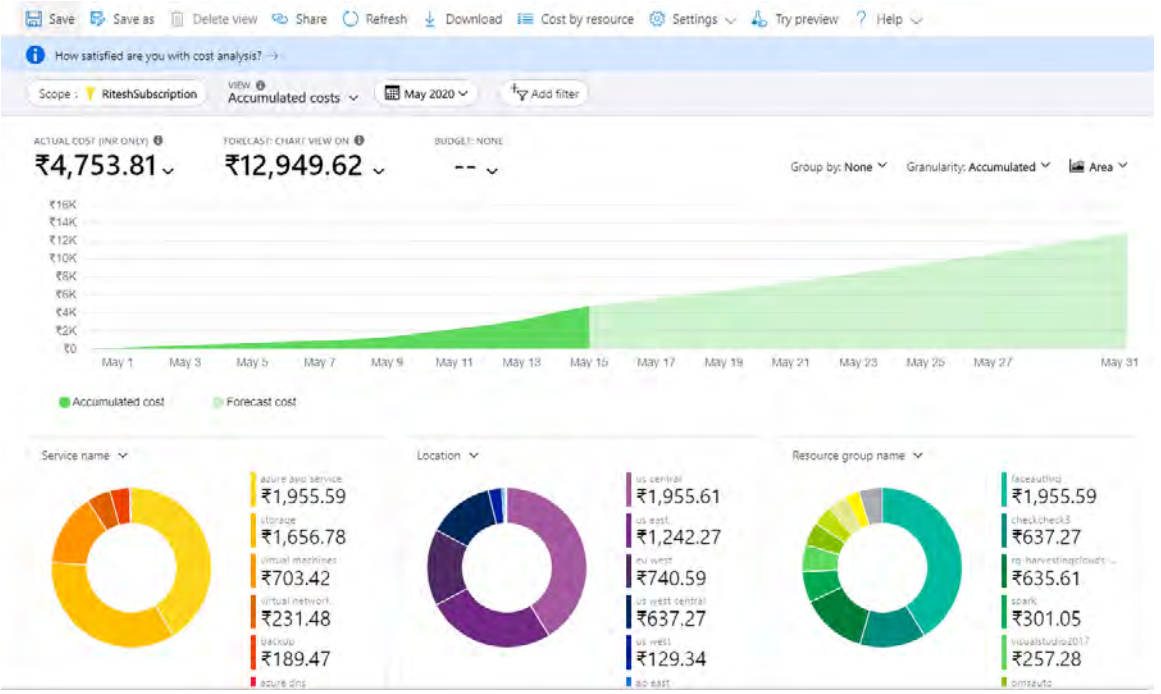


Figure 6.4 : analyse des coûts d'abonnement via l'option Analyse des coûts

Le tableau de bord affiche non seulement le coût actuel, mais prévoit également les coûts et les décompose en fonction de plusieurs dimensions. Les champs **Service name** (Nom du service), **Location** (Emplacement) et **Resource group name** (Nom du groupe de ressources) sont renseignés par défaut, mais ils peuvent également être remplacés par d'autres dimensions. Chaque vue est toujours associée à une portée. Certaines des portées disponibles sont le compte de facturation, le groupe de gestion, l'abonnement et le groupe de ressources. Vous pouvez modifier la portée en fonction du niveau que vous souhaitez analyser.

Le menu **Budget** à gauche nous permet de définir le budget pour optimiser la gestion des coûts. Il propose des fonctions d'alerte au cas où le coût réel ne respecte pas les estimations budgétaires :

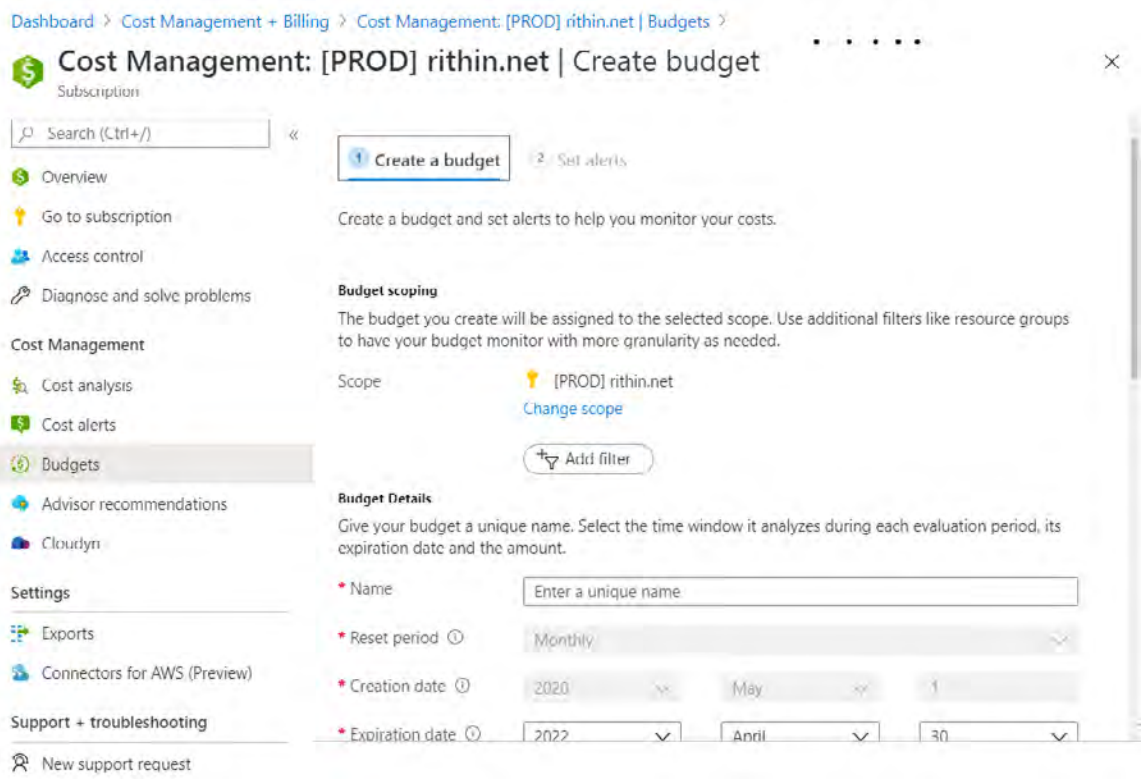


Figure 6.5 : création d'un budget

Cost Management nous permet également d'extraire des données de coût provenant d'autres Clouds, tels que AWS, dans les tableaux de bord actuels, en gérant ainsi les coûts de plusieurs Clouds à partir d'un panneau et d'un tableau de bord uniques. Toutefois, cette fonctionnalité est toujours en version préliminaire au moment de la rédaction de ce livre. Ce connecteur est devenu facturable depuis le 25 août 2020.

Vous devez renseigner les détails de votre rôle AWS et d'autres détails pour extraire les informations sur les coûts, comme illustré à la *Figure* 6.5. Si vous ne savez pas comment créer la stratégie et le rôle dans AWS, accédez à l'adresse https://docs.microsoft.com/azure/cost-management-billing/costs/aws-integration-set-up-configure#create-a-role-and-policy-in-aws :

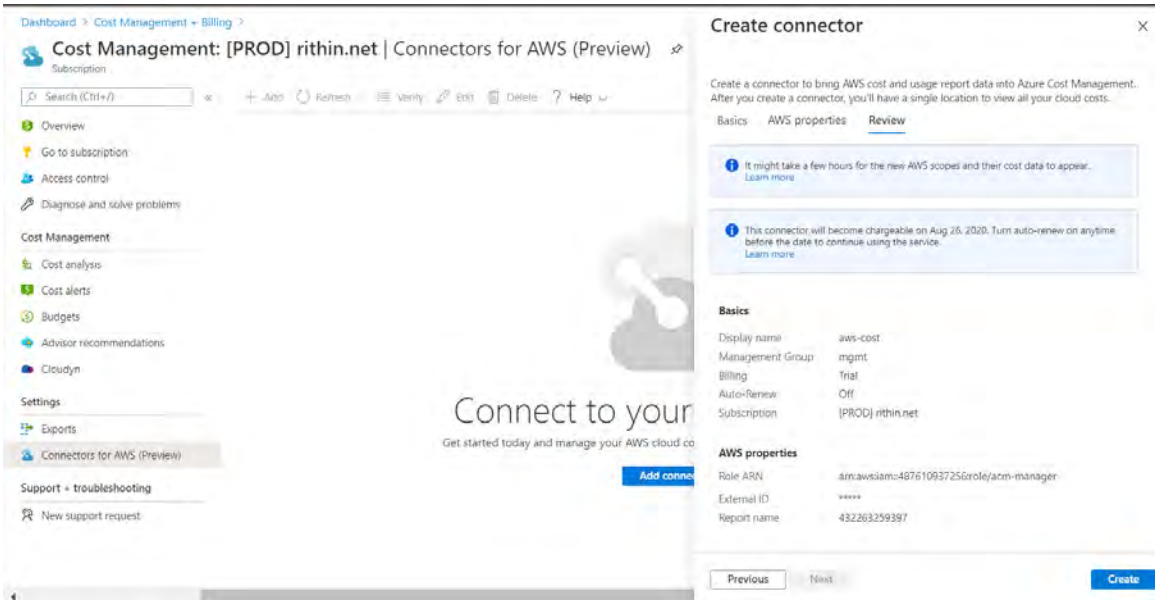


Figure 6.6 : création d'un connecteur AWS dans Cost Management

Les rapports de coûts peuvent également être exportés régulièrement vers un compte de stockage.

Une partie de l'analyse des coûts est également disponible dans le panneau **Subscriptions** (Abonnements). Les ressources et leurs coûts sont affichés dans la section **Overview** (Présentation). En outre, il existe un autre graphique, où sont affichés vos dépenses courantes, vos prévisions et votre solde de crédit (si vous utilisez un abonnement basé sur le crédit).

La *figure* 6.7 affiche les informations sur les coûts :

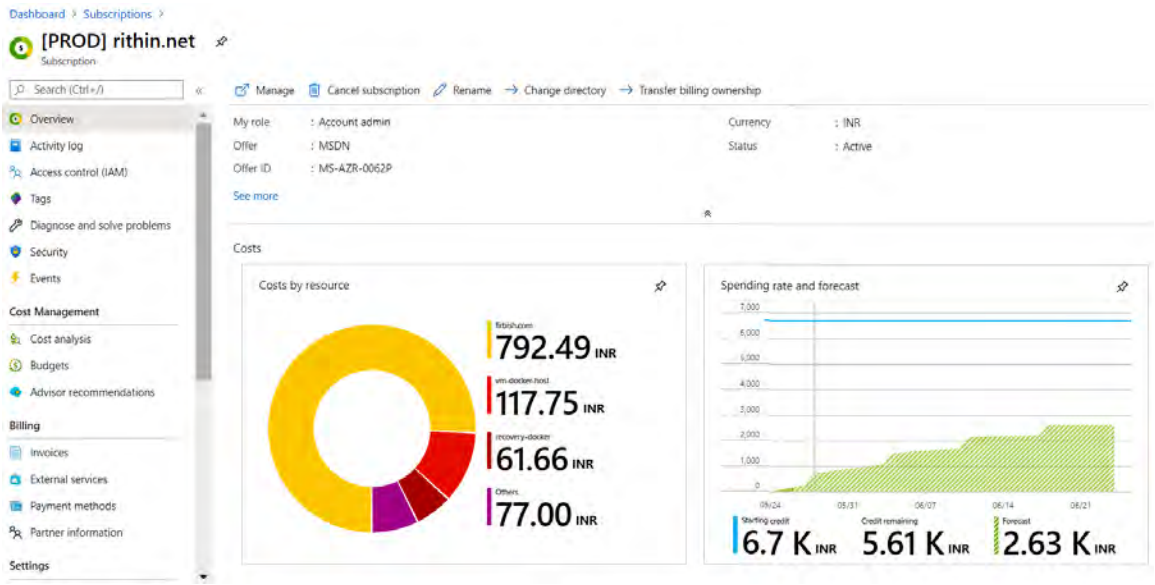


Figure 6.7 : analyse de coûts de l'abonnement

En cliquant sur l'un des coûts de la *Figure* 6.7 vous serez redirigé vers la section **Cost Management** – **Cost Analysis** (Analyse des coûts). Cost Management propose de nombreuses dimensions qui permettent de regrouper les données à des fins d'analyse. Les dimensions disponibles varient en fonction de la portée sélectionnée. Voici certaines des dimensions les plus utilisées :

- Types de ressources
- Groupe de ressources
- Balise
- Emplacement des ressources
- ID de ressource
- Catégorie de compteurs
- Sous-catégorie de compteurs
- Service

Au début du chapitre, nous avons mentionné que les balises peuvent être utilisées pour la gestion des coûts. Imaginons que vous ayez une balise appelée Department (Service) avec des valeurs relatives à l'IT, les RH et les finances. En marquant les ressources de manière appropriée, vous pourrez déterminer le coût encouru par chaque service. Vous pouvez également télécharger le rapport des coûts au format CSV, Excel ou PNG en utilisant le bouton **Download** (Télécharger) :

Cost Management prend également en charge plusieurs vues. Vous pouvez créer votre propre tableau de bord et l'enregistrer. Les clients ayant souscrit un contrat d'entreprise bénéficient d'un avantage supplémentaire : le connecteur Cost Management ou Power BI. Grâce au connecteur, les utilisateurs peuvent extraire les statistiques d'utilisation de Power BI et créer des visualisations.

Jusqu'à présent, nous avons discuté du suivi de notre utilisation à l'aide de Cost Management. La section suivante porte sur le fonctionnement de la facturation des services utilisés.

Facturation

Le système de facturation Azure fournit également des informations au sujet des factures mensuelles.

Selon le type d'offre, la méthode de facturation peut varier. Pour les utilisateurs qui ont choisi le paiement basé sur la consommation, les factures seront envoyées mensuellement à l'administrateur de compte. Toutefois, pour les clients ayant souscrit des contrats d'entreprise, la facture sera envoyée au contact indiqué lors de l'inscription.

En cliquant sur le menu **Invoices** (Factures), vous accéderez à la liste de toutes les factures générées, et en cliquant sur l'une de ces factures, vous obtiendrez les détails de cette facture. La *Figure 6.8* montre comment les factures sont affichées dans le portail Azure :

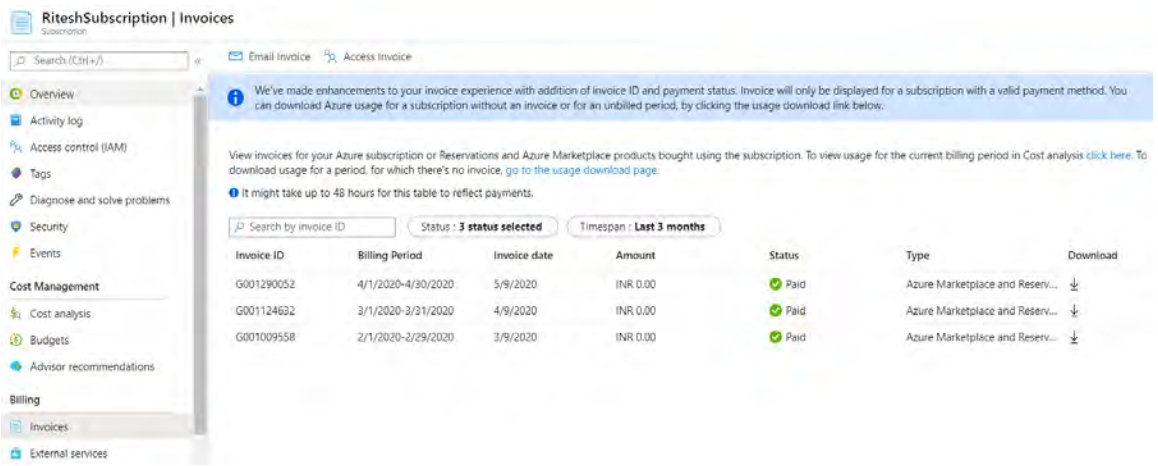


Figure 6.8 : liste des factures et leurs détails

Il existe deux types de factures : l'une concerne les services Azure tels que SQL, les machines virtuelles et la mise en réseau. L'autre type de facture concerne Azure Marketplace et les réservations. Azure Marketplace propose aux clients des services de partenaires auprès de différents fournisseurs. Nous évoquerons les réservations Azure ultérieurement.

L'administrateur du compte d'un abonnement basé sur la consommation a accès aux factures par défaut. S'il le souhaite, il peut déléguer l'accès à d'autres utilisateurs, tels que l'équipe financière de l'organisation, en choisissant l'option **Access invoice** (Accéder aux factures), comme illustré à la *Figure* 6.8. En outre, l'administrateur de compte peut sélectionner les adresses électroniques auxquelles il souhaite envoyer les copies des factures.

L'option **Email Invoice** (Envoyer la facture par e-mail) n'est pas disponible pour le plan de support à l'heure actuelle. Vous pouvez également visiter le portail des comptes et télécharger la facture. Microsoft abandonne progressivement ce portail, et la plupart des fonctions deviennent obsolètes à mesure qu'elles sont intégrées au portail Azure.

Jusqu'à présent, nous avons discuté des abonnements et du fonctionnement de la facturation. Microsoft a introduit récemment une nouvelle plateforme : Modern Commerce. Grâce à cette nouvelle expérience commerciale, le processus et l'expérience d'achat ont été simplifiés. Examinons de plus près Modern Commerce et découvrons en quoi cette plateforme diffère de la plateforme héritée dont nous avons discuté jusqu'à présent.

L'expérience Modern Commerce

Si votre entreprise travaille déjà avec Microsoft, vous savez déjà qu'il existe plusieurs accords pour chaque offre, notamment Web Direct, les contrats d'entreprise, CSP, **accord sur les services et produits Microsoft** (**MSPA**), **abonnement au Cloud et au serveur** (**SCE**), etc. Parallèlement à cela, chacun dispose de son propre portail ; par exemple, les contrats d'entreprise ont le portail EA, le CSP dispose du portail Espace partenaires et le contrat de licences en volume dispose également de son propre portail.

Chaque offre est associée à des conditions générales spécifiques, et les clients doivent les respecter chaque fois qu'ils effectuent un achat. La transition d'une offre à une autre n'est pas aisée, car chacune est régie par des conditions générales spécifiques. Imaginons que vous ayez déjà un abonnement EA et que vous souhaitiez le convertir en abonnement CSP. Il vous faudra peut-être supprimer certains services partenaires car ils ne seront pas pris en charge dans CSP. Chaque offre applique des règles différentes selon les produits. Du point de vue des clients, il est très difficile de comprendre ce qui est pris en charge et en quoi les règles diffèrent.

Pour résoudre ce problème, Microsoft a récemment publié un nouvel accord appelé **Microsoft Customer Agreement** (**MCA**). Il s'agira des conditions générales de base. Vous pouvez y apporter des modifications chaque fois que cela est nécessaire lorsque vous vous inscrivez à un nouveau programme.

Trois programmes **Go-to-Market** (**GTM**) sont proposés pour Azure :

- **Orienté terrain :** les clients interagissent directement avec l'équipe de comptes Microsoft et la facturation sera directement gérée par Microsoft. Ce programme remplacera les contrats d'entreprise à terme.
- **Orienté partenaire :** cela équivaut au programme Azure dans le CSP, où un partenaire gère votre facturation. Il existe différents partenaires à travers le monde. Vous pourrez trouver rapidement les partenaires à proximité en effectuant une recherche Web. Ce programme remplacera le programme Azure dans le CSP. Lors de la première étape de Modern Commerce, un partenaire signera un **accord de partenariat Microsoft** (**MPA**) avec Microsoft afin que ses clients existants signent le MCA. Au moment de la rédaction de cet ouvrage, de nombreux partenaires ont transféré leurs clients vers Modern Commerce, et la nouvelle expérience commerciale est disponible dans 139 pays.
- **Libre-service :** ce programme remplace Web Direct. Il ne nécessite aucune implication de la part du partenaire ou de l'équipe de comptes Microsoft. Les clients peuvent l'acheter directement auprès de microsoft.com. Ils signeront le MCA au cours de l'achat.

La facturation sera exécutée sur le plan Azure et elle sera toujours alignée sur le mois calendaire. L'achat d'un plan Azure est très similaire à l'achat d'un autre abonnement. La différence réside dans le fait que la signature du MCA est effectuée cours du processus.

Un plan Azure peut héberger plusieurs abonnements, et il agira comme un conteneur de niveau racine. Toute l'utilisation est reliée à un seul plan Azure. Tous les abonnements d'un plan Azure feront office de conteneurs pour les services hôtes, tels que les machines virtuelles, SQL Database et la mise en réseau.

Voici certains des changements et des progrès que nous avons pu observer suite à l'introduction de Modern Commerce :

- Les portails deviendront obsolètes à terme. Par exemple, les clients ayant souscrit un contrat d'entreprise n'étaient en mesure de télécharger les informations sur l'utilisation de l'inscription à partir du portail EA. Microsoft les a désormais intégrées à Azure Cost Management, en proposant une meilleure expérience par rapport à celle du portail EA.
- Les tarifs seront exprimés en USD et facturés dans la devise locale. Si votre devise n'est pas USD, le **taux de change** (**FX**) sera appliqué et indiqué dans votre facture. Microsoft utilise les tarifs FX de Thomson Reuters, et ces tarifs seront appliqués le premier de chaque mois. Cette valeur sera cohérente tout au long du mois, quel que soit le taux du marché.
- Les clients du CSP qui souscrivent au nouveau plan Azure seront en mesure d'utiliser Cost Management. L'accès à Cost Management offre de nombreuses possibilités en matière de suivi des coûts, car il permet d'accéder à toutes les fonctions natives de Cost Management.

Tous les abonnements dont nous avons discuté jusqu'à présent seront ensuite transférés vers un plan Azure, qui est l'avenir d'Azure. Maintenant que vous comprenez les notions de base de Modern Commerce, abordons un autre sujet qui joue un rôle très important lorsque nous créons des solutions. Des limites sont appliquées par défaut à la plupart des services. Certaines peuvent être augmentées, tandis que certaines sont des limites strictes. Lorsque nous concevons une solution, nous devons veiller à établir des quotas suffisants. La planification de la capacité est un élément essentiel de la conception architecturale. La prochaine section porte sur les limites des abonnements.

Utilisation et quotas

Comme nous l'avons mentionné dans la section précédente, la planification de la capacité doit être l'une des premières étapes de l'architecture d'une solution. Nous devons vérifier si l'abonnement dispose de quotas suffisants pour s'adapter aux nouvelles ressources conçues. Si ce n'est pas le cas, des problèmes risquent de survenir pendant le déploiement.

Chaque abonnement dispose d'un quota limité pour chaque type de ressource. Par exemple, 10 adresses IP publiques maximum peuvent être mises en service avec un compte MSDN Microsoft. De même, toutes les ressources sont assujetties à une limite maximale par défaut pour chaque type de ressource. Ces limites d'abonnement peuvent être augmentées en contactant le support Azure ou en cliquant sur le bouton **Request Increase** (Demander une augmentation) dans le panneau **Usage** (Utilisation) **+ Quota** sur la page **Subscription** (Abonnement).

Il sera difficile de passer en revue toute la liste compte tenu du nombre de ressources de chaque région. Le portail fournit des options pour filtrer le jeu de données et rechercher les éléments souhaités. La *Figure 6.9* illustre le scénario suivant : si nous filtrons l'emplacement à **Central US** (Centre des États-Unis) et si nous définissons le fournisseur de ressources sur **Microsoft.Storage**, nous pouvons vérifier les quotas disponibles pour les comptes de stockage :

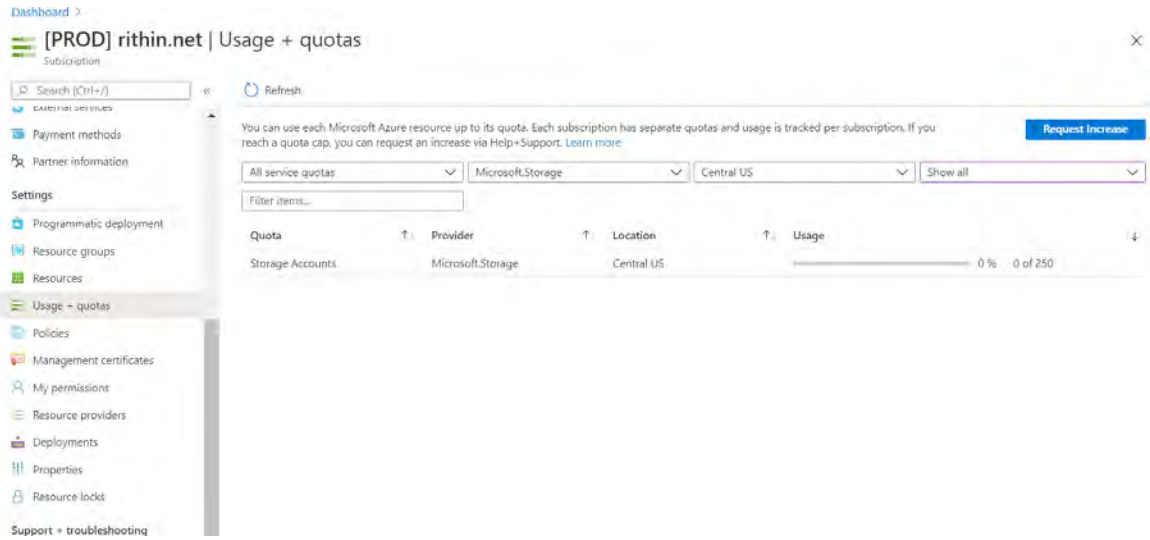


Figure 6.9 : utilisation et quota pour un emplacement et un fournisseur de ressources donnés

La *Figure 6.9* montre clairement que nous n'avons créé aucun compte de stockage dans le Centre des États-Unis. Le quota s'élève donc à 250 comptes. Si la solution que nous concevons requiert plus de 250 comptes, nous devons cliquer sur **Request Increase** (Demander une augmentation) en contactant le support Azure.

Ce panneau nous permet de planifier la capacité avant le déploiement.

Lors du filtrage du rapport, nous avons utilisé le terme **fournisseur de ressources** et sélectionné **Microsoft.Storage**. Ce terme est abordé en détail dans la section suivante.

Fournisseurs de ressources et types de ressources

Que vous interagissiez avec le portail Azure, les services de filtrage ou le filtrage du rapport d'utilisation de la facturation, vous devrez peut-être travailler avec des fournisseurs de ressources et des types de ressources. Lorsque vous créez une machine virtuelle, vous interagissez avec le fournisseur de ressources `Microsoft.Compute` et le type de ressource `virtualMachines`. Le bouton **Create** (Créer) sur lequel vous cliquez pour créer des machines virtuelles communique avec le fournisseur de ressources via une API en vue d'effectuer le déploiement. Cette action est toujours exprimée au format `{resource-provider}/{resource-type}`. Ainsi, le type de ressource de la machine virtuelle est `Microsoft.Compute/virtualMachines`. En bref, les fournisseurs de ressources permettent de créer des types de ressources.

Les fournisseurs de ressources doivent être enregistrés dans des abonnements Azure. Les types de ressources ne seront pas disponibles au sein d'un abonnement si les fournisseurs de ressources ne sont pas enregistrés. Par défaut, la plupart des fournisseurs sont automatiquement enregistrés ; cela dit, certains scénarios nécessitent une inscription manuelle.

Le tableau de bord indiqué à la *Figure 6.10* permet d'obtenir la liste des fournisseurs disponibles, de ceux qui sont enregistrés et de ceux qui ne le sont pas et d'enregistrer des fournisseurs non enregistrés et vice versa. Pour effectuer cette opération, vous devez disposer des rôles nécessaires : les rôles de propriétaire ou de contributeur suffisent. La *Figure 6.10* montre l'aspect du tableau de bord :

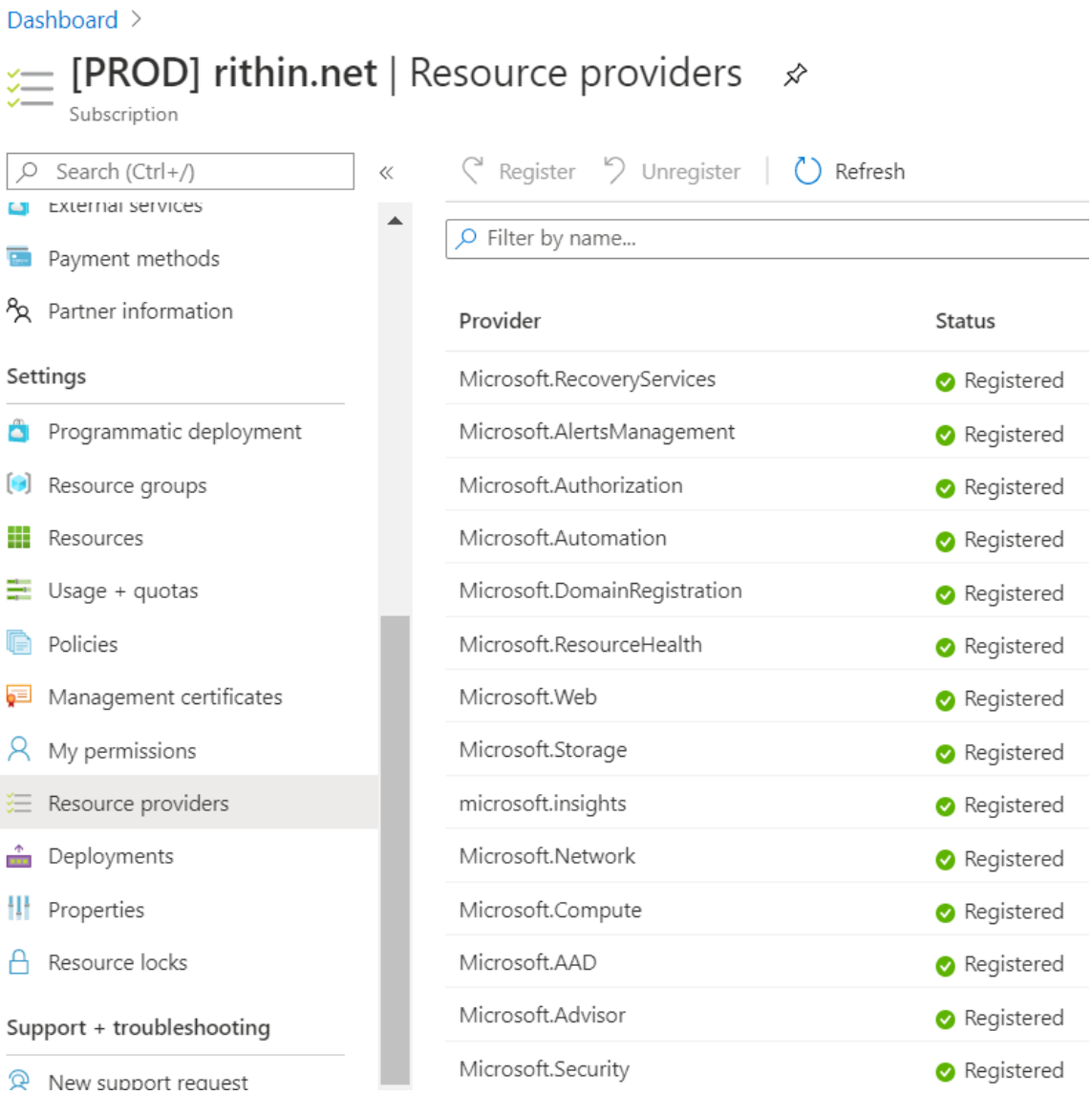


Figure 6.10 : liste des fournisseurs de ressources enregistrés et non enregistrés

Dans la section précédente, nous avons discuté du téléchargement des factures et des informations d'utilisation. Si vous avez besoin de télécharger les données par programme et de les enregistrer, vous pouvez utiliser des API. La section suivante porte sur les API de facturation Azure.

Utilisation et API de facturation

Le portail est un excellent moyen de trouver des informations sur la consommation et sur la facturation des services, toutefois Azure propose également les API suivantes pour récupérer les détails par programme et créer des tableaux de bord, ainsi que des rapports personnalisés. Les API varient selon le type d'abonnement utilisé. Il existe de nombreuses API, nous allons donc partager la documentation Microsoft sur chaque API afin de toutes les passer en revue.

API de facturation d'entreprise Azure

Les clients ayant souscrit des contrats d'entreprise disposent d'un ensemble dédié d'API leur permettant de travailler avec les données de facturation. Les API suivantes utilisent la clé API du portail EA pour l'authentification. Elles ne fonctionneront pas avec les jetons d'Azure Active Directory :

- **API de solde et de synthèse :** comme nous l'avons mentionné plus tôt, les contrats d'entreprise fonctionnent avec un engagement monétaire, et il est très important de suivre le solde, le surplus, les ajustements de crédit et les frais d'Azure Marketplace. À l'aide de cette API, les clients peuvent extraire le solde et la synthèse d'une période de facturation.
- **API dédiée aux informations sur l'utilisation :** cette API permet d'obtenir des informations sur l'utilisation quotidienne au sujet de l'inscription, avec une précision permettant d'atteindre le niveau de l'instance. La réponse de cette API sera similaire au rapport d'utilisation qui peut être téléchargé à partir du portail EA.
- **API dédiée aux frais du magasin Marketplace :** il s'agit d'une API dédiée permettant d'extraire les frais liés aux achats sur le marché.
- **API de fiche de prix :** chaque inscription est associée à une fiche de prix spéciale, et les remises varient d'un client à l'autre. L'API de fiche de prix peut extraire la liste de prix.
- **API dédiée aux informations de l'instance réservée :** nous n'avons pas encore discuté des réservations Azure. Ce sujet sera abordé à la fin de ce chapitre. Cette API permet d'obtenir des informations d'utilisation sur les réservations et une liste des réservations dans l'inscription.

Voici le lien vers la documentation dédiée aux API des contrats d'entreprise : https://docs.microsoft.com/azure/cost-management-billing/manage/enterprise-api.

Intéressons-nous désormais aux API de consommation Azure.

API de consommation Azure

Les API de consommation Azure peuvent être utilisées avec des abonnements de contrats d'entreprise et Web Direct (à quelques exceptions près). Elles nécessitent un jeton, qui doit être généré au cours de l'authentification sur Azure Active Directory. Étant donné que ces API prennent également en charge les contrats d'entreprise, ce jeton ne doit pas être confondu avec la clé des API de contrats d'entreprise mentionnée dans la section précédente. Voici les API de clés qui sont explicites :

- API dédiée aux informations d'utilisation
- API dédiée aux frais Marketplace
- API de recommandations de réservation
- API dédiée aux informations et à la synthèse des réservations

Les clients ayant souscrit un contrat d'entreprise bénéficient de la prise en charge supplémentaire des API suivantes :

- Fiche de prix
- Budgets
- Soldes

La documentation est disponible ici :https://docs.microsoft.com/azure/cost-management-billing/manage/consumption-api-overview.

En outre, il existe un autre ensemble d'API réservé exclusivement aux clients Web Direct :

- **API dédiée à l'utilisation des ressources Azure :** cette API peut être utilisée avec un abonnement de contrat d'entreprise ou basé sur la consommation et permet de télécharger les données d'utilisation.
- **API Azure Resource RateCard :** cette API s'applique uniquement à Web Direct. Les contrats d'entreprise ne sont pas pris en charge. Les clients Web Direct peuvent l'utiliser pour télécharger des feuilles de prix.
- **API de téléchargement de factures Azure :** cette API s'applique uniquement aux clients Web Direct. Elle est utilisé pour télécharger des factures par programme.

Les noms peuvent vous sembler familiers. La seule différence est le point de terminaison appelé. Pour les API de facturation d'entreprise Azure, l'URL commencera par `https://consumption.azure.com`, et par `https://management.azure.com` pour les API de consommation Azure. C'est ainsi que vous pourrez les différencier. La section suivante porte sur un nouvel ensemble d'API spécifiquement utilisées par Cost Management.

API Azure Cost Management

Azure Cost Management a introduit un nouvel ensemble d'API pour les clients. Ces API sont l'épine dorsale de Cost Management, que nous avons déjà utilisé dans le portail Azure. Voici les API de clé :

- **API d'utilisation des requêtes :** il s'agit de la même API que celle utilisée pour l'analyse des coûts dans le portail Azure. Nous pouvons personnaliser la réponse à notre guise à l'aide d'une charge utile. Elle s'avère très utile pour personnaliser un rapport. La plage de dates ne peut pas dépasser 365 jours.
- **API de budget :** les budgets sont une autre caractéristique d'Azure Cost Management et cette API nous permet d'interagir avec les budgets par programme.
- **API de prévision :** cette API peut être utilisée pour obtenir la prévision d'une portée. Pour le moment, l'API de prévision est réservée aux clients ayant souscrit des contrats d'entreprise.
- **API de dimensions :** nous avons déjà mentionné que Cost Management prend en charge plusieurs dimensions en fonction de la portée. Cette API vous permet d'obtenir la liste des dimensions prises en charge en fonction d'une portée spécifique.
- **API d'exportation :** Cost Management nous permet également d'automatiser l'exportation du rapport vers un compte de stockage. L'API d'exportation peut être utilisée pour interagir avec la configuration d'exportation, notamment le nom du compte de stockage, la personnalisation, la fréquence, etc.

Consultez la documentation officielle à l'adresse https://docs.microsoft.com/rest/api/cost-management.

Étant donné que Modern Commerce est une extension du MCA, un tout nouvel ensemble d'API est proposé ici : https://docs.microsoft.com/rest/api/billing.

Vous avez peut-être remarqué que CSP n'a été mentionné dans aucun de ces scénarios. Dans CSP, le client n'a pas accès à la facturation car elle est gérée par un partenaire. Par conséquent, les API ne sont pas exposées. Toutefois, une transition vers un plan Azure permettrait au client CSP d'utiliser les API Azure Cost Management pour consulter les tarifs de distribution.

Tout langage de programmation ou de script peut être employé pour utiliser ces API et les mélanger afin de créer des solutions de facturation complètes. Dans la section suivante, nous allons nous concentrer sur la calculatrice de tarifs Azure, qui permet au client ou à l'architecte à comprendre le coût d'un déploiement.

Calculatrice de tarifs Azure

Azure met une calculatrice des coûts à la disposition des utilisateurs et des clients afin que ces derniers puissent estimer leurs frais et leur consommation. Cette calculatrice est disponible à l'adresse https://azure.microsoft.com/pricing/calculator :

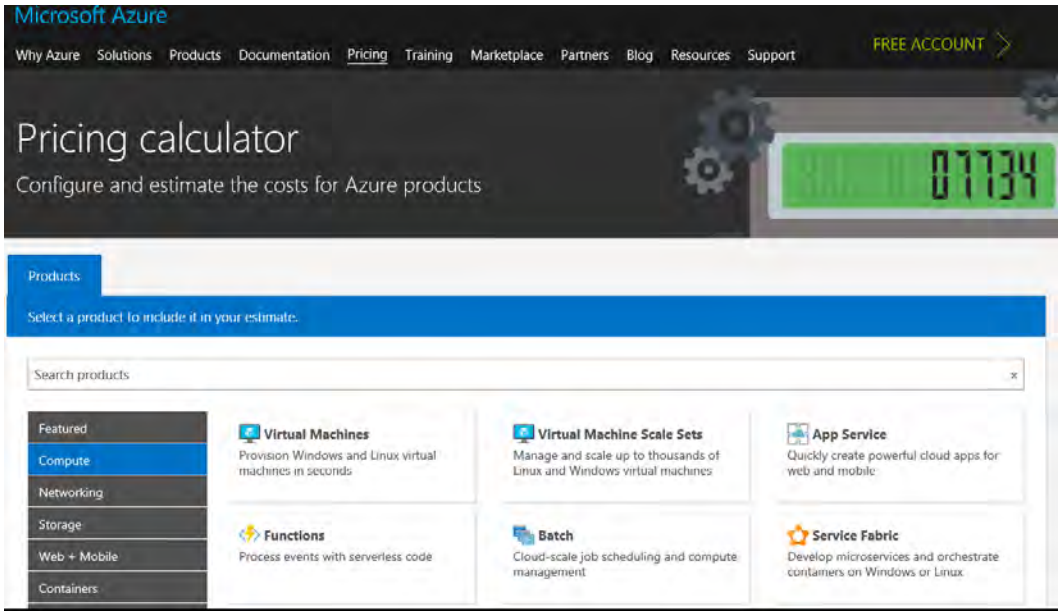


Figure 6.11 : calculatrice de tarifs Azure

Les utilisateurs peuvent sélectionner plusieurs ressources à partir du menu de gauche, qui seront ajoutées à la calculatrice. Dans l'exemple suivant, une machine virtuelle est ajoutée. Une configuration supplémentaire, tenant compte de la région de la machine virtuelle, du système d'exploitation, du type, du niveau, de la taille de l'instance, du nombre d'heures et du décompte doit être fournie afin de pouvoir calculer les coûts :

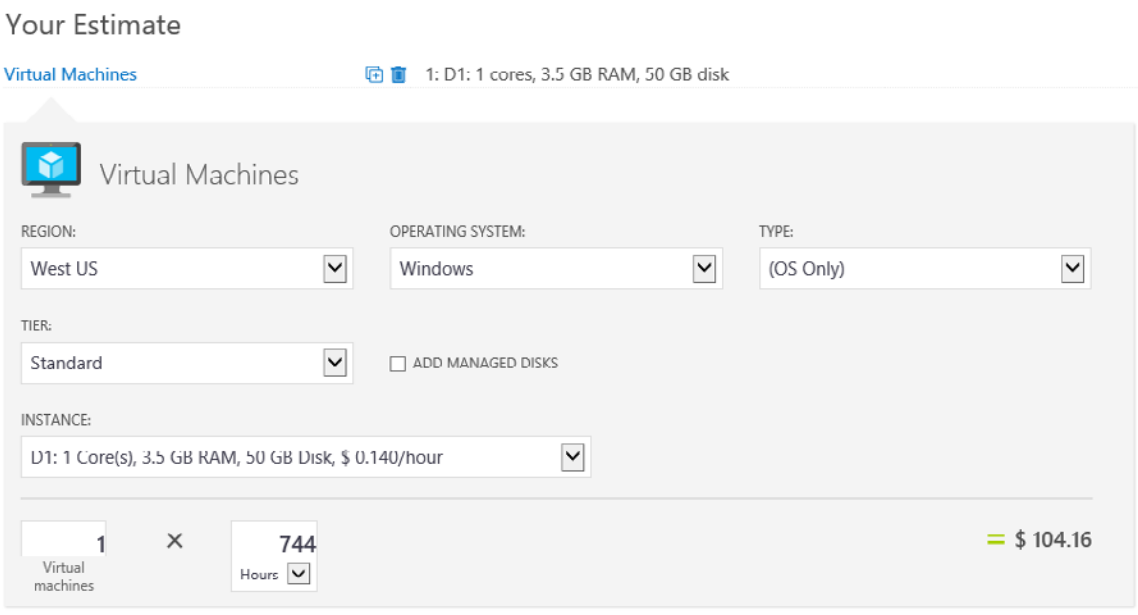


Figure 6.12 : saisie des détails de configuration pour calculer les coûts des ressources

De même, les coûts liés à Azure Functions concernant la taille de la mémoire des machines virtuelles, les durées d'exécution et les exécutions par seconde, sont indiqués à la *Figure 6.13* :

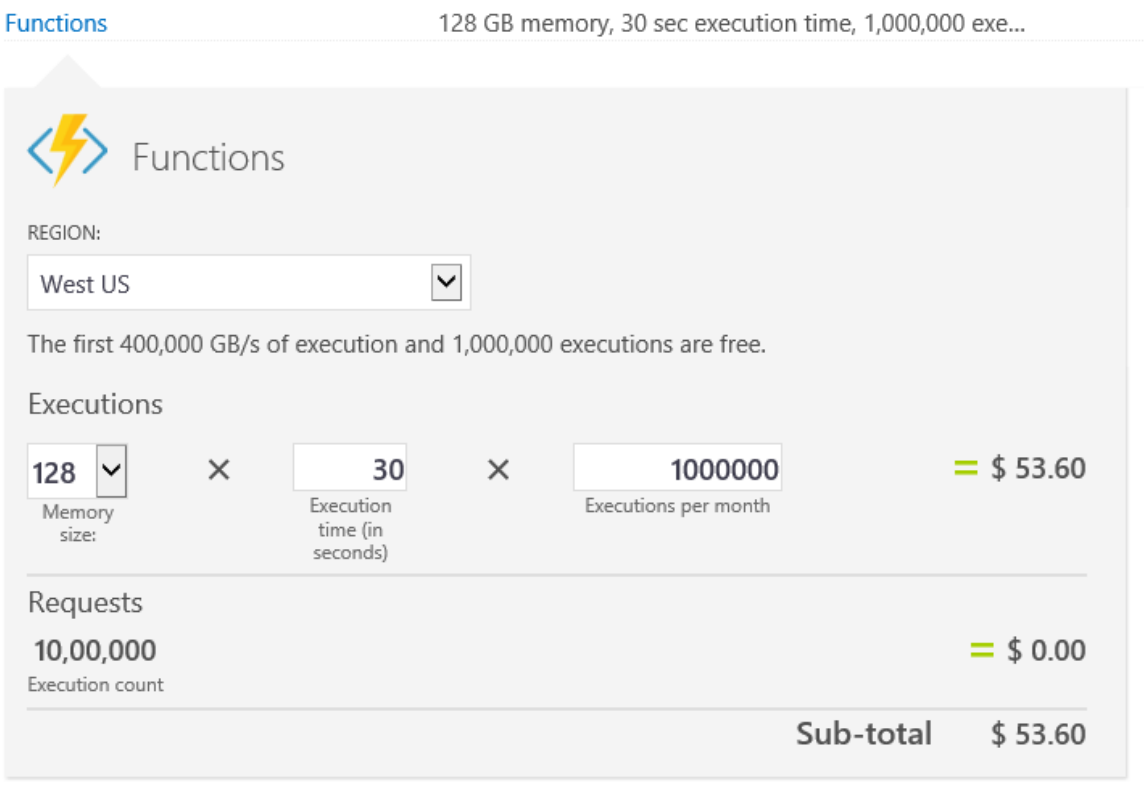


Figure 6.13 : calcul des coûts pour Azure Functions

Azure fournit différents niveaux et plans de support :

- Support par défaut : gratuit
- Support développeur : 29 USD par mois
- Support standard : 300 USD par mois
- Profession direct : 1 000 USD par mois

Pour consulter la comparaison complète des plans de support, accédez à l'adresse https://azure.microsoft.com/support/plans.

Vous pouvez sélectionner le support que vous souhaitez en fonction de vos besoins. Enfin, le coût global total est indiqué :

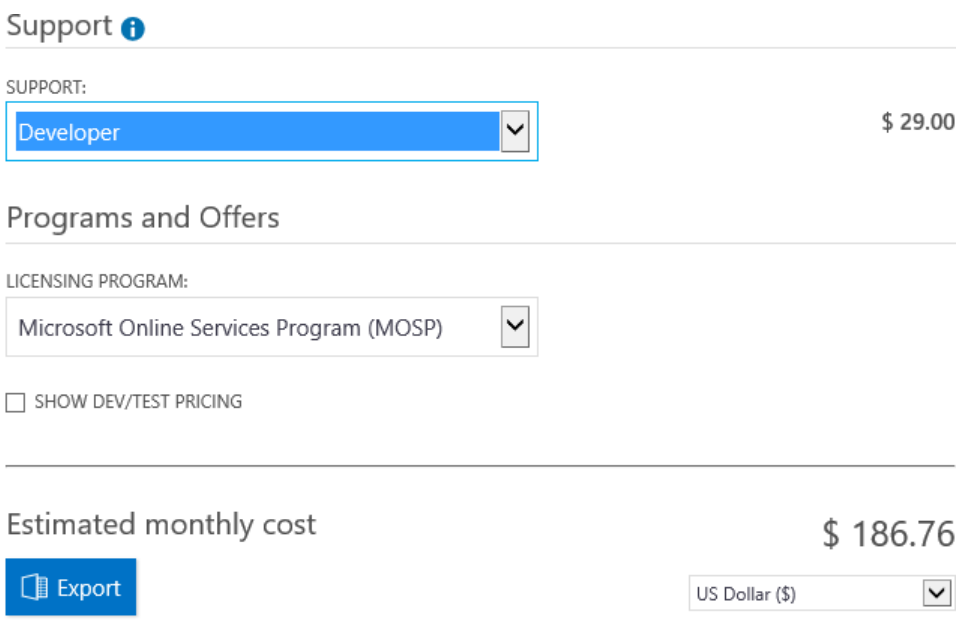


Figure 6.14 : estimation des coûts pour un plan de support sélectionné

Les architectes doivent être familiarisés avec chaque fonction Azure utilisée dans l'architecture et la solution Azure. L'efficacité de la calculatrice Azure dépend des ressources sélectionnées et de leur configuration. Toute donnée erronée peut entraîner une estimation biaisée et incorrecte, qui différera de la facturation effective.

Cette section conclut ce chapitre. Nous avons abordé les bases de la facturation, le moment est venu de découvrir les bonnes pratiques. C'est en appliquant les bonnes pratiques que vous pourrez optimiser les coûts.

Bonnes pratiques

Les architectes doivent mieux comprendre leur architecture ainsi que les composants Azure utilisés. Ils doivent, sur la base d'un suivi actif, d'audits et de la consommation, déterminer l'offre Microsoft la plus appropriée en termes de références, de taille et de fonctionnalités. Cette section abordera certaines des bonnes pratiques à adopter du point de vue de l'optimisation des coûts.

Azure Governance

La solution Azure Governance peut être définie comme un ensemble de processus ou de mécanismes utilisés pour maintenir un contrôle total sur les ressources déployées dans Azure. Voici quelques-uns des points clés :

- Configurez une convention de nommage pour tous les types et groupes de ressources. Veillez à respecter en permanence la convention de nommage pour l'ensemble des ressources et groupes de ressources. Pour ce faire, vous pouvez établir des stratégies Azure.

- Configurez une organisation logique et plusieurs taxonomies pour les ressources, les groupes de ressources et les abonnements en leur appliquant des balises. Les balises catégorisent les ressources et permettent également d'évaluer les coûts selon différentes perspectives. Des stratégies Azure peuvent être établies pour y parvenir. Par ailleurs, plusieurs stratégies peuvent être combinées en initiatives. Ces initiatives peuvent être appliquées, et à leur tour appliqueront toutes les stratégies de vérification de la conformité et de génération de rapport.
- Utilisez directement Azure Blueprints au lieu de modèles ARM. Ainsi, le déploiement de nouveaux environnements, ressources et groupes de ressources sera normalisé selon les normes de l'entreprise, y compris les conventions de nommage et l'utilisation des balises.

Bonnes pratiques au niveau des ressources de traitement

Les ressources de traitement contribuent à assurer la bonne exécution des services. Voici quelques-unes des bonnes pratiques en matière de ressources de traitement qui doivent être appliquées par les architectes Azure pour optimiser l'utilisation des ressources et la rentabilité :

- Utilisez Azure Advisor pour connaître les options permettant de réaliser des économies sur les machines virtuelles et déterminer si une machine virtuelle est sous-utilisée. Advisor utilise des modèles de Machine Learning et l'intelligence artificielle pour analyser votre utilisation et fournir des recommandations. Ces recommandations jouent un rôle important dans l'optimisation des coûts.
- Utilisez Azure **Reserved Instances** (**RI**). Les instances réservées permettent de réaliser des économies potentielles sur les coûts de calcul en payant le coût de la machine virtuelle à l'avance ou mensuellement. Une instance réservée peut durer un ou trois ans. Si vous achetez une instance réservée, vous réduirez le coût de calcul et seuls le disque, le réseau et la licence (le cas échéant) seront facturés pour une machine virtuelle. Si vous avez cinq machines virtuelles, vous pouvez opter pour cinq instances réservées afin de supprimer complètement le coût de calcul. Les instances réservées recherchent automatiquement des machines virtuelles avec la référence correspondante et s'associent à cette dernière. Les économies potentielles peuvent varier entre 20 et 40 % en fonction de la taille de la machine virtuelle.
- **Azure Hybrid Benefit** (**AHUB**) vous permet d'utiliser vos propres licences Windows Server ou SQL afin de réduire les coûts de licence. En combinant les instances réservées et Azure Hybrid Benefit, vous réaliserez des économies considérables.
- Sélectionnez l'emplacement le plus approprié pour vos services de traitement, par exemple des machines virtuelles. Sélectionnez un emplacement où toutes les fonctions Azure seront regroupées dans la même région. Cela évitera tout trafic de sortie.
- Sélectionnez la taille optimale des machines virtuelles. Une machine virtuelle de grande taille coûte plus cher qu'une machine plus petite, et une machine virtuelle volumineuse peut s'avérer inutile.

- Redimensionnez les machines virtuelles lors des pics de demande et réduisez leur taille lorsque celle-ci recule. Azure lance de nouvelles références régulièrement. Si une nouvelle taille est plus adaptée à vos besoins, vous devez l'utiliser.
- Éteignez les services de traitement en dehors des heures d'activité ou lorsqu'ils ne sont pas utilisés. Cette exigence s'applique aux environnements hors production.
- Annulez l'assignation des machines virtuelles au lieu de les éteindre. Cela libérera toutes les ressources et le système de mesure de la consommation s'arrêtera également.
- Utilisez des laboratoires de développement et de test aux fins de développement et de test. Ceux-ci fournissent des stratégies ainsi que des fonctionnalités d'arrêt et de démarrage automatiques.
- Avec virtual machine scale sets, commencez par un faible nombre de machines virtuelles, puis augmentez-les à mesure que la demande évolue.
- Sélectionnez la taille appropriée (petite, moyenne ou large) pour les passerelles d'application. Celles-ci sont soutenues par des machines virtuelles et peuvent contribuer à réduire les coûts.
- Sélectionnez la passerelle d'application de niveau basique si le pare-feu d'application Web n'est pas nécessaire.
- Sélectionnez les niveaux appropriés pour les passerelles VPN (VPN basique, standard, hautes performances et ultra-hautes performances).
- Réduisez le trafic réseau entre les régions Azure.
- Utilisez un équilibreur de charge avec une adresse IP publique pour accéder à plusieurs machines virtuelles au lieu d'assigner une adresse IP publique à chaque machine virtuelle.
- Surveillez les machines virtuelles et calculez leurs performances et leur consommation. En fonction de ces calculs, déterminerez si vous souhaitez redimensionner horizontalement ou verticalement vos machines virtuelles. Vous pouvez par exemple décider de réduire la taille ou le nombre de vos machines virtuelles.

Si vous appliquez ces bonnes pratiques lorsque vous concevez vos solutions, vous pourrez réaliser des économies. Maintenant que nous avons couvert le calcul, nous allons adopter une approche similaire pour le stockage.

Bonnes pratiques en termes de stockage

Étant donné que nous hébergeons nos applications dans le Cloud, le stockage Azure sera utilisé pour stocker les données relatives à ces applications. Si nous ne suivons pas les bonnes pratiques, nous risquons de rencontrer des problèmes. Voici certaines bonnes pratiques en matière de stockage :

- Sélectionnez le type de redondance de stockage approprié (GRS, LRS, RA-GRS). Le type GRS est plus onéreux que le type LRS.

- Archivez les données de stockage au niveau froid ou archives. Conservez les données fréquemment consultées dans le niveau chaud.
- Retirez les objets blob qui ne sont pas nécessaires.
- Supprimez les disques du système d'exploitation de la machine virtuelle de manière explicite après avoir supprimé les machines virtuelles si celles-ci ne sont pas nécessaires.
- Les comptes de stockage doivent être mesurés en fonction de leur taille et des opérations de lecture, d'écriture, de création de liste et de conteneur.
- Privilégiez les disques standard et non les disques premium. Utilisez des disques premium uniquement si votre entreprise en a besoin.
- Utilisez CDN et la mise en cache pour les fichiers statiques au lieu de les récupérer chaque fois dans le stockage.
- Azure offre une capacité réservée pour réaliser des économies sur les données blob.

Appliquez ces bonnes pratiques afin de concevoir des solutions de stockage rentables. La section suivante porte sur les bonnes pratiques en matière de déploiement de services PaaS.

Bonnes pratiques relatives au PaaS

Azure propose de nombreux services PaaS. Si ceux-ci sont mal configurés, vous risquez de devoir payer des frais imprévus. Pour éviter ce scénario, appliquez les bonnes pratiques suivantes :

- Sélectionnez un niveau Azure SQL approprié (basique, standard, premium, RS ou premium) ainsi que les niveaux de performance adaptés.
- Faites votre choix entre une base de données unique et une base de données élastique. S'il existe de nombreuses bases de données, il sera plus rentable d'utiliser des bases de données élastiques.
- Repensez l'architecture de vos solution afin d'utiliser des solutions PaaS (Serverless ou microservices dans les conteneurs) au lieu de solutions IaaS. Ces solutions PaaS permettent de réaliser des économies en termes de maintenance et proposent une consommation à la minute. Si vous ne consommez pas ces services, aucun coût n'est applicable, même si votre code et vos services sont disponibles en permanence.

Il existe également des optimisations des coûts spécifiques aux ressources et il n'est pas possible de toutes les aborder dans un seul chapitre. Vous êtes invité à lire la documentation relative aux coûts de chaque fonctionnalité et à leur utilisation.

Bonnes pratiques générales

Jusqu'à présent, nous avons examiné les bonnes pratiques spécifiques au service, et nous allons clore cette section en formulant quelques directives générales :

- Le coût des ressources diffère d'une région à une autre. Essayez une autre région, à condition que cela ne nuise pas aux performances ou à la latence.
- Les contrats d'entreprise proposent des remises plus importantes que les autres offres. Contactez l'équipe de compte Microsoft afin de découvrir les avantages offerts par un contrat d'entreprise.
- Si les coûts Azure peuvent être prépayés, des remises sont alors offertes pour tous types d'abonnement.
- Supprimez ou retirez toute ressource non utilisée. Déterminez quelles ressources sont sous-utilisées et réduisez leur référence ainsi que leur taille. Si elles ne sont pas nécessaires, supprimez-les.
- Faites appel à un conseiller Azure et tenez compte de ses recommandations.

Comme nous l'avons mentionné plus tôt, il s'agit de quelques directives génériques. À mesure que vous concevrez des solutions, vous établirez vos propres bonnes pratiques. En attendant, appliquez celles-ci. Chaque composant d'Azure est associé à ses propres bonnes pratiques. Il est judicieux de consulter la documentation lorsque vous concevez vos solutions car cela vous permettra de créer des solutions rentables.

Résumé

Dans ce chapitre, nous avons appris l'importance de la gestion des coûts et de l'administration dans un environnement Cloud. Nous avons également abordé les différentes options de tarification et les diverses fonctionnalités d'optimisation de prix proposées par Azure La gestion des coûts d'un projet est primordiale principalement parce que les dépenses mensuelles peuvent être très faibles, mais sont susceptibles d'augmenter si les ressources ne sont pas surveillées périodiquement. Les architectes du Cloud doivent concevoir leur application d'une manière rentable. Ils doivent utiliser les ressources Azure, les références, les niveaux et les tailles appropriés et savoir à quel moment démarrer, interrompre, redimensionner, transférer les données, etc. Une gestion des coûts appropriée permettra de s'assurer que les frais réguliers sont conformes aux budgets.

Le chapitre suivant porte sur différentes fonctions Azure associées aux services de données ,notamment Azure SQL, Cosmos DB et le partitionnement.

7

Solutions OLTP Azure

Azure fournit des services **Infrastructure en tant que Service** (**IaaS**) et **Plateforme en tant que Service** (**PaaS**). Ces types de services fournissent aux organisations des niveaux et des contrôles différents sur le stockage, le calcul et les réseaux. Le stockage est la ressource utilisée lors de l'utilisation du stockage et de la transmission des données. Azure offre de nombreuses options de stockage des données, telles que les objets Blob de stockage Azure, les tables, Cosmos DB, Azure SQL Database, Azure Data Lake et bien plus encore. Bien que certains de ces options soient destinées au stockage Big Data, à l'analytique et à la présentation, d'autres sont destinées aux applications qui traitent les transactions. Azure SQL est la ressource principale dans Azure qui fonctionne avec les données transactionnelles.

Ce chapitre se concentrera sur divers aspects de l'utilisation des banques de données transactionnelles, tels qu'Azure SQL Database et d'autres bases de données open source, généralement utilisés dans les systèmes de **traitement transactionnel en ligne** (**OLTP**), et couvriront les sujets suivants :

- Applications OLTP
- Bases de données relationnelles
- Modèles de déploiement
- Azure SQL Database
- Instance unique
- Pools élastiques
- Instance gérée
- Cosmos DB

Nous allons commencer ce chapitre en examinant les applications OLTP et en répertoriant les services OLTP d'Azure, ainsi que leurs cas d'utilisation.

Applications OLTP

Comme nous l'avons mentionné précédemment, les applications OLTP sont des applications qui aident au traitement et à la gestion des transactions. Voici quelques-unes des mises en œuvre OLTP les plus répandues dans les ventes de distribution, les systèmes de transactions financières et la saisie de commandes. Ces applications effectuent la capture, le traitement, la récupération, la modification et le stockage des données. Cependant, cela n'est pas tout. Les applications OLTP traitent ces tâches de données comme des transactions. Les transactions ont quelques propriétés importantes et les applications OLTP tiennent compte de ces propriétés. Ces propriétés sont regroupées sous l'acronyme **ACID**. Discutons de ces propriétés en détail :

- **Atomicité** : cette propriété indique qu'une transaction doit consister en des instructions et que toutes ces instructions doivent se terminer avec succès ou aucune d'entre elles. Si plusieurs instructions sont regroupées, elles forment une transaction. Atomicité signifie que chaque transaction est traitée comme l'unité d'exécution unique la plus basse, qui soit se termine avec succès, soit échoue.
- **Cohérence** : cette propriété se concentre sur l'état des données dans une base de données. Elle stipule que tout changement d'état devrait être complet et basé sur les règles et les contraintes de la base de données, et que les mises à jour partielles ne devraient pas être autorisées.

- **Isolation** : cette propriété indique qu'il peut y avoir plusieurs transactions simultanées exécutées sur un système et que chaque transaction doit être traitée isolément. Une transaction ne doit pas avoir connaissance ou interférer avec une autre transaction. Si les transactions doivent être exécutées en séquence, à la fin, l'état des données devrait être le même qu'avant.
- **Durabilité** : cette propriété indique que les données doivent être rendues persistantes et disponibles, même après l'échec, une fois qu'elles sont validées dans la base de données. Une transaction engagée devient un fait.

Vous savez désormais en quoi consistent les applications OLTP, le moment est donc venu d'évoquer le rôle des bases de données relationnelles dans les applications OLTP.

Bases de données relationnelles

Les applications OLTP s'appuient généralement sur des bases de données relationnelles pour la gestion et le traitement des transactions. Les bases de données relationnelles proviennent généralement d'un format tabulaire composé de lignes et de colonnes. Le modèle de données est converti en plusieurs tables où chaque table est connectée à une autre table (basée sur des règles) à l'aide de relations. Ce processus est également désigné sous le terme de normalisation.

Plusieurs services Azure prennent en charge les applications OLTP et le déploiement des bases de données relationnelles. La section suivante porte sur les services Azure qui sont reliés aux applications OLTP.

Services Cloud Azure

Si vous recherchez **sql** dans le portail Azure, vous obtiendrez plusieurs résultats. J'ai marqué certains d'entre eux pour montrer les ressources qui peuvent être utilisées directement pour les applications OLTP :

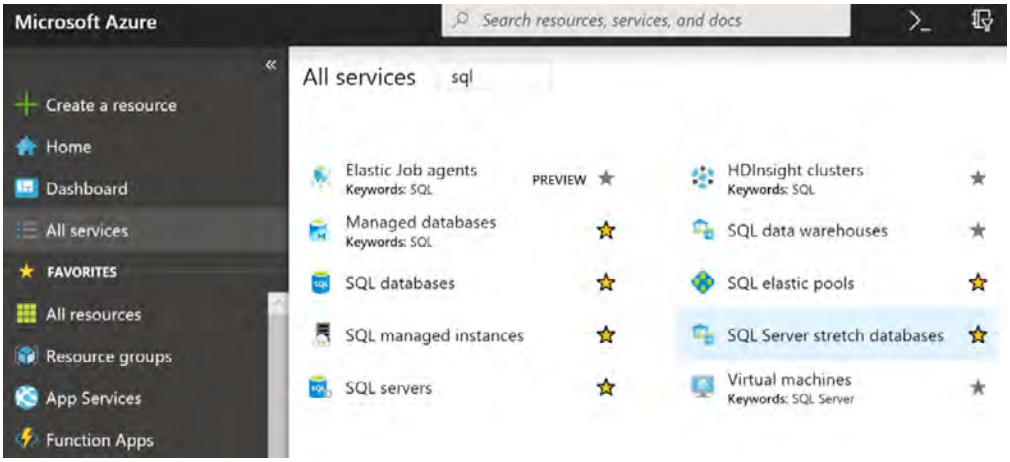


Figure 7.1 : liste des services Azure SQL

La Figure 7.1 montre les fonctions variées et les options disponibles pour créer des bases de données SQL Server sur Azure.

Encore une fois, une recherche rapide sur la **base de données** du portail Azure fournit plusieurs ressources, et celles marquées dans la *Figure 7.2* peuvent être utilisées pour les applications OLTP :

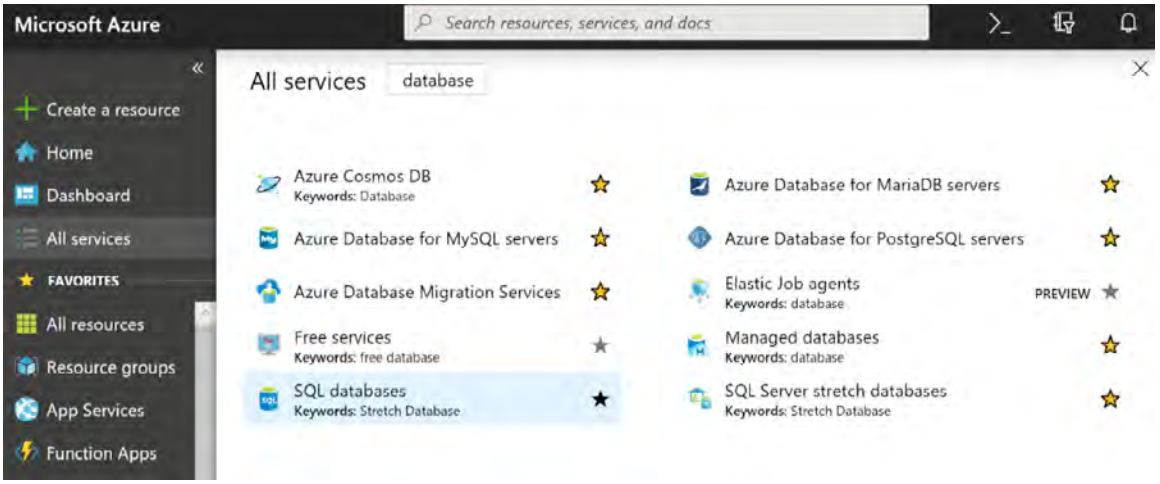


Figure 7.2 : liste des services Azure utilisés pour les applications OLTP

La Figure 7.2 montre les ressources fournies par Azure qui peuvent héberger des données dans plusieurs bases de données, notamment les suivantes :

- Base de données MySQL
- Bases de données MariaDB
- Bases de données PostgreSQL
- Cosmos DB

Intéressons-nous maintenant aux modèles de déploiement.

Modèles de déploiement

Les modèles de déploiement dans Azure sont classés en fonction du niveau de gestion ou de contrôle. C'est l'utilisateur qui sélectionne le niveau de gestion ou de contrôle qu'il préfère. Il peut opter pour un contrôle complet à l'aide de services tels que des machines virtuelles, ou utiliser des services gérés afin de déléguer la gestion des ressources à Azure.

Il existe deux modèles de déploiement pour le déploiement de bases de données sur Azure :

- Bases de données sur les machines virtuelles Azure (IaaS)
- Bases de données hébergées en tant que services gérés (PaaS)

Étudions à présent la différence entre le déploiement sur les machines virtuelles Azure et les instances gérées. Commençons par les machines virtuelles.

Bases de données sur les machines virtuelles Azure

Azure fournit plusieurs **références** (**SKU**) pour les machines virtuelles. Il existe des machines à haut débit (IOPS) dotées de capacités de calcul hautes performances qui sont également disponibles avec des machines virtuelles à usage général. Au lieu d'héberger un serveur SQL Server, MySQL ou toute autre base de données sur des serveurs locaux, il est possible de déployer ces bases de données sur ces machines virtuelles. Le déploiement et la configuration de ces bases de données ne sont pas différents des déploiements sur site. La seule différence réside dans le fait que la base de données est hébergée sur le Cloud plutôt que sur des serveurs locaux. Les administrateurs doivent effectuer les mêmes activités et les mêmes étapes qu'ils le feraient normalement pour le déploiement sur site. Bien que cette option soit idéale lorsque les clients souhaitent un contrôle total sur leur déploiement, il existe des modèles qui peuvent être plus rentables, évolutifs et hautement disponibles par rapport à cette option. Nous aborderons ces modèles plus en détail ultérieurement dans ce chapitre.

Les étapes pour déployer n'importe quelle base de données sur les machines virtuelles Azure sont les suivantes :

1. Créer une machine virtuelle dont la taille répond aux exigences de performances de l'application.
2. Déployer la base de données dessus.
3. Configurer la configuration de la machine virtuelle et de la base de données.

Cette option ne fournit pas de haute disponibilité par défaut, sauf si plusieurs serveurs sont provisionnés. Elle ne fournit pas non plus de fonctions pour la mise à l'échelle automatique, sauf si l'automatisation personnalisée la prend en charge.

La récupération après sinistre relève également de la responsabilité du client. Les serveurs doivent être déployés sur plusieurs régions connectées à l'aide de services tels que l'appairage mondial, les passerelles VPN, ExpressRoute ou Virtual WAN. Il est possible que ces machines virtuelles soient connectées à un datacenter local via des VPN de site à site ou ExpressRoute sans aucune exposition au monde extérieur.

Ces bases de données sont également appelées **bases de données non gérées**. En revanche, les bases de données hébergées avec Azure, autres que les machines virtuelles, sont gérées par Azure et sont connues sous le nom de **services gérés**. Nous allons aborder ces sujets en détail dans la section suivante.

Bases de données hébergées en tant que services gérés

Les services gérés signifient qu'Azure fournit des services de gestion pour les bases de données. Ces services gérés incluent l'hébergement de la base de données, ce qui garantit que l'hôte est hautement disponible, en veillant à ce que les données soient répliquées en interne en vue d'une disponibilité pendant la récupération d'urgence, pour assurer une évolutivité dans la contrainte d'une référence donnée, surveiller les hôtes et les bases de données et générer des alertes pour les notifications ou exécuter des actions, fournir des services de journalisation et d'audit pour le dépannage et prendre en charge la gestion des performances et les alertes de sécurité.

En bref, il existe beaucoup de services que les clients peuvent utiliser immédiatement lors de l'utilisation de services gérés d'Azure, sans avoir besoin d'effectuer une gestion active sur ces bases de données. Dans ce chapitre, nous allons examiner Azure SQL Database en détail et fournir des informations sur d'autres bases de données, telles que MySQL et Postgre. Nous allons également couvrir les bases de données non relationnelles telles que Cosmos DB, qui est une base de données NoSQL.

Azure SQL Database

Azure SQL Server fournit une base de données relationnelle hébergée en tant que PaaS. Les clients peuvent provisionner ce service, apporter leur propre schéma et données de base de données, et y connecter leurs applications. Cette solution fournit toutes les fonctions de SQL Server lors du déploiement sur une machine virtuelle. Ces services ne fournissent pas d'interface utilisateur pour créer des tables et son schéma, ni ne fournissent directement des fonctions d'interrogation. Vous devez utiliser SQL Server Management Studio et les outils de l'interface de ligne de commande SQL pour vous connecter à ces services et travailler directement avec eux.

Azure SQL Database est fourni avec trois modèles de déploiement distincts :

- **Instance unique** : dans ce modèle de déploiement, une base de données unique est déployée sur un serveur logique. Cela implique également la création de deux ressources sur Azure : un serveur logique SQL et une base de données SQL.
- **Pool élastique** : dans ce mode de déploiement, plusieurs bases de données sont déployées sur un serveur logique. Cela implique également la création de deux ressources sur Azure : un serveur logique SQL et un pool de base de données élastique SQL, qui contient toutes les bases de données.
- **Instance gérée :** il s'agit d'un modèle de déploiement relativement nouveau de l'équipe SQL Azure. Ce déploiement reflète une collection de bases de données sur un serveur logique fournissant un contrôle complet sur les ressources en termes de bases de données système. En règle générale, les bases de données système ne sont pas visibles dans d'autres modèles de déploiement, mais elles sont disponibles dans le modèle. Ce modèle est très proche du déploiement de SQL Server sur site :

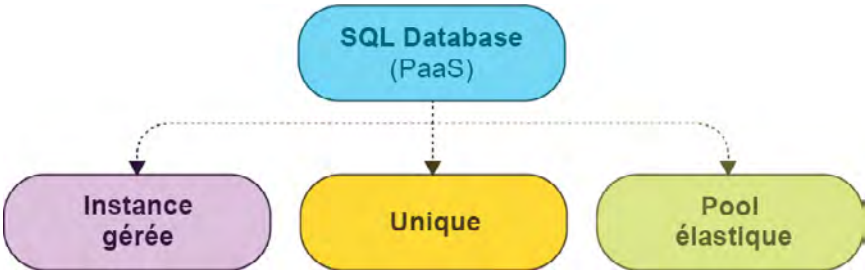


Figure 7.3 : modèles de déploiement Azure SQL Database

Si vous vous demandez quel modèle utiliser et quand l'utiliser, consultez la comparaison des fonctions entre SQL Database et SQL Managed instance. Une comparaison complète des fonctions est disponible à l'adresse https://docs.microsoft.com/azure/azure-sql/database/features-comparison.

Nous allons ensuite aborder certaines des fonctions de SQL Database. Commençons par les fonctions d'application.

Fonctions d'application

Azure SQL Database fournit plusieurs fonctions spécifiques à l'application qui répondent aux différentes exigences des systèmes OLTP :

- **Magasin colonnes** : cette fonction permet au stockage des données d'être dans un format colonnes plutôt que dans un format de ligne.
- **OLTP en mémoire** : en général, les données sont stockées dans les fichiers backend dans SQL et les données sont extraites chaque fois que l'application les requiert. Contrairement à cela, OLTP en mémoire met toutes les données en mémoire et il n'y a pas de latence dans la lecture du stockage pour les données. Le stockage des données OLTP en mémoire sur SSD offre les meilleures performances possibles pour Azure SQL.
- Toutes les fonctions de SQL Server sur site.

La prochaine fonction que nous allons aborder est la haute disponibilité.

Haute disponibilité

Azure SQL, par défaut, est hautement disponible à 99,99 %. Cette solution dispose de deux architectures différentes pour maintenir la haute disponibilité basée sur les références. Pour les références de base, standard et générale, l'ensemble de l'architecture est divisé dans les deux couches suivantes.

- Couche de calcul
- Couche de stockage

Il existe une redondance intégrée pour ces deux couches pour fournir une haute disponibilité :

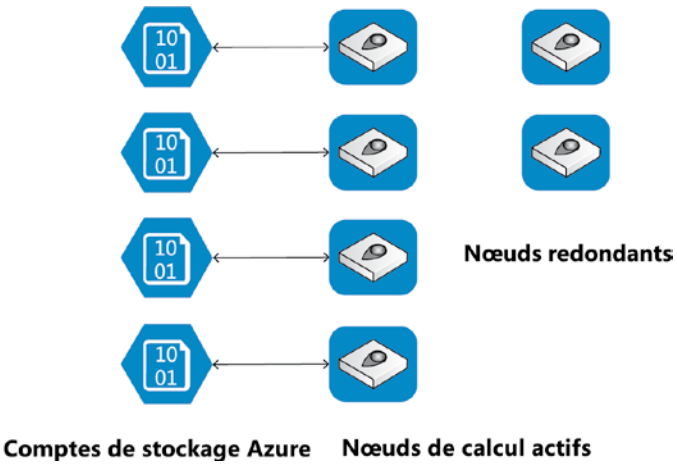


Figure 7.4 : couches de calcul et de stockage dans des références standard

Pour les références Premium et essentielles à l'entreprise, le calcul et le stockage se trouvent sur la même couche. La haute disponibilité est obtenue par réplication du calcul et du stockage déployés dans un cluster à quatre nœuds, à l'aide de la technologie similaire aux groupes de disponibilité AlwaysOn de SQL Server :

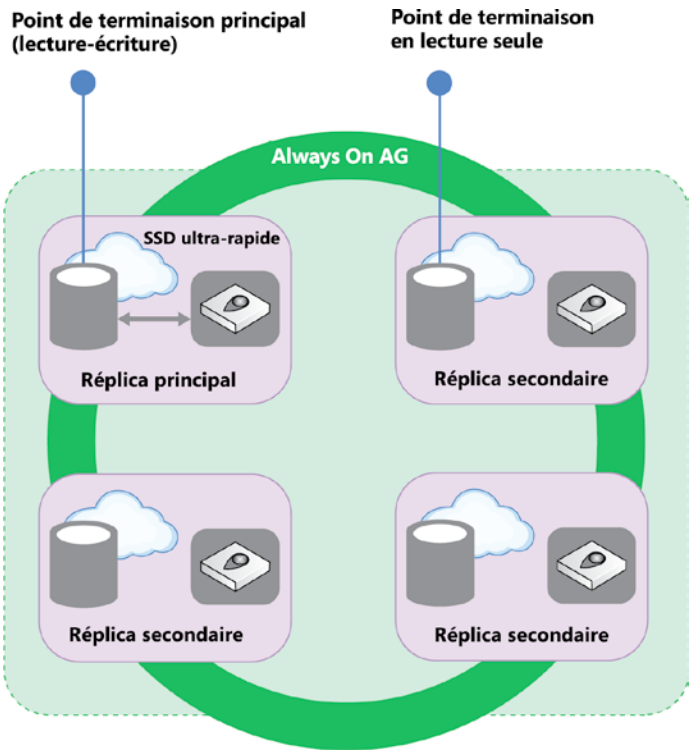


Figure 7.5 : déploiement d'un cluster à quatre nœuds

Maintenant que nous avons évoqué la gestion de la haute disponibilité, passons à la fonction suivante : les sauvegardes.

Sauvegardes

Azure SQL Database fournit également des fonctions permettant de sauvegarder automatiquement les bases de données et de les stocker sur des comptes de stockage. Cette fonction est particulièrement importante dans les cas où une base de données devient corrompue ou si un utilisateur supprime accidentellement une table. Cette fonction est disponible au niveau du serveur, comme illustré à la *Figure 7.6* :

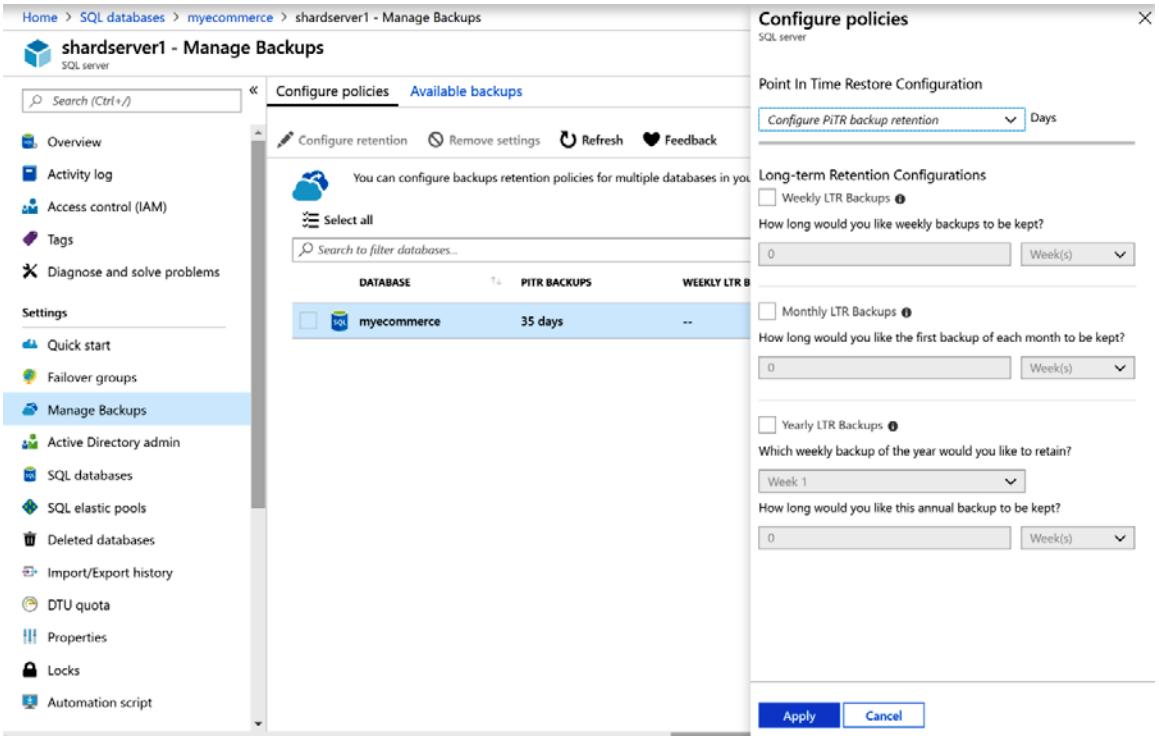


Figure 7.6 : sauvegarde de bases de données dans Azure

Les architectes doivent préparer une stratégie de sauvegarde afin que celles-ci puissent être utilisées en cas de besoin. Lors de la configuration des sauvegardes, assurez-vous que vos sauvegardes ne se produisent ni trop rarement, ni trop fréquemment. En fonction des besoins de l'entreprise, une sauvegarde hebdomadaire ou même une sauvegarde quotidienne doit être configurée, ou même plus fréquemment que cela, si nécessaire. Ces sauvegardes peuvent être utilisées à des fins de restauration.

Les sauvegardes contribuent à la continuité de service et à la récupération des données. Vous pouvez également opter pour la géoréplication pour récupérer les données lors d'une défaillance de région. La section suivante porte sur la géoréplication.

Géoréplication

Azure SQL Database offre également l'avantage de pouvoir répliquer une base de données vers une autre région, également appelée région secondaire. Cet avantage est entièrement basé sur le plan que vous avez sélectionné. La base de données de la région secondaire peut être lue par les applications. Azure SQL Database permet de fournir des bases de données secondaires lisibles. Il s'agit d'une excellente solution de continuité d'activité car une base de données lisible est disponible à tout moment. Avec la géoréplication, il est possible d'avoir jusqu'à quatre régions secondaires d'une base de données sur différentes régions ou dans la même région. Avec la géoréplication, il est également possible de basculer vers une base de données secondaire en cas de sinistre. La géoréplication est configurée au niveau de la base de données, comme illustré à la *Figure 7.7* :

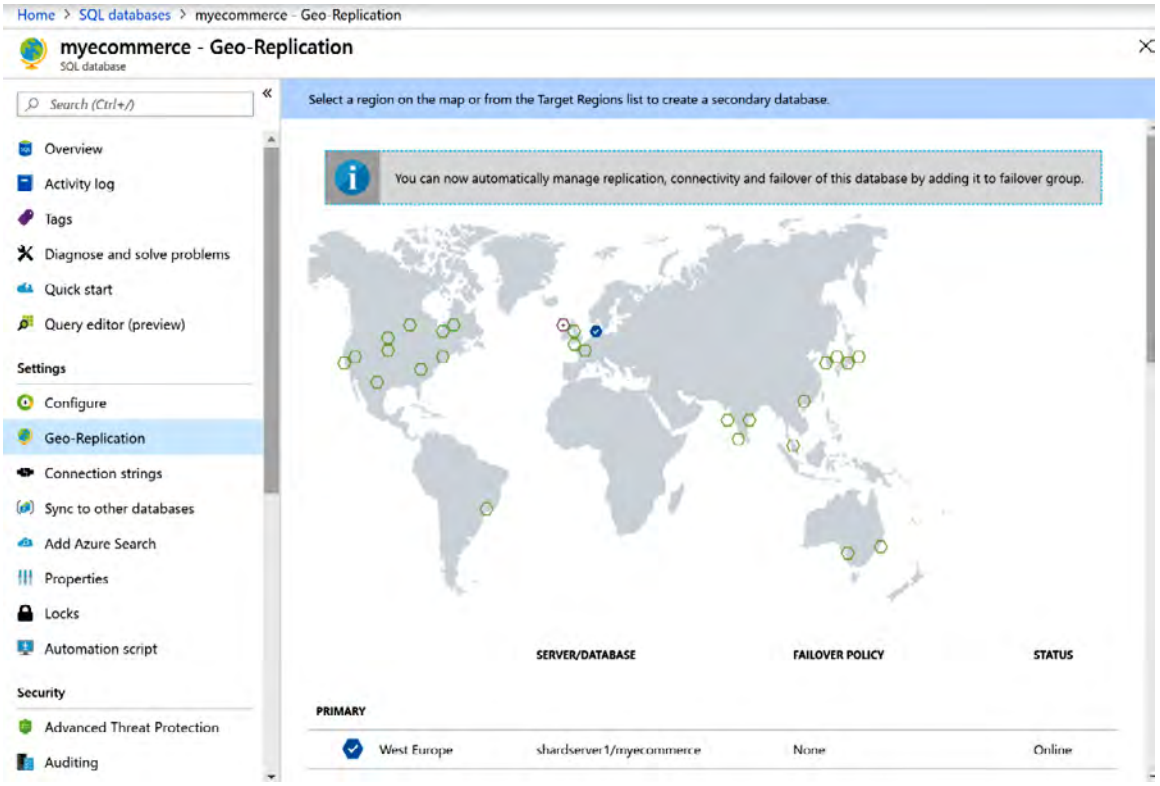


Figure 7.7 : géoréplication dans Azure

Si vous faites défiler cet écran vers le bas, les régions qui peuvent agir en tant que régions secondaires sont répertoriées, comme illustré à la *Figure* 7.8 :

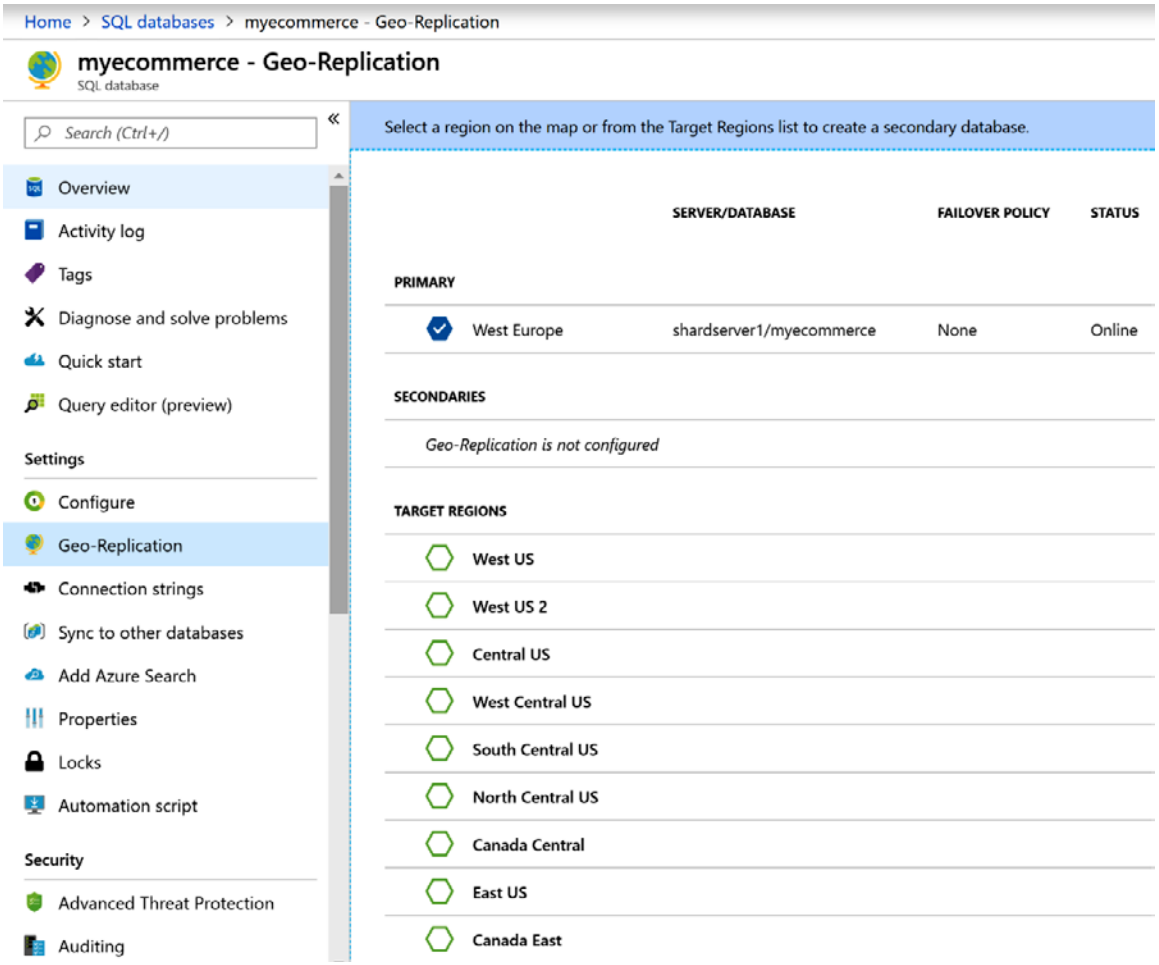


Figure 7.8 : liste des secondaires disponibles pour la géoréplication

Avant de concevoir des solutions qui impliquent la géoréplication, nous devons valider les réglementations en matière de résidence et de conformité des données. Si les données client ne sont pas autorisées à être stockées en dehors d'une région pour des raisons de conformité, nous ne devons pas les répliquer dans d'autres régions.

Nous découvrirons les options d'évolutivité dans la section suivante.

Évolutivité

Azure SQL Database fournit une évolutivité verticale en ajoutant plus de ressources (telles que le calcul, la mémoire et les IOPS). Pour ce faire, vous pouvez augmenter le nombre de **Database Throughput Units** (**DTU**) ou les ressources de calcul et de stockage dans le cas du modèle **vCore** :

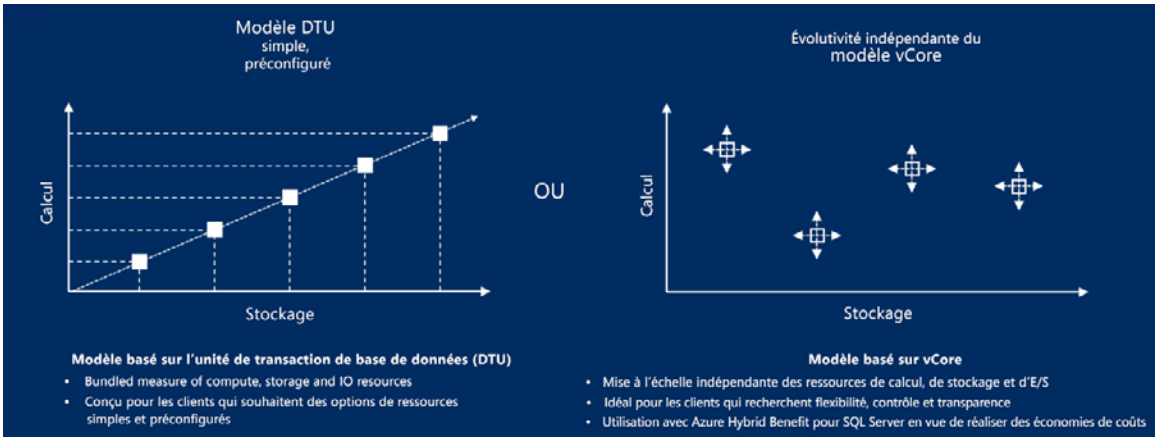


Figure 7.9 : évolutivité dans Azure SQL Database

Les différences entre le modèle basé sur les DTU et le modèle basé sur vCore seront abordées plus loin dans ce chapitre.

La section suivante porte sur la sécurité et vous permettra de comprendre comment créer des solutions de données sécurisées dans Azure.

Sécurité

La sécurité est un facteur important pour toute solution de base de données et service. Azure SQL fournit une sécurité de classe entreprise pour Azure SQL, et cette section répertorie certaines des fonctions de sécurité importantes dans Azure SQL.

Pare-feu

Azure SQL Database, par défaut, ne fournit pas d'accès à des demandes. Les adresses IP sources doivent être explicitement acceptées pour l'accès à SQL Server. Il existe une option permettant à tous les services Azure d'accéder à une base de données SQL également. Cette option inclut les machines virtuelles hébergées sur Azure.

Le pare-feu peut être configuré au niveau du serveur au lieu du niveau de la base de données. L'option Autoriser l'accès aux services Azure permet à tous les services, y compris les machines virtuelles, d'accéder à une base de données hébergée sur un serveur logique.

Par défaut, cette option est désactivée pour des raisons de sécurité. En l'activant vous pourrez accéder à tous les services Azure :

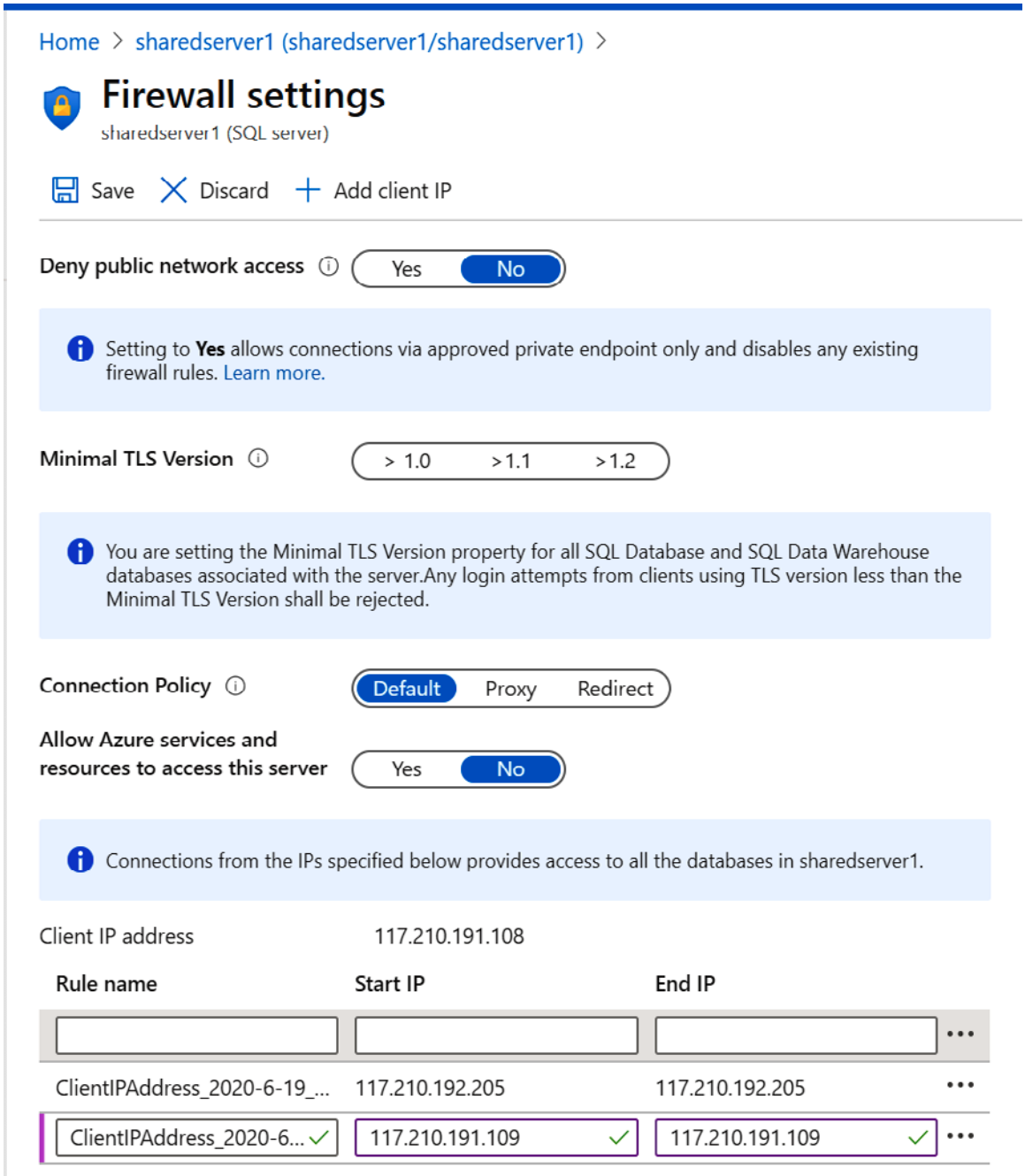


Figure 7.10 : configuration d'un pare-feu au niveau du serveur dans Azure

Azure SQL Server sur des réseaux dédiés

Bien que l'accès à SQL Server soit généralement disponible via Internet, il est possible que l'accès à SQL Server puisse être limité aux demandes provenant de réseaux virtuels. Il s'agit d'une fonction relativement nouvelle dans Azure. Cela permet d'accéder aux données dans SQL Server à partir d'une application sur un autre serveur du réseau virtuel sans que la demande ne passe par Internet.

Pour cela, un point de terminaison de type **Microsoft.Sql** doit être ajouté au sein du réseau virtuel, et ce dernier doit se trouver dans la même région que celle d'Azure SQL Database :

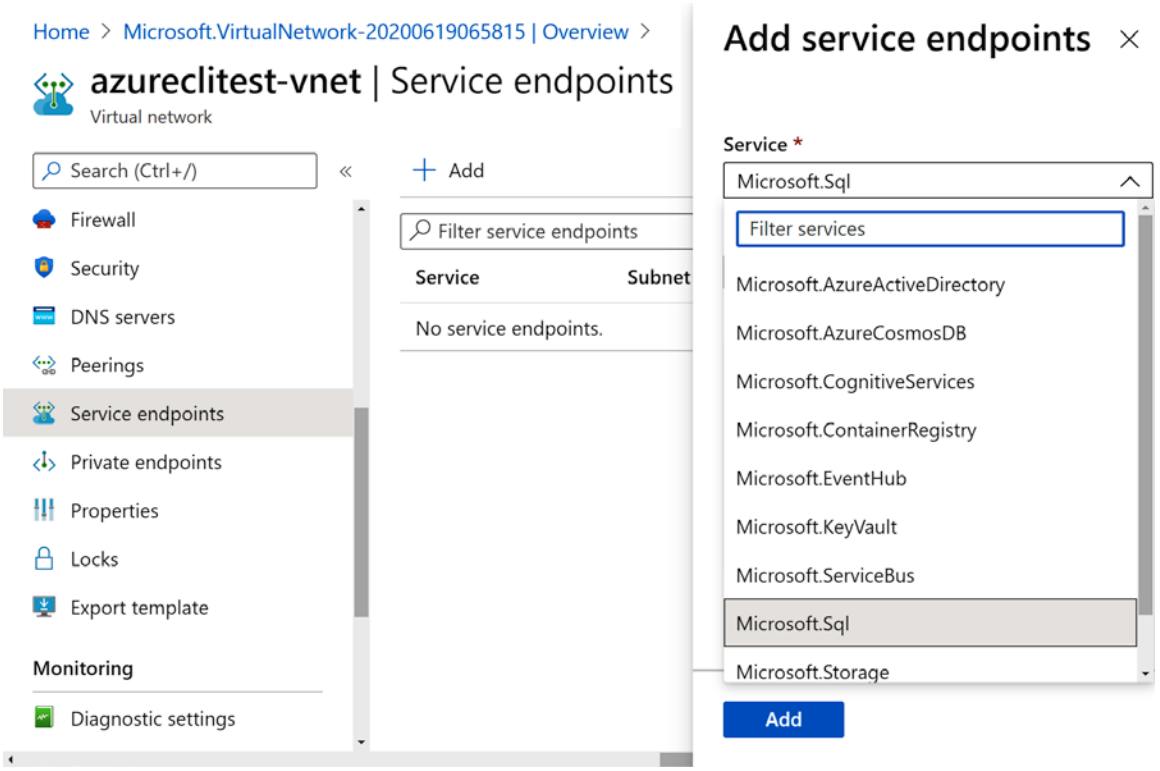


Figure 7.11 : ajout d'un point de terminaison de service Microsoft.Sql

Un sous-réseau approprié dans le réseau virtuel doit être choisi :

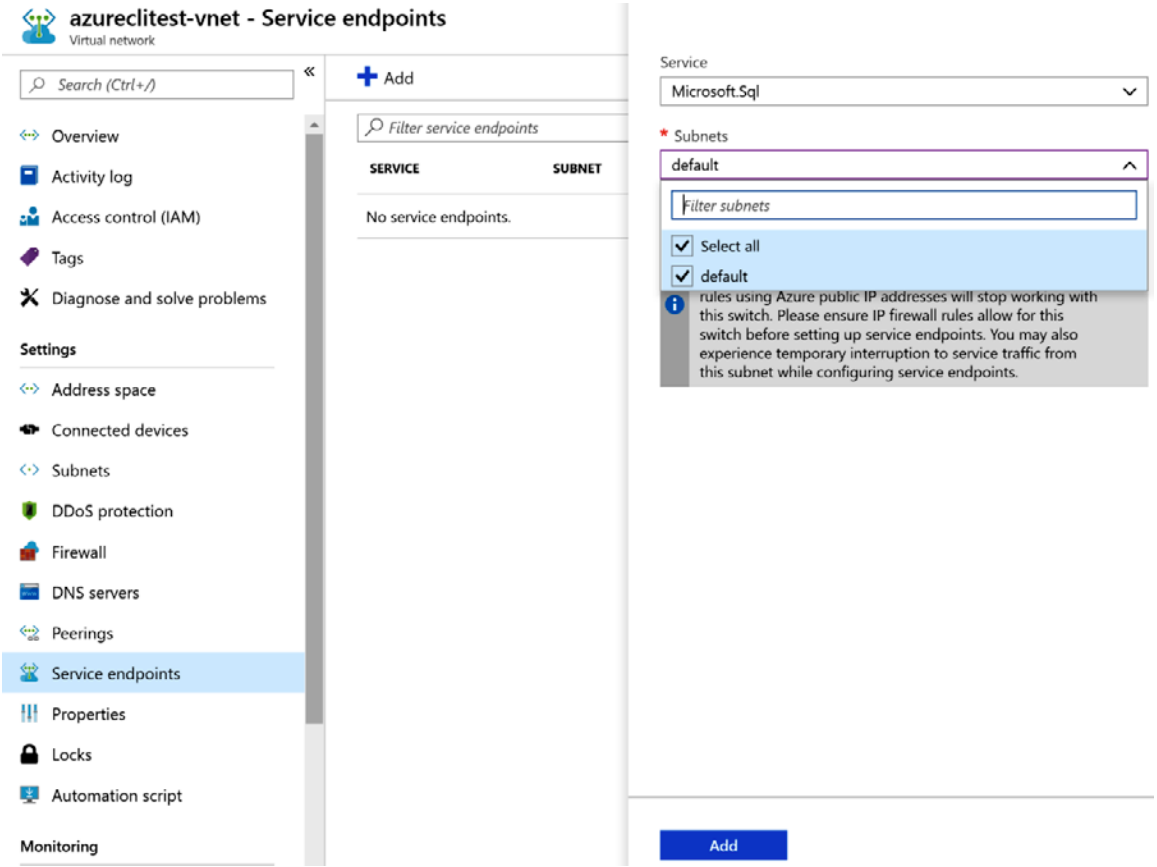


Figure 7.12 : choix d'un sous-réseau pour le service Microsoft.Sql

Enfin, à partir du panneau de configuration d'Azure SQL Server, un réseau virtuel existant doit être ajouté et un point de terminaison de service **Microsoft.Sql** est activé :

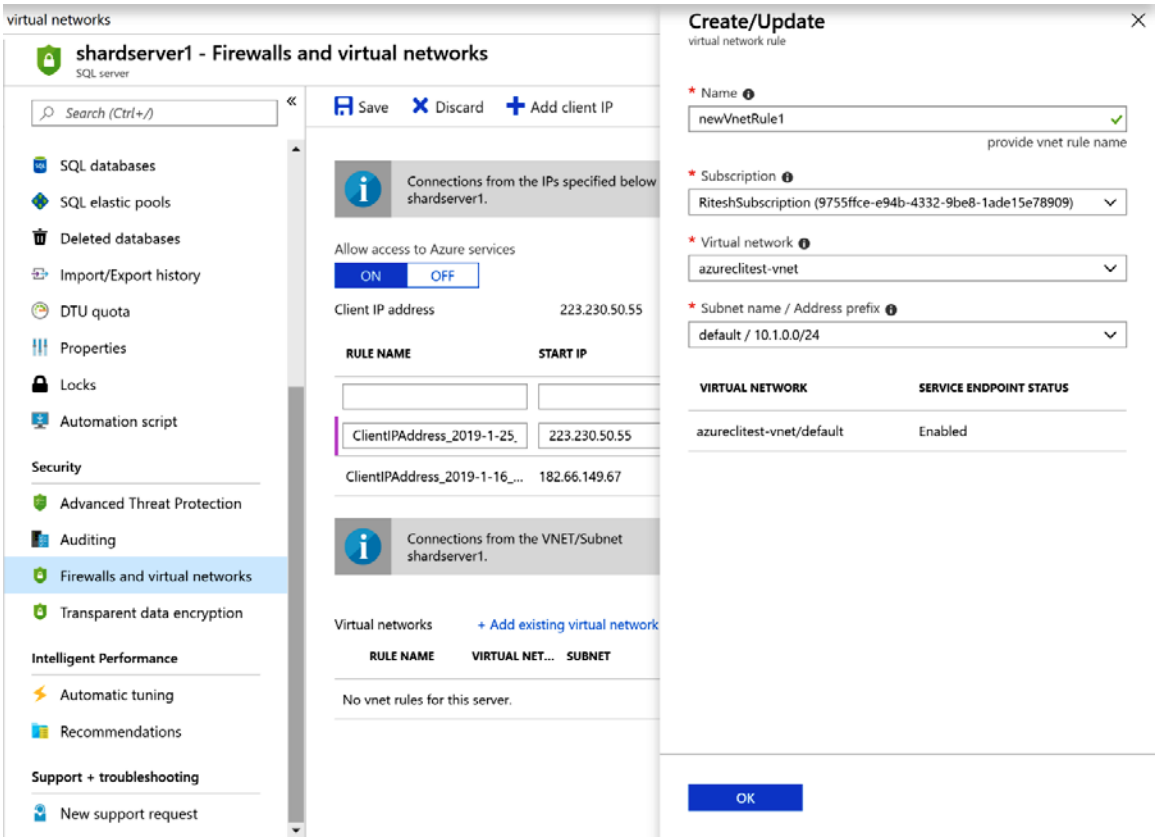


Figure 7.13 : ajout d'un réseau virtuel avec le point de terminaison de service Microsoft.Sql

Bases de données chiffrées au repos

Les bases de données doivent être sous une forme cryptée au repos. **Le terme au repos** signifie que les données se trouvent dans l'emplacement de stockage de la base de données. Bien que vous n'ayez peut-être pas accès à SQL Server et à sa base de données, il est préférable de chiffrer le stockage de la base de données.

Les bases de données sur un système de fichiers peuvent être chiffrées à l'aide de clés. Ces clés doivent être stockées dans Azure Key Vault et le coffre-fort doit être disponible dans la même région que celle d'Azure SQL Server. Le système de fichiers peut être chiffré à l'aide de l'élément de menu de **chiffrement de données transparent** du panneau de configuration SQL Server et en sélectionnant **Yes** (Oui) pour **Use your own key** (Utiliser votre propre clé).

La clé est une clé RSA 2048 et doit exister dans le coffre-fort. SQL Server déchiffrera les données au niveau de la page lorsqu'il souhaite les lire et les envoyer à l'appelant, puis il les chiffrera après avoir écrit dans la base de données. Il n'est pas nécessaire demodifier les applications, et celles-ci sont totalement transparentes :

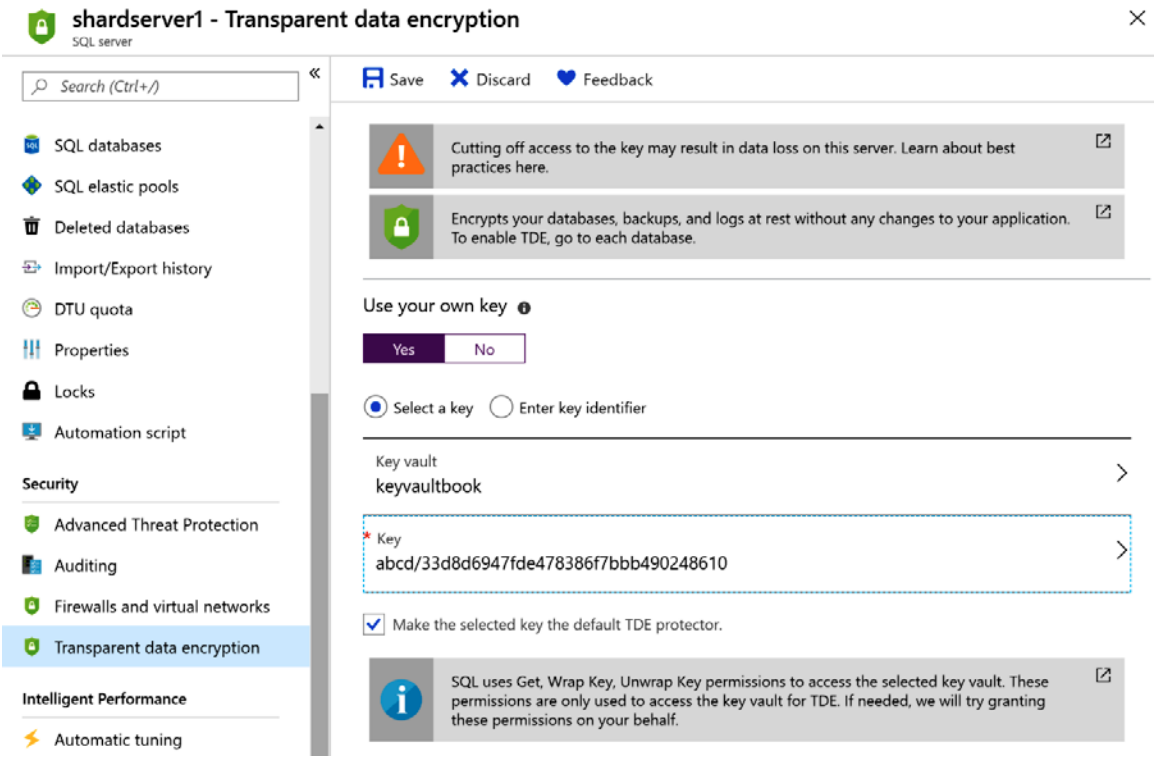


Figure 7.14 : chiffrement des données transparent dans SQL Server

Dynamic data masking (masquage dynamique des données)

SQL Server fournit également une fonction qui masque les colonnes individuelles qui contiennent des données sensibles afin que personne, à l'exception des utilisateurs privilégiés, ne puisse consulter les données réelles en les interrogeant dans SQL Server Management Studio. Les données resteront masquées et ne seront démasquées que lorsqu'une application ou un utilisateur autorisé interrogera la table. Les architectes doivent s'assurer que les données sensibles, telles que les informations de carte bancaire, les numéros de sécurité sociale, les numéros de téléphone, les adresses e-mail et d'autres détails financiers, sont masquées.

Les règles de masquage peuvent être définies sur la colonne d'une table. Il existe quatre principaux types de masques, vous pouvez les consulter ici https://docs.microsoft.com/sql/relational-databases/security/dynamic-data-masking?view=sql-server-ver15#defining-a-dynamic-data-mask.

La *Figure* 7.15 illustre l'ajout du masquage des données :

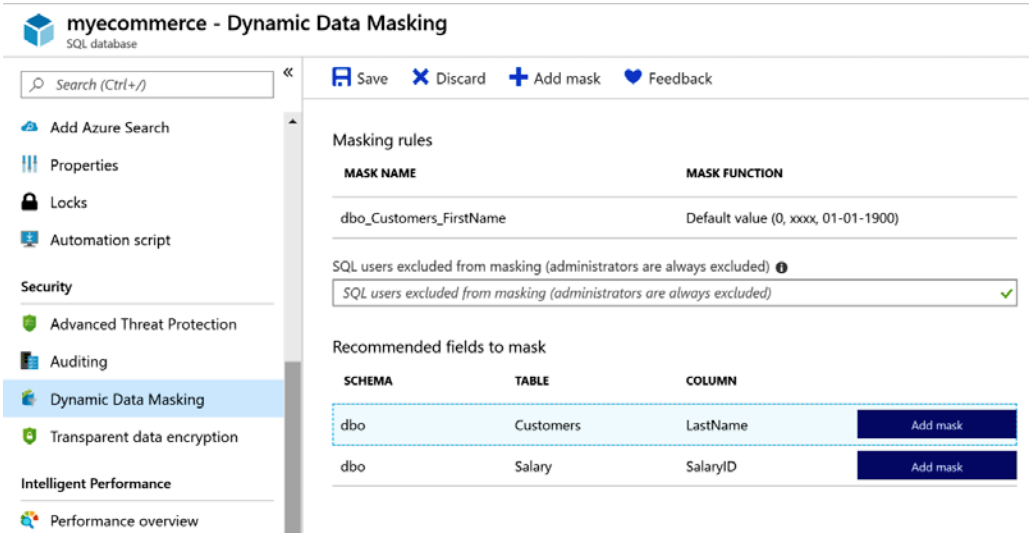


Figure 7.15 : masquage dynamique des données dans SQL Database

Intégration Azure Active Directory

L'intégration avec Azure **Active Directory** (**AD**) est une autre fonction importante d'Azure SQL, à des fins d'authentification. Sans intégration avec Azure AD, le seul mécanisme d'authentification disponible pour SQL Server est l'authentification par nom d'utilisateur et mot de passe ; c'est-à-dire l'authentification SQL. Il n'est pas possible d'utiliser l'authentification Windows intégrée. La chaîne de connexion pour l'authentification SQL se compose à la fois du nom d'utilisateur et du mot de passe en texte brut et elle n'est pas sécurisée. L'intégration à Azure AD permet d'authentifier les applications avec l'authentification Windows, par nom de principal de service ou par jeton. Il est recommandé d'utiliser Azure SQL Database intégré à Azure AD.

Il existe d'autres fonctions de sécurité, telles que la protection avancée contre les menaces, l'audit de l'environnement et la surveillance, qui doivent être activées sur tous les déploiements Azure SQL Database au niveau de l'entreprise.

C'est sur cette note que nous concluons notre découverte des fonctions d'Azure SQL Database. Intéressons-nous maintenant aux types de bases de données SQL.

Instance unique

Les bases de données à instance unique sont hébergées en tant que base de données unique sur un seul serveur logique. Ces bases de données n'ont pas accès aux fonctions complètes fournies par SQL Server. Chaque base de données est isolée et portable. Les instances uniques prennent en charge les modèles d'achat basés sur les vCPU et les DTU que nous avons abordés précédemment.

Une base de données unique présente un autre avantage : la rentabilité. Si vous utilisez un modèle basé sur vCore, vous pouvez opter pour des ressources de calcul et de stockage inférieures afin d'optimiser les coûts. Si vous avez besoin d'une puissance de calcul ou de stockage accrue, vous pouvez toujours effectuer une montée en charge. L'évolutivité dynamique est une caractéristique importante des instances uniques qui permet d'adapter les ressources de manière dynamique en fonction des besoins de l'entreprise. Les instances uniques permettent aux clients SQL Server existants de répliquer leurs applications sur site vers le Cloud.

Parmi les autres fonctions proposées figurent la disponibilité, la surveillance et la sécurité.

Nous avons commencé la section sur Azure SQL Database en mentionnant les pools élastiques. Vous pouvez également faire migrer une base de données unique vers un pool élastique pour partager les ressources. Si vous ignorez ce que sont le partage de ressources et les pools élastiques, ces sujets sont abordés dans la section suivante.

Pools élastiques

Un pool élastique est un conteneur logique qui peut héberger plusieurs bases de données sur un seul serveur logique. Les pools élastiques sont disponibles dans les modèles d'achat basés sur vCore et DTU. Le modèle d'achat basé sur les vCPU est la méthode de déploiement par défaut recommandée. Vous pourrez en effet choisir vos ressources de calcul et de stockage en fonction des charges de travail de votre entreprise. Comme illustré à la *Figure* 7.16 , vous pouvez sélectionner le nombre de cœurs et la quantité de stockage requis par votre base de données :

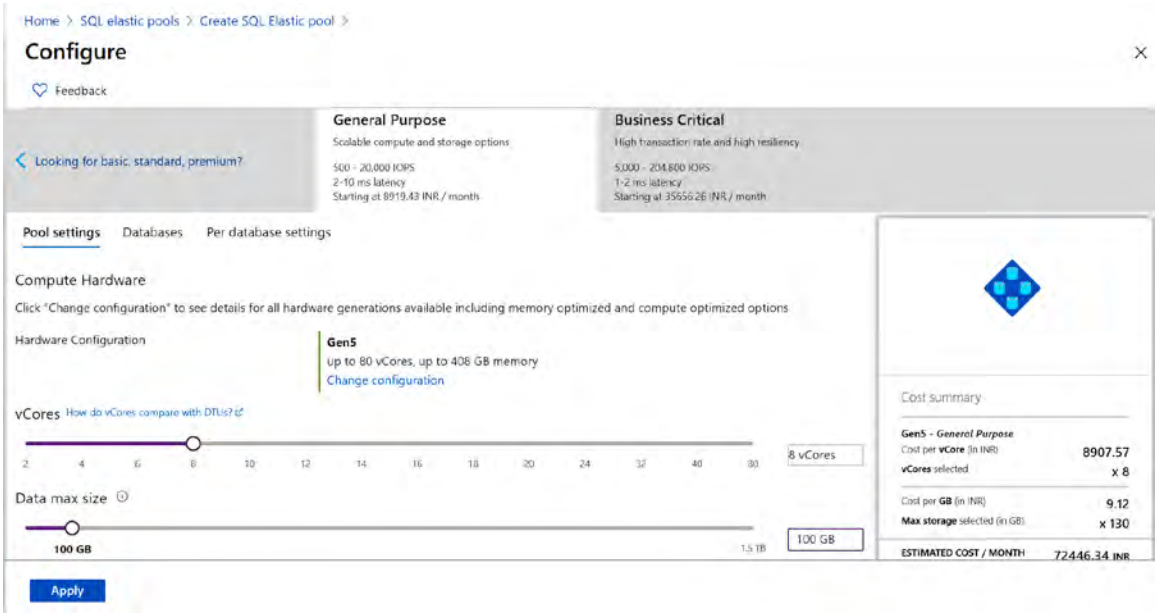


Figure 7.16 : configuration de pools élastiques dans le modèle basé sur vCore

La partie supérieure de la figure précédente montre une option indiquant **Vous recherchez une option basique, standard, premium ?** Si vous sélectionnez cette option, le modèle sera basculé vers le modèle DTU.

Les références disponibles pour les pools élastiques dans le modèle basé sur DTU sont les suivantes :

- Basique
- Standard
- Premium

La Figure 7.17 montre la quantité maximale de DTU pouvant être provisionnées pour chaque référence :

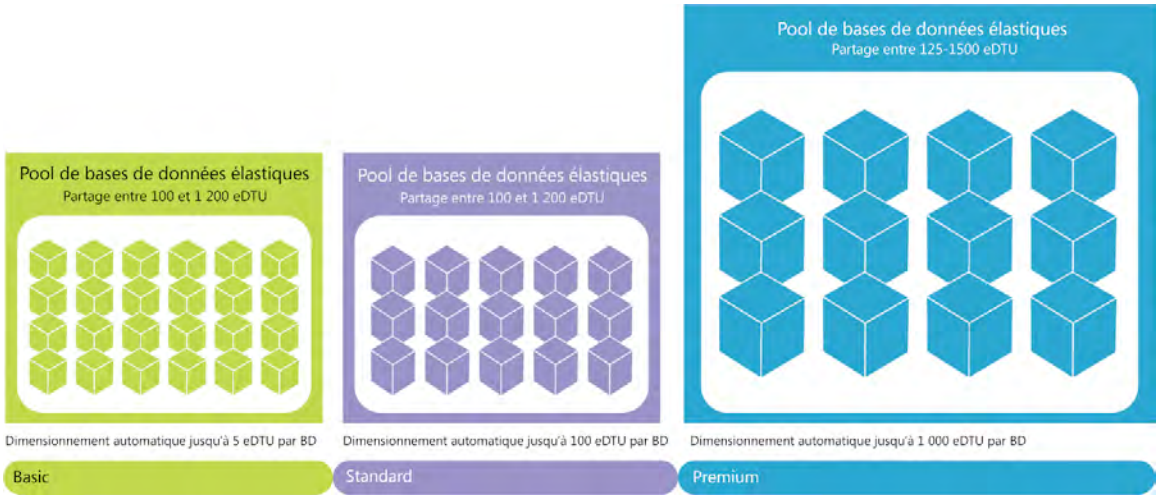


Figure 7.17 : nombre de DTU par référence dans un pool élastique

Toutes les fonctions abordées dans le cadre des instances uniques Azure SQL sont également disponibles pour les pools élastiques ; toutefois, l'évolutivité horizontale est une fonction supplémentaire qui assure le partitionnement. Le partitionnement fait référence au partitionnement vertical ou horizontal des données et au stockage de celles-ci dans des bases de donnée distinctes. Il est également possible de mettre en place une mise à l'échelle automatique des bases de données individuelles dans un pool élastique en consommant plus de DTU que ce qui est réellement alloué à cette base de données.

Les pools élastiques offrent également un autre avantage en termes de coût. Vous verrez dans une section ultérieure qu'Azure SQL Database est évalué à l'aide du concept de DTU, lesquels sont provisionnés dès que le service SQL Server est provisionné. Les DTU sont facturés, qu'ils soient consommés ou non. S'il existe plusieurs bases de données, alors il est possible de mettre ces bases de données dans des pools élastiques ; ils partageront les DTU entre eux.

Toutes les informations relatives à la mise en œuvre du partitionnement avec les pools élastiques d'Azure SQL sont disponibles à l'adresse https://docs.microsoft.com/azure/sql-database/sql-database-elastic-scale-introduction.

Nous allons ensuite discuter de l'option de déploiement d'une instance gérée, qui est une base de données évolutive, intelligente, basée sur le Cloud et entièrement gérée.

Instance gérée

Une instance gérée est un service unique qui fournit un serveur SQL géré similaire à ce qui est disponible sur les serveurs locaux. Les utilisateurs ont accès à des bases de données maître, modèle et à d'autres systèmes. L'utilisation d'instance gérée est idéale lorsque plusieurs bases de données et clients migrent leurs instances vers Azure. Les instances gérées se composent de plusieurs bases de données.

Azure SQL Database fournit un nouveau modèle de déploiement appelé instance gérée Azure SQL Database, qui offre une compatibilité quasiment à 100 % avec le moteur de base de données SQL Server Enterprise Edition. Ce modèle fournit une implémentation de réseau virtuel natif traitant des problèmes de sécurité courants et est un modèle d'entreprise hautement recommandé pour les clients SQL Server locaux. Une instance gérée permet aux clients SQL Server existants de lever et de déplacer leurs applications locales vers le Cloud en procédant à des modifications minimales sur l'application et la base de données, tout en préservant toutes les fonctions PaaS. Ces capacités de PaaS réduisent considérablement la surcharge de gestion et le coût total de propriété, comme illustré à la *Figure 7.18* :

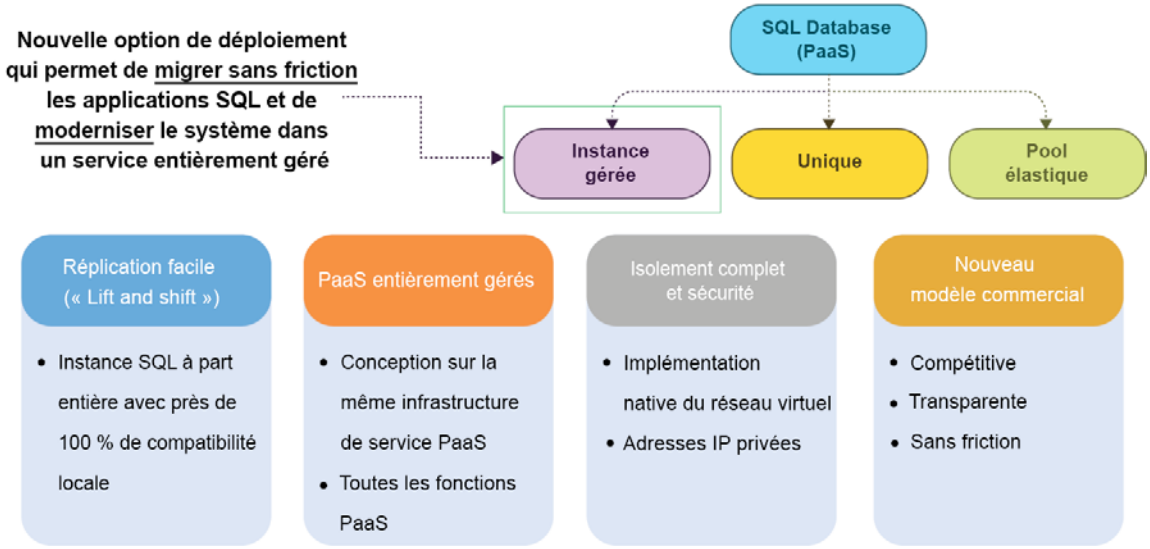


Figure 7.18 : Azure SQL Database Managed Instance

Une comparaison exhaustive entre Azure SQL Database, Azure SQL Managed Instance et SQL Server sur une machine virtuelle Azure est disponible à l'adresse : https://docs.microsoft.com/azure/azure-sql/azure-sql-iaas-vs-paas-what-is-overview#comparison-table.

Les principales fonctions des instances gérées sont affichées dans la *Figure 7.19* :

Feature	Description
SQL Server version / build	SQL Server Database Engine (latest stable)
Managed automated backups	Yes
Built-in instance and database monitoring and metrics	Yes
Automatic software patching	Yes
The latest Database Engine features	Yes
Number of data files (ROWS) per the database	Multiple
Number of log files (LOG) per database	1
VNet - Azure Resource Manager deployment	Yes
VNet - Classic deployment model	No
Portal support	Yes
Built-in Integration Service (SSIS)	No - SSIS is a part of Azure Data Factory PaaS
Built-in Analysis Service (SSAS)	No - SSAS is separate PaaS
Built-in Reporting Service (SSRS)	No - use Power BI or SSRS IaaS

Figure 7.19 : fonctions de SQL Database Managed Instance

Nous avons mentionné les modèles de tarification basé sur les vCPU et modèles de tarification basé sur les DTU à plusieurs reprises tout au long du chapitre. Il est temps d'examiner de plus près ces modèles de tarification.

Tarification de base de données SQL

Azure SQL ne proposait auparavant qu'un seul modèle de tarification : un modèle basé sur les unités de débit de base de données (Database Throughput Units, DTU). Toutefois, un modèle de tarification alternatif basé sur les vCPU a également été lancé. Le modèle de tarification est sélectionné en fonction des exigences du client. Le modèle basé sur DTU convient aux clients qui recherchent des options de ressources simples et préconfigurées. D'autre part, le modèle basé sur vCore offre la souplesse nécessaire pour choisir les ressources de calcul et de stockage. Il confère également contrôle et transparence.

Étudions chaque modèle d'un peu plus près.

Tarification selon les DTU

Le DTU est la plus petite unité de mesure de performance pour Azure SQL Database. Chaque DTU correspond à une certaine quantité de ressources. Ces ressources incluent le stockage, les cycles CPU, les IOPS et la bande passante réseau. Par exemple, un seul DTU peut fournir trois IOPS, quelques cycles CPU et des latences d'E-S de 5 ms pour les opérations de lecture et de 10 ms pour les opérations d'écriture.

Azure SQL Database fournit plusieurs références pour la création de bases de données, et chacune de ces références a défini des contraintes pour la quantité maximale de DTU. Par exemple, la référence de base fournit seulement **5** DTU avec un maximum de **2 Go** de données, comme illustré à la *Figure* 7.20 :

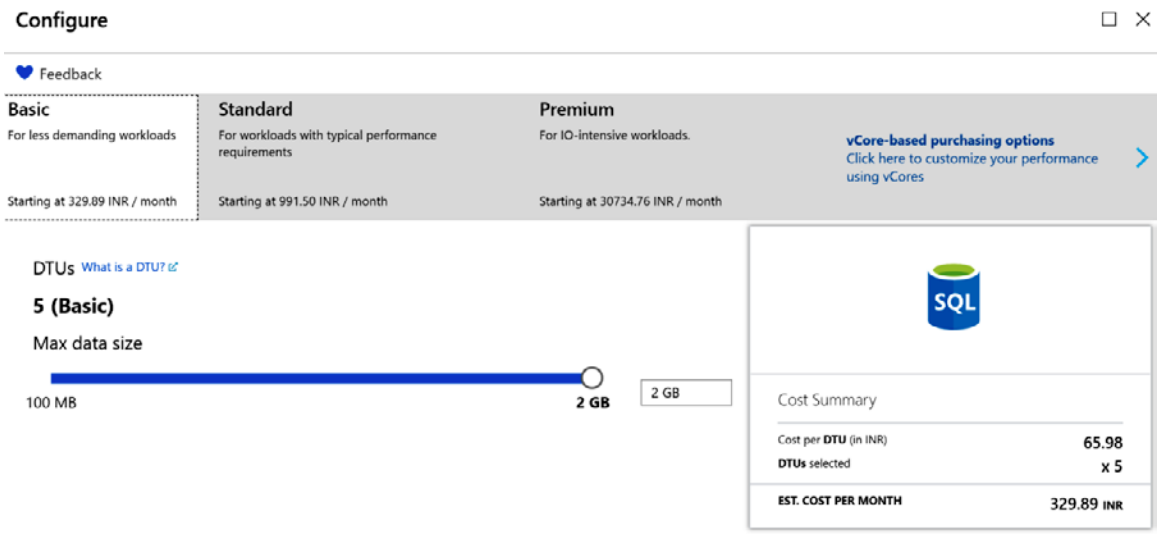


Figure 7.20 : DTU adaptés à différentes références

D'autre part, la référence standard fournit entre **10** DTUs et **300** DTUs avec un maximum de **250** Go de données. Comme vous pouvez le voir ici, chaque DTU coûte environ 991 roupies ou autour de 1,40 USD :

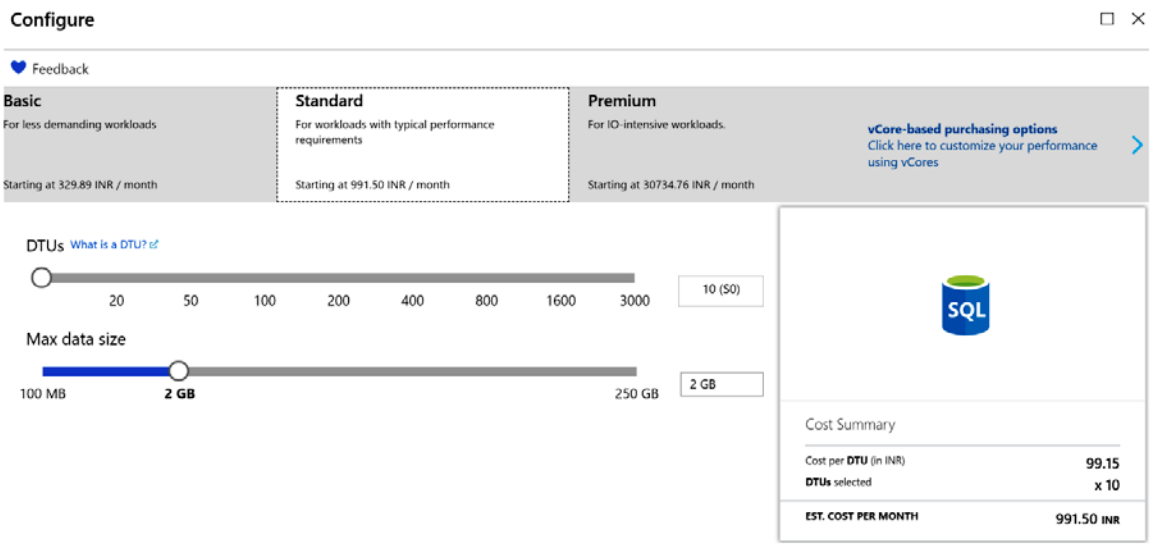


Figure 7.21 : résumé des coûts pour le nombre de DTU sélectionné dans la référence standard

Une comparaison entre ces références en termes de performances et de ressources est fournie par Microsoft et est illustrée à la *Figure* 7.22 :

	Basic	Standard	Premium
Target workload	Development and production	Development and production	Development and production
Uptime SLA	99.99%	99.99%	99.99%
Backup retention	7 days	35 days	35 days
CPU	Low	Low, Medium, High	Medium, High
IO throughput (approximate)	2.5 IOPS per DTU	2.5 IOPS per DTU	48 IOPS per DTU
IO latency (approximate)	5 ms (read), 10 ms (write)	5 ms (read), 10 ms (write)	2 ms (read/write)
Columnstore indexing	N/A	S3 and above	Supported
In-memory OLTP	N/A	N/A	Supported

Figure 7.22 : comparaison des références dans Azure

Une fois que vous provisionnez un certain nombre de DTU, les ressources backend (CPU, IOPS et mémoire) sont allouées et sont facturées, qu'elles aient été consommées ou non. Si plus de DTU que nécessaire ont été achetés, cela entraîne un gaspillage, tout comme un nombre insuffisant de DTU provisionnés nuit aux performances.

Azure fournit également des pools élastiques à cet effet. Comme vous le savez, il existe plusieurs bases de données dans un pool élastique et les DTU sont attribués à des pools élastiques au lieu de bases de données individuelles. Toutes les bases de données au sein d'un pool peuvent partager les DTU. Cela signifie que si une base de données est peu utilisée et consomme uniquement 5 DTU, une autre base de données consommera 25 DTUs pour compenser.

Il est important de noter que, collectivement, la consommation des DTU ne peut pas dépasser la quantité de DTU provisionnés pour le pool élastique. En outre, une quantité minimale de DTU doit être affectée à chaque base de données dans le pool élastique, et ce nombre minimum de DTU est pré-alloué pour la base de données.

Un pool élastique est livré avec ses propres SKU :

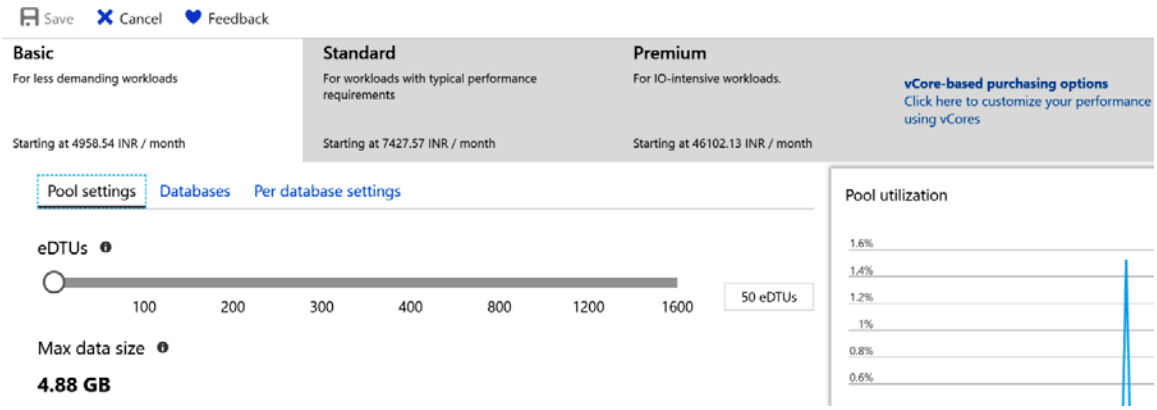


Figure 7.23 : références dans un pool élastique

En outre, il existe une limite concernant le nombre maximal de bases de données qui peuvent être créées dans un pool élastique unique. L'ensemble des limites est affiché ici : https://docs.microsoft.com/azure/azure-sql/database/resource-limits-dtu-elastic-pools.

Tarification selon les vCPU

Il s'agit du nouveau modèle de tarification pour Azure SQL. Ce modèle de tarification propose des options permettant de se procurer le nombre de **CPU virtuels (vCPU)** alloués pour le serveur au lieu de définir le nombre de DTU requis pour une application. Un vCPU est un CPU logique avec un matériel rattaché comme le stockage, la mémoire, et les cœurs de CPU.

Dans ce modèle, il existe trois références : **Usage général**, **Hyperscale** et **Essentiel à l'entreprise**, chacune présentant un nombre varié de vCPU et de ressources disponibles. Cette tarification est disponible pour tous les modèles de déploiement SQL :

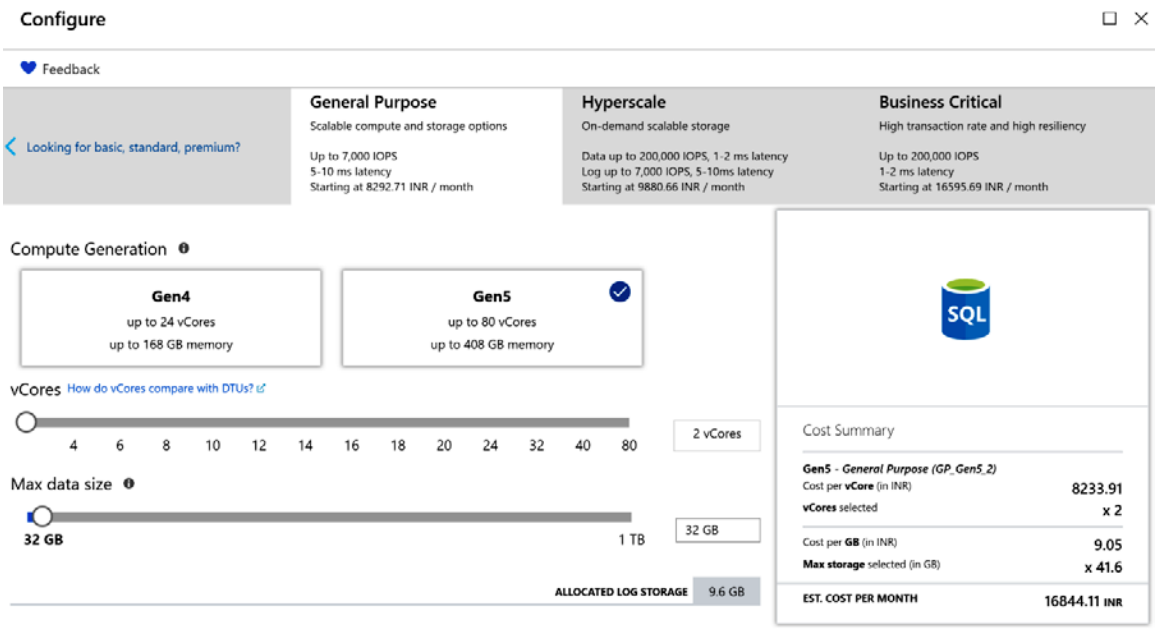


Figure 7.24 : tarification vCPU pour la référence à usage général

Comment choisir le modèle de tarification approprié

Les architectes devraient être en mesure de choisir un modèle de tarification approprié pour Azure SQL Database. Les DTU constituent un excellent mécanisme de tarification lorsqu'il existe un modèle d'utilisation applicable et disponible pour la base de données. Étant donné que la disponibilité des ressources dans le schéma des éléments DTU est linéaire, comme illustré dans le schéma suivant, il est tout à fait possible que les capacités de mémoire sont beaucoup plus intensément utilisées que celles du CPU. Dans ce cas, il est possible de choisir différents niveaux de CPU, de mémoire et de stockage pour une base de données.

Dans les DTU, les ressources sont emballées, et il n'est pas possible de les configurer à un niveau granulaire. Avec un modèle vCPU, il est possible de choisir différents niveaux de mémoire et de CPU pour différentes bases de données. Si le modèle d'utilisation d'une application est connu, l'utilisation du modèle de tarification vCPU pourrait être une option à privilégier par rapport au modèle DTU. En fait, le modèle vCPU offre également l'avantage des licences hybrides si une organisation possède déjà des licences SQL Server locales. Une remise allant jusqu'à 30 % s'applique à ces instances de SQL Server.

Dans la *Figure* 7.25, vous pouvez voir à partir du graphique de gauche qu'à mesure que la quantité de DTU augmente, la disponibilité des ressources augmente également de manière linéaire ; toutefois, avec la tarification vCPU (dans le graphique à droite), il est possible de choisir des configurations indépendantes pour chaque base de données :

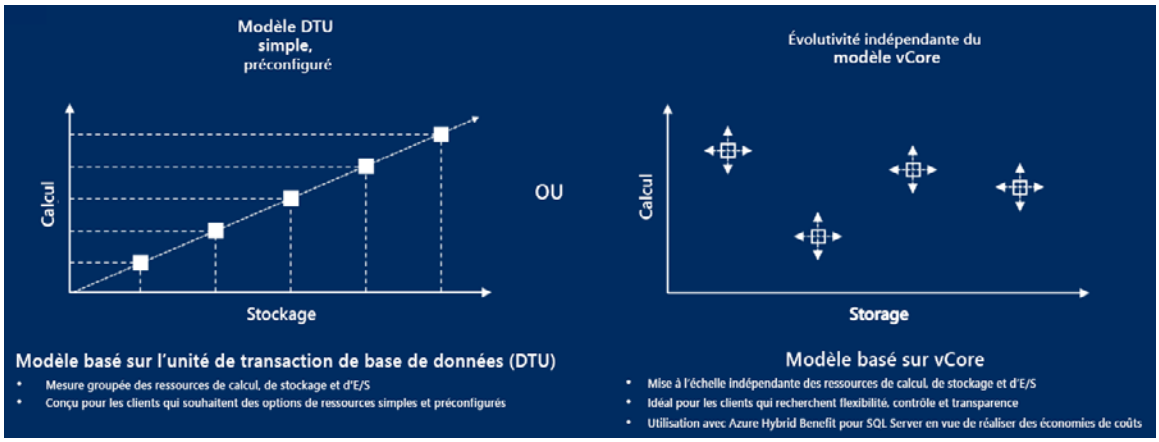


Figure 7.25 : graphique de calcul-stockage pour les modèles DTU et vCore

C'est sur cette note que nous concluons cette section sur Azure SQL Database. Nous avons discuté des différents plans, méthodes de déploiement, fonctionnalités et tarifs relatifs à Azure SQL Database. La section suivante porte sur Cosmos DB, qui est un service de base de données NoSQL.

Azure Cosmos DB

Cosmos DB est le service de base de données multi-modèles, distribué, inter-régions et hautement disponible d'Azure. Cosmos DB est fait pour vous si vous souhaitez que votre solution soit hautement réactive et toujours disponible. Étant donné qu'il s'agit d'une base de données multimodèle et inter-régions, nous pouvons déployer des applications plus près de l'emplacement de l'utilisateur, afin de garantir une faible latence et une haute disponibilité.

En cliquant sur un bouton, le débit et le stockage peuvent être mis à l'échelle dans n'importe quel nombre de régions Azure. Quelques modèles de base de données différents couvrent presque toutes les exigences de base de données non relationnelles, notamment :

1. SQL (documents)
2. MongoDB
3. Cassandra
4. Table
5. Graphique Gremlin

La hiérarchie des objets dans Cosmos DB commence par le compte Cosmos DB. Un compte peut présenter plusieurs bases de données, et chaque base de données peut avoir plusieurs conteneurs. Selon le type de base de données, le conteneur peut être constitué de documents, comme dans le cas de SQL : des données de valeur de clé semi-structurées dans le stockage en table, ou des entités et des relations entre ces entités, si vous utilisez Gremlin et Cassandra pour stocker des données NoSQL.

Cosmos DB peut aussi être utilisé pour stocker des données OLTP. Il tient compte des ACID en qui concerne les données de transaction, avec quelques mises en garde.

Cosmos DB fournit des exigences ACID au niveau du document unique. Par conséquent, les données contenues dans un document, lorsqu'elles sont mises à jour, supprimées ou insérées, seront maintenues à leur niveau d'atomicité, de cohérence, d'isolation et de durabilité. Toutefois, au-delà des documents, la cohérence et l'atomicité doivent être gérées par le développeur.

La tarification de Cosmos DB est disponible ici : https://azure.microsoft.com/pricing/details/cosmos-db.

La *Figure* 7.26 montre certaines fonctions d'Azure Cosmos DB :

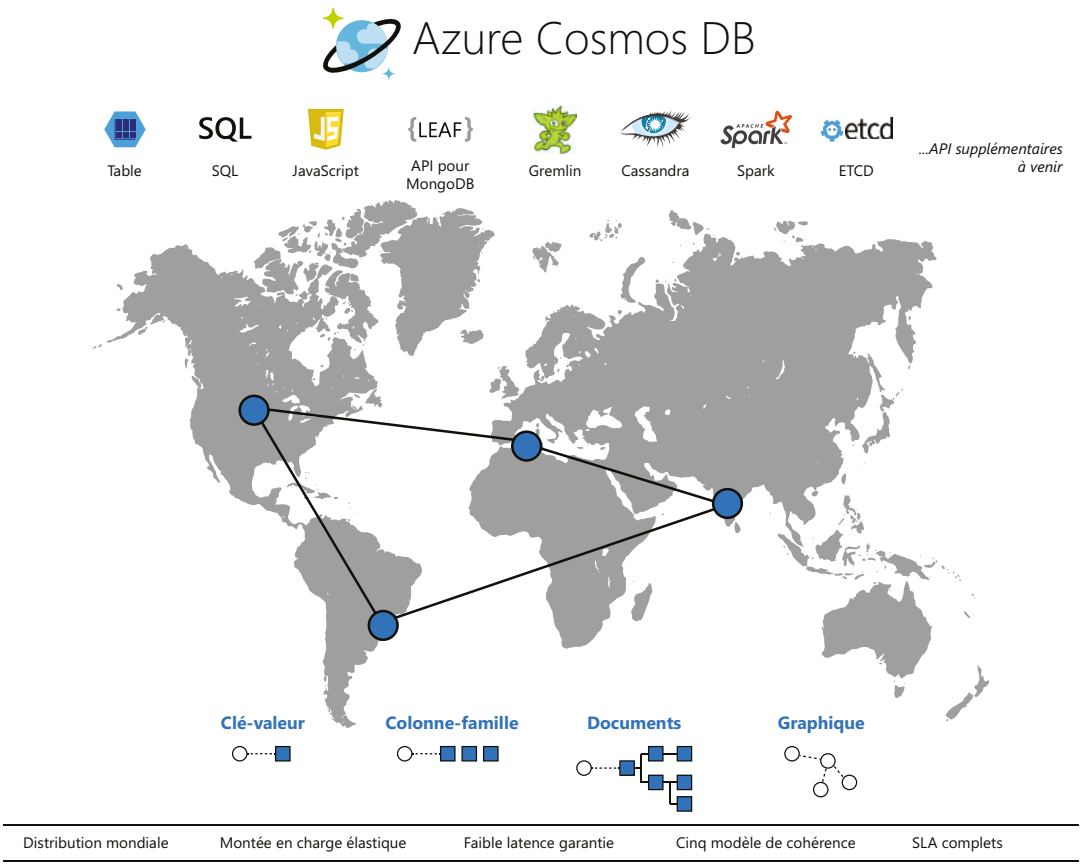


Figure 7.26 : une présentation d'Azure Cosmos DB

La section suivante porte sur certaines fonctions clés d'Azure Cosmos DB.

Fonctions

Voici certains des principaux avantages d'Azure Cosmos DB :

- **Distribution mondiale** : les applications hautement réactives et disponibles peuvent être développées dans le monde entier à l'aide d'Azure Cosmos DB. À l'aide de la réplication, les réplicas de données peuvent être stockés dans des régions Azure proches des utilisateurs afin de réduire la latence, tout en garantissant une distribution mondiale.
- **Réplication :** vous pouvez opter pour la réplication ou la refuser dans une région à votre guise. Imaginons que vous disposiez d'une réplique de vos données disponible dans la région est des États-Unis, et que votre organisation envisage d'interrompre ses processus dans l'est des États-Unis et de migrer vers le sud du Royaume-Uni. En quelques clics seulement, l'est des États-Unis peut être supprimé et le sud du Royaume-Uni peut être ajouté au compte à des fins de réplication.
- **Always on** : Cosmos DB garantit une haute disponibilité de 99,999 % pour la lecture et l'écriture. Le basculement régional d'un compte Cosmos DB vers une autre région peut être invoqué via le portail Azure ou par programme. La continuité de service et la planification de la récupération d'urgence sont ainsi garanties pour votre application lors d'une défaillance de région.
- **Évolutivité** : Cosmos DB offre une évolutivité élastique inégalée pour les écritures et les lectures dans le monde entier. La réponse à l'évolutivité est considérable. Vous pouvez donc évoluer et traiter des milliers, voire des centaines de millions de requêtes/seconde avec un seul appel d'API. Fait intéressant : vous pouvez le faire dans le monde entier et vous ne payer que le débit et le stockage. Ce niveau d'évolutivité est idéal pour gérer les pics inattendus.
- **Faible latence** : comme nous l'avons mentionné précédemment, la réplication des copies des données vers des emplacements plus proches des utilisateurs réduit considérablement la latence. Ainsi, les utilisateurs peuvent accéder à leurs données en quelques millisecondes. Cosmos DB garantit une latence inférieure à 10 ms pour les lectures et les écritures dans le monde entier.
- **Économies sur le coût total de possession** : Cosmos DB est un service entièrement géré, par conséquent, le niveau de gestion requis par le client est faible. En outre, le client n'a pas besoin de configurer de datacenters dans le monde entier pour héberger les utilisateurs d'autres régions.
- **SLA** : il offre un SLA de 99,999 % de haute disponibilité.
- **Prise en charge des API OSS (Open-Source Software)** : la prise en charge des API OSS est un autre avantage supplémentaire de Cosmos DB. Cosmos DB met en œuvre des API pour Cassandra, Mongo DB, Gremlin et le stockage de table Azure.

Scénarios de cas d'utilisation

Si votre application implique des niveaux élevés de lecture et d'écriture de données à l'échelle mondiale, Cosmos DB est le choix idéal. Les types d'applications courants qui présentent ces exigences incluent les applications Web, mobiles, de jeu et d'Internet des objets. Ces applications peuvent ainsi bénéficier d'une haute disponibilité, d'une faible latence et de la présence mondiale de Cosmos DB.

En outre, le temps de réponse fourni par Cosmos DB est proche du temps réel. Les SDK Cosmos DB peuvent être exploités pour développer des applications iOS et Android à l'aide de l'infrastructure Xamarin.

Quelques jeux populaires utilisent Cosmos DB, notamment **The Walking Dead : no Man's Land**, par Next Games, et **Halo 5 : Guardians**.

La liste complète des scénarios d'utilisation et des exemples est disponible ici : https://docs.microsoft.com/azure/cosmos-db/use-cases.

Cosmos DB est le service de prédilection d'Azure pour stocker des données semi-structurées dans le cadre d'applications OLTP. Je pourrais écrire tout un livre sur les fonctions de Cosmos DB. Toutefois, l'objectif de cette section consistait à vous présenter Cosmos DB, en vous expliquant son rôle dans la gestion des applications OLTP.

Résumé

Dans ce chapitre, vous avez appris qu'Azure SQL Database est l'un des services phares d'Azure. Une multitude de clients utilisent ce service aujourd'hui, lequel fournit toutes les capacités d'entreprise nécessaires à un système de gestion de base de données essentiel à l'entreprise.

Vous avez découvert qu'il existe plusieurs types de déploiement pour Azure SQL Database, tels que les instances uniques, les instances gérées et les pools élastiques. Les architectes doivent effectuer une évaluation complète de leurs besoins et choisir le modèle de déploiement approprié. Après avoir choisi un modèle de déploiement, ils doivent opter pour une stratégie de tarification basée sur les DTU et les vCPU. Ils doivent également configurer toutes les exigences de sécurité, de disponibilité, de récupération d'urgence, de surveillance, de performance et d'évolutivité dans Azure SQL Database relatives aux données.

Le chapitre suivant porte sur la sécurisation des applications dans Azure. Nous allons aborder les pratiques et les fonctionnalités de sécurité de la plupart des services.

8

Architecture d’applications sécurisées sur Azure

Au chapitre précédent, nous avons découvert les services de données Azure. Nous traitons des données sensibles, la sécurité est donc une préoccupation majeure. La sécurité est sans aucun doute l’exigence non fonctionnelle la plus importante à mettre en œuvre pour les architectes. Les entreprises s'efforcent de mettre en place une stratégie de sécurité de manière appropriée. En effet, la sécurité est l’une des plus grandes préoccupations de chaque partie prenante d’une application, au niveau de son développement, de son déploiement et de sa gestion. Cet enjeu est d’autant plus important lorsqu’une application est conçue en vue d’un déploiement dans le Cloud.

Afin que vous compreniez comment sécuriser vos applications sur Azure selon la nature du déploiement, les sujets suivants seront abordés dans ce chapitre :

- Présentation de la sécurité dans Azure
- Sécurité au niveau de l'infrastructure
- Sécurité au niveau de l'application
- Authentification et autorisation pour les applications Azure
- Travailler avec OAuth, Azure Active Directory et d'autres méthodes d'authentification à l'aide des identités fédérées, notamment les fournisseurs d'identité tiers tels que Facebook
- Comprendre les identités gérées et les utiliser pour accéder aux ressources

Sécurité

Nous l'avons déjà dit, la sécurité est un élément important pour tout logiciel ou service. L'application doit être pourvue d'une sécurité adaptée afin que seules les personnes habilitées à y accéder puissent l'utiliser et que les utilisateurs aient uniquement la possibilité d'effectuer des opérations autorisées. De même, l'ensemble du mécanisme de demande-réponse doit s'appuyer sur des méthodes garantissant que seules les parties prévues peuvent comprendre les messages, afin de détecter rapidement tout message frauduleux.

La sécurité dans Azure est d'autant plus importante, et ce pour les raisons suivantes. Tout d'abord, les organisations déployant leurs applications ne contrôlent pas entièrement les équipements et les réseaux sous-jacents. De plus, la sécurité doit être intégrée à chaque couche, y compris les équipements, les réseaux, les systèmes d'exploitation, les plateformes et les applications. Toute omission ou erreur de configuration peut rendre l'application vulnérable aux intrusions. Vous avez peut-être entendu parler de la vulnérabilité qui a récemment affecté les réunions Zoom, au cours desquelles les pirates ont pu enregistrer des réunions, même lorsque l'hôte de la réunion avait désactivé l'enregistrement pour les participants. Selon certaines sources, des millions de comptes Zoom ont été vendus sur le dark Web. L'entreprise a pris les mesures nécessaires pour remédier à cette vulnérabilité.

La sécurité est une préoccupation majeure de nos jours, en particulier lors de l'hébergement d'applications dans le Cloud, et peut entraîner des conséquences désastreuses si elle est mal gérée. Par conséquent, il est nécessaire de connaître les bonnes pratiques appliquées à la sécurisation de vos charges de travail. Nous progressons dans le domaine du DevOps, où les équipes de développement et des opérations collaborent efficacement à l'aide d'outils et de pratiques, en accordant la priorité à la sécurité.

Pour tenir compte des principes et des pratiques de sécurité et en faire une partie intégrante du DevOps sans affecter la productivité globale et l'efficacité du processus, une nouvelle culture connue sous le nom de **DevSecOps** a été introduite. DevSecOps nous aide à identifier les problèmes de sécurité dès le début de la phase de développement et non après la livraison, où ces problèmes doivent alors être corrigés. Dans un processus de développement qui intègre la sécurité à chaque phase, DevSecOps réduit le coût lié au recrutement des professionnels de la sécurité à un stade ultérieur pour trouver des failles de sécurité dans les logiciels.

La sécurisation d'une application revient à interdire son accès aux entités inconnues et non autorisées. Cela signifie également que la communication avec l'application est sécurisée et authentique. Cela inclut les mesures de sécurité suivantes :

- **Authentification** : l'authentification vérifie l'identité d'un utilisateur et s'assure que cette même identité peut accéder à l'application ou au service. L'authentification s'effectue dans Azure à l'aide d'OpenID Connect, qui est un protocole d'authentification basé sur OAuth 2.0.
- **Autorisation** : l'autorisation désigne le fait d'autoriser et d'attribuer des permissions à une identité au sein d'une application ou d'un service. L'autorisation s'effectue dans Azure à l'aide de la technologie oAuth.
- **Confidentialité** : la confidentialité garantit que la communication entre l'utilisateur et l'application reste sécurisée. L'échange de charge utile entre entités est chiffrée, de sorte à n'avoir un sens que pour l'expéditeur et le récepteur, personne d'autre. La confidentialité des messages s'effectue à l'aide d'un chiffrement symétrique et asymétrique. Des certificats sont utilisés pour mettre en œuvre la cryptographie, à savoir le chiffrement et le déchiffrement des messages.

  Le chiffrement symétrique utilise une clé unique, qui est communiquée à l'expéditeur et au récepteur, tandis que le chiffrement asymétrique utilise une paire de clés privée et publique pour le chiffrement. Cette dernière méthode est plus sécurisé. Les paires de clés SSH dans Linux, qui sont utilisées pour l'authentification, sont un très bon exemple du chiffrement asymétrique.
- **Intégrité** : l'intégrité veille à ce que la charge utile et l'échange de messages entre l'expéditeur et le récepteur demeurent authentiques. Le récepteur reçoit le même message envoyé par l'expéditeur. Les signatures numériques et le hachage se rapportent au mécanisme de mise en œuvre destiné à vérifier l'intégrité des messages entrants.

La sécurité est un partenariat entre le fournisseur de service et le consommateur. Les deux parties disposent de niveaux de contrôle différents sur des piles de déploiement entières et chacune doit mettre en œuvre de bonnes pratiques de sécurité pour s'assurer que toutes les menaces sont identifiées et sous contrôle. Dans le *chapitre 1*, *Prise en main d'Azure*, nous avons appris que le Cloud fournit trois paradigmes : IaaS, PaaS et SaaS, chacun disposant de niveaux de contrôle collaboratif différents sur la pile de déploiement. Chaque partie doit appliquer des pratiques de sécurité pour les composants qu'elle contrôle et qui sont à sa portée. En l'absence de mesures de sécurité sur les couches de la pile ou de la part de l'une des parties, l'intégralité du déploiement et de l'application sera vulnérable aux attaques. Chaque organisation doit disposer d'un modèle de cycle de vie pour la sécurité, tout comme pour tout autre processus. Ainsi, les pratiques de sécurité sont continuellement améliorées et permettent d'éviter toute faille de sécurité. Dans la section suivante, nous allons discuter du cycle de vie de la sécurité et de son utilisation.

Cycle de vie de la sécurité

La sécurité est souvent considérée comme une exigence non fonctionnelle pour une solution. Cependant, compte tenu du nombre croissant de cyber-attaques, celle-ci est devenue une exigence fonctionnelle pour chaque solution de nos jours.

Chaque organisation suit et gère un certain cycle de vie d'application. Lorsque la sécurité est traitée comme une nécessité fonctionnelle, elle doit suivre le même processus de développement d'applications. La sécurité ne doit pas constituer une préoccupation secondaire, elle doit au contraire faire partie intégrante de l'application dès le départ. La sécurité doit être également planifiée lors de la phase de planification globale d'une application. Selon la nature de l'application, différents types et catégories de menaces doivent être identifiés, puis documentés afin d'adopter une approche et de déterminer une portée appropriée pour s'en prémunir. Un exercice de modélisation des menaces est nécessaire pour illustrer toutes les menaces auxquelles chaque composant est exposé. Cela permettra par la suite de mettre en place des normes et des stratégies de sécurité pour cette application. Il s'agit généralement de la phase de conception de la sécurité. La phase suivante est dénommée Prévention des menaces ou phase de développement. Lors de cette phase, la sécurité est appliquée au code et à la configuration afin de minimiser toute menace et tout risque de sécurité.

Un système ne peut pas être sécurisé tant qu'il n'a pas été testé. Des tests de pénétration ainsi que d'autres tests de sécurité doivent être réalisés pour identifier les mesures d'atténuation des menaces qui n'ont pas été implémentées ou négligées. Les bugs détectés lors des tests sont corrigés et le cycle se poursuit jusqu'au lancement de l'application. Ce processus de gestion du cycle de vie des applications illustré à la *Figure 8.1* doit être suivi du point de vue de la sécurité :

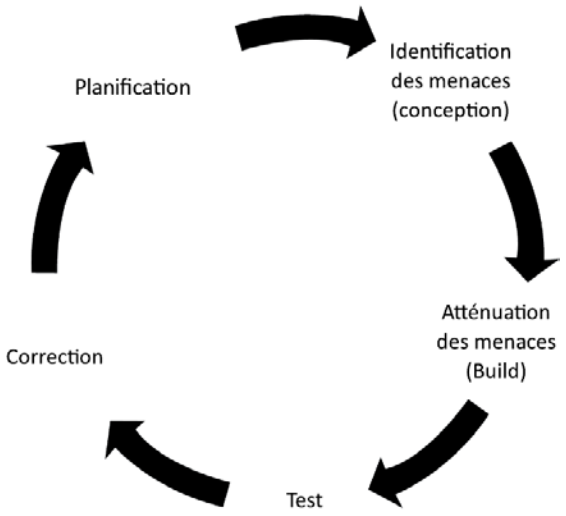


Figure 8.1 : cycle de vie de la sécurité

La planification, la modélisation des menaces, l'identification, la prévention, le test et la correction des menaces sont des processus itératifs qui se poursuivent même lorsqu'une application ou un service est opérationnel. Les environnements et les applications doivent faire l'objet d'un suivi actif dans leur intégralité, afin d'identifier de manière proactive les menaces et y remédier. Le suivi doit également permettre d'activer des alertes et de générer des journaux d'audit afin de favoriser un diagnostic réactif, le dépannage et l'élimination des menaces et autres vulnérabilités.

Le cycle de vie de la sécurité de n'importe quelle application démarre par la phase de planification, qui mène à terme à la phase de conception. Lors de la phase Conception, l'architecture applicative est décomposée en composants granulaires, avec une communication distincte et des limites en termes d'hébergement. Les menaces sont identifiées sur la base de leur interaction avec les autres composants, au sein des limites d'hébergement et en dehors de celles-ci. Les menaces sont limitées grâce à la mise en place de fonctions de sécurité appropriées au sein de l'architecture globale. Une fois les mesures de prévention mises en œuvre, des tests supplémentaires sont réalisés afin de vérifier si la menace perdure. Une fois l'application opérationnelle déployée dans l'environnement de production, elle fait l'objet d'un suivi pour identifier toute faille et vulnérabilité en termes de sécurité, suite de quoi des mesures de prévention, proactives ou réactives, sont implémentées.

Comme nous l'avons mentionné précédemment, différentes organisations appliquent divers processus et méthodes pour mettre en œuvre le cycle de vie de la sécurité. De même, Microsoft fournit des conseils complets et des informations sur le cycle de vie de la sécurité, lesquels sont disponibles à l'adresse https://www.microsoft.com/securityengineering/sdl/practices. À l'aide des pratiques communiquées par Microsoft, chaque organisation peut se concentrer sur la création de solutions plus sécurisées. À mesure que nous progressons à l'ère du cloud computing et que nous migrons nos données d'entreprise et de clients vers le Cloud, il est essentiel d'apprendre à sécuriser ces données. La section suivante porte sur la sécurité Azure et les différents niveaux de sécurité, qui nous permettront de créer des solutions sécurisées dans Azure.

Sécurité Azure

Azure offre tous ses services via des datacenters dans plusieurs régions. Ces datacenters sont reliés entre eux au sein des régions, et entre elles. Azure sait bien que les applications, services et données hébergés sont essentiels et critiques pour les clients. Le système doit s'assurer que la sécurité reste la plus grande propriété dans ses datacenters et ses régions.

Les clients déploient des applications dans le Cloud car ils font confiance à Azure et estiment qu'ainsi, leurs applications et données seront protégées de toute vulnérabilité et faille. Ils n'adopteront pas le Cloud si cette confiance est ébranlée, par conséquent Azure met en place la sécurité à tous les niveaux, comme indiqué à la *Figure* 8.2, depuis le périmètre des datacenters physiques aux composants logiques du logiciel. Chaque couche est protégée et même l'équipe du datacenter Azure n'y a pas accès :

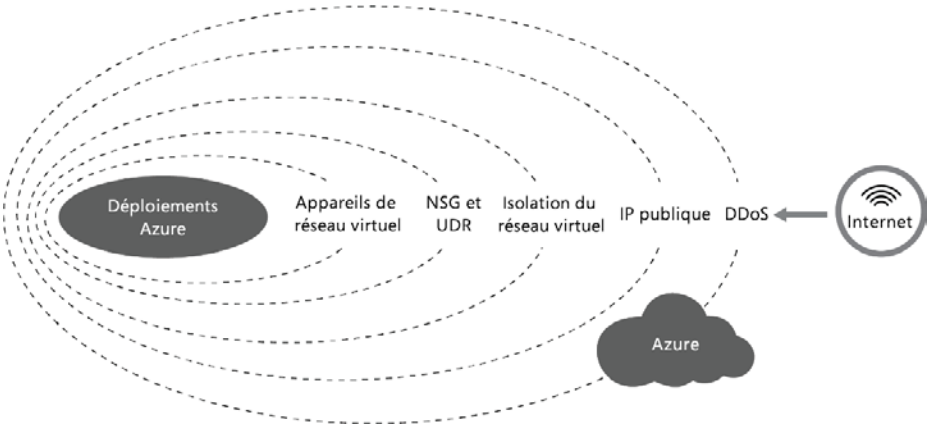


Figure 8.2 : fonctions de sécurité à différents niveaux dans les datacenters Azure

La sécurité est d'une importance primordiale à la fois pour Microsoft et pour Azure. Microsoft veille à maintenir une relation de confiance avec ses clients, en assurant une sécurité totale des déploiements, des solutions et des données des clients, tant sur le plan physique que virtuel. Les utilisateurs n'utiliseront pas une plateforme Cloud si celle-ci n'est pas sécurisée, physiquement et numériquement.

Afin de s'assurer que les clients font confiance en Azure, chaque activité de développement d'Azure est planifiée, documentée, auditée et fait l'objet d'un suivi du point de vue de la sécurité. Les datacenters Azure physiques sont protégés contre toute intrusion et contre tout accès non autorisé. En effet, même le personnel de Microsoft et l'équipe des opérations n'ont pas accès aux solutions et aux données des clients. Certaines des fonctions de sécurité fournies par Azure sont répertoriées ici :

- **Accès utilisateur sécurisé** : seul le client peut accéder à son déploiement, à sa solution et à ses données. Même le personnel du datacenter Azure n'a pas accès aux composants des clients. Les clients peuvent autoriser l'accès à des tiers, à leur discrétion.

- **Chiffrement au repos** : Azure chiffre toutes ses données de gestion, comprenant une variété de solutions de stockage de classe entreprise pour répondre à différents besoins. Microsoft assure également le chiffrement de services gérés tels qu'Azure SQL Database, Azure Cosmos DB et Azure Data Lake. Étant donné que les données sont chiffrées au repos, personne ne peut les consulter. Il met également à disposition cette fonction aux clients, si ceux-ci souhaitent chiffrer leurs données au repos.
- **Chiffrement en transit** : Azure chiffre toutes les données qui sont issues de son réseau. Il s'assure également que son réseau de base est protégé contre tout accès non autorisé.
- **Surveillance active et audit** : Azure surveille tous ses datacenters activement et en continu. Il identifie activement toute infraction, menace et risque et les minimise.

Azure respecte à la fois des normes de conformité nationales, locales et sectorielles. La liste complète des offres de conformité Microsoft est disponible à l'adresse https://www.microsoft.com/trustcenter/compliance/complianceofferings. Utilisez cette liste comme référence lors du déploiement de solutions conformes dans Azure. Nous connaissons désormais les principales fonctions de sécurité dans Azure, le moment est venu d'aborder la sécurité IaaS. La section suivante explique comment les clients peuvent tirer parti des fonctions de sécurité disponibles pour IaaS dans Azure.

Sécurité IaaS

Azure est une plateforme mature permettant de déployer des solutions IaaS. De nombreux utilisateurs d'Azure souhaitent contrôler pleinement leurs déploiements, et utilisent ainsi IaaS pour leurs solutions. Il est essentiel que ces déploiements et solutions soient sécurisés par défaut et de par leur conception. Azure offre des fonctions de sécurité enrichies afin de sécuriser les solutions IaaS. Dans cette section, certaines des principales fonctions seront abordées.

Groupes de sécurité réseau

Au strict minimum, un déploiement IaaS se compose de machines virtuelles et de réseaux virtuels. Une machine virtuelle peut être exposée à Internet via l'application d'une adresse IP publique à son interface réseau, ou bien elle peut être à la disposition de ressources internes uniquement. Certaines de ces ressources internes peuvent à leur tour être exposées à Internet. Dans tous les cas, les machines virtuelles doivent être sécurisées de sorte que les requêtes non autorisées ne puissent les atteindre. Les machines virtuelles doivent être sécurisées à l'aide d'installations capables de filtrer les requêtes au niveau du réseau en soi, plutôt que de laisser ces dernières atteindre la machine virtuelle et entreprendre une action en conséquence.

Le Ring-fencing, ou barrière, est l'un des mécanismes de sécurité utilisés par les machines virtuelles. Cette barrière peut autoriser ou refuser des requêtes en fonction de leur protocole, de l'IP d'origine et de destination et des ports d'origine et de destination. Cette fonction est déployée à l'aide de la ressource des **groupes de sécurité de réseau** d'Azure (Network Security Groups, **NSG**) Les NSG se composent de règles qui ont été évaluées à la fois pour les requêtes entrantes et sortantes. Selon l'exécution et l'évaluation de ces règles, il sera déterminé si l'accès de ces requêtes doit être autorisé ou refusé.

Les NSG sont flexibles et peuvent être appliqués à un sous-réseau du réseau virtuel ou des interfaces réseau individuelles. Lorsqu'elles sont appliquées à un sous-réseau, les règles de sécurité sont appliquées aux machines virtuelles hébergées sur ce sous-réseau. L'application à une interface réseau en revanche a une incidence uniquement sur les requêtes d'une machine virtuelle spécifique sur cette interface réseau. Il est également possible d'appliquer des NSG à la fois au sous-réseau et aux interfaces réseau simultanément. En règle générale, cette conception doit être utilisée pour appliquer des règles de sécurité communes au niveau du sous-réseau et des règles de sécurité distinctes au niveau de l'interface réseau. Cela permet de concevoir des règles de sécurité modulaires.

Le flux permettant d'évaluer les NSG est illustré à la *Figure* 8.3 :

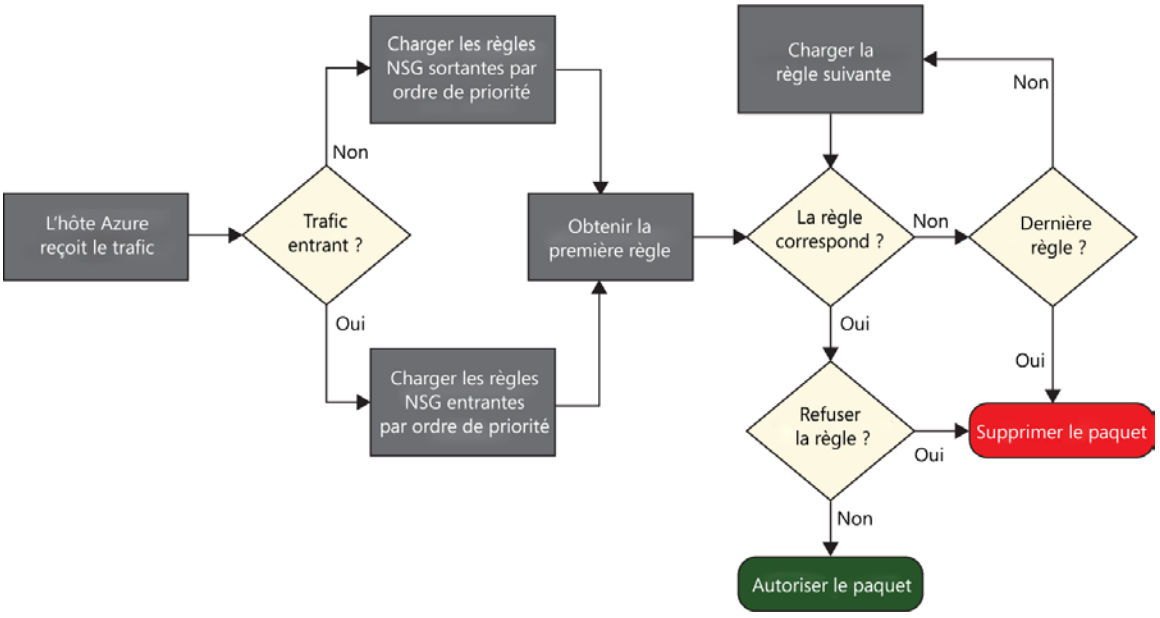


Figure 8.3 : diagramme de flux représentant l'évaluation des NSG

Lorsqu'une demande parvient à un hôte Azure, selon qu'il s'agit d'une demande entrante ou sortante, les règles appropriées sont chargées et exécutées par rapport à la demande/réponse. Si la règle correspond à la requête/réponse, cette dernière est autorisée ou refusée. La correspondance de règles se compose d'importantes informations de demande/réponse, telles que l'adresse IP source, l'adresse IP de destination, le port source, le port de destination et le protocole utilisé. Les NSG prennent également en charge les balises de service. Une balise de service désigne un groupe de préfixes d'adresse IP sur un service Azure donné. Microsoft gère les préfixes d'adresse et les met à jour automatiquement. Ainsi, les règles de sécurité ne doivent plus être mises à jour chaque fois qu'un préfixe d'adresse est modifié.

L'ensemble des balises de service disponibles figurent à l'adresse https://docs.microsoft.com/azure/virtual-network/service-tags-overview#available-service-tags. Les balises de service peuvent être utilisées avec des NSG et le pare-feu Azure. Maintenant que nous connaissons le fonctionnement des NSG, passons à la conception des NSG. Vous pourrez ainsi déterminer les principaux points à prendre en compte lors de la création des règles NSG en vue de renforcer la sécurité.

Conception de réseau virtuel

La première étape de conception d'un NSG consiste à déterminer les exigences de sécurité de la ressource. Les éléments suivants doivent être déterminés ou pris en compte :

- La ressource est-elle accessible depuis Internet uniquement ?
- La ressource est-elle accessible à la fois depuis les ressources internes et Internet ?
- La ressource est-elle accessible depuis les ressources internes uniquement ?
- En fonction de l'architecture de la solution déployée, déterminez les ressources dépendantes, les équilibreurs de charge, les passerelles et les machines virtuelles utilisés.
- Configuration d'un réseau virtuel et de son sous-réseau.

Une conception NSG adaptée devra être créée sur la base des résultats de ces examens. Idéalement, plusieurs sous-réseaux du réseau devraient être mis en place pour chaque charge de travail et type de ressource. Il n'est pas recommandé de déployer les équilibreurs de charge et les machines virtuelles sur le même sous-réseau.

Selon les besoins du projet, des règles communes à différents sous-réseaux et différentes charges de travail des machines virtuelles doivent être établies. Par exemple, concernant un déploiement SharePoint, l'application frontale et les serveurs SQL sont déployés sur des sous-réseaux distincts. Des règles doivent être déterminées pour chaque sous-réseau.

Une fois que des règles au niveau du sous-réseau sont identifiées, il est nécessaire d'identifier des règles inhérentes aux ressources individuelles, lesquelles doivent être appliquées au niveau de l'interface réseau. Il est important de comprendre que si une règle autorise une requête entrante sur un port, ce port peut également être utilisé pour les requêtes sortantes sans aucune configuration.

Si les ressources sont accessibles depuis Internet, des règles doivent être créées avec des plages et des ports IP spécifiques, dans la mesure du possible, au lieu d'autoriser le trafic de toutes les plages IP (généralement représentées sous le format 0.0.0.0/0). Veillez à mener des tests fonctionnels et de sécurité afin de vous assurer que les règles NSG adéquates et optimales sont ouvertes et fermées.

Pare-feux

NSG fournit des périmètres de sécurité extérieurs pour les requêtes. Cependant, les machines virtuelles doivent tout de même mettre en place des mesures de sécurité supplémentaires. Il est toujours préférable d'implémenter la sécurité tant au niveau interne qu'externe. Les machines virtuelles, qu'elles exécutent Linux ou Windows, fournissent un mécanisme permettant de filtrer les requêtes au niveau du système d'exploitation. Un tel système est dénommé pare-feu, dans Windows et dans Linux.

Il est conseillé de mettre en place des pare-feux pour les systèmes d'exploitation. Cela permet de bâtir une barrière de sécurité virtuelle, qui permettra de n'autoriser que les requêtes considérées comme fiables. Toute requête non fiable se verra refuser l'accès. Il existe également des dispositifs de pare-feu physique, toutefois dans un logiciel Cloud, les pare-feux de système d'exploitation sont utilisés. La *Figure 8.4* illustre la configuration du pare-feu pour un système d'exploitation Windows :

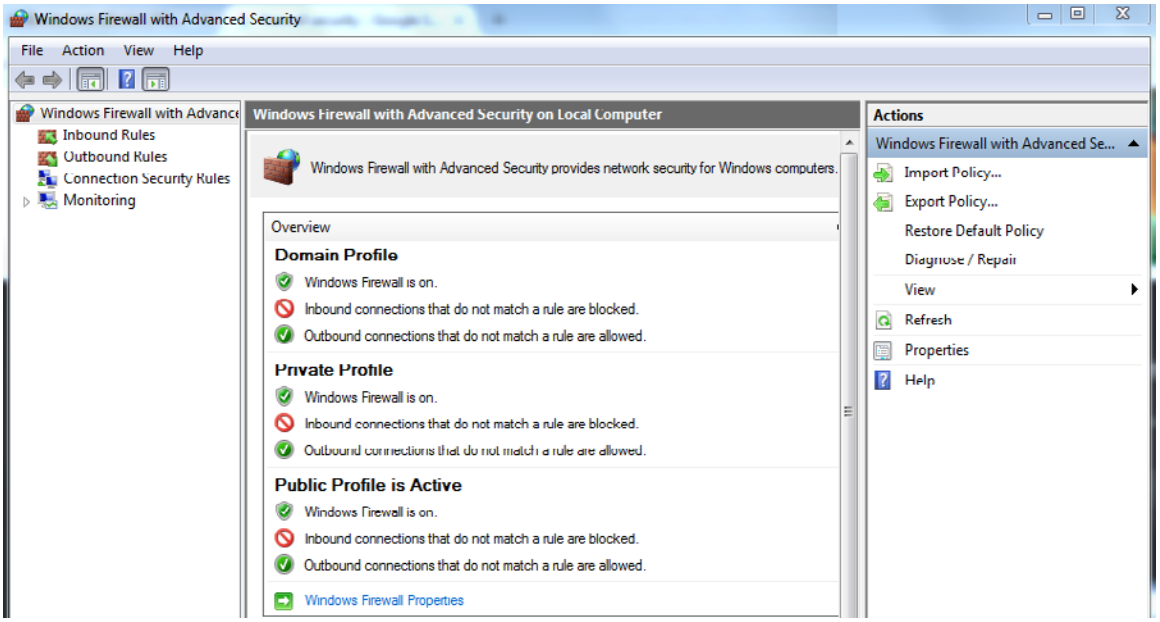


Figure 8.4 : configuration du pare-feu

Les pare-feux filtrent les paquets réseau et identifient les ports entrants et les adresses IP. Selon les informations de ces paquets, le pare-feu évalue les règles et détermine si l'accès doit être autorisé ou refusé.

Différentes solutions de pare-feu sont disponibles sous Linux. Certaines des offres de pare-feu sont très spécifiques à la distribution utilisée ; par exemple, SUSE utilise SuSefirewall2 et Ubuntu utilise ufw. Les mises en œuvre les plus couramment utilisées sont firewalld et iptables, qui sont disponibles sur chaque distribution.

Conception du pare-feu

À titre de bonne pratique, les pare-feux doivent être évalués pour les différents systèmes d'exploitation. Chaque machine virtuelle a une responsabilité distincte au sein de la solution et du déploiement global. Les règles inhérentes à ces responsabilités individuelles doivent être identifiées et les pare-feux doivent être ouverts et fermés en conséquence.

Lors de l'évaluation des règles de pare-feu, il est important de tenir compte en tout temps des règles relatives aux groupes de sécurité, tant au niveau du sous-réseau qu'au niveau de l'interface réseau. Dans le cas contraire, il est possible que certaines règles soient rejetées au niveau du groupe NSG, mais qu'elles restent actives au niveau du pare-feu, et vice versa. Si une requête est autorisée au niveau de la règle NSG et refusée au niveau du pare-feu, l'application ne fonctionnera pas correctement, tandis que les risques de sécurité augmenteront si la requête est refusée au niveau de la règle NSG et autorisée au niveau du pare-feu.

Un pare-feu permet de mettre en place plusieurs réseaux isolés par leurs règles de sécurité. Veillez à mener des tests fonctionnels et de sécurité afin de vous assurer que les règles de pare-feu adéquates et optimales sont ouvertes et fermées.

Il est judicieux d'utiliser le pare-feu Azure, qui est un service réseau basé dans le Cloud en plus des NSG. Il est très facile à configurer, offre une gestion centrale de l'administration et ne nécessite aucune maintenance. Ensemble, le pare-feu Azure et les NSG garantissent la sécurité entre les machines virtuelles, les réseaux virtuels et même les différents abonnements Azure. Cela dit, si une solution nécessite un niveau de sécurité supplémentaire, un pare-feu peut être mise en œuvre au niveau du système d'exploitation. Nous discuterons d'un pare-feu Azure plus en détail dans l'une des sections suivantes, *Pare-feu Azure*.

Groupes de sécurité d'application

Les NSG sont flexibles au niveau du sous-réseau du réseau virtuel ou directement au niveau des interfaces réseau individuelles. Bien qu'il soit suffisant d'appliquer des NSG au niveau du sous-réseau, certaines situations exigent une mise en œuvre supplémentaire. Différents types de charges de travail sont disponibles dans un seul sous-réseau et chacun d'entre eux nécessite un groupe de sécurité différent. Il est possible d'attribuer des groupes de sécurité aux **cartes d'interface réseau** (**NIC**) individuelles des machines virtuelles. Toutefois, cette opération peut facilement engendrer une maintenance excessive si les machines virtuelles sont nombreuses.

Azure dispose d'une fonction relativement récente, appelée groupes de sécurité d'application. Nous pouvons créer des groupes de sécurité d'application et les attribuer directement à plusieurs cartes réseau, même lorsque ces cartes réseau appartiennent à des machines virtuelles relevant de différents sous-réseaux et groupes de ressources. La fonctionnalité des groupes de sécurité d'application est similaire aux NSG, sauf qu'elle offre une autre méthode plus souple pour attribuer des groupes aux ressources réseau, car elle permet de les affecter à différents groupes de ressources et sous-réseaux. Les groupes de sécurité d'application peuvent simplifier les NSG. Il existe toutefois une limite principale. Un groupe de sécurité d'application peut appartenir à la source et la destination d'une règle de sécurité, ce qui n'est pas le cas pour l'instant pour plusieurs groupes de sécurité d'application.

L'une des bonnes pratiques appliquées à la création de règles consiste à toujours minimiser le nombre de règles de sécurité nécessaires, afin d'éviter le maintien de règles explicites. La section précédente portait sur l'utilisation des balises de service avec les NSG pour éviter de maintenir les préfixes d'adresse IP individuels de chaque service. De même, lorsque vous utilisez des groupes de sécurité d'application, nous pouvons réduire la complexité des adresses IP explicites et des différentes règles. Cette pratique est recommandée chaque fois que cela est possible. Appliquez-la uniquement si votre solution exige une règle explicite avec une adresse IP individuelle ou une plage d'adresses IP.

Pare-feu Azure

Dans la section précédente, nous avons discuté de l'utilisation du pare-feu Azure au sein d'un système d'exploitation Windows/Linux pour autoriser ou interdire les demandes et les réponses au travers de ports et de services particuliers. Bien que les pare-feux de système d'exploitation jouent un rôle important du point de vue de la sécurité et doivent être mis en œuvre dans tous les déploiements des entreprises, Azure fournit une ressource de sécurité appelée Pare-feu Azure. Cette solution présente une fonctionnalité similaire de filtrage des demandes en fonction des règles et détermine si une demande doit être autorisée ou rejetée.

Le pare-feu Azure présente un avantage : il évalue une requête avant qu'elle n'atteigne un système d'exploitation. Le pare-feu Azure est une ressource réseau et un service autonome qui protège les ressources au niveau du réseau virtuel. Toutes les ressources, y compris les machines virtuelles et les équilibreurs de charge, qui sont directement associées à un réseau virtuel peuvent être protégées à l'aide du pare-feu Azure.

Le pare-feu Azure est un service hautement disponible et évolutif qui peut protéger non seulement les requêtes HTTP, mais aussi tout type de requêtes entrantes et sortantes d'un réseau virtuel, notamment FTP, SSH et RDP. Le pare-feu Azure peut également couvrir plusieurs zones de disponibilité pendant le déploiement pour optimiser la disponibilité.

Il est fortement recommandé que le pare-feu Azure soit déployé pour les charges de travail stratégiques sur Azure, parallèlement à d'autres mesures de sécurité. Il est également important de noter que le pare-feu Azure doit être utilisé, même si d'autres services, tels qu'Azure Application Gateway et Azure Front Door, sont utilisés. En effet, tous ces outils présentent des portées et des fonctions différentes. En outre, le pare-feu Azure prend en charge les balises de service et les renseignements sur les menaces. Dans la section précédente, nous avons discuté des avantages des balises de service. Les renseignements sur les menaces peuvent être utilisés pour générer des alertes lorsque le trafic provient ou est dirigé vers des adresses IP et domaines malveillants connus, lesquels sont enregistrés dans le flux Microsoft Threat Intelligence.

Réduire la surface d'attaque

Les groupes NSG et les pare-feux permettent de gérer les requêtes autorisées dans l'environnement. Cependant, l'environnement ne doit pas être ouvertement exposé aux attaques de sécurité. La surface du système doit être activée de sorte à être à même d'assurer un fonctionnement optimal tout en étant suffisamment désactivée pour que les pirates informatiques ne détectent pas des failles et des accès à exploiter de par leur manque de sécurité. La sécurité doit être suffisamment rigoureuse pour que les pirates informatiques ne puissent pas aisément s'infiltrer dans le système.

Voici certaines des configurations à définir :

- Supprimer tous les groupes et utilisateurs inutiles du système d'exploitation.
- Identifier l'appartenance à un groupe pour tous les utilisateurs.
- Mettre en œuvre des stratégies de groupe à l'aide des services de répertoire.
- Bloquer l'exécution de scripts sauf si ceux-ci sont signés par une autorité de confiance.
- Consigner et auditer toutes les activités.
- Installer des logiciels antivirus et programmes anti-logiciels malveillants et planifier des analyses et des mises à jour fréquemment.
- Désactiver ou mettre fin aux services qui ne sont pas nécessaires.
- Verrouiller le système de fichiers pour permettre uniquement un accès autorisé.
- Verrouiller les modifications apportées au registre.
- Le pare-feu doit être configuré selon les exigences.
- L'exécution du script PowerShell doit être défini sur **Restricted** (Restreint) ou sur **RemoteSigned**. Les commandes PowerShell **`Set-ExecutionPolicy -ExecutionPolicy Restricted`** ou **`Set-ExecutionPolicy -ExecutionPolicy RemoteSigned`** permettent de le faire.
- Protection renforcée activée à l'aide d'Internet Explorer.
- Restreindre la possibilité de créer de nouveaux utilisateurs et groupes.

- Supprimer l'accès Internet et mettre en place des serveurs intermédiaires pour RDP.
- Supprimer l'autorisation de consignation dans les serveurs à l'aide de RDP via Internet. Au lieu de cela, utilisez le VPN site à site, le VPN point à site ou Express Route vers RDP dans des ordinateurs distants du réseau.
- Déployer régulièrement toutes les mises à jour de sécurité.
- Exécuter l'outil de gestionnaire de la conformité à la sécurité sur l'environnement et mettre en œuvre toutes ses recommandations.
- Surveiller activement l'environnement à l'aide de Security Center et d'Operations Management Suite.
- Déployer des appareils de réseau virtuel pour acheminer le trafic proxy interne et proxy inversé.
- Toutes les données sensibles, telles que la configuration, les chaînes de connexion et les informations d'identification, doivent être chiffrées.

Les éléments susmentionnés sont quelques-uns des points clés qui doivent être pris en compte du point de vue de la sécurité. La liste continuera de croître, et nous devons constamment améliorer la sécurité pour prévenir tout type de violation de sécurité.

Mise en œuvre de serveurs intermédiaires

Il est recommandé de supprimer l'accès à Internet sur les machines virtuelles. Il est également conseillé de limiter l'accessibilité des services de bureau à distance sur Internet, toutefois, comment accéder dans ce cas aux machines virtuelles ? Vous pouvez par exemple autoriser uniquement des ressources internes à accéder au RDP dans les machines virtuelles à l'aide des options de VPN Azure. Toutefois, vous pouvez également utiliser des **serveurs intermédiaires**.

Les serveurs intermédiaires sont des serveurs déployés sur une **zone démilitarisée (DMZ)**. Cela signifie qu'ils ne se trouvent pas sur le réseau hébergeant les solutions et applications principales. Au lieu de cela, ils se trouvent sur un réseau ou un sous-réseau distinct. L'objectif principal des serveurs intermédiaires est d'accepter les requêtes des utilisateurs RDP et de les aider à s'y connecter. À partir d'un serveur intermédiaire, les utilisateurs peuvent accéder à d'autres machines virtuelles, à l'aide du RDP. Celui-ci a accès à deux réseaux ou plus : l'un d'eux est connecté à l'environnement externe et l'autre est interne, dans la solution. Le serveur intermédiaire implémente toutes les restrictions en matière de sécurité et fournit aux clients une connexion sécurisée aux autres serveurs. Normalement, l'accès aux e-mails et à Internet est désactivé sur les serveurs intermédiaires.

Vous trouverez un exemple de déploiement d'un serveur intermédiaire avec **VMSS (Virtual machine scale sets)** qui utilise des modèles Azure Resource Manager, disponibles à l'adresse https://azure.microsoft.com/resources/templates/201-vmss-windows-jumpbox.

Azure Bastion

Dans la section précédente, nous avons discuté de la mise en œuvre de serveurs intermédiaires. Azure Bastion est un service entièrement géré qui peut être configuré dans un réseau virtuel pour fournir un accès RDP/SSH à vos machines virtuelles directement dans le portail Azure sur TLS. L'hôte Bastion servira de serveur intermédiaire. Vous n'aurez donc pas besoin d'utiliser des adresses IP publiques pour vos machines virtuelles. Le concept de l'utilisation de Bastion est le même que celui de la mise en œuvre d'un serveur intermédiaire. Toutefois, étant donné qu'il s'agit d'un service géré, il est entièrement géré par Azure.

Étant donné que Bastion est un service entièrement géré par Azure et qu'il est renforcé en interne, nous n'avons pas besoin d'appliquer des NSG supplémentaires sur le sous-réseau Bastion. En outre, étant donné que nous n'attachons aucune adresse IP publique à nos machines virtuelles, celles-ci sont protégées contre l'analyse de port.

Sécurité des applications

Les applications Web peuvent être hébergées dans des solutions basées sur IaaS, sur des machines virtuelles, et dans des services gérés fournis par Azure, tels qu'App Service. App Service fait partie du paradigme de déploiement PaaS, sujet que nous aborderons dans la section suivante. Cette section porte sur la sécurité au niveau de l'application.

SSL/TLS

Secure Socket layer (**SSL**) est devenu obsolète et a été remplacé par **Transport Layer security** (**TLS**). Le protocole TLS offre une sécurité end-to-end au moyen de la cryptographie. Il propose deux types de cryptographie :

- Symétrique : la même clé est disponible à la fois pour l'expéditeur et le destinataire du message, et elle est utilisée pour le chiffrement et le déchiffrement du message.
- Asymétrique : chaque partie prenante possède deux clés : une clé privée et une clé publique. La clé privée reste sur le serveur ou avec l'utilisateur et demeure secrète, tandis que la clé publique est distribuée librement à tout le monde. Les titulaires de la clé publique l'utilisent pour chiffrer le message, qui ne peut être déchiffré que par la clé privée correspondante. Étant donné que la clé privée reste avec le propriétaire, lui-seul peur déchiffrer le message. **Rivest–Shamir–Adleman** (**RSA**) est l'un des algorithmes utilisés pour générer ces paires de clés publiques et privées.
- Les clés sont également disponibles dans les certificats communément appelés certificats X.509. Toutefois, les certificats présentent d'autres détails que les clés et sont généralement émis par des autorités de certification de confiance.

Le protocole TLS doit être utilisé par les applications Web pour garantir la sécurité et la confidentialité des échanges de messages entre les utilisateurs et le serveur, tout en protégeant les identités. Ces certificats doivent être achetés auprès d'une autorité de certification de confiance et ne pas être des certificats auto-signés.

Identités gérées

Avant d'aborder les identités gérées, il est important de savoir comment les applications ont été créées sans elles.

La méthode traditionnelle de développement d'applications consiste à utiliser des secrets, tels qu'un nom d'utilisateur, un mot de passe ou des chaînes de connexion SQL, dans des fichiers de configuration. L'intégration de ces secrets dans les fichiers de configuration permet de faciliter et d'assouplir la modification des applications en fonction de ces secrets sans modifier le code. Cette méthode nous permet de respecter le principe « Ouvert à l'extension, fermé aux modifications ». Toutefois, cette approche présente un inconvénient du point de vue de la sécurité. Les secrets peuvent être consultés par tous ceux qui ont accès aux fichiers de configuration, car ces secrets y sont généralement répertoriés en texte brut. Quelques astuces permettent de les chiffrer, même si elles ne sont pas infaillibles.

Pour mieux utiliser les secrets et les informations d'identification au sein d'une application, ceux-ci doivent être stocker dans un référentiel de secrets, tel qu'Azure Key Vault. Azure Key Vault assure une sécurité complète à l'aide du **module de sécurité matérielle** (**HSM**). Les secrets sont stockés de manière chiffrée avec le déchiffrement à la demande à l'aide de clés stockées sur du matériel distinct. Les secrets peuvent être stockés dans Key Vault, et chacun présente un nom complet et une clé. La clé se présente sous la forme d'un URI qui peut être utilisé pour désigner le secret des applications, comme illustré à la *Figure* 8.5 :

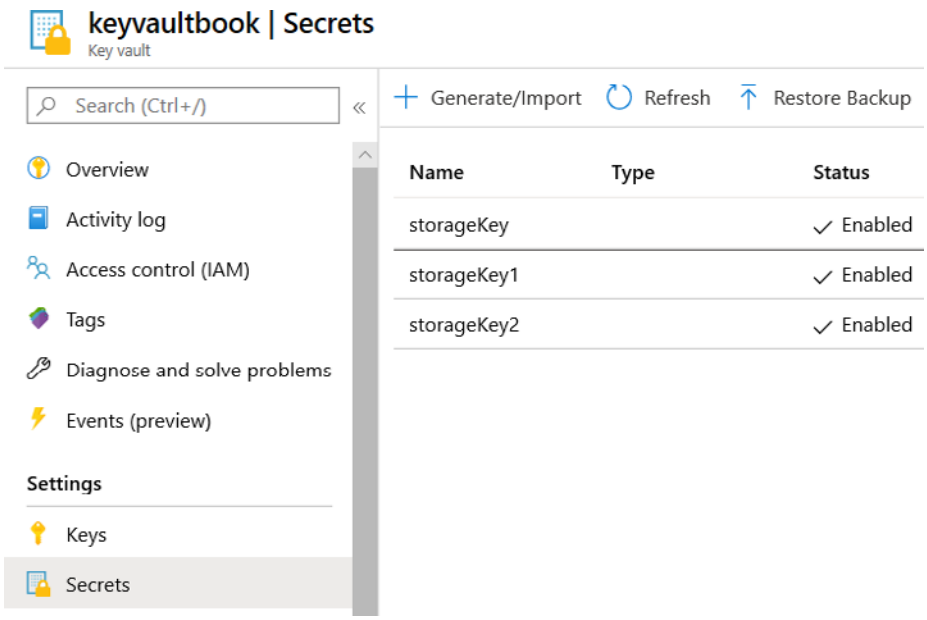


Figure 8.5 : stockage de secrets dans un coffre de clés

Pour consulter un secret dans les fichiers de configuration de l'application, nous pouvons utiliser le nom ou la clé. Un autre défi se pose néanmoins. Comment l'application se connecte-t-elle et s'authentifie-t-elle avec le coffre de clés ?

Les coffres de clés appliquent des stratégies d'accès qui définissent les autorisations pour un utilisateur ou un groupe concernant l'accès aux secrets et aux informations d'identification au sein du coffre de clés. Les utilisateurs, les groupes ou les applications de service qui peuvent obtenir un accès sont mis en place et hébergés dans **Azure Active Directory** (**Azure AD**). Bien que les comptes d'utilisateurs individuels puissent bénéficier d'un accès à l'aide des stratégies d'accès Key Vault, il est recommandé d'utiliser un principal de service pour accéder au coffre de clés. Un principal de service possède un identificateur, également appelé ID d'application ou ID client, ainsi qu'un mot de passe. L'ID client, ainsi que son mot de passe, peuvent être utilisés pour l'authentification auprès d'Azure Key Vault. Ce principal de service peut être autorisé à accéder aux secrets. Les stratégies d'accès appliquées à Azure Key Vault sont accordées dans le volet **Access policies** (Stratégies d'accès) de votre coffre de clés. Dans *la Figure 8.6* , vous pouvez voir que le principal de service (**https://keyvault.book.com**) a accordé l'accès au coffre de clés appelé **keyvaultbook** :

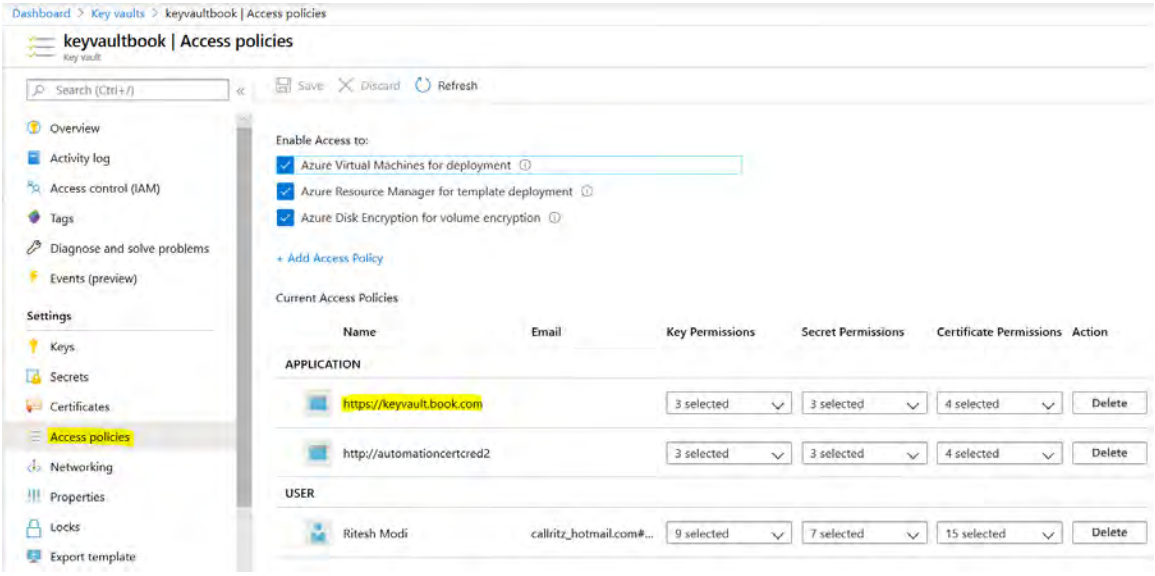


Figure 8.6 : accès accordé à un principal de service à un coffre de clés

Cela nous amène à relever un autre défi : pour accéder au coffre de clés, nous devons utiliser l'ID client et le secret dans nos fichiers de configuration pour nous connecter au coffre de clés, saisir le secret et récupérer sa valeur. Cela équivaut presque à utiliser un nom d'utilisateur, un mot de passe et une chaîne de connexion SQL dans les fichiers de configuration.

C'est là que les identités gérées s'avèrent utiles. Azure a lancé des identités de service géré, en les renommant par la suite Identités gérées. Les identités gérées sont des identités gérées par Azure. Les identités gérées créent également un principal de service ainsi qu'un mot de passe en arrière-plan. Aucune information d'identification ne doit être entrée dans les fichiers de configuration avec les identités gérées.

Les identités gérées peuvent uniquement être utilisées pour l'authentification auprès de services qui ont choisi Azure AD comme fournisseur d'identité. Les identités gérées sont destinées uniquement à l'authentification. Si le service cible ne fournit pas d'autorisation de **contrôle d'accès en fonction du rôle** (**RBAC**) à l'identité, l'identité est susceptible de ne pas effectuer l'activité prévue sur le service cible.

Les identités gérées sont divisées en deux catégories :

- Identités gérées attribuées au système
- Identités gérées attribuées à l'utilisateur

Les identités attribuées au système sont générées par le service lui-même. Par exemple, si un service d'application souhaite se connecter à Azure SQL Database, il peut générer l'identité gérée attribuée au système dans le cadre de ses options de configuration. Ces identités gérées sont également supprimées lorsque la ressource ou le service parent est supprimé(e). Comme illustré à la *Figure* 8.7 , une identité attribuée au système peut être utilisée par App Service pour se connecter à Azure SQL Database :

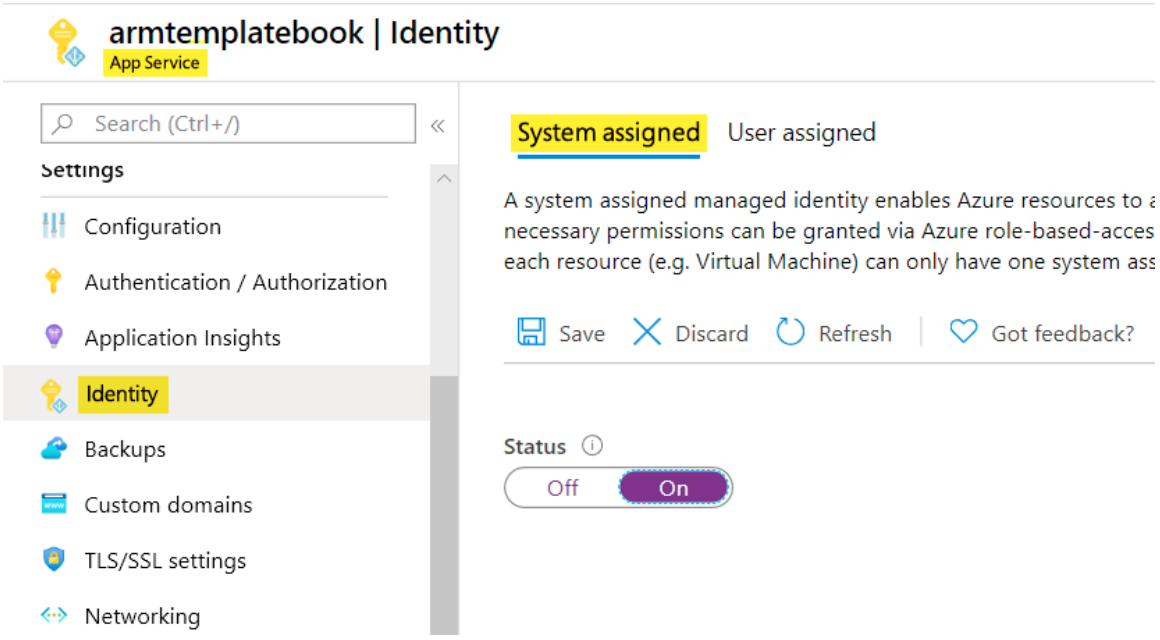


Figure 8.7 : activation d'une identité gérée attribuée au système pour App Service

Les identités gérées attribuées à l'utilisateur sont créées en tant qu'identités autonomes distinctes et ensuite attribuées aux services Azure. Elles peuvent être appliquées et réutilisées avec plusieurs services Azure, car leurs cycles de vie ne dépendent pas de la ressource à laquelle elles sont attribuées.

Une fois qu'une identité gérée est créée et que des contrôles RBAC ou des autorisations d'accès sont accordés à celle-ci sur la ressource cible, elle peut être utilisée dans les applications pour accéder aux ressources et aux services cibles.

Azure propose un SDK et une API REST pour communiquer avec Azure AD et obtenir un jeton d'accès pour les identités gérées. Ce jeton peut ensuite être utilisé pour accéder aux ressources cibles et les consommer.

Le SDK est inclus dans le package NuGet **`Microsoft.Azure.Services.AppAuthentication`** pour C#. Une fois le jeton d'accès disponible, il peut être utilisé pour consommer la ressource cible.

Voici le code permettant d'obtenir le jeton d'accès :

```
var tokenProvider = new AzureServiceTokenProvider();

string token = await tokenProvider.GetAccessTokenAsync
("https://vault.azure.net");
```

Ou encore :

```
string token = await tokenProvider.GetAccessTokenAsync
("https://database.windows.net");
```

Il convient de noter que le code de l'application doit s'exécuter dans le contexte d'App Service ou d'une application de fonction, car l'identité y est associée et n'est disponible que dans le code lorsqu'elle est exécutée à partir de ceux-ci.

Le code précédent couvre deux cas d'utilisation différents. Les codes permettant d'accéder au coffre de clés et à Azure SQL Database sont affichés ensemble.

Il est important de noter que les applications ne fournissent aucune information relative aux identités gérées dans le code et sont entièrement gérées à l'aide de la configuration. Les développeurs, les administrateurs d'applications individuels et les opérateurs n'accéderont à aucune information d'identification relative aux identités gérées. En outre, celles-ci ne sont pas non plus mentionnées dans le code. La rotation des informations d'identification est entièrement régie par le fournisseur de ressources qui héberge le service Azure. La rotation par défaut se produit tous les 46 jours. C'est au fournisseur de ressources d'appeler de nouvelles informations d'identification si nécessaire. Le fournisseur peut donc attendre plus de 46 jours.

La prochaine section porte sur **gestionnaire des informations et des événements de sécurité** (**SIEM**) natif du Cloud : Azure Sentinel.

Azure Sentinel

Azure propose un SIEM et une **réponse automatisée à l'orchestration de sécurité (SOAR)** en tant que service autonome qui peut être intégrée à n'importe quel déploiement personnalisé sur Azure. La *Figure 8.8* illustre quelques fonctions clés d'Azure Sentinel :

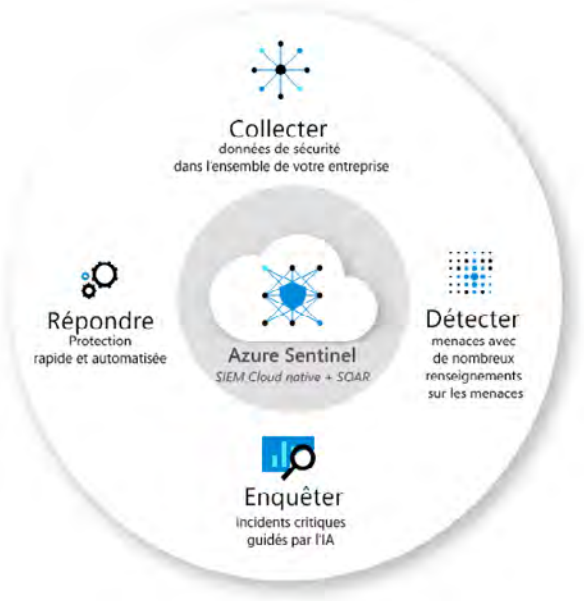


Figure 8.8 : principales fonctions d'Azure Sentinel

Azure Sentinel recueille des journaux d'informations à partir de déploiements et de ressources et effectue des analyses pour identifier des modèles d'application et des tendances liés à divers problèmes de sécurité qui sont tirés des sources de données.

Une surveillance active de l'environnement doit être mise en œuvre, les journaux doivent être collectés et les informations doivent être extraites de ces journaux dans le cadre d'une activité distincte de l'implémentation de code. C'est là que le service SIEM entre en jeu. De nombreux connecteurs peuvent être utilisés avec Azure Sentinel. Chacun d'entre eux sera utilisé pour ajouter des sources de données à Azure Sentinel. Azure Sentinel fournit des connecteurs pour les services Microsoft, tels que Office 365, Azure AD et Azure Threat Protection. Les données collectées seront transmises à un espace de travail log Analytics et vous pourrez écrire des requêtes pour rechercher ces journaux.

Les outils SIEM tels qu'Azure Sentinel peuvent être activés sur Azure afin de rechercher tous les journaux de Log Analytics et d'Azure Security Center, lesquels peuvent à leur tour en obtenir à partir de plusieurs sources, déploiements et services. SIEM peut ensuite exécuter ses renseignements en plus de ces données collectées et générer des informations. Il peut générer des rapports et des tableaux de bord en fonction des renseignements découverts pour la consommation, mais également enquêter sur les activités suspectes et les menaces, et prendre des mesures à leur égard.

Bien qu'Azure Sentinel puisse sembler très similaire à la fonctionnalité d'Azure Security Center, Azure Sentinel offre bien plus de fonctions qu'Azure Security Center. Il se distingue d'Azure Security Center grâce à sa capacité à collecter des journaux à partir d'autres sources à l'aide de connecteurs.

Sécurité PaaS

Azure fournit de nombreux services PaaS, chacun disposant de ses propres fonctions de sécurité. En général, les services PaaS sont accessibles à l'aide des informations d'identification, des certificats et des jetons. Les services PaaS permettent de générer des jetons d'accès de sécurité à durée limitée. Les applications clientes peuvent envoyer ces jetons d'accès sécurisé pour représenter les utilisateurs approuvés. Dans cette section, nous allons aborder certains des services PaaS les plus importants, qui sont utilisés dans chaque solution.

Azure Private Link

Azure Private Link permet d'accéder aux services Azure PaaS et aux services clients ou partenaires hébergés/détenus/partagés sur Azure sur un point de terminaison privé de votre réseau virtuel. Nous n'avons pas à exposer nos services sur l'Internet publique lorsque nous utilisons Azure Private Link, et tout le trafic entre notre service et le réseau virtuel passe par le réseau principal de Microsoft.

Azure Private Endpoint est l'interface réseau qui garantit une connexion privée et sécurisée à un service alimenté par Azure Private Link. Étant donné que le point de terminaison privé est mappé à l'instance du service PaaS, et non à l'intégralité du service, les utilisateurs ne peuvent se connecter qu'à la ressource. Les connexions à n'importe quel autre service sont refusées, en assurant une protection contre les fuites de données. Private Endpoint offre également un accès local sécurisé via ExpressRoute ou des tunnels VPN. Ainsi, vous n'avez plus besoin de configurer l'appairage public ou de passer par l'Internet public pour atteindre le service.

Azure Application Gateway

Azure fournit un équilibreur de charge de niveau 7 connu sous le nom d'Azure Application Gateway. Ce dernier, en plus d'équilibrer les charges, contribue au routage à l'aide de valeurs de l'URL. Il présente également une fonctionnalité connue sous le nom de Pare-feu d'application Web. Azure Application Gateway prend en charge la terminaison TLS à la passerelle, de sorte que les serveurs back-end reçoivent le trafic non chiffré. Cela présente plusieurs avantages, tels que de meilleures performances, une meilleure utilisation des serveurs dorsaux et un routage intelligent des paquets. Dans la section précédente, nous avons discuté du pare-feu Azure et de la façon dont il protège les ressources au niveau du réseau. En revanche, le pare-feu d'application Web protège le déploiement au niveau de l'application.

Toute application déployée qui est exposée sur Internet est confrontée à de nombreux défis de sécurité. Voici certaines des menaces de sécurité importantes :

- Scripts intersites
- Exécution de code à distance
- Injection SQL
- Attaques de **déni de service** (**DoS**)
- Attaques de **déni de service distribué** (**DDoS**)

Il existe beaucoup d'autres menaces cependant.

Un grand nombre de ces attaques peut être traité par les développeurs via l'écriture de code défensif et l'application des bonnes pratiques. Toutefois, le code ne permet pas à lui-seul d'identifier ces problèmes sur un site en direct. Le pare-feu d'application Web configure des règles qui peuvent identifier ces problèmes, comme mentionné précédemment, et refuser les requêtes.

Il est conseillé d'utiliser les fonctions d'Application Gateway Web Application Firewall pour protéger les applications contre les menaces de sécurité en direct. Le pare-feu d'application Web autorisera le transfert des requêtes ou les arrêtera, selon la façon dont il est configuré.

Azure Front Door

Azure a lancé un service relativement nouveau connu sous le nom d'Azure Front Door. Le rôle d'Azure Front Door est assez similaire à celui d'Azure Application Gateway, à une différence près toutefois : la portée. Application Gateway fonctionne au sein d'une seule région, tandis qu'Azure Front Door fonctionne au niveau mondial, dans différents datacenters et régions . Il dispose également d'un pare-feu d'application Web qui peut être configuré pour protéger les applications déployées dans plusieurs régions contre diverses menaces de sécurité, telles que l'injection SQL, l'exécution de code à distance et les scripts inter-sites.

Application Gateway peut être déployée derrière Front Door pour résoudre le drainage des connexions. En outre, en déployant Application Gateway derrière Front Door, vous pourrez répondre à vos besoins en matière d'équilibrage de charge. En effet, Front Door ne peut effectuer que l'équilibrage de charge basé sur les chemins d'accès au niveau mondial. L'ajout d'Application Gateway à l'architecture apportera un équilibrage de charge supplémentaire aux serveurs back-end du réseau virtuel.

Azure App Service Environment

Azure App Service est déployé sur des réseaux partagés en coulisses. Toutes les références d'App Service utilisent un réseau virtuel, qui peut potentiellement être utilisé par d'autres locataires. Afin d'accroître votre contrôle et de sécuriser le déploiement d'App Service sur Azure, les services peuvent être hébergés sur des réseaux virtuels dédiés. Pour ce faire, vous pouvez utiliser **Azure App service Environment** (**ASE**), qui fournit une isolation complète pour exécuter votre App Service à grande échelle. Cette méthode offre également une sécurité supplémentaire en vous permettant de déployer le pare-feu Azure, les groupes de sécurité d'application, les NSG, la passerelle d'application, le pare-feu d'application Web et Azure Front Door. Tous les plans App Service créés dans App Service Environment figureront dans un niveau de tarification isolé, sans que nous puissions choisir un autre niveau.

Tous les journaux de ce réseau virtuel et de ce calcul peuvent ensuite être rassemblés dans Azure log Analytics et Security Center, et enfin avec Azure Sentinel.

Azure Sentinel peut ensuite fournir des informations et exécuter des classeurs et des procédures opérationnelles pour répondre aux menaces de sécurité de manière automatisée. Les playbooks de sécurité peuvent être exécutés dans Azure Sentinel en réponse aux alertes. Chaque playbook de sécurité comprend des mesures qui doivent être prises en cas d'alerte. Les playbooks sont basés sur Azure Logic Apps. Vous pouvez ainsi utiliser et personnaliser les modèles prédéfinis intégrés et disponibles avec Logic Apps.

Log Analytics

Log Analytics est une nouvelle plateforme de gestion des déploiements dans le Cloud, des datacenters sur site et des solutions hybrides.

Elle fournit plusieurs solutions modulaires, une fonction spécifique qui permet d'implémenter une fonction. Par exemple, les solutions de sécurité et d'audit permettent de définir une vue d'ensemble de la sécurité dans le cadre du déploiement des organisations. De même, il existe de nombreuses solutions, telles que l'automatisation, le suivi des modifications, etc. que vous pouvez implémenter du point de vue de la sécurité. Les services d'audit et de sécurité de Log Analytics fournissent des informations sur les cinq catégories suivantes :

- **Domaines de sécurité** : ils offrent la possibilité d'afficher les enregistrements de sécurité, les évaluations de logiciels malveillants, les évaluations de mise à jour, la sécurité réseau, les informations d'identité et d'accès et les ordinateurs avec des événements de sécurité. L'accès au tableau de bord Azure Security Center est également fourni.
- **Évaluation des logiciels malveillants** : celle-ci aide à identifier les serveurs qui ne sont pas protégés contre les logiciels malveillants et présentent des problèmes de sécurité. Cette méthode permet d'identifier les problèmes de sécurité potentiels et leur gravité. Les utilisateurs peuvent prendre des mesures proactives sur la base de ces recommandations. Les sous-catégories de l'Azure Security Center fournissent des informations recueillies par ce dernier.

- **Problèmes notoires** : cela permet d'identifier rapidement les problèmes actifs et leur gravité.
- **Détections** : cette catégorie est en mode de prévisualisation. Elle permet d'identifier des modèles d'attaque en visualisant les alertes de sécurité.
- **Renseignement sur les menaces** : cela permet d'identifier des modèles d'attaques en visualisant le nombre total de serveurs enregistrant un trafic sortant d'IP malveillantes, le type de menace malveillante, ainsi qu'une carte indiquant d'où proviennent ces IP.

Les détails susmentionnés, s'affichent dans le portail comme l'illustre la *Figure* 8.9 :

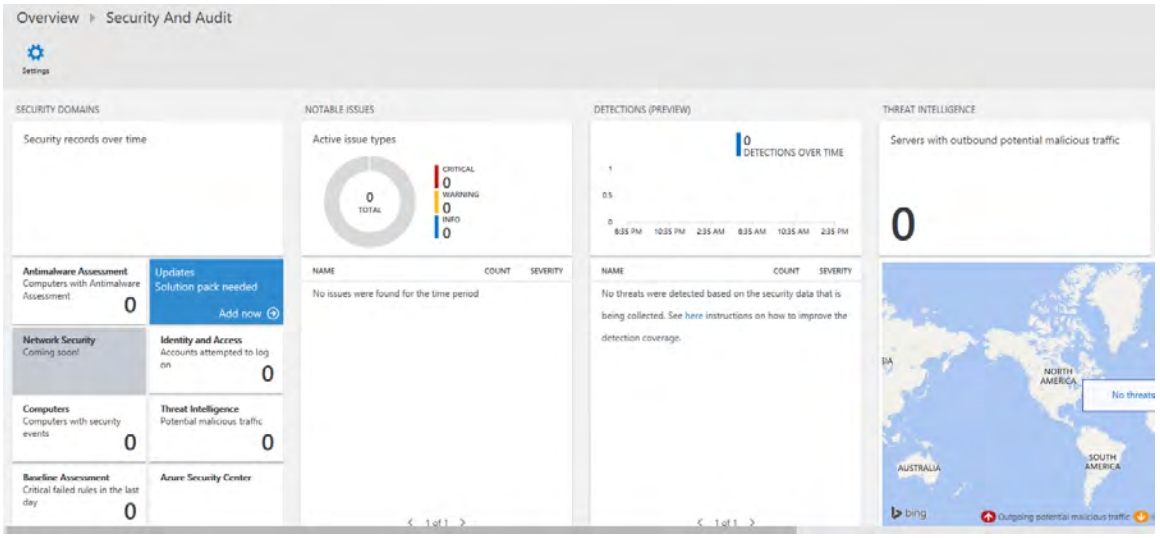


Figure 8.9 : les informations sont affichées dans le volet de sécurité et d'audit de Log Analytics

Maintenant que vous en savez un peu plus sur la sécurité des services PaaS, nous allons découvrir comment sécuriser les données stockées dans le stockage Azure.

Stockage Azure

Un compte de stockage constitue un élément important dans l'architecture globale de la solution. Les comptes de stockage peuvent stocker des informations importantes comme les **informations personnelles identifiables** (**PII**) des utilisateurs, les transactions commerciales et d'autres données sensibles ou confidentielles. Il est essentiel de sécuriser les comptes de stockage et de n'autoriser qu'un accès limité à des utilisateurs autorisés. Les données enregistrées sont chiffrées et transmises via des canaux sécurisés. Le stockage, les utilisateurs et les applications client consommant le compte de stockage et ses données doivent jouer un rôle crucial dans la sécurité globale des données. Les données doivent rester chiffrées en tout temps. Cela inclut également les informations d'identification et les chaînes de connexion aux magasins de données.

Azure fournit une fonction RBAC permettant de contrôler qui gère les comptes de stockage Azure. Ces autorisations RBAC sont accordées aux utilisateurs et aux groupes dans Azure AD. Toutefois, lorsqu'une application à déployer sur Azure est créée, ses utilisateurs et clients ne seront pas disponibles dans Azure AD. Pour permettre aux utilisateurs d'accéder au compte de stockage, le stockage Azure fournit des clés d'accès au stockage. Il existe deux types de clés d'accès au niveau du compte de stockage : primaire et secondaire. Les utilisateurs en possession de ces clés peuvent se connecter au compte de stockage. Ces clés d'accès au stockage sont utilisées pour s'authentifier afin d'accéder au compte de stockage. Les applications peuvent accéder aux comptes de stockage à l'aide de clés principales ou secondaires. Deux clés sont fournies de sorte que si la clé principale est compromise, les applications peuvent être mises à jour afin d'utiliser la clé secondaire pendant que la clé principale sera regénérée. Ceci permet de minimiser les temps d'arrêt des applications. En outre, ce système renforce la sécurité, en supprimant la clé compromise sans que cela n'ait d'impact sur les applications. Les détails de la clé de stockage, tels qu'ils apparaissent dans le portail Azure, sont indiqués à la *Figure 8.10* :

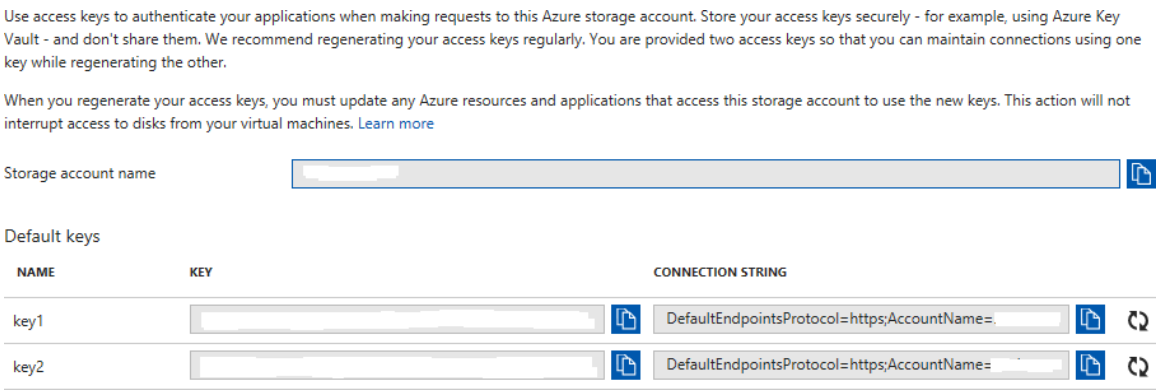


Figure 8.10 : clés d'accès d'un compte de stockage

le stockage Azure propose quatre services : blob, fichiers, files d'attente et tables dans un compte. Chacun de ces services fournit également des infrastructures destinées à leur propre sécurité via des jetons d'accès sécurisés.

Une **signature d'accès partagé** (**SAS**) est une URI qui octroie des droits d'accès limités aux services de stockage Azure : blob, fichiers, files d'attente et tables. Ces jetons SAS peuvent être partagés avec les clients qui ne sont pas de confiance sur l'ensemble de la clé du compte de stockage, pour limiter l'accès à certaines ressources de compte de stockage. En distribuant une URI de signature d'accès partagé à ces clients, l'accès aux ressources est accordé pour une période déterminée.

Les jetons SAS existent à la fois aux niveaux du compte de stockage, du blob, du fichier, de la table et de la file d'attente. La signature au niveau du compte de stockage est plus puissante et a la capacité d'accorder ou de rejeter des autorisations au niveau du service individuel. Elle peut également être utilisée à la place des niveaux de services de ressources individuelles.

Les jetons SAS fournissent un accès granulaire aux ressources et peuvent également être combinés. Ces jetons incluent la lecture, l'écriture, la suppression, la liste, l'ajout, la création, la mise à jour et le traitement. En outre, même l'accès aux ressources peut être déterminé lors de la génération de jetons SAS. Il peut s'agir de blobs, de tables, de files d'attente et de fichiers, individuels ou d'une combinaison de ces éléments. Les clés de compte de stockage sont destinées à l'ensemble du compte et ne peuvent se limiter à des services individuels. Elles ne peuvent pas non plus être limitées du point de vue des autorisations. Il est beaucoup plus simple de créer et de révoquer des jetons SAS que des clés d'accès au compte de stockage. Les jetons SAS peuvent être créés de sorte à être utilisés pour une période déterminée, suite de quoi ils deviendront automatiquement non valides.

Il convient de noter que si des clés de compte de stockage sont regénérées, le jeton SAS qui en dépend sera invalidé et un nouveau jeton SAS devra être créé et partagé avec les clients. La *Figure 8.11* illustre une option permettant de sélectionner la portée, les autorisations, la date de début, la date de fin, l'adresse IP autorisée, les protocoles autorisés et la clé de signature pour créer un jeton SAS :

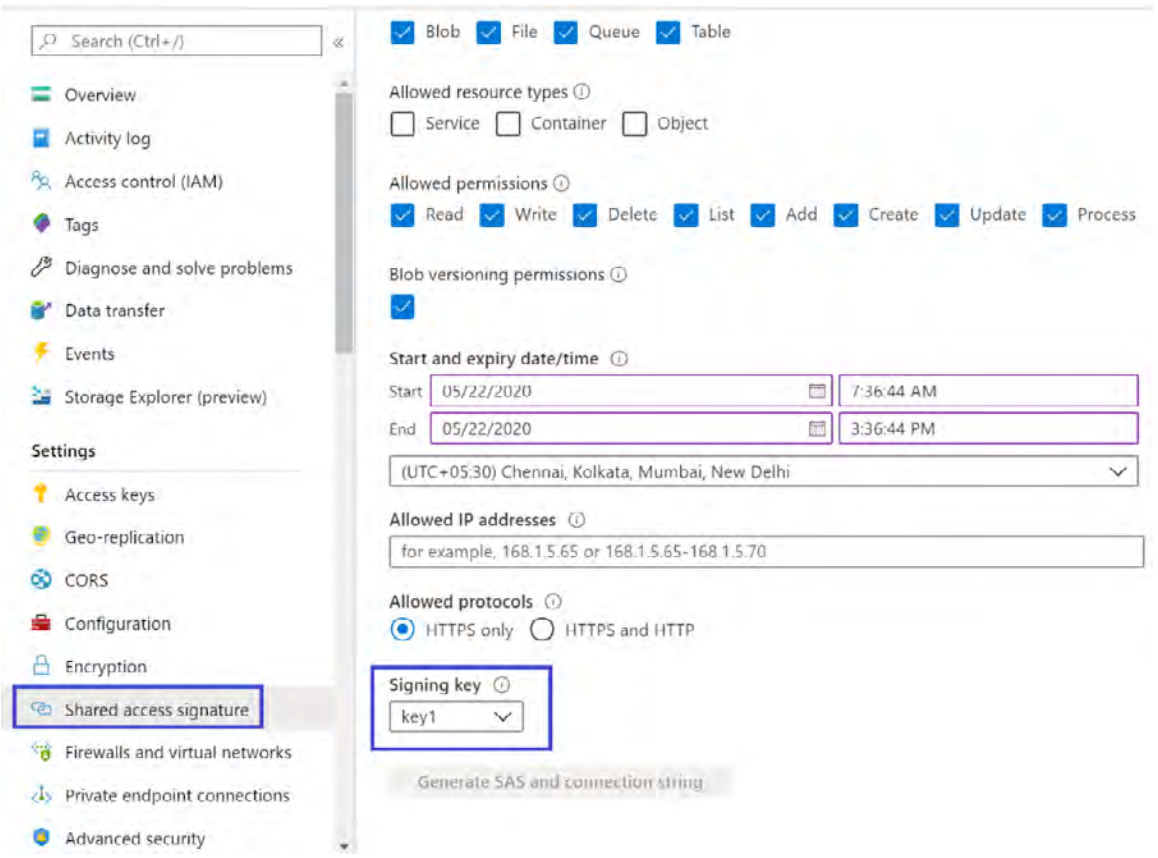


Figure 8.11 : création d'un jeton SAS

Si nous régénérons **key1**, que nous avons utilisé pour signer le jeton SAS dans l'exemple précédent, nous devons créer un jeton SAS avec **Key2** ou le nouveau **key1**.

Le vol de cookie, l'injection de scripts et les attaques DoS sont des méthodes couramment employées par les pirates informatiques afin de perturber un environnement et de dérober des données. Les navigateurs et le protocole HTTP implémentent un mécanisme intégré qui bloque l'exécution de toute activité malveillante. En règle générale, tout ce qui a trait à des interdomaines n'est pas autorisé, tant sur HTTP que sur les navigateurs. Un script s'exécutant dans un domaine ne peut pas demander les ressources d'un autre domaine. Cependant, il existe des cas d'utilisation valides où de telles requêtes peuvent être autorisées. Le protocole HTTP implémente **cross-origin resource sharing** (**CORS**). Avec CORS, il est possible d'accéder à des ressources sur plusieurs domaines et de les exploiter. Le stockage Azure permet de configurer des règles CORS pour les ressources de blobs, de fichiers, de files d'attente et de tables. Le stockage Azure permet de créer des règles qui sont examinées pour chaque requête authentifiée. Si ces règles sont respectées, la requête est autorisée à accéder à la ressource. La figure 8,12 illustre l'ajout des règles CORS à chacun des services de stockage :

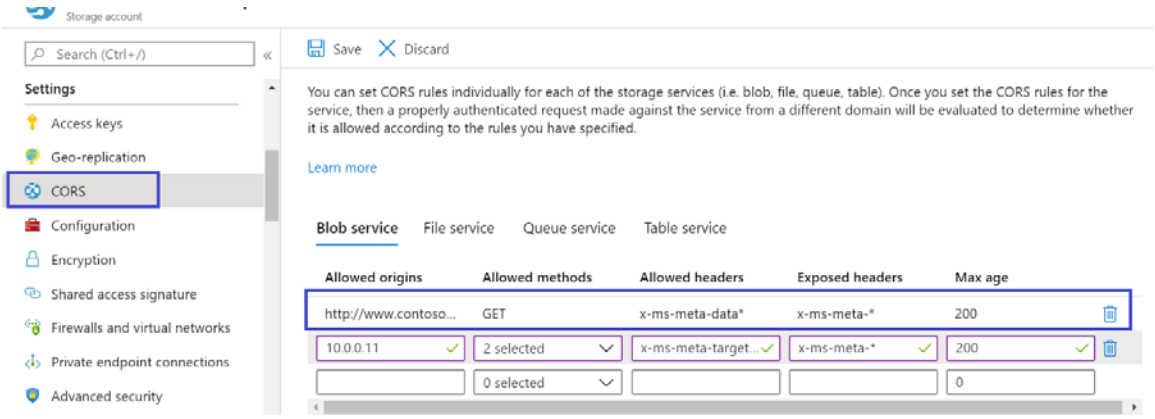


Figure 8.12 : création de règles CORS pour un compte de stockage

Les données doivent non seulement être protégées durant leur circulation, mais également lorsqu'elles sont au repos. Si les données au repos ne sont pas chiffrées, n'importe qui ayant accès au disque physique du datacenter pourra les lire. Bien que cette violation de données ne soit que très peu probable, il est conseillé aux clients de chiffrer leurs données. Le chiffrement du service de stockage permet également de protéger les données au repos. Ce service fonctionne de manière transparente et s'injecte sans que les utilisateurs ne le sachent. Il chiffre les données lorsque celles-ci sont enregistrées dans un compte de stockage et les déchiffre automatiquement lorsqu'elles sont lues. L'intégralité de ce processus se déroule sans qu'aucune activité supplémentaire ne soit requise de la part des utilisateurs.

Les clés de compte Azure doivent être alternées périodiquement. Cela permettra de s'assurer qu'un pirate informatique n'a pas la possibilité d'accéder aux comptes de stockage.

Il est également conseillé de regénérer les clés, toutefois leur utilisation dans les applications existantes doit être prise en compte. Si une telle action provoque une rupture dans l'application existante, celle-ci doit avant tout faire l'objet d'une gestion des changements, lesquels doivent être implémentés progressivement.

Il est toujours recommandé de générer des jetons SAS de niveau de service individuels avec des délais limités. Ces jetons ne doivent être fournis qu'aux utilisateurs autorisés à accéder aux ressources. Appliquez toujours le principe du moindre privilège et n'accordez que les autorisations nécessaires.

Les clés SAS et les clés de compte de stockage doivent être stockées dans Azure Key Vault. Elles garantissent un stockage sécurisé et en régulent l'accès. Ces clés peuvent être lues lors de l'exécution par des applications de Key Vault au lieu d'être stockées dans des fichiers de configuration.

En outre, vous pouvez également utiliser Azure AD pour autoriser les requêtes au stockage blob et en file d'attente. Nous allons utiliser RBAC pour accorder les autorisations nécessaires à un principal de service. Une fois que nous aurons authentifié le principal de service à l'aide d'Azure AD, un jeton OAuth 2.0 sera généré. Ce jeton peut être ajouté à l'en-tête d'autorisation de vos appels d'API pour autoriser une requête au niveau du stockage blob ou en file d'attente. Microsoft recommande d'utiliser l'autorisation Azure AD tout en travaillant avec des applications blob et en file d'attente en raison de la sécurité accrue fournie par Azure AD et de sa simplicité par rapport aux jetons SAS.

La section suivante porte sur l'évaluation des options de sécurité disponibles pour Azure SQL Database.

Azure SQL

SQL Server stocke des données relationnelles sur Azure, qui est un service de base de données relationnelle gérée. Il est également connu sous le nom de **base de données en tant que service** (**DBaaS**), qui fournit une plateforme hautement disponible, évolutive, axée sur la performance et sécurisée pour le stockage des données. Il est accessible depuis n'importe quel emplacement, quel que soit le langage ou la plateforme utilisé(e). Les clients ont besoin d'une chaîne de connexion comprenant les informations de serveur, la base de données et les données de sécurité pour s'y connecter.

SQL Server fournit des paramètres de pare-feu qui interdisent l'accès à tous les utilisateurs par défaut. Les plages et les adresses IP doivent être inscrites sur une liste blanche pour accéder à SQL Server. Les architectes ne doivent autoriser que les adresses IP qu'ils estiment de confiance et qui appartiennent aux clients ou aux partenaires. Certains déploiements dans Azure comportent un grand nombre d'adresses IP, ou des adresses IP inconnues, comme les applications déployées dans Azure Functions ou Logic Apps. Afin que ces applications aient accès à SQL Azure, Azure SQL fournit des listes blanches de toutes les adresses IP des services Azure sur tous les abonnements.

Il convient de noter que la configuration du pare-feu s'effectue au niveau du serveur et non pas de la base de données. Cela signifie que toutes les modifications apportées ici affecteront toutes les bases de données d'un serveur. La *Figure 8.13* illustre l'ajout des IP client au pare-feu en vue d'accorder l'accès au serveur :

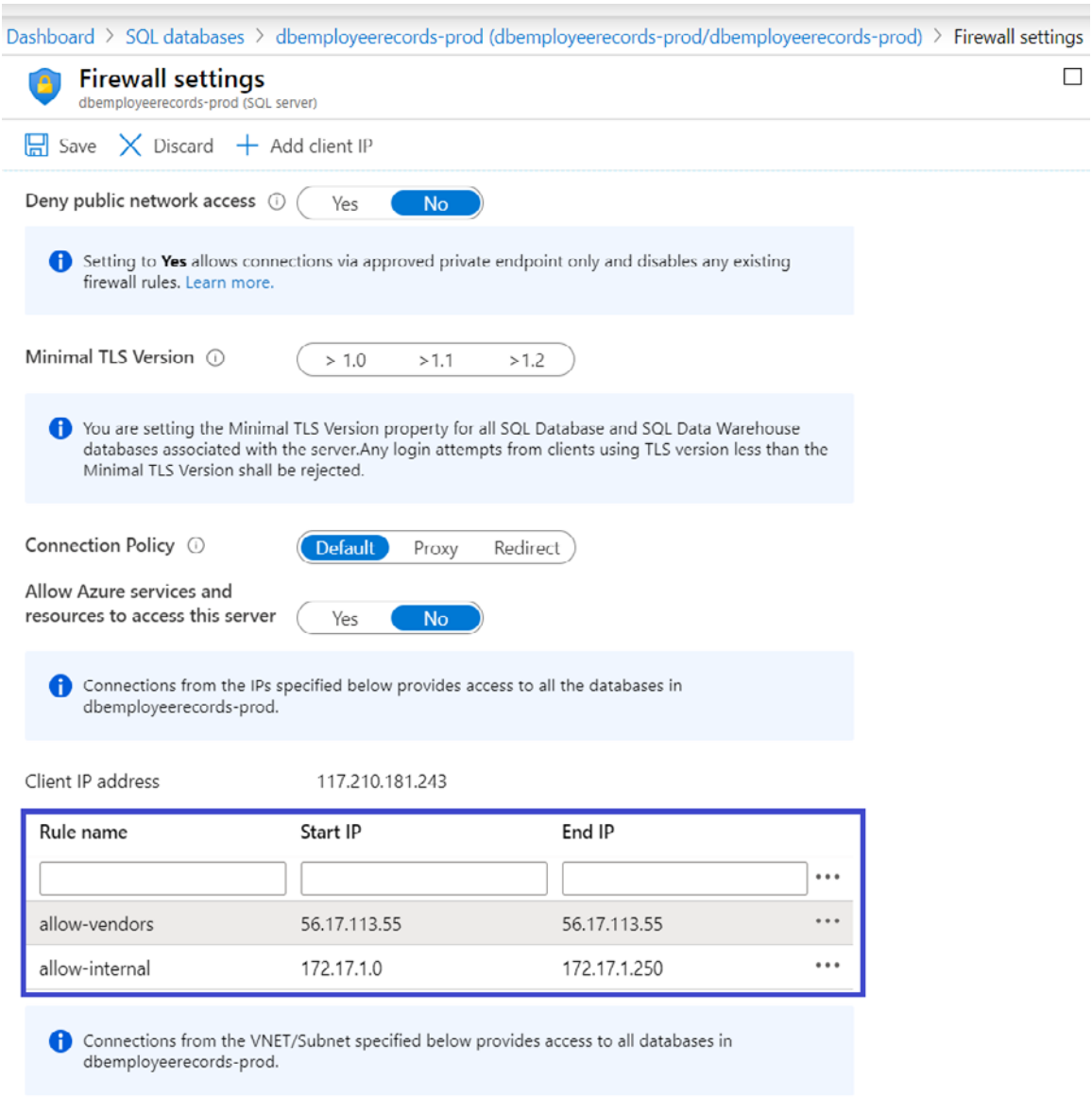


Figure 8.13 : configuration des règles de pare-feu

Azure SQL fournit également une sécurité renforcée en chiffrant les données au repos. Ainsi, personne, pas même les administrateurs du datacenter Azure, ne peut accéder aux données stockées dans SQL Server. La technologie utilisée par SQL Server pour chiffrer les données au repos est dénommée **Transparent Data Encryption** (**TDE**). Aucune modification n'est requise au niveau de l'application pour implémenter la technologie TDE. SQL Server chiffre et déchiffre les données en toute transparence lorsque l'utilisateur enregistre et lit ces données. Cette fonction est disponible au niveau de la base de données. Nous pouvons également intégrer TDE à Azure Key Vault pour activer l'option **Bring Your Own Key** (**BYOK**). À l'aide de BYOK, nous pouvons activer TDE à l'aide d'une clé gérée par le client dans Azure Key Vault.

SQL Server fournit également un **masquage dynamique des données** (Dynamic Data Masking **DDM**), ce qui est particulièrement utile pour certains types de données, telles que les informations de cartes bancaires ou les données utilisateur PII. Le masquage diffère du chiffrement. Le masquage ne chiffre pas les données, il les masque simplement, s'assurant ainsi qu'elles ne sont pas dans un format consultable de visu. Les utilisateurs doivent masquer et chiffrer les données sensibles dans Azure SQL Server.

SQL Server fournit également un service d'audit et de détection des menaces pour tous les serveurs. Il s'agit de services d'informations et de collecte de données avancés qui s'exécutent sur ces bases de données afin de détecter toute menace et vulnérabilité et prévenir les utilisateurs en conséquence. Des journaux d'audit sont maintenus par Azure dans les comptes de stockage et peuvent être consultés par les administrateurs en vue d'entreprendre une action. Des menaces, telles que l'injection SQL et les connexions client anonymes peuvent générer des alertes, et les administrateurs pourront en être informés par e-mail. La *Figure 8.14* illustre l'activation de la détection des menaces :

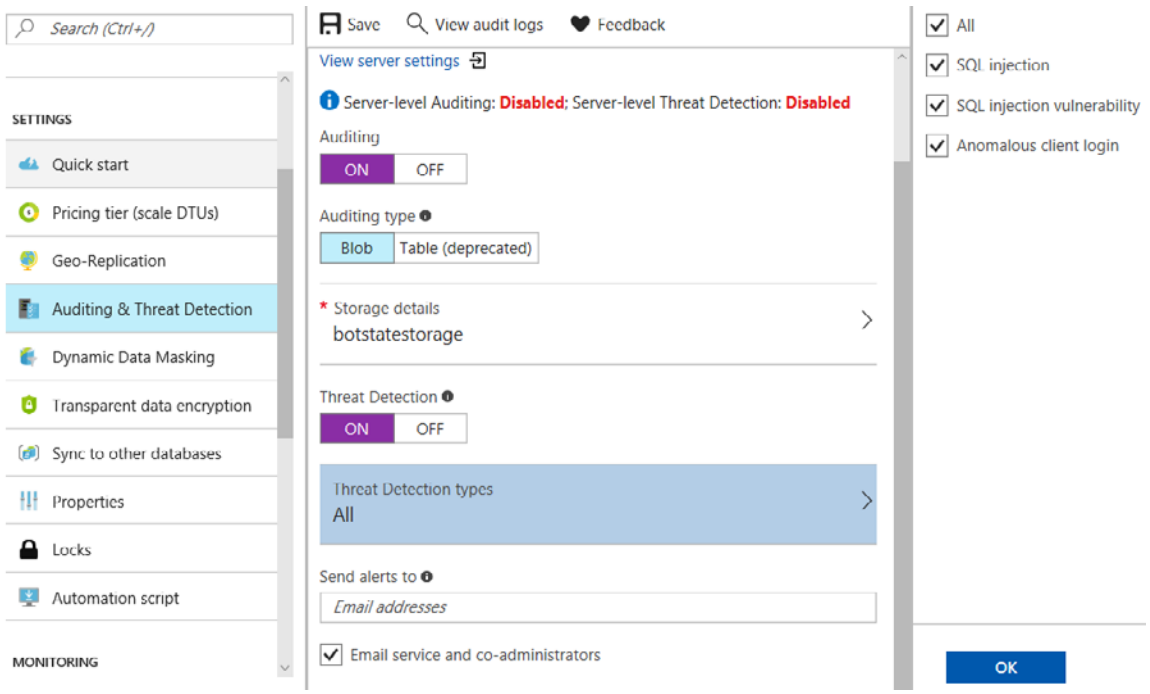


Figure 8.14 : activation de la protection contre les menaces et sélection des types de menaces à détecter

Les données peuvent être masquées dans Azure SQL. Cela permet de stocker des données dans un format illisible par les humains :

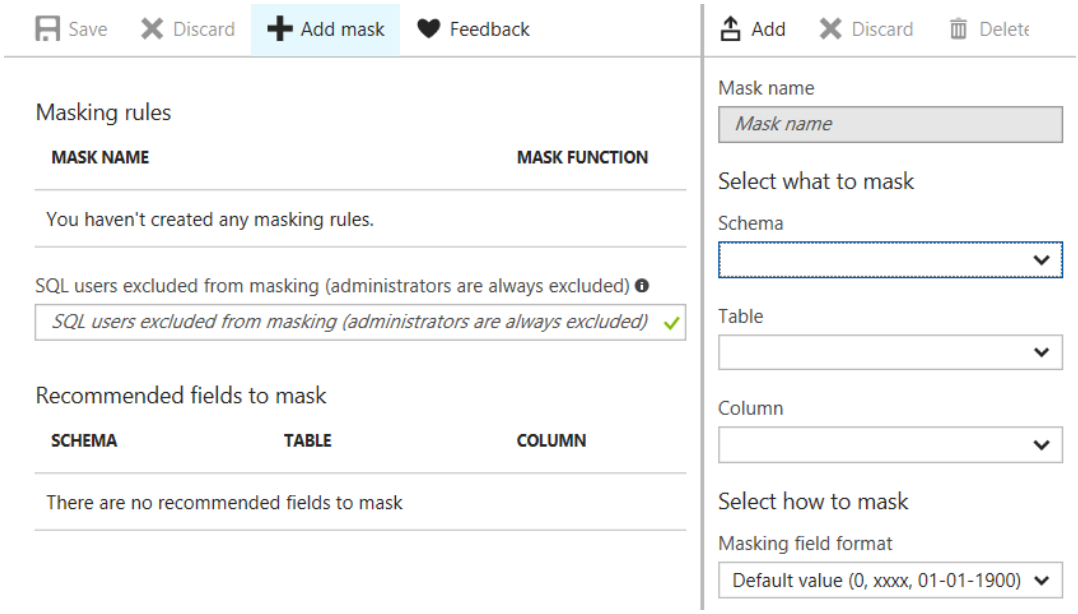


Figure 8.15 : configuration des paramètres de masquage de données

Azure SQL fournit également une fonction TDE pour chiffrer les données au repos, comme l'illustre la *Figure 8.16* :

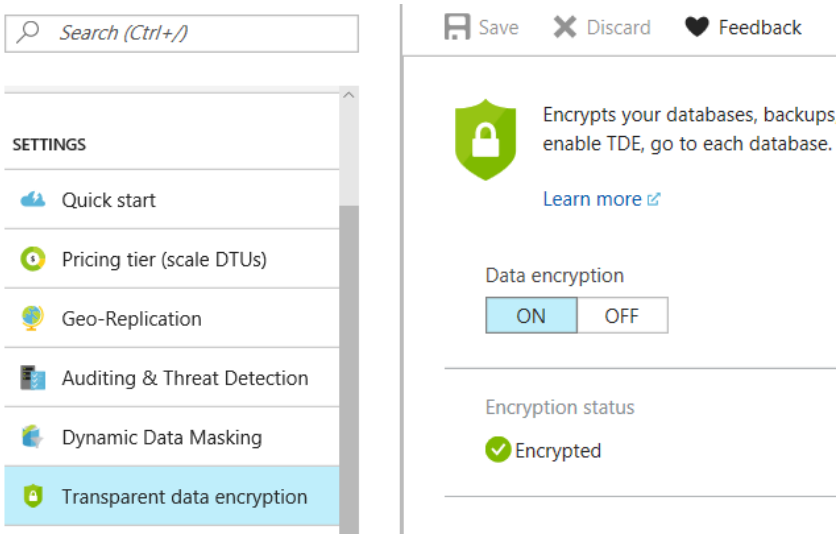


Figure 8.16 : activation de TDE

Pour effectuer une évaluation des vulnérabilités sur SQL Server, vous pouvez tirer parti de SQL Vulnerability Assessment, qui fait partie du package unifié des fonctionnalités avancées de sécurité SQL, appelées Sécurité avancée des données. SQL Vulnerability Assessment peut être utilisée par les clients de manière proactive afin d'améliorer la sécurité de la base de données en découvrant, en effectuant le suivi et en remédiant aux vulnérabilités potentielles de la base de données.

Nous avons évoqué Azure Key Vault plusieurs fois dans les sections précédentes, notamment lorsque nous avons discuté des identités gérées, de SQL Database, etc. Maintenant que vous connaissez l'objectif d'Azure Key Vault, abordons quelques méthodes permettant de sécuriser le contenu de votre coffre de clés dans la section suivante.

Azure Key Vault

La sécurisation des ressources à l'aide de mots de passe, de clés, d'informations d'identification, de certificats et d'identifiants uniques est un élément important pour tout environnement et pour toute application du point de vue de la sécurité. Ceux-ci doivent être protégés et le maintien de cette sécurité représente un pilier important de l'architecture de sécurité. La gestion et les opérations, qui conservent des secrets et sécurisent des clés, qui ne sont disponibles que lorsque nécessaire, représentent un autre élément clé à ne pas négliger. En règle générale, ces secrets sont utilisés partout, dans le code source, le fichier de configuration, sur papier et dans d'autres formats numériques. Afin de relever ces défis et de stocker tous les secrets de manière uniforme dans un stockage sécurisé et centralisé, des chambres fortes Azure doivent être utilisées.

Une chambre forte Azure est bien intégrée aux autres services Azure. Par exemple, vous pouvez simplement créer un certificat stocké dans Azure Key Vault et le déployer dans le magasin de certificats de machines virtuelles Azure. Toutes sortes de clés, y compris les clés de stockage, d'IoT et d'événements, ainsi que des chaînes de connexion, peuvent être stockées dans Azure Key Vault. Elles peuvent être récupérées et utilisées en toute transparence sans que personne ne les voit ou ne les stocke temporairement ailleurs. Des informations d'identification pour SQL Server et d'autres services peuvent également être stockées dans Azure Key Vault.

Azure Key Vault fonctionne dans une région donnée. Cela signifie qu'une ressource d'Azure Key Vault doit être mise en service dans la même région où l'application et le service sont déployés. Si un déploiement compte plusieurs régions et a besoin d'utiliser les services d'Azure Key Vault, plusieurs instances de chambres fortes Azure doivent être mises en service.

Dans Azure Key Vault, les secrets, les clés et les certificats ne sont pas stockés dans le stockage général. Ces données sensibles sont sauvegardées par le module de sécurité matérielle (HSM). Cela signifie que ces données sont stockées dans un matériel distinct sur Azure qui ne peut être déverrouillé que par les clés détenues par les utilisateurs. Pour renforcer la sécurité, vous pouvez également mettre en œuvre des points de terminaison de service de réseau virtuel dans Azure Key Vault. Ainsi, l'accès au coffre de clés sera limité à des réseaux virtuels spécifiques. Vous pouvez également restreindre l'accès à une plage d'adresses IPv4.

Dans la section *Stockage Azure*, nous avons discuté de l'utilisation d'Azure AD pour autoriser les requêtes aux blobs et aux files d'attente. Nous avons indiqué que nous utilisons un jeton OAuth, qui est obtenu à partir d'Azure AD, pour authentifier les appels d'API. Dans la section suivante, vous découvrirez comment l'authentification et l'autorisation sont réalisées à l'aide d'OAuth. Une fois que vous aurez terminé la section suivante, vous serez en mesure de relier son contenu à celui de la section *Stockage Azure*.

Autorisation et authentification à l'aide d'OAuth

Azure AD est un fournisseur d'identité capable d'authentifier les utilisateurs en fonction des utilisateurs et des principaux services déjà disponibles au sein du locataire. Azure AD met en œuvre le protocole OAuth et prend en charge l'autorisation sur Internet. Il met en œuvre un serveur d'autorisation et des services pour activer les flux d'autorisation OAuth, implicites et ceux relatifs aux informations d'identification client. Il s'agit de différents flux d'interaction OAuth bien documentés entre les applications clientes, les points de terminaison d'autorisation, les utilisateurs et les ressources protégées.

Azure AD prend également en charge l'**authentification unique** (**SSO**), qui renforce la sécurité et facilite la connexion aux applications enregistrées avec Azure AD. Vous pouvez utiliser OpenID Connect, OAuth, SAML, une option basée sur un mot de passe ou la méthode SSO liée ou désactivée lors du développement de nouvelles applications. Si vous ne savez pas quelle méthode utiliser, consultez l'organigramme de Microsoft ici : https://docs.microsoft.com/azure/active-directory/manage-apps/what-is-single-sign-on#choosing-a-single-sign-on-method.

Les applications Web, les applications basées sur JavaScript et les applications clientes natives (telles que les applications mobiles et de bureau) peuvent utiliser Azure AD à la fois pour l'authentification et l'autorisation. Certaines plateformes de réseaux sociaux, telles que Facebook, Twitter, etc., qui prennent en charge le protocole OAuth pour l'autorisation.

L'une des façons les plus simples d'activer l'authentification pour les applications Web à l'aide de Facebook est illustrée ci-après. D'autres méthodes qui utilisent des binaires de sécurité peuvent être utilisées. Ce sujet n'est toutefois pas couvert dans ce livre.

Dans cette procédure pas à pas, un service Azure App Service sera configuré ainsi qu'un plan App Service pour héberger une application Web personnalisée. Un compte Facebook valide sera nécessaire en tant que condition préalable afin de rediriger les utilisateurs vers celui-ci pour l'authentification et l'autorisation.

Un groupe de ressources peut être créé à l'aide du portail Azure, comme illustré à la *Figure* 8.17 :

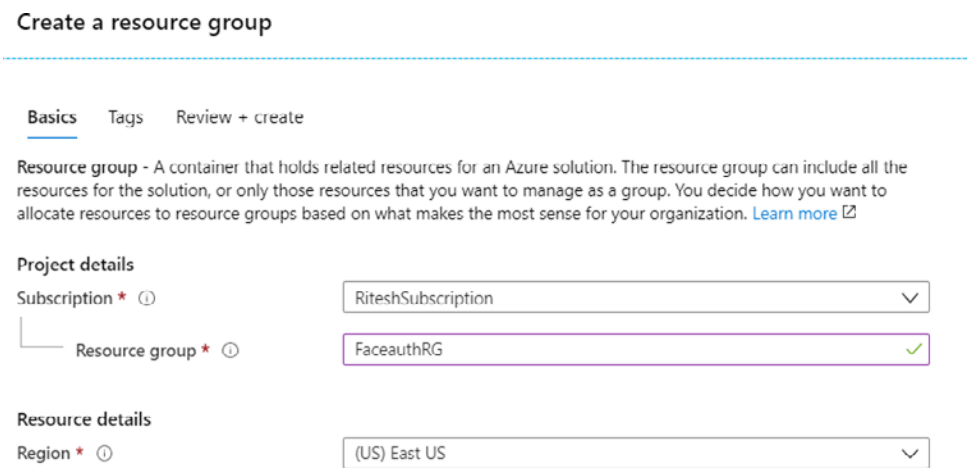


Figure 8.17 : création d'un groupe de ressources

Une fois que le groupe de ressources a été créé, un service d'application peut être créé à l'aide du portail, comme illustré à la *Figure* 8.18 :

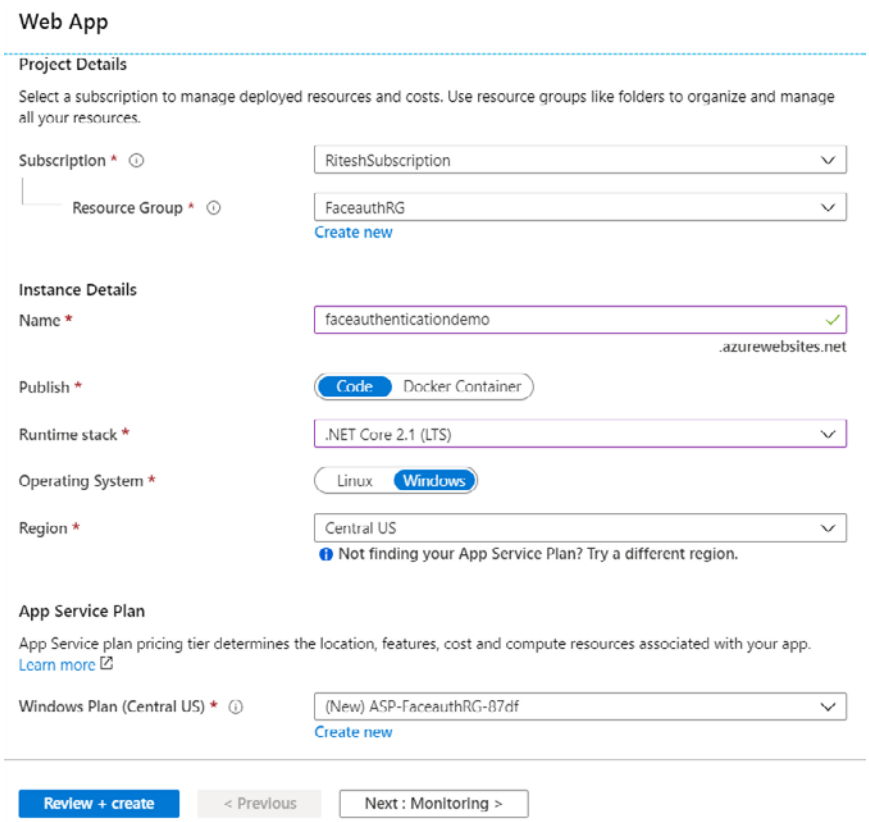


Figure 8.18 : création d'une application

Il est important de noter l'URL de l'application Web. En effet, celle-ci sera nécessaire par la suite lors de la configuration de l'application Facebook.

Une fois l'application Web configurée dans Azure, l'étape suivante consiste à créer une application dans Facebook. Cette étape permet de représenter votre application Web dans Facebook et de générer des informations d'identification client appropriées pour l'application Web. C'est ainsi que Facebook reconnaît l'application Web.

Accédez à developers.facebook.com et connectez-vous à l'aide des informations d'identification appropriées. Créez une application en sélectionnant l'option **Create App** (Créer une application) dans **My Apps** (Mes applications) dans le coin supérieur droit, comme illustré à la *Figure* 8.19 :

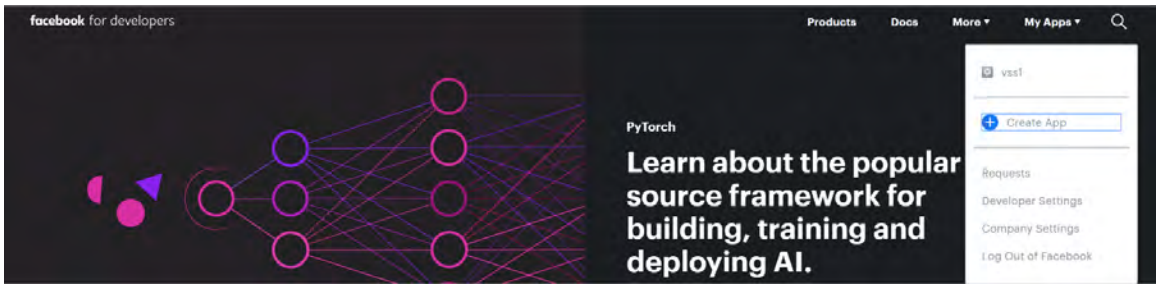


Figure 8.19 : création d'une application à partir du portail de développement Facebook

La page Web vous invitera à saisir un nom à l'application Web pour créer une application dans Facebook :

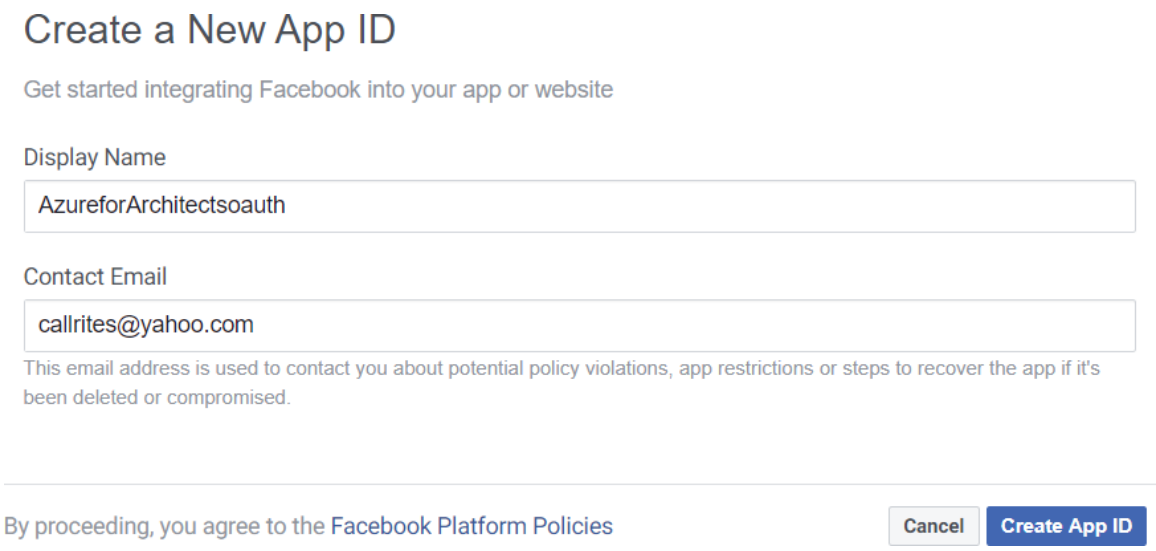


Figure 8.20 : ajout d'une nouvelle connexion

Ajoutez un nouveau produit **Facebook Login** (Connexion via Facebook) et cliquez sur **Set Up** (Configurer) pour configurer la connexion de manière à ce que l'application Web personnalisée soit hébergée sur Azure App Service :

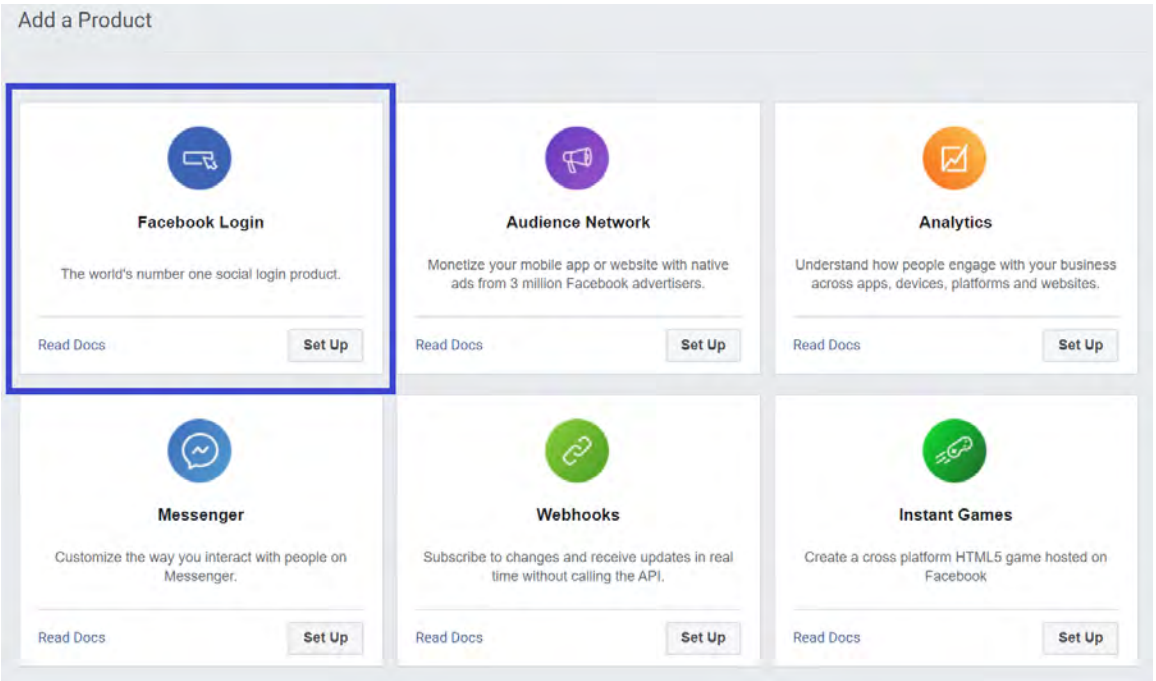


Figure 8.21 : ajout d'une connexion Facebook à l'application

Le bouton **Set Up** (Configurer) propose quelques options, comme illustré à la *Figure 8.22*. Ces options configurent le flux OAuth, tel que le flux d'autorisation, le flux implicite ou le flux d'informations d'identification du client. Sélectionnez l'option **Web** car elle est nécessaire à l'autorisation Facebook :

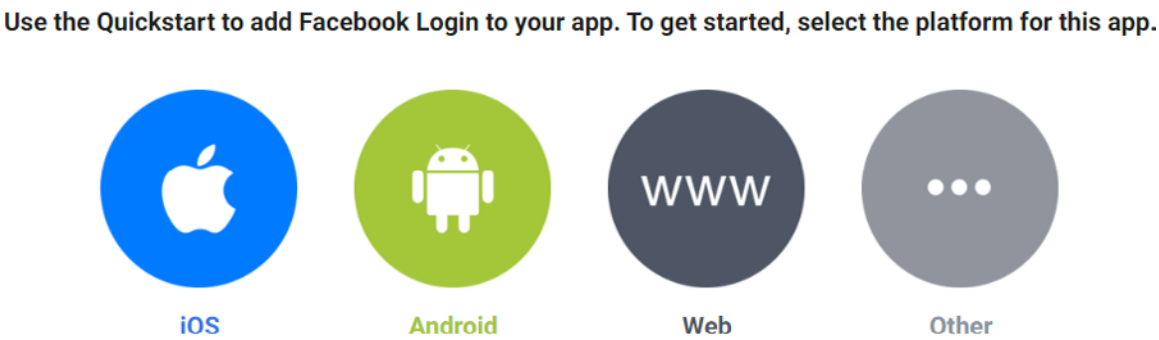


Figure 8.22 : sélection de la plateforme

Indiquez l'URL de l'application Web que nous avons notée plus tôt après la mise en service de l'application Web sur Azure :

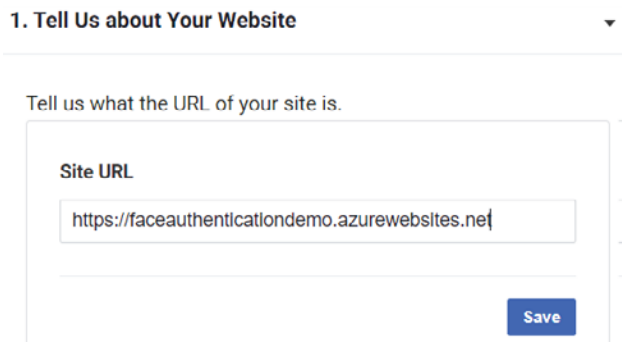


Figure 8.23 : saisie de l'URL du site sur l'application

Cliquez sur l'option **Settings** (Paramètres) dans le menu de gauche et indiquez l'URL de redirection OAuth pour l'application. Azure dispose déjà d'URL de rappel bien définies pour chacune des plateformes de réseaux sociaux populaires, et celle utilisée pour Facebook est **`domain name/.auth/login/facebook/callback`** :

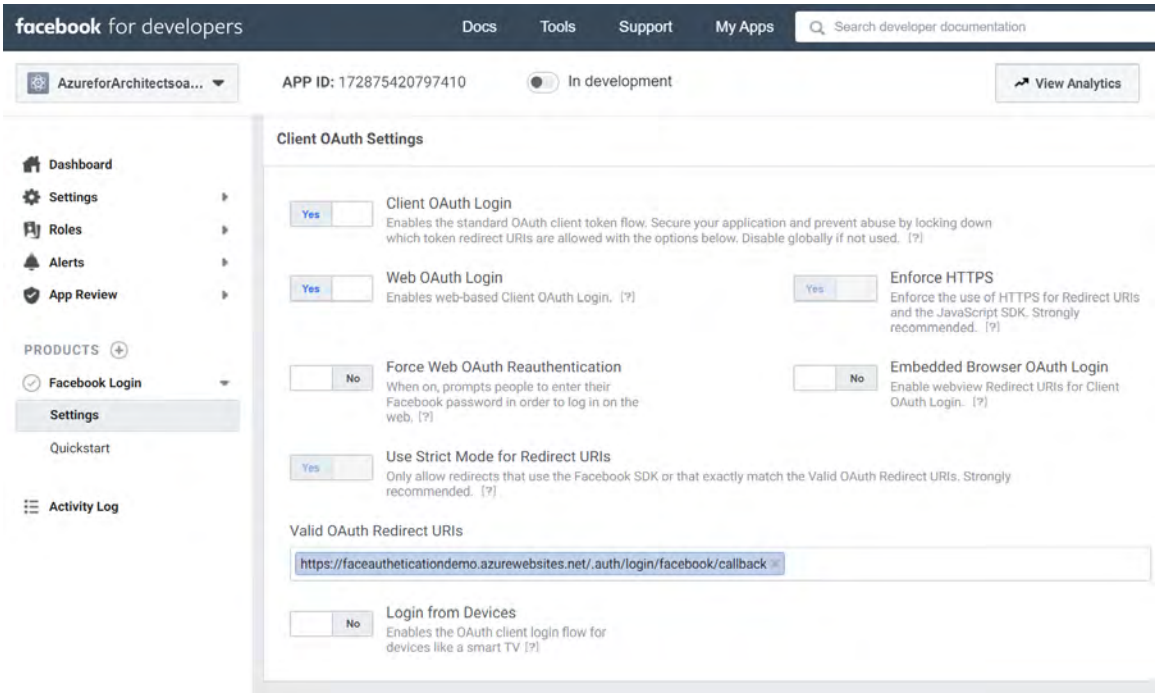


Figure 8.24 : ajout d'URI de redirection OAuth

Accédez aux paramètres **Basic** (Bases) dans le menu de gauche et notez les valeurs des champs **App ID** (ID d'application) et **App Secret** (Secret d'application). Ces éléments sont nécessaires à la configuration de l'authentification/autorisation Azure App Services :

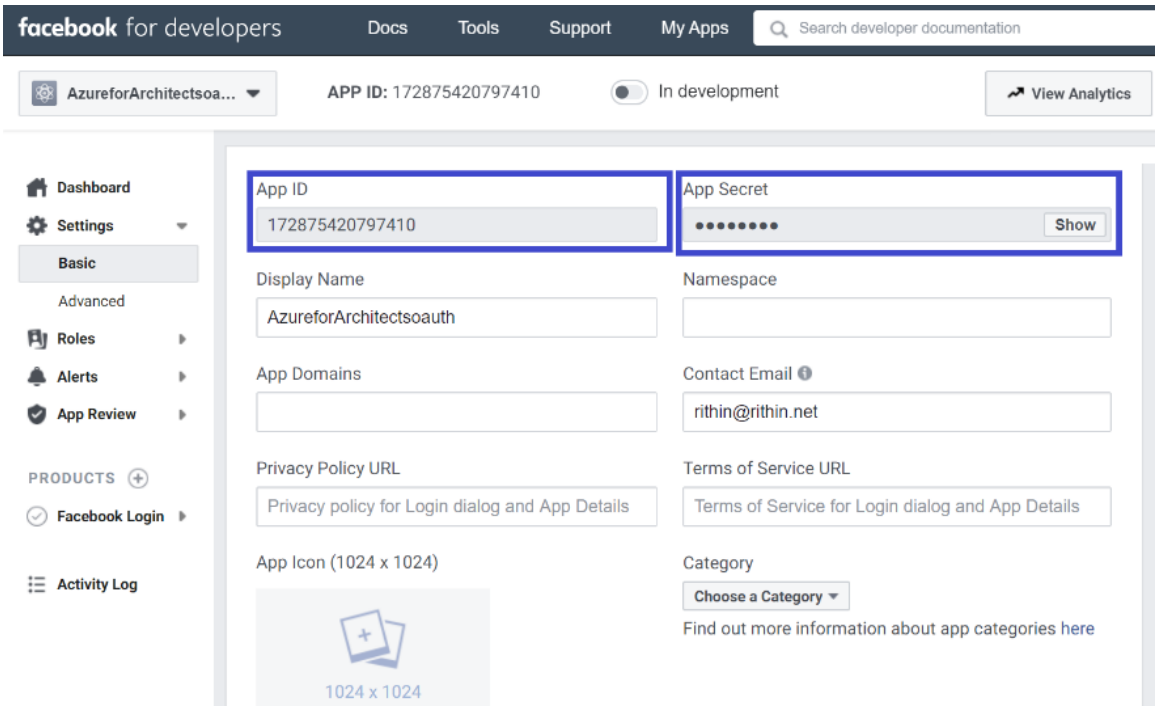


Figure 8.25 : recherche de l'ID d'application et du secret d'application

Dans le portail Azure, retournez au service Azure App Service créé lors des premières étapes de cette section et accédez au panneau authentification/autorisation. Activez l'option **App Services Authentication**, sélectionnez **Log in with Facebook** (Se connecter avec Facebook) pour l'authentification, puis cliquez sur l'élément **Facebook** dans la liste :

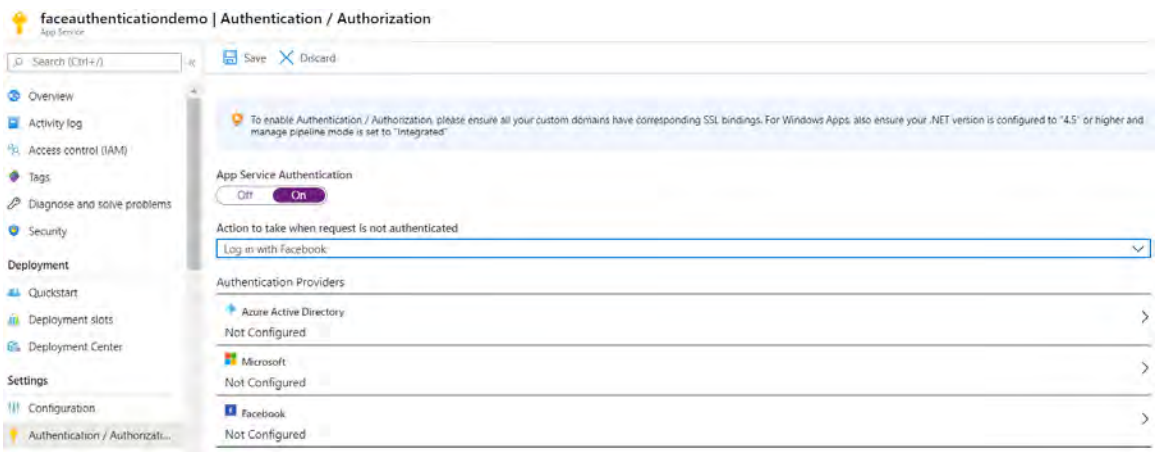


Figure 8.26 : activation de l'authentification Facebook dans App Service

Sur la page résultante, indiquez l'ID d'application et le secret d'application déjà notés, et sélectionnez la portée. La portée détermine les informations communiquées par Facebook à l'application Web :

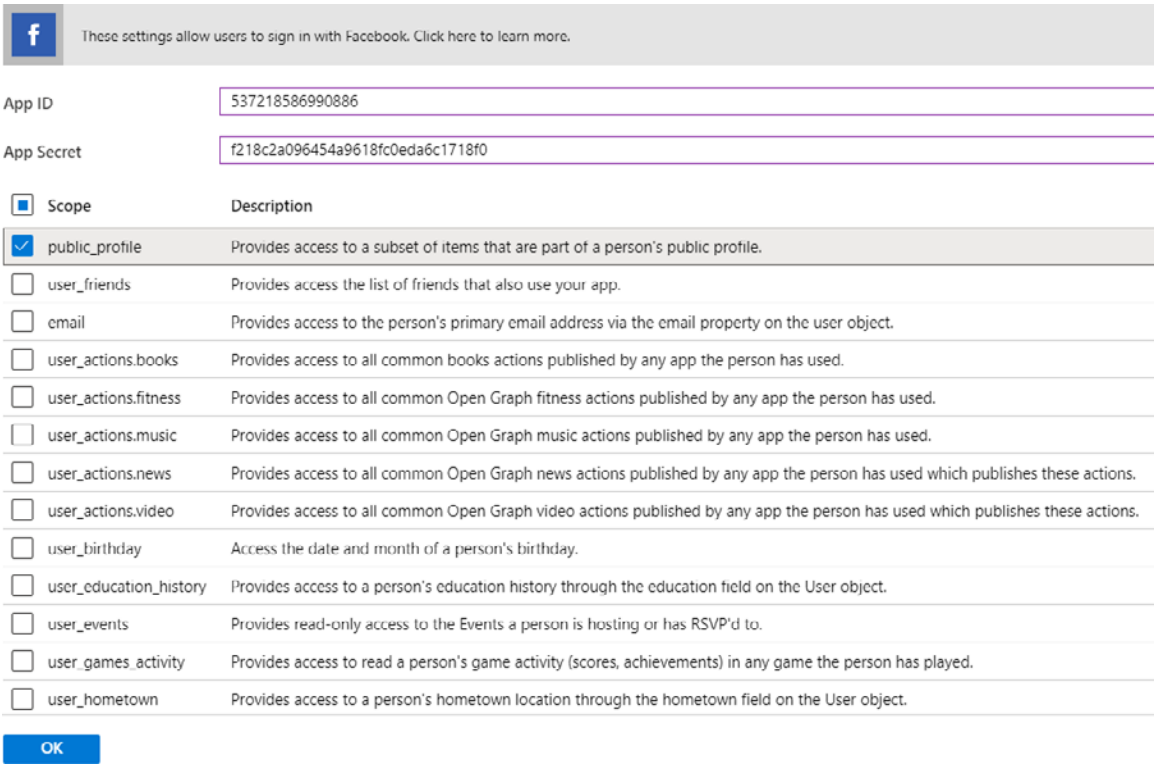


Figure 8.27 : sélection de la portée

Cliquez sur **OK** et sur le bouton **Save** (Enregistrer) pour enregistrer les paramètres d'authentification/d'autorisation.

Maintenant, si une nouvelle session de navigation privée est lancée et si vous accédez à l'application Web personnalisée, la demande doit être redirigée vers Facebook. Comme vous l'avez peut-être vu sur d'autres sites Web, vous serez invité à fournir vos informations d'identification si vous sélectionnez **Log in avec Facebook** (Se connecter avec Facebook) :

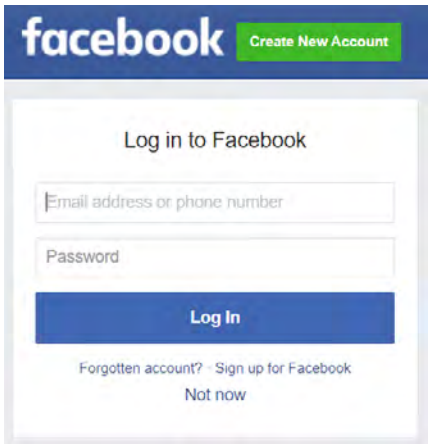


Figure 8.28 : connexion au site Web à l'aide de Facebook

Une fois vos informations d'identification saisies, une boîte de dialogue relative au consentement de l'utilisateur vous demandera l'autorisation de partager les données issues de Facebook avec l'application Web :

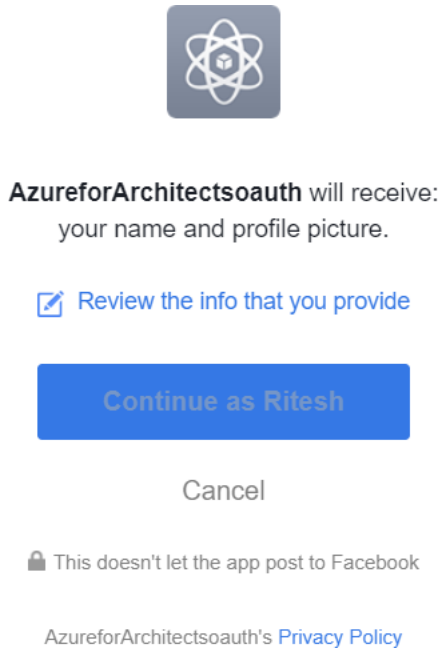


Figure 8.29 : consentement de l'utilisateur concernant le partage d'informations avec l'application

Si le consentement est accordé, la page Web de l'application Web doit apparaître :

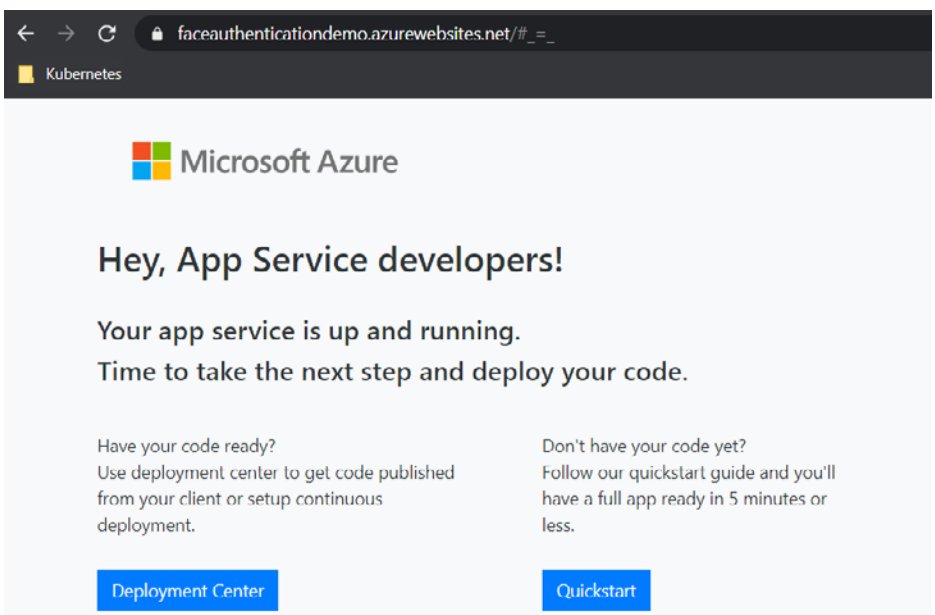


Figure 8.30 : accès à la page de destination

Une approche similaire peut être utilisée pour protéger votre application Web à l'aide d'Azure AD, de Twitter, de Microsoft et de Google. Vous pouvez également intégrer votre propre fournisseur d'identité si nécessaire.

L'approche indiquée ici illustre uniquement l'une des façons de protéger un site Web à l'aide d'informations d'identification stockées ailleurs et l'autorisation d'applications externes pour accéder aux ressources protégées. Azure propose également des bibliothèques JavaScript et des ensembles .NET dédiés à l'utilisation de la méthode de programmation impérative pour consommer les points de terminaison OAuth fournis par Azure AD et d'autres plateformes de réseaux sociaux. Il est recommandé d'utiliser cette approche pour optimiser le contrôle et la flexibilité de l'authentification et de l'autorisation au sein de vos applications.

Jusqu'à présent, nous avons discuté des fonctions de sécurité et de leur mise en œuvre. Il est également utile de mettre en place un système de surveillance et d'audit. La mise en œuvre d'une solution d'audit permettra à votre équipe de sécurité d'auditer les journaux et de prendre des mesures de précaution. La section suivante porte sur les solutions de surveillance et d'audit de sécurité dans Azure.

Surveillance et audit de la sécurité

Chaque activité effectuée dans votre environnement, qu'il s'agisse des e-mails ou de la modification d'un pare-feu, peut être catégorisée comme un événement de sécurité. Du point de vue de la sécurité, il est nécessaire de mettre en place un système de journalisation central pour surveiller et suivre les modifications apportées. Si des activités suspectes sont identifiées au cours d'un audit, vous pouvez déterminer le défaut de l'architecture et la façon d'y remédier. En outre, en cas de violation de données, les journaux aideront les professionnels de la sécurité à comprendre le modèle d'une attaque et la façon dont il a été exécuté. Des mesures préventives nécessaires peuvent également être prises pour éviter des incidents similaires à l'avenir. Azure fournit deux ressources de sécurité importantes pour gérer tous les aspects de la sécurité d'un abonnement Azure, des groupes de ressource et des ressources :

- Azure Monitor
- Azure Security Center

Commençons par étudier Azure Monitor.

Azure Monitor

Azure Monitor est un emplacement unique destiné au suivi des ressources Azure. Il fournit des informations sur les ressources Azure et sur leur état. Il fournit également une interface de requête enrichie, où les informations peuvent être décomposées et classées à l'aide des données aux niveaux de l'abonnement, du groupe de ressources, de la ressource individuelle et du type de ressource. Azure Monitor recueille les données provenant de nombreuses sources de données, y compris les mesures et les journaux d'Azure, les applications clientes et les agents s'exécutant dans des machines virtuelles. D'autres services, tels qu'Azure Security Center et Network Watcher, ingèrent également des données dans l'espace de travail Log Analytics, lesquelles peuvent être analysées à l'aide d'Azure Monitor. Vous pouvez utiliser les API REST pour envoyer des données personnalisées à Azure Monitor.

Azure Monitor peut être utilisé à via le portail Azure, PowerShell, l'interface CLI et les API REST :

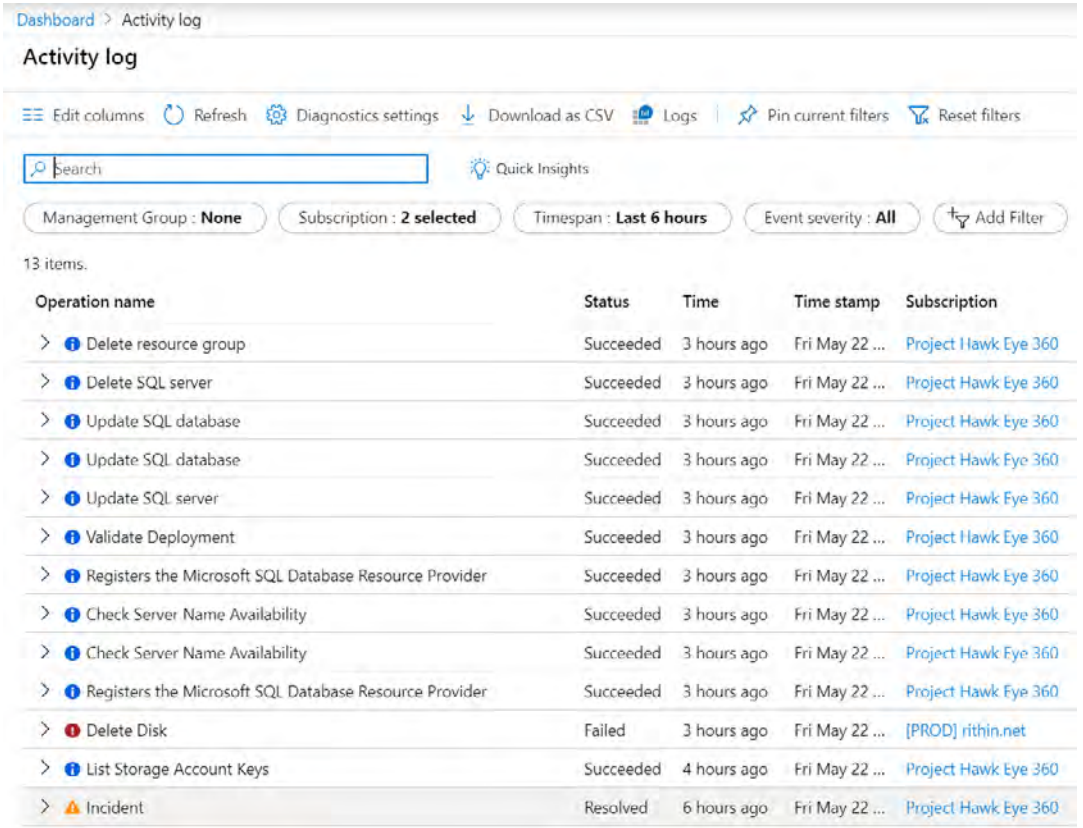


Figure 8.31 : exploration des journaux d'activité

Les journaux suivants sont fournis par Azure Monitor :

- **Journal Activité** : celui-ci fournit toutes les opérations au niveau de la gestion effectuées sur les ressources. Il fournit des informations sur le moment de la création, le créateur, le type de ressource et l'état des ressources.
- **Journal des opérations (classique)** : il fournit des informations sur toutes les opérations réalisées sur les ressources au sein d'un groupe de ressources et d'un abonnement.
- **Mesures** : celles-ci aident à obtenir des informations au niveau de la performance des ressources individuelles et définit des alertes en fonction.
- **Paramètres de diagnostics** : ces paramètres permettent de configurer les journaux d'effets en définissant le stockage Azure pour stocker les journaux, les envoyer en temps réel dans les hubs d'événements Azure, puis à Log Analytics.
- **Recherche de fichiers journaux** : elle permet d'intégrer Log Analytics avec Azure Monitor.

Azure Monitor permet d'identifier les incidents liés à la sécurité et de prendre des mesures appropriées. Il est important que seules les personnes autorisées puissent accéder à Azure Monitor, qui peut contenir des informations sensibles.

Azure Security Center

Azure Security Center, comme son nom l'indique, est un emplacement unique pour tous les besoins de sécurité. Il existe généralement deux activités liées à la sécurité : la mise en œuvre de la sécurité et la surveillance des menaces et des violations. Le centre de sécurité a été principalement conçu pour soutenir ces deux activités. Azure Security Center permet aux utilisateurs de définir leurs stratégies de sécurité et de les appliquer sur les ressources Azure. Selon l'état actuel des ressources Azure, Azure Security Center fournit des recommandations de sécurité afin de renforcer la solution et les ressources individuelles d'Azure. Ces recommandations incluent quasiment toutes les bonnes pratiques de sécurité pour les ressources Azure, notamment le chiffrement des données et des disques, la protection du réseau, la protection du point de terminaison, les listes de contrôle d'accès, la mise en liste blanche des requêtes entrantes, ainsi que le blocage des requêtes non autorisées. Les ressources comprennent aussi bien les composants de l'infrastructure, comme les équilibreurs de charge, les groupes de sécurité réseau et les réseaux virtuels que les ressources PaaS comme Azure SQL et le stockage Azure. Voici un extrait du volet **Overview** (Aperçu) d'Azure Security Center, qui illustre le degré de sécurité global de l'abonnement, l'hygiène de la sécurité des ressources et bien plus encore :

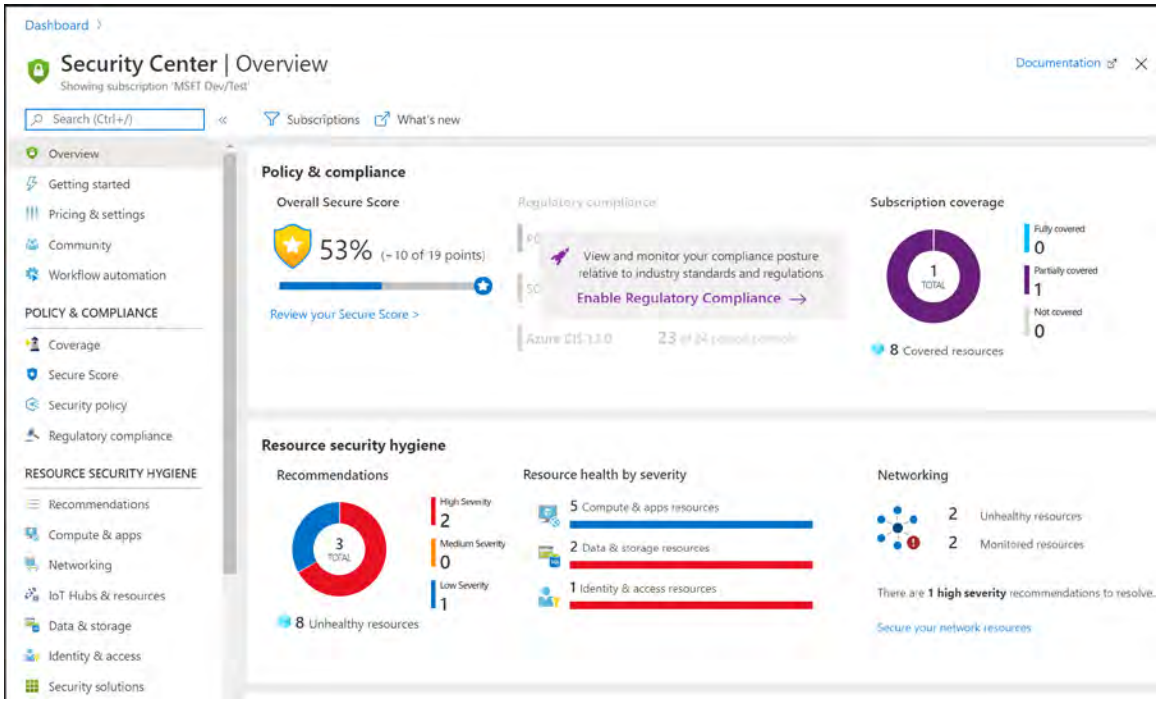


Figure 8.32 : aperçu d'Azure Security Center

Azure Security Center est une plateforme enrichie qui fournit des recommandations pour plusieurs services, comme illustré à la *Figure* 8.33. Ces recommandations peuvent également être exportées vers des fichiers CSV à titre de référence :

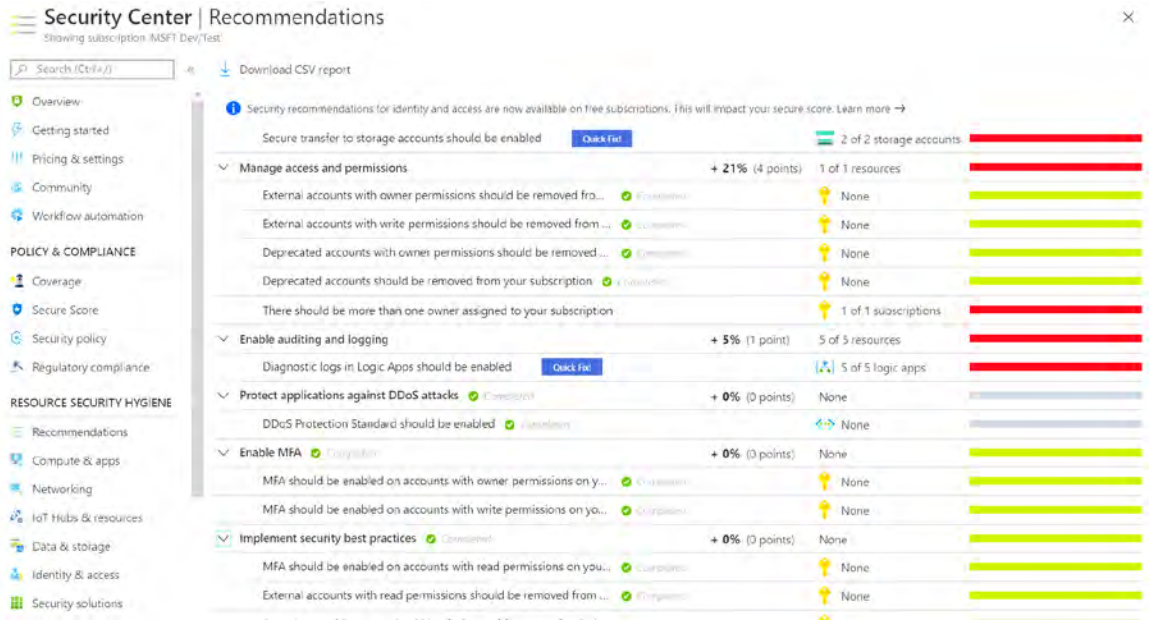


Figure 8.33 : recommandations formulées par Azure Security Center

Comme nous l'avons mentionné au début de cette section, la surveillance et l'audit sont cruciaux dans un environnement d'entreprise. Azure Monitor peut présenter plusieurs sources de données et peut être utilisé pour auditer les journaux à partir de ces sources. Azure Security Center fournit des évaluations continues et des recommandations de sécurité hiérarchisées, ainsi que le degré de sécurité global.

Résumé

La sécurité constitue toujours un aspect important pour n'importe quel déploiement ou solution. Celle-ci a gagné en importance et en pertinence avec les déploiements dans le Cloud. En outre, les cyberattaques font peser de plus en plus de menaces. Par conséquent, la sécurité est devenue un enjeu essentiel pour les organisations. Quel que soit le type de déploiement et la solution, IaaS, PaaS ou SaaS, la sécurité est essentielle. Les datacenters Azure sont complètement sécurisés et disposent d'une dizaine de certifications de sécurité internationales. Ils sont sécurisés par défaut. Ils prévoient des ressources de sécurité individuelles telles que les groupes NSG, les traductions d'adresse réseau, les points de terminaison sécurisés, les certificats, les chambres fortes, le chiffrement du stockage et des machines virtuelles et les fonctions de sécurité PaaS pour les ressources individuelles PaaS. La sécurité suit un cycle de vie complet autonome, et elle doit être correctement planifiée, conçue, implémentée et testée, au même titre que toute autre fonction de l'application.

Nous avons discuté des pare-feux du système d'exploitation et du pare-feu Azure, tout en évoquant la façon dont ils permettent de renforcer la sécurité globale de votre solution. Nous avons également évoqué les nouveaux services Azure, tels qu'Azure bastion, Azure Front Door et Azure Private Link.

La sécurité des applications était un autre point essentiel, nous avons discuté de l'authentification et de l'autorisation réalisées à l'aide d'OAuth. Nous avons démontré brièvement comment créer un service d'application et intégrer la connexion Facebook. Facebook n'était qu'un exemple ; vous pouvez utiliser Google, Twitter, Microsoft, Azure AD ou n'importe quel fournisseur d'identité personnalisé.

Nous avons également exploré les options de sécurité offertes par Azure SQL, qui est un service de base de données géré et proposé par Azure. Nous avons discuté de la mise en œuvre des fonctions de sécurité, et dans la section finale, nous avons conclu avec la surveillance et l'audit à l'aide d'Azure Monitor et d'Azure Security Center. La sécurité joue un rôle essentiel dans votre environnement. Un architecte doit toujours concevoir et créer des solutions dont l'architecture repose sur la sécurité. Azure offre de nombreuses options pour y parvenir.

Maintenant que vous savez comment sécuriser vos données dans Azure, passons au chapitre suivant pour étudier les solutions Big Data d'Hadoop, de Data Lake Storage, de Data Lake Analytics et de Data Factory.

9

Solutions de Big Data Azure

Le chapitre précédent portait sur les différentes stratégies de sécurité qui peuvent être mises en œuvre sur Azure. Une application sécurisée nous permet de gérer de vastes quantités de données. Ces dernières années, le Big Data a suscité beaucoup d'intérêt. Des outils, des logiciels et des stockages spécialisés sont requis pour le gérer. Il est intéressant de noter que ces outils, plateformes et options de stockage n'étaient pas disponibles en tant que services il y a quelques années. Toutefois, avec la nouvelle technologie Cloud, Azure fournit de nombreux outils, plateformes et ressources pour créer facilement des solutions de Big Data. Ce chapitre détaille l'architecture complète de l'ingestion, du nettoyage, du filtrage et de la visualisation des données d'une manière significative.

Dans ce chapitre, nous allons aborder les thèmes suivants :

- Présentation du Big Data
- Intégration des données
- **Extract-Transform-Load** (**ETL**)
- Data Factory
- Data Lake Storage
- Écosystèmes d'outils, notamment Spark, Databricks et Hadoop
- Databricks

Big Data

Avec l'afflux d'appareils bon marché, tels que les appareils de l'Internet des objets et les appareils portables, la quantité des données générées et collectées a considérablement augmenté. Presque toutes les organisations disposent d'une grande quantité de données et sont prêtes à en acheter davantage si nécessaire. Lorsque de grandes quantités de données sont collectées dans différents formats et que leur nombre augmente, nous les désignons alors sous le terme Big Data. Le Big Data présente trois caractéristiques clés :

- **Volume** : par volume, nous entendons la quantité de données à la fois en termes de taille (en Go, To et Po, par exemple) et en termes de nombre d'enregistrements (comme dans un million de lignes dans une banque de données hiérarchique, 100 000 images, un demi-milliard de documents JSON, etc.).
- **Vitesse** : la vitesse désigne la vitesse à laquelle les données sont reçues ou ingérées. Si les données ne changent pas fréquemment ou si de nouvelles données ne sont pas collectées fréquemment, la vitesse des données est faible, tandis que s'il y a des mises à jour fréquentes et que beaucoup de nouvelles données sont reçues régulièrement, on dit qu'elles présentent une vitesse élevée.
- **Variété** : la variété fait référence à différents types et formats de données. Les données peuvent provenir de différentes sources dans différents formats. Les données peuvent être reçues en tant que données structurées (comme dans les fichiers séparés par des virgules, les fichiers JSON ou les données hiérarchiques), en tant que bases de données semi-structurées (comme dans les documents présentant des schémas sans donnée NoSQL), ou en tant que données non structurées (telles que les blobs les, images, fichiers PDF, etc.). Avec autant de variantes, il est important de définir un processus pour le traitement des données ingérées.

Dans la section suivante, nous allons examiner le processus général du Big Data.

Processus pour le Big Data

Lorsque les données proviennent de plusieurs sources dans divers formats et à des vitesses différentes, il est important de définir un processus de stockage, d'assimilation, de filtrage et de nettoyage des données afin que nous puissions faciliter le traitement de ces données et les rendre utiles pour d'autres processus. Un processus bien défini est nécessaire pour la gestion des données. Le processus général de Big Data qui doit être suivi est illustré à la *Figure 9.1* :

Figure 9.1 : processus du Big Data

Le traitement du Big Data comporte quatre étapes principales. Examinons-les en détail :

- **Ingestion** : il s'agit du processus d'introduction et d'ingestion des données dans l'environnement Big Data. Les données peuvent provenir de plusieurs sources, et les connecteurs doivent être utilisés pour ingérer ces données au sein de la plateforme Big Data.
- **Stockage** : une fois ingérées, les données doivent être stockées dans le pool de données pour le stockage à long terme. Le stockage doit être appliqué pour les données historiques et en temps réel et doit englober les données structurées, semi-structurées et non structurées. Des connecteurs doivent pouvoir lire les données à partir de sources de données, ou les sources de données doivent être en mesure de pousser les données vers le stockage.
- **Analyse** : une fois les données lues à partir du stockage, elles doivent être analysées. Ce processus nécessite le filtrage, le regroupement, la liaison et la transformation des données pour recueillir des informations.
- **Visualisation** : l'analyse peut être envoyée en tant que rapports à l'aide de plusieurs plateformes de notification ou utilisée pour générer des tableaux de bord avec des graphiques et des diagrammes.

Auparavant, les outils nécessaires pour collecter, ingérer, stocker et analyser les Big Data n'étaient pas facilement disponibles pour les entreprises. Ils nécessitaient en effet du matériel coûteux et de grands investissements. En outre, aucune plateforme n'était en mesure de les traiter. Avec l'avènement du Cloud, il est devenu plus facile pour les entreprises de collecter, d'ingérer, de stocker et d'effectuer des analyses de Big Data à l'aide de leurs outils et infrastructures préférés. Elles peuvent payer le fournisseur de Cloud et utiliser son infrastructure en évitant toute dépense en capital. En outre, le coût du Cloud est très bon marché comparé à n'importe quelle solution sur site.

Le Big Data exige une immense quantité de ressources de calcul, de stockage et de réseau. En règle générale, la quantité de ressources nécessaire ne peut être hébergée sur une machine ou un serveur unique. Même si suffisamment de ressources sont disponibles sur un seul serveur, le traitement d'un pool entier de Big Data est très long, car chaque tâche est effectuée en séquence et chaque étape dépend de l'étape précédente. Il est nécessaire de disposer d'infrastructures et d'outils spécialisés capables de distribuer le travail sur plusieurs serveurs et de retourner les résultats, tout en les présentant à l'utilisateur après avoir correctement combiné les résultats de tous les serveurs. Ces outils sont des outils spécialisés pour le Big Data et garantissent la disponibilité, l'évolutivité et la distribution afin de s'assurer qu'une solution Big Data puisse être optimisée pour fonctionner rapidement avec une robustesse et une stabilité intégrées.

Les deux principaux services Big Data proposés par Azure sont HD Insights et Databricks. Découvrons les différents outils disponibles dans le paysage du Big Data.

Outils dédiés au Big Data

De nombreux outils et services sont disponibles dans l'espace Big Data, et nous allons découvrir certains d'entre eux dans ce chapitre.

Azure Data Factory

Azure Data Factory est le service ETL phare d'Azure. Il définit les données entrantes (en termes de format et de schéma), transforme les données selon les règles et les filtres métier, augmente les données existantes et transfère les données vers un magasin de destination qui est facilement consommable par d'autres services en aval. Il est en mesure d'exécuter des pipelines (contenant une logique ETL) sur Azure, ainsi que des infrastructures personnalisées. Il peut également exécuter des packages SQL Server Integration Services.

Azure Data Lake Storage

Azure Data Lake Storage est un stockage de Big Data prêt à l'emploi de niveau entreprise qui est résilient, hautement disponible et sécurisé. Il est compatible avec Hadoop et peut prendre en charge plusieurs pétaoctets de données stockées. Il repose sur les comptes de stockage Azure et bénéficie ainsi directement de tous les avantages du compte de stockage. La version actuelle est appelée Gen2, et regroupe les fonctionnalités du stockage Azure et de Data Lake Storage Gen1.

Hadoop

Hadoop a été créé par la base logicielle Apache. Il s'agit d'une infrastructure distribuée, évolutive et fiable dédiée au traitement de Big Data qui permet de diviser le Big Data en petits segments de données et de les distribuer au sein d'un cluster. Un cluster Hadoop comprend deux types de serveurs : maîtres et esclaves. Le serveur maître contient les composants administratifs d'Hadoop, tandis que les serveurs esclaves traitent les données. Hadoop est responsable des données de partition logique entre les esclaves. Les esclaves effectuent toute la transformation sur les données, recueillent les informations et les transmettent aux nœuds maîtres qui les rassemblent pour générer la sortie finale. Hadoop peut englober des milliers de serveurs, chaque serveur assurant le calcul et le stockage des tâches. Hadoop est disponible en tant que service à l'aide du service **HDInsight** dans Azure.

Trois composants principaux constituent le système de base d'Hadoop :

HDFS : le système de fichiers distribués Hadoop est un système de fichiers dédié au stockage de Big Data. Il s'agit d'une infrastructure distribuée qui permet de décomposer de grands fichiers Big Data en petits segments et de les placer sur différents esclaves dans un cluster. HDFS est un système de fichiers tolérant aux pannes. Ainsi, même si différents segments de données sont mis à la disposition de divers esclaves dans le cluster, les données sont également répliquées entre les esclaves. Par conséquent, en cas de défaillance d'un esclave, ces données seront disponibles sur un autre serveur. Il garantit également un accès rapide et efficace aux données au demandeur.

MapReduce : MapReduce est une autre infrastructure importante qui permet à Hadoop de traiter les données en parallèle. Cette infrastructure est responsable du traitement des données stockées dans les esclaves HDFS et de leur mappage aux esclaves. Une fois que les esclaves ont terminé le traitement, la partie « réduite » apporte des informations à chaque esclave et les rassemble pour générer la sortie finale. En règle générale, les HDFS et MapReduce sont disponibles sur le même nœud, de sorte que les données n'ont pas besoin de se déplacer entre les esclaves. Ainsi, leur traitement est plus efficace.

YARN : **Yet Another Resource Negotiator** (**YARN**) est un composant architectural Hadoop important qui planifie les tâches associées aux applications et à la gestion des ressources au sein d'un cluster. YARN a été publié dans la version Hadoop 2.0. Beaucoup l'assimile au successeur de MapReduce, car il est plus efficace en termes de traitement par lots et d'allocation de ressources.

Apache Spark

Apache Spark est une plateforme d'analyse distribuée et fiable pour le traitement des données à grande échelle. Il fournit un cluster capable d'exécuter des tâches de transformation et de Machine Learning sur de grandes quantités de données en parallèle et de retourner un résultat consolidé au client. Il comprend des nœuds maîtres et de travail. Les nœuds maîtres sont chargés de diviser et de distribuer les actions au sein des tâches et des données entre les nœuds de travail, ainsi que de regrouper les résultats de tous les nœuds de travail et de renvoyer les résultats au client. Voici un point essentiel à retenir lors de l'utilisation de Spark : la logique ou les calculs doivent être facilement parallèles, et la quantité de données est trop volumineuse pour une seule machine. Spark est disponible dans Azure en tant que service à partir de HDInsight et Databricks.

Databricks

Databricks repose sur Apache Spark. Il s'agit d'une plateforme en tant que service qui met un cluster Spark géré à la disposition des utilisateurs. Il fournit de nombreuses fonctions supplémentaires, telles qu'un portail complet pour gérer le cluster Spark et ses nœuds. Il permet également de créer des blocs-notes, de planifier et d'exécuter des tâches, tout en assurant la sécurité et la prise en charge de plusieurs utilisateurs.

Le moment est venu de découvrir comment intégrer les données provenant de plusieurs sources et les traiter à l'aide des outils susmentionnés.

Intégration des données

Nous sommes bien conscients de la façon dont les modèles d'intégration sont utilisés pour les applications ; les applications composées de plusieurs services sont intégrées ensemble à l'aide de divers modèles. Cependant, il existe un autre paradigme qui est une exigence clé pour de nombreuses organisations, connu sous le nom d'intégration des données. L'augmentation de l'intégration des données s'est essentiellement produite au cours de la dernière décennie, lorsque la génération et la disponibilité des données ont été incroyablement élevées. La vitesse, la variété et le volume des données générées ont considérablement augmenté, et les données sont quasiment partout.

Chaque organisation dispose d'un grand nombre de types d'applications différents et ils génèrent tous des données dans leur propre format propriétaire. Souvent, les données sont également achetées sur le marché. Même pendant les fusions et regroupements d'organisations, les données doivent être migrées et combinées.

L'intégration des données fait référence au processus d'introduction de données provenant de sources multiples et à la génération d'une nouvelle sortie qui a plus de sens et d'utilisabilité.

Il existe un besoin certain d'intégration de données pour les scénarios suivants :

- Migrer des données d'une source ou d'un groupe de sources vers une destination cible. Cela est nécessaire pour rendre les données disponibles dans différents formats, pour différents consommateurs et parties prenantes.
- Extraire des informations des données. Compte tenu de la disponibilité croissante et rapide des données, les organisations souhaitent en tirer des informations exploitables. Elles souhaitent créer des solutions qui fournissent des informations, aussi les données provenant de plusieurs sources doivent être fusionnées, nettoyées, améliorées et stockées dans un entrepôt de données.
- Générer des tableaux de bord et des rapports en réel.
- Créer des solutions analytiques.

L'intégration d'application a un comportement d'exécution lorsque les utilisateurs consomment l'application ; par exemple, dans le cas de la validation et de l'intégration de cartes bancaires. D'autre part, l'intégration des données se produit comme un exercice backend et n'est pas directement liée à l'activité de l'utilisateur.

Découvrons à présent le processus ETL avec Azure Data Factory.

ETL

Un processus connu sous le nom ETL aide à construire une source de données cible pour les données sur site consommables par des applications. En général, les données sont dans un format brut, et pour les rendre consommables, elles doivent passer par les trois phases suivantes :

- **Extraction** : au cours de cette phase, les données sont extraites de plusieurs emplacements. Par exemple, il pourrait y avoir plusieurs sources qui doivent toutes être connectées ensemble afin de récupérer les données. Les phases d'extraction utilisent généralement des connecteurs de données constitués d'informations de connexion liées à la source de données cible. Ils peuvent également disposer d'un stockage temporaire pour déplacer les données depuis la source de données et pour les stocker en vue d'une récupération plus rapide. Cette phase est responsable de l'ingestion de données.
- **Transformation** : les données disponibles après la phase d'extraction peuvent ne pas être directement consommables par les applications. Diverses raisons peuvent expliquer une telle situation ; par exemple, les données peuvent avoir des irrégularités, ou elles peuvent être manquantes ou erronées. Ou bien, certaines données peuvent ne pas être nécessaires du tout. De même, le format des données peut ne pas être propice à la consommation par les applications cibles. Dans tous ces cas, la transformation doit être appliquée aux données de telle manière qu'elles puissent être consommées efficacement par les applications.

- **Chargement** : après la transformation, les données doivent être chargées dans la source de données cible dans un format et un schéma qui permettent une disponibilité plus rapide, simple et axée sur les performances pour les applications. Encore une fois, une telle solution se composerait de connecteurs de données pour les sources de données de destination et du chargement des données en leur sein.

Poursuivant en abordant la relation entre Azure Data Factory et le processus ETL.

Une amorce sur Azure Data Factory

Azure Data Factory est un outil entièrement géré, hautement disponible, hautement évolutif et facile à utiliser pour la création de solutions d'intégration et la mise en œuvre de phases ETL. Data Factory vous aide à créer de nouveaux pipelines dans un mode glisser-déposer à l'aide d'une interface utilisateur, sans écrire de code ; cependant, cette solution fournit toujours des fonctions pour vous permettre d'écrire du code dans votre langage préféré.

Il existe quelques concepts importants à découvrir avant d'utiliser le service Data Factory, que nous explorerons plus en détail dans les sections suivantes :

- **Activités** : les activités sont des tâches individuelles qui permettent l'exécution et le traitement de la logique dans un pipeline Data Factory. Il existe plusieurs types d'activités. Il existe des activités liées au déplacement des données, à leur transformation et au contrôle. Chaque activité possède une stratégie grâce à laquelle elle peut décider du mécanisme de nouvelle tentative et de l'intervalle de cette dernière.
- **Pipelines** : les pipelines de Data Factory sont constitués de groupes d'activités et sont chargés de rassembler les activités. Les pipelines sont les charges de travail et les orchestrateurs qui permettent l'exécution des phases ETL. Les pipelines permettent de tisser ensemble des activités et de déclarer des dépendances entre elles. En utilisant des dépendances, il est possible d'exécuter certaines tâches en parallèle et d'autres tâches de manière séquentielle.
- **Jeux de données** : les jeux de données sont les sources et les destinations des données. Il peut s'agir de comptes de stockage Azure, de stockage Data Lake ou d'un hôte d'autres sources.
- **Services liés** : ce sont des services qui contiennent les informations de connexion et de connectivité pour les jeux de données et qui sont utilisés par des tâches individuelles pour s'y connecter.
- **Exécution d'intégration** : le moteur principal responsable de l'exécution de Data Factory est appelé l'exécution d'intégration. L'exécution d'intégration est disponible sur les trois configurations suivantes :
- **Azure** : dans cette configuration, la solution Data Factory s'exécute sur les ressources de calcul fournies par Azure.

- **Auto-hébergé** : dans cette configuration, la solution Data Factory s'exécute lorsque vous apportez vos propres ressources de calcul. Il peut s'agir de serveurs de machines virtuelles sur site ou basés sur le Cloud.
- **Azure SQL Server Integration Services (SSIS)** : cette configuration permet l'exécution de packages SSIS traditionnels écrits à l'aide de SQL Server.
- **Versions** : Data Factory est fourni dans deux versions différentes. Il est important de comprendre que tous les nouveaux développements se produiront sur la version v2, et que la version v1 restera telle quelle, ou disparaîtra à terme. Le version v2 est préférable pour les raisons suivantes :

  Elle offre la possibilité d'exécuter des packages d'intégration SQL Server.

  Elle possède des fonctions améliorées par rapport à la version v1.

  Elle est fournie avec une surveillance améliorée par rapport à la version v1.

Maintenant que vous avez une bonne compréhension de Data Factory, découvrons les différentes options de stockage disponibles sur Azure.

Une amorce sur Azure Data Lake Storage

Azure Data Lake Store assure le stockage des solutions Big Data. Cette solution est spécialement conçue pour stocker les vastes quantités de données qui sont généralement nécessaires dans les solutions de Big Data. Il s'agit d'un service géré et fourni par Azure. Les clients n'ont qu'à apporter leurs données et à les stocker dans un lac de données.

Il existe deux versions de la solution Azure Data Lake Storage : la version 1 (Gen1) et la version actuelle, la version 2 (Gen2). Gen2 possède toutes les fonctions de Gen1, à la différence près qu'il est construit sur le stockage Blob Azure.

Comme le stockage Blob Azure est hautement disponible, réplicable plusieurs fois, préparé aux sinistres et à faible coût, ces avantages sont transférés à Gen2 Data Lake. Data Lake peut stocker n'importe quel type de données, y compris les données relationnelles, non relationnelles, basées sur le système de fichiers et hiérarchiques.

La création d'une instance Data Lake Gen2 est aussi simple que la création d'un compte de stockage. La seule modification nécessaire consiste à activer l'espace de noms hiérarchique à partir de l'onglet **Advanced** (Avancé) de votre compte de stockage. Il est important de noter qu'il n'y a pas de migration ou de conversion directe d'un compte de stockage général vers Azure Data Lake ou vice versa. En outre, les comptes de stockage servent à stocker les fichiers, tandis que Data Lake est optimisé pour la lecture et l'ingestion de grandes quantités de données.

Intéressons-nous désormais au processus et aux phases principales tout en travaillant avec le Big Data. Il s'agit de phases distinctes et chacune est responsable de différentes activités sur les données.

Migration de données du stockage Azure vers Data Lake Storage Gen2

Dans cette section, nous allons migrer des données à partir du stockage Blob Azure vers un autre conteneur Azure de la même instance de stockage d'objets Blob Azure, et nous allons également migrer des données vers une instance Azure Data Lake Gen2 à l'aide d'un pipeline Azure Data Factory. Les sections suivantes présentent les étapes nécessaires à la création d'une solution end-to-end.

Préparation du compte de stockage source

Avant de pouvoir créer des pipelines Azure Data Factory et de les utiliser pour la migration, nous devons créer un compte de stockage, composé d'un certain nombre de conteneurs, et charger les fichiers de données. Dans le monde réel, ces fichiers et la connexion de stockage seraient déjà préparés. La première étape de création d'un compte de stockage Azure consiste à créer un groupe de ressources ou à choisir un groupe de ressources existant dans un abonnement Azure.

Provisionnement d'un nouveau groupe de ressources

Chaque ressource dans Azure est associée à un groupe de ressources. Avant de provisionner un compte de stockage Azure, nous devons créer un groupe de ressources qui hébergera le compte de stockage. Les étapes de création d'un groupe de ressources sont mentionnées ici. Il est à noter qu'un groupe de ressources peut être créé lors de l'approvisionnement d'un compte de stockage Azure ou un groupe de ressources existant peut être utilisé :

1. Accédez au portail Azure, connectez-vous et cliquez sur **+ Create a resource** (+ Créer une ressource) puis, recherchez le **`Resource group`** (Groupe de ressources).
2. Sélectionnez **Resource group** (Groupe de ressources) dans les résultats de la recherche et créez un groupe de ressources. Indiquez un nom et choisissez un emplacement approprié. Notez que toutes les ressources doivent être hébergées dans le même groupe de ressources et emplacement afin qu'il soit facile de les supprimer.

Après avoir mis en service le groupe de ressources, nous allons configurer un compte de stockage dans celui-ci.

Provisionnement d'un compte de stockage

Dans cette section, nous allons parcourir les étapes de création d'un compte de stockage Azure. Ce compte de stockage extraira la source de données à partir de laquelle les données seront migrées. Pour créer un compte de stockage, procédez comme suit :

1. Cliquez sur **+ Create a resource** (+ Créer une ressource) et recherchez le **`Storage Account`** (Compte de stockage). Sélectionnez **Storage Account** (Compte de stockage) dans les résultats de la recherche, puis créez un compte de stockage.
2. Indiquez un nom et un emplacement, puis sélectionnez un abonnement basé sur le groupe de ressources créé précédemment.
3. Sélectionnez **StorageV2 (general purpose v2)** (StorageV2 [usage général v2]) pour **Account kind** (Type de compte), **Standard** pour **Performance** et **Locally-redundant storage (LRS)** (Stockage localement redondant [LRS]) pour **Replication** (Réplication), comme illustré à la *Figure 9.2* :

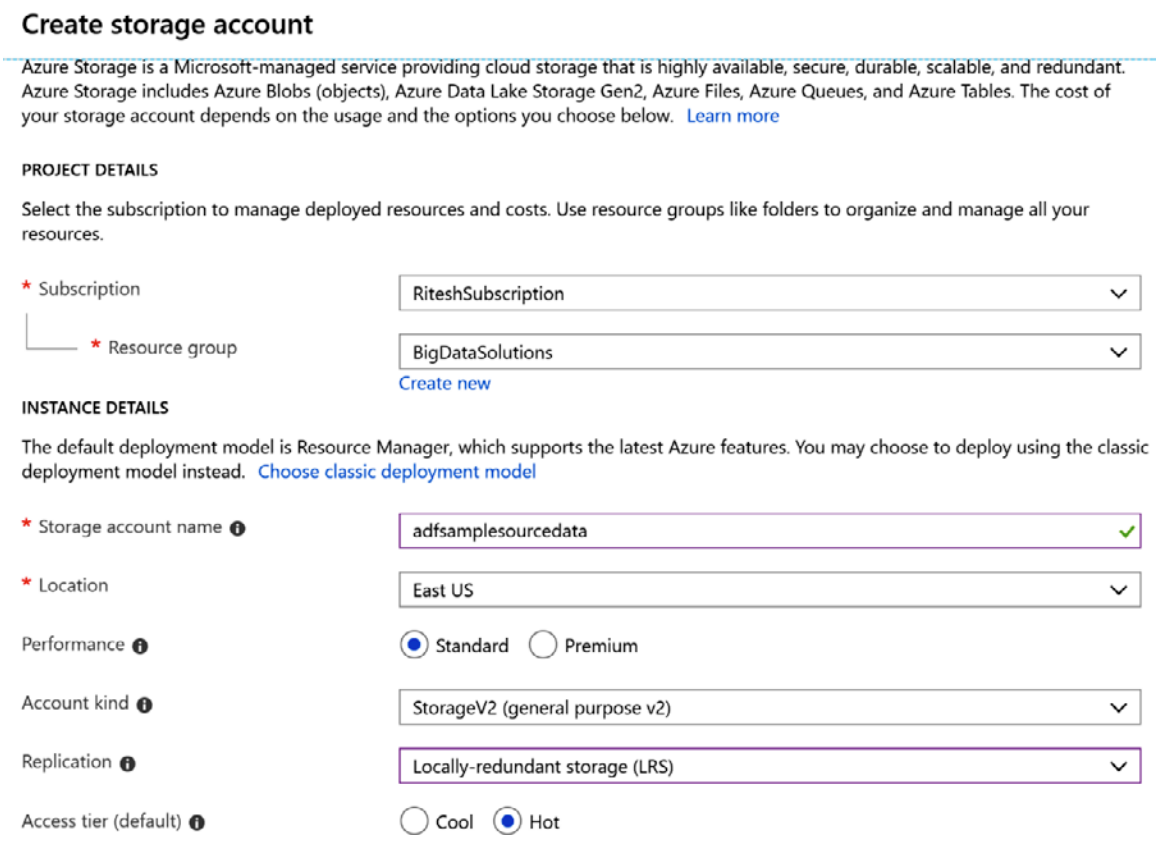


Figure 9.2 : configuration du compte de stockage

4. Maintenant , créez deux conteneurs dans le compte de stockage. Le conteneur **`rawdata`** contient les fichiers qui seront extraits par le pipeline Data Factory et agira en tant que jeu de données source, tandis que **finaldata** contiendra des fichiers où les pipelines Data Factory écriront des données, et agira comme jeu de données de destination :

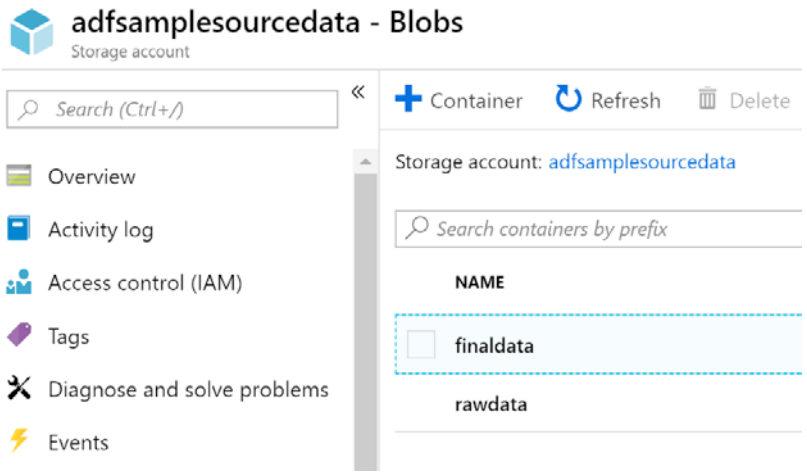


Figure 9.3 : création de conteneurs

5. Chargez un fichier de données (ce fichier est disponible avec le code source) dans le conteneur **rawdata**, comme indiqué à la *Figure 9.4* :

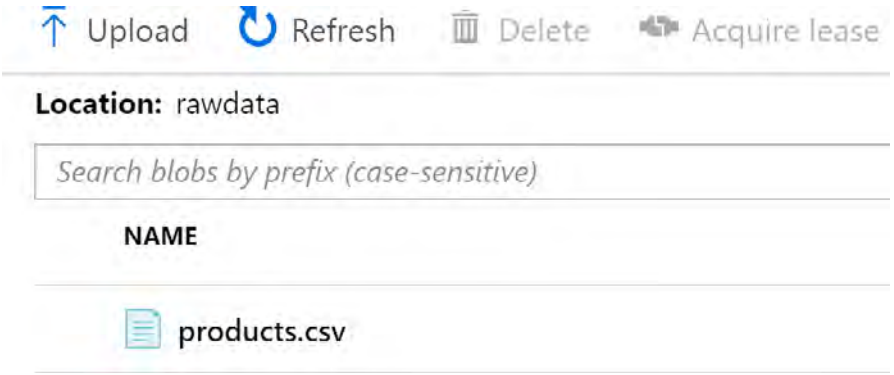


Figure 9.4 : chargement d'un fichier de données

Une fois ces étapes terminées, les activités de préparation des données source sont achevées. Nous pouvons désormais aborder la création d'une instance Data Lake.

Provisionnement du service Data Lake Gen2

Comme nous le savons déjà, le service Data Lake Gen2 est bâti sur le compte de stockage Azure. Pour cette raison, nous allons créer un compte de stockage de la même manière que nous l'avons fait plus tôt, à la seule différence que nous allons sélectionner **Enabled** (Activer) pour **Hierarchical namespace** (Espace de noms hiérarchique) dans l'onglet **Advanced** (Avancé) du nouveau compte de stockage Azure. Cela créera le nouveau service Data Lake Gen2 :

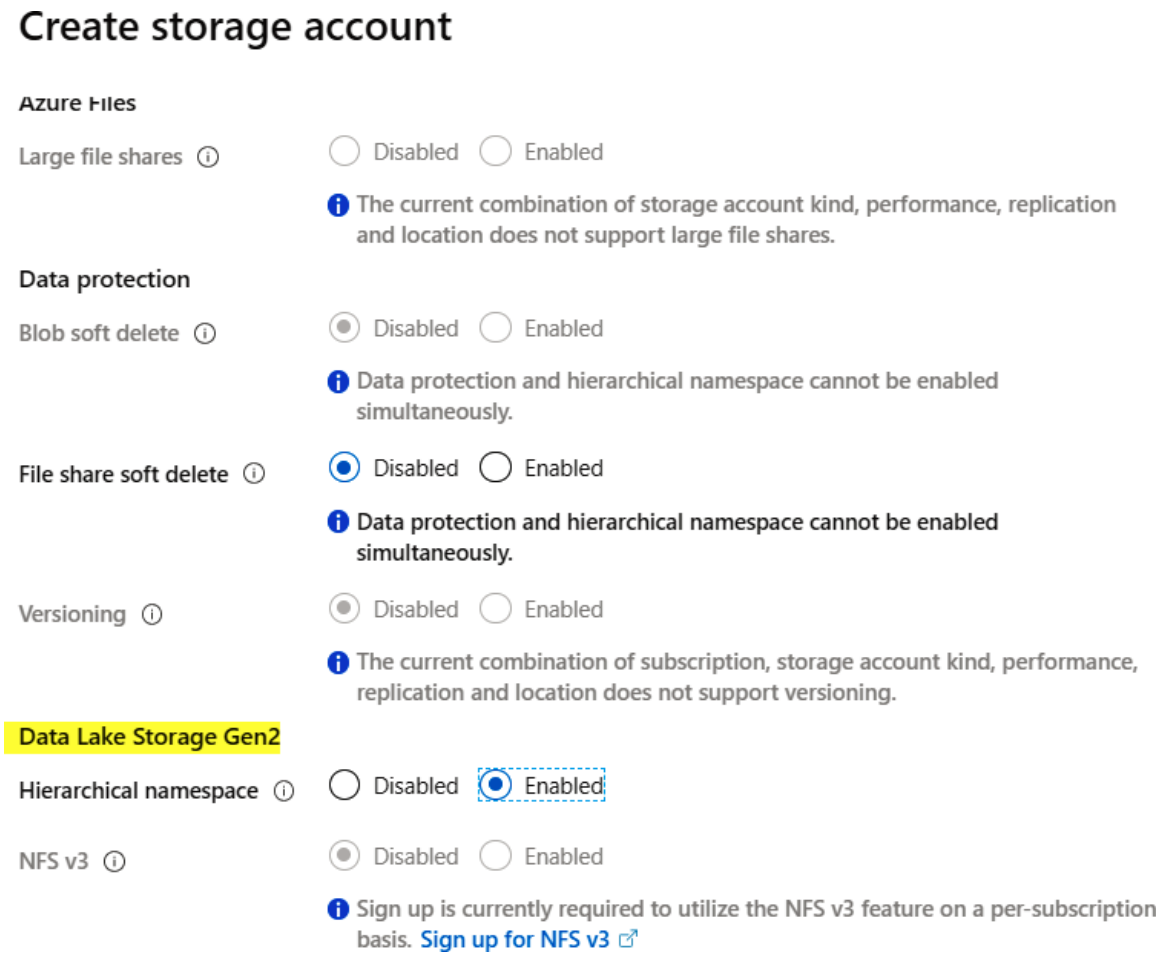


Figure 9.5 : création d'un compte de stockage

Après la création du lac de données, nous allons créer un pipeline Data Factory.

Configuration d'Azure Data Factory

Maintenant que nous avons déjà provisionné le groupe de ressources et le compte de stockage Azure, il est temps de créer une ressource Data Factory :

1. Créez un pipeline Data Factory en sélectionnant **V2** et en fournissant un nom et un emplacement ainsi qu'un groupe de ressources et une sélection d'abonnement.

   Data Factory a trois versions différentes, comme illustré à la *Figure* 9.6. Nous avons déjà discuté des versions **V1** et **V2** :

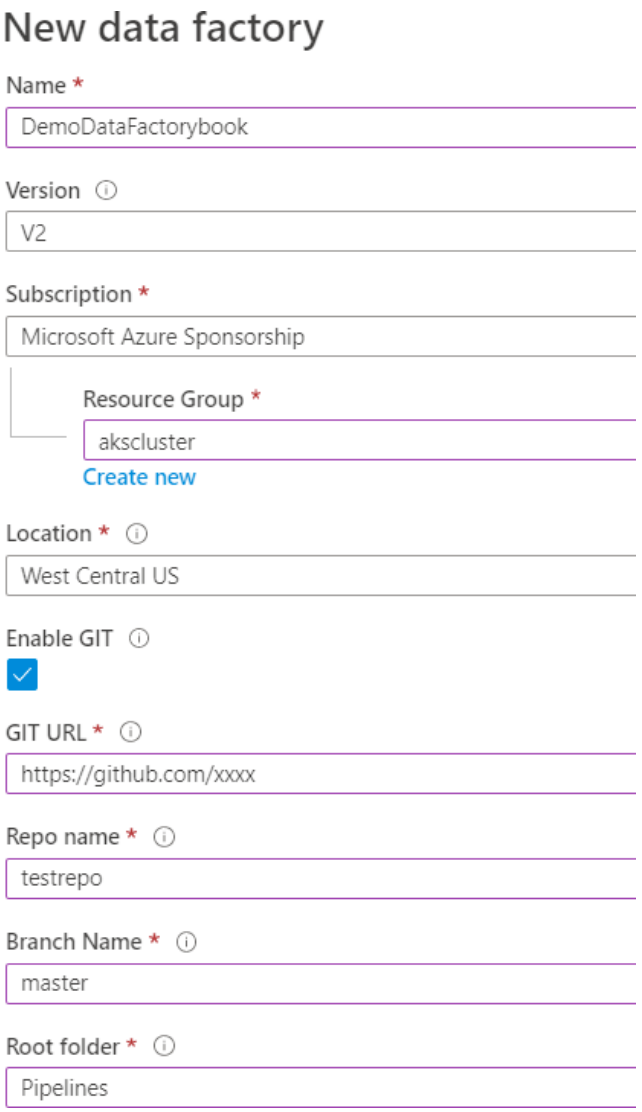


Figure 9.6 : sélection de la version Data Factory

2. Une fois la ressource Data Factory créée, cliquez sur le lien **Author & Monitor** (Créer et surveiller) dans le panneau central.

   Une nouvelle fenêtre s'affichera, composée du concepteur de Data Factory pour les pipelines.

Le code des pipelines peut être stocké dans les référentiels de contrôle de version, afin qu'il puisse être suivi pour les modifications de code et favoriser la collaboration entre les développeurs. Si vous avez oublié de configurer les paramètres du référentiel au cours de ces étapes, vous pourrez le faire ultérieurement.

La section suivante porte sur la configuration des paramètres du référentiel de contrôle de version si votre ressource Data Factory a été créée sans qu'aucun paramètre de référentiel n'ait été configuré.

Paramètres du référentiel

Avant de créer des éléments Data Factory, tels que des jeux de données et des pipelines, il est judicieux de configurer le référentiel de code pour héberger des fichiers liés à Data Factory :

1. Sur la page **Authoring** (Création), cliquez sur le bouton **Manage** (Gérer), puis sur **Git Configuration** (Configuration Git) dans le menu de gauche. Un autre volet s'affiche, cliquez sur le bouton **Set up code repository** (Configurer le référentiel de code) :

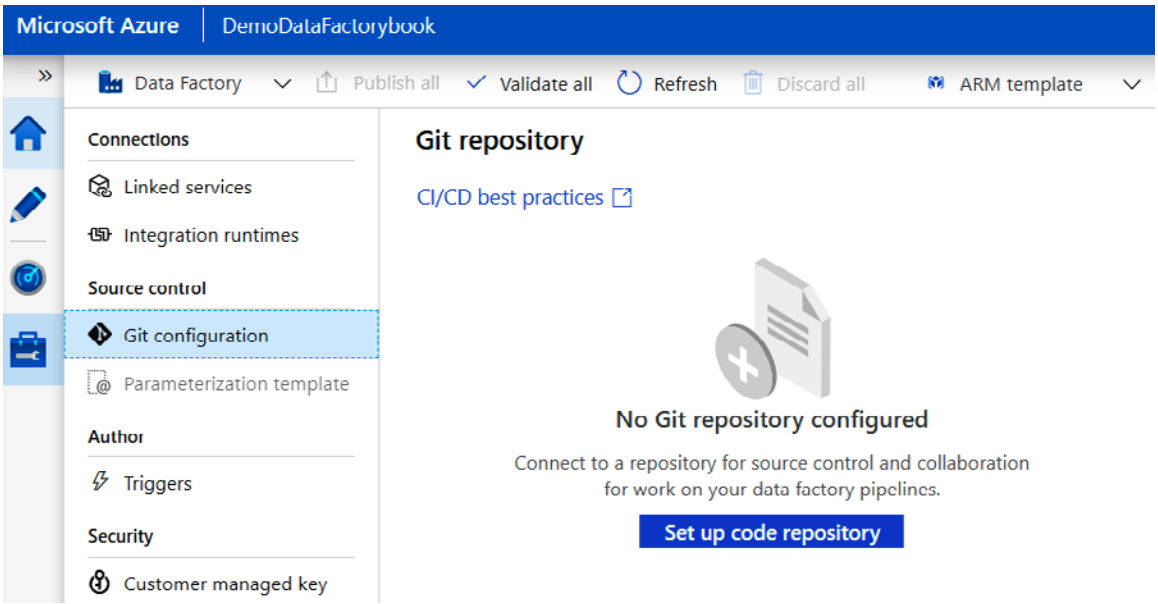


Figure 9.7 : configuration d'un référentiel Git

2. Dans le panneau qui s'affiche, sélectionnez l'un des types de référentiels dans lesquels vous souhaitez stocker les fichiers de code Data Factory. Dans ce cas, nous allons sélectionner **Azure DevOps Git** :

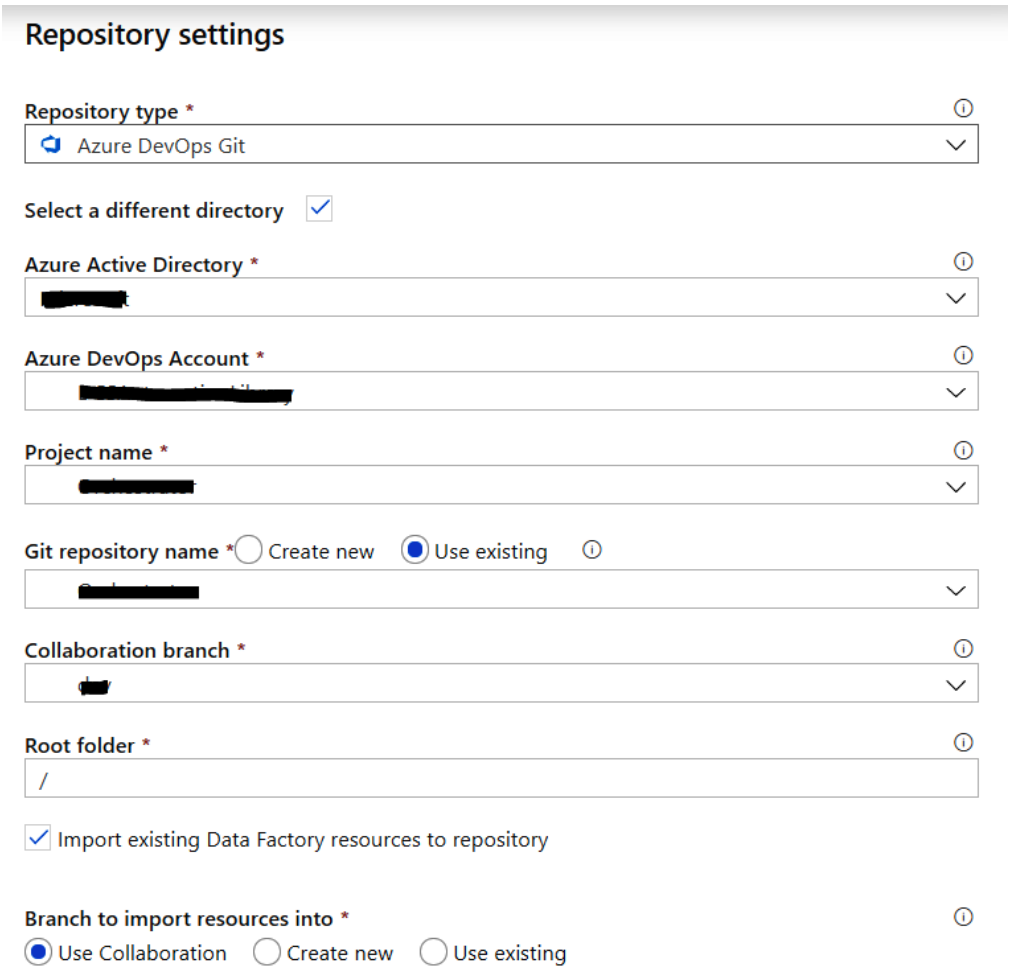


Figure 9.8 : sélection du type de référentiel Git approprié

3. Créez un référentiel ou réutilisez un référentiel existant à partir d'Azure DevOps. Vous devez déjà avoir un compte dans Azure DevOps. Si ce n'est pas le cas, visitez https://dev.azure.com et utilisez le compte avec lequel vous vous êtes connecté sur le portail Azure et créé une organisation, ainsi qu'un projet au sein de celui-ci. Reportez-vous au *chapitre 13, Intégration d'Azure DevOps* , pour en savoir plus sur la création d'organisations et de projets dans Azure DevOps.

   Nous pouvons à présent revenir à la fenêtre de création de Data Factory et commencer à créer des artefacts pour notre nouveau pipeline.

La section suivante porte sur la préparation de jeux de données qui seront utilisés dans nos pipelines Data Factory.

Jeux de données Data Factory

Maintenant, nous pouvons revenir à la pipeline Data Factory. Tout d'abord, créez un jeu de données qui agira en tant que jeu de données source. Ce sera le premier compte de stockage que nous créons et sur lequel nous chargeons l'exemple de fichier **`products.csv`** sur :

1. Cliquez sur **+ Datasets -> New DataSet** (+ Jeux de données -> Nouveau jeu de données) dans le menu de gauche et sélectionnez **Azure Blob Storage as data store** (Stockage Blob Azure comme magasin de données) et **delimitedText** comme format pour le fichier source. Créez un service lié en fournissant un nom et en sélectionnant un abonnement et un compte de stockage Azure. Par défaut, **`AutoResolveIntegrationRuntime`** est utilisé pour l'environnement d'exécution. C'est donc Azure qui fournira l'environnement d'exécution sur le calcul géré par Azure. Les services liés fournissent plusieurs méthodes d'authentification et nous utilisons la méthode**SAS (Shared Access Signature) URI (Uniform Resource Locator)**. Il est également possible d'utiliser une clé de compte, un principal de service et une identité gérée comme méthodes d'authentification :

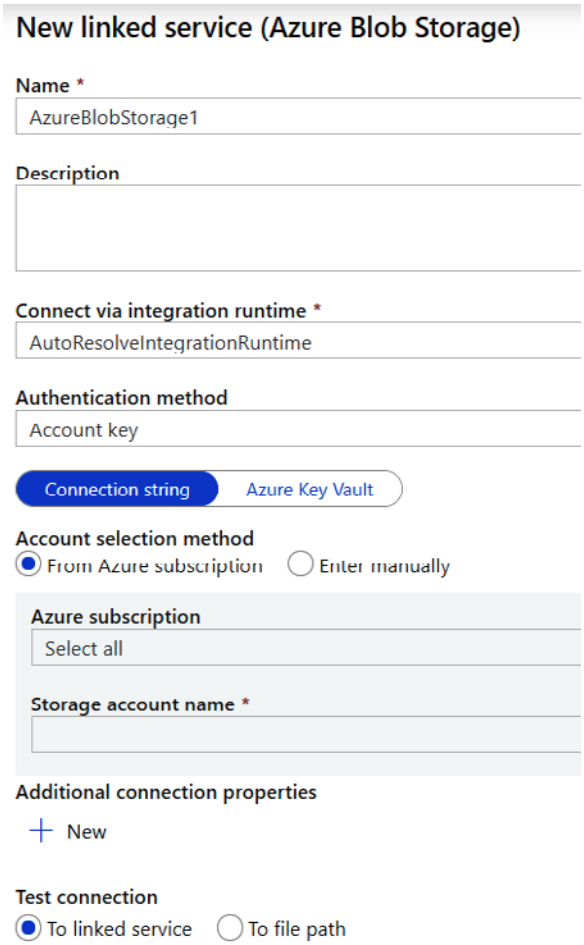


Figure 9.9 : mise en œuvre de la méthode d'authentification

2. Ensuite, dans le panneau inférieur résultant de l'onglet **General** (Général), cliquez sur le lien **Open properties** (Ouvrir les propriétés) et indiquez un nom pour le jeu de données :

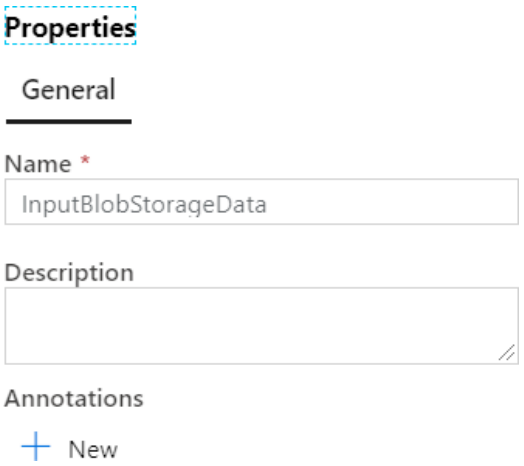


Figure 9.10 : attribution d'un nom au jeu de données

3. À partir de l'onglet **Connection** (Connexion), indiquez les informations relatives au conteneur, au nom du fichier blob dans le compte de stockage, au séparateur de ligne, au séparateur de colonne et d'autres informations qui aideront Data Factory à lire les données source de manière appropriée.

   Après la configuration, l'onglet **Connection** (Connexion) doit ressembler à la *Figure 9.11*. Notez que le chemin d'accès inclut le nom du conteneur et le nom du fichier :

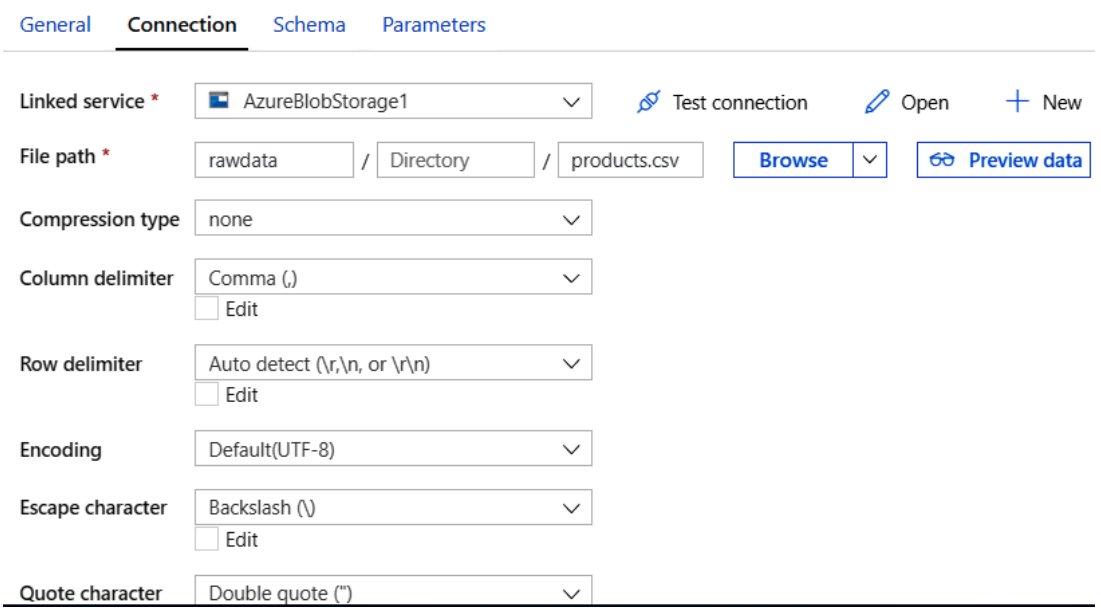


Figure 9.11 : configuration de la connexion

4. À ce stade, si vous cliquez sur le bouton **Preview data** (Aperçu des données), les données d'aperçu s'afficheront à partir du fichier`product.csv`. Sous l'onglet **Schema** (Schéma), ajoutez deux colonnes et nommez-les `ProductID` et `ProductPrice`. Le schéma permet de fournir un identificateur aux colonnes et de mapper également les colonnes sources dans le jeu de données source sur les colonnes cibles du jeu de données cible lorsque les noms ne sont pas les mêmes.

Maintenant que le premier jeu de données est créé, créons le deuxième.

Création du second jeu de données

Créez un jeu de données et un service lié pour le compte de stockage Blob de destination de la même manière que vous l'avez fait auparavant. Notez que le compte de stockage est le même que la source, mais le conteneur est différent. Assurez-vous que les données entrantes ont des informations de schéma qui leur sont également associées, comme illustré à la *Figure* 9.12 :

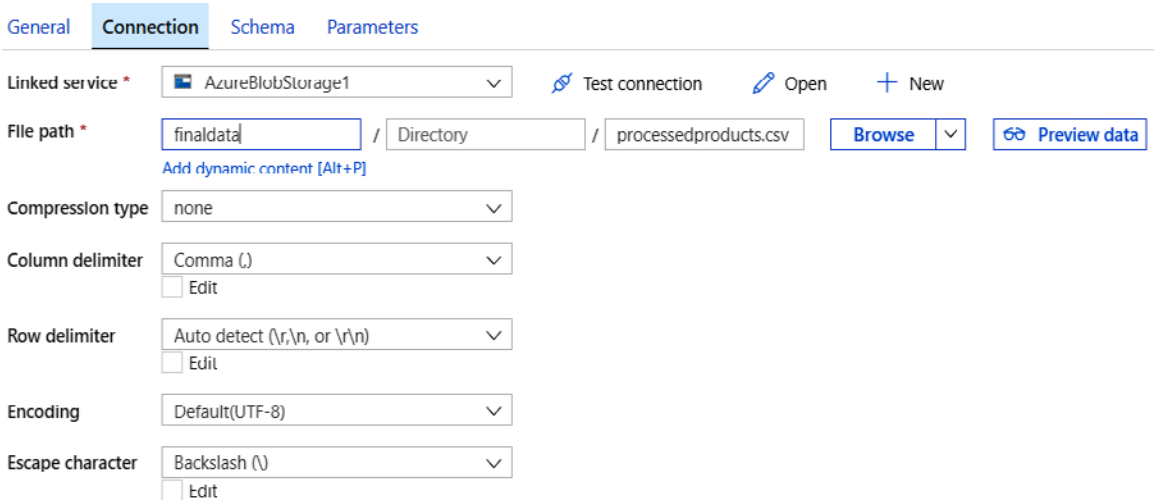


Figure 9.12 : création du second jeu de données

Nous allons ensuite créer un troisième jeu de données.

Création d'un troisième jeu de données

Créez un jeu de données pour l'instance de stockage Data Lake Gen2 en tant que jeu de données cible. Pour ce faire, sélectionnez le nouveau jeu de données, puis **Azure Data Lake Storage Gen2 (Preview)**(Azure Data Lake Storage Gen2 [aperçu]).

Donnez un nom au nouveau jeu de données et créez un service lié dans l'onglet **Connection** (Connexion). Choisissez **Use account key** (Utiliser la clé de compte) comme méthode d'authentification ; le reste de la configuration sera automatiquement renseigné après avoir sélectionné le nom du compte de stockage. Ensuite, testez la connexion en cliquant sur le bouton **Test connection** (Tester la connexion). Conservez la configuration par défaut pour le reste des onglets, comme illustré à la *Figure* 9.13 :

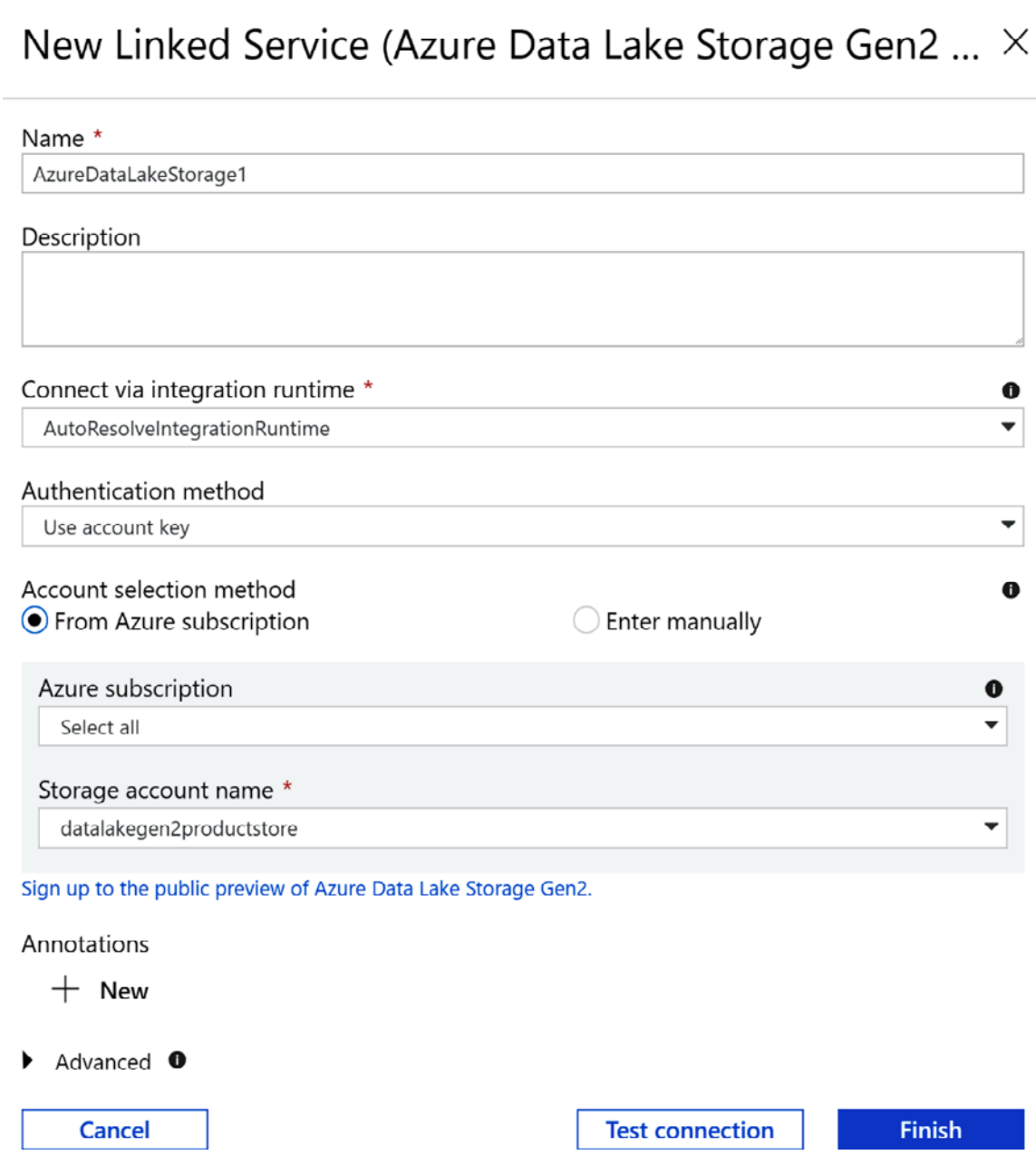


Figure 9.13 : configuration dans les onglets Connexion

Maintenant que nous avons établi la connexion aux données source, ainsi qu'aux banques de données source et de destination, il est temps de créer les pipelines qui contiendront la logique de la transformation des données.

Création d'un pipeline

Une fois tous les jeux de données créés, nous pouvons créer un pipeline qui consommera ces jeux de données. Les étapes de création d'un pipeline sont mentionnées ci-après :

1. Cliquez sur le menu **+ Pipeline => New Pipeline** (+ Pipelines => Nouveau Pipeline) dans le menu supérieur pour créer un pipeline. Ensuite, glissez et déposez l'activité **Copy Data** (Copier les données) depuis le menu **Move & Transform** (Déplacer et transformer), comme illustré à la *Figure 9.14* :

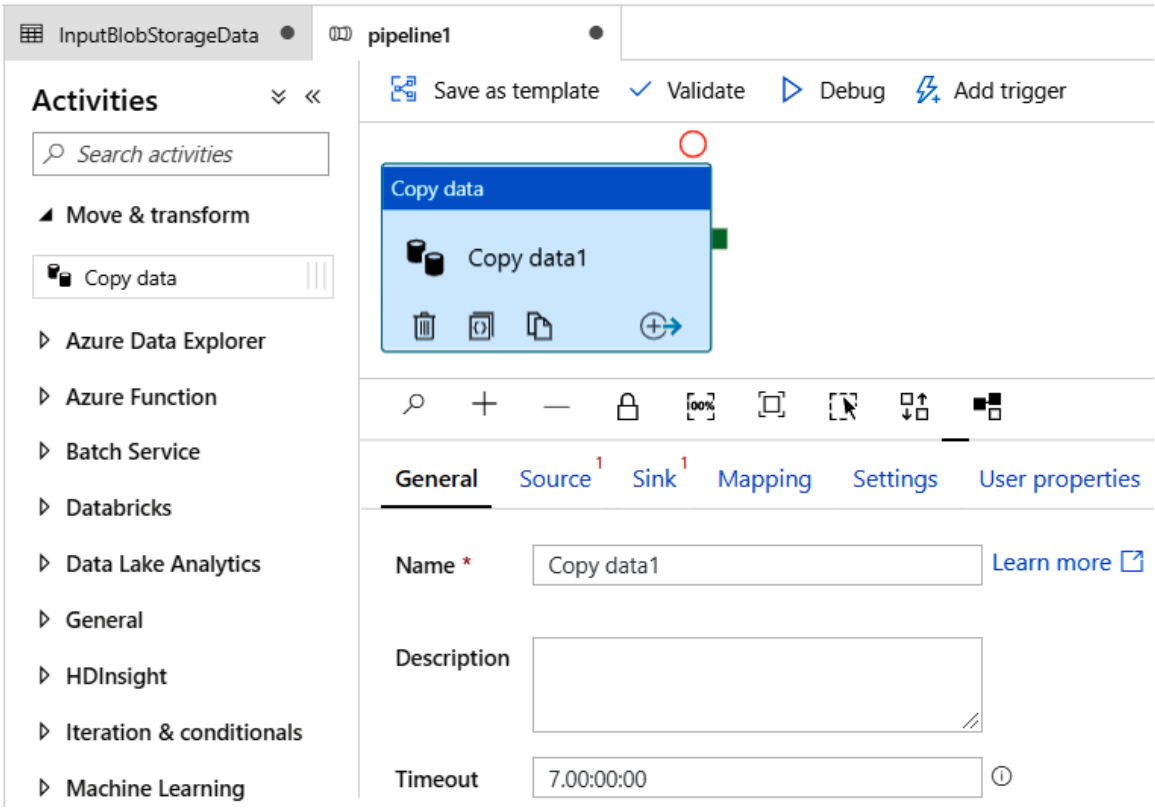


Figure 9.14 : menu Pipeline

2. L'onglet **General** (Général) résultant peut être laissé tel quel, mais l'onglet **Source** doit être configuré pour utiliser le jeu de données source que nous avons configuré précédemment :

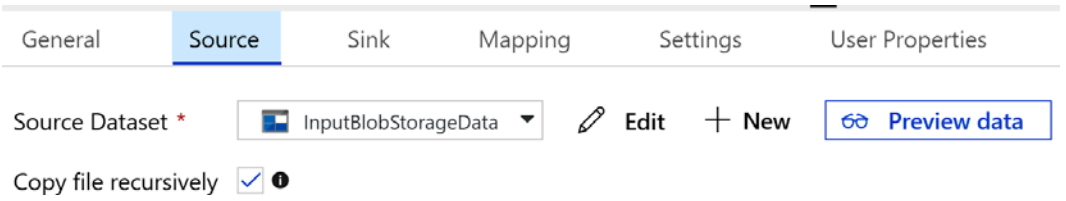


Figure 9.15 : onglet Source

3. L'onglet **Sink** (Récepteur) est utilisé pour configurer la banque de données de destination et le jeu de données, et il doit être configuré pour utiliser le jeu de données cible que nous avons configuré précédemment :

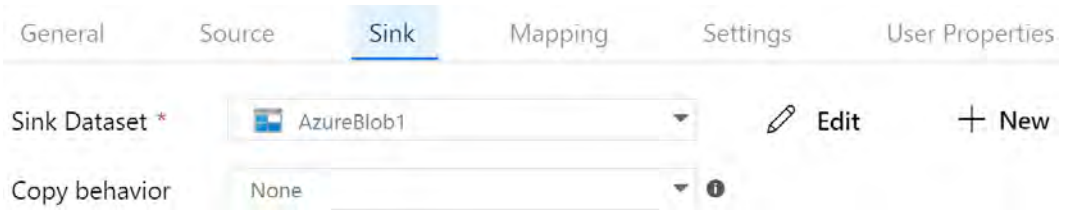


Figure 9.16 : onglet Récepteur

4. Dans l'onglet **Mapping** (Mappage), mappez les colonnes de la source aux colonnes du jeu de données de destination, comme illustré à la *Figure 9.17* :

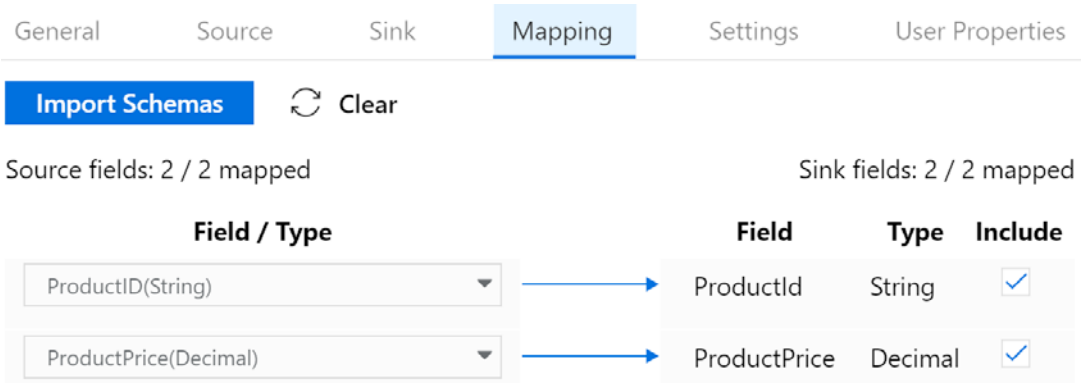


Figure 9.17 : onglet Mappage

Ajouter une autre activité de copie de données

Dans notre pipeline, nous pouvons ajouter plusieurs activités, chacune étant responsable d'une tâche de transformation particulière. La tâche mentionnée dans cette section est responsable de la copie des données du compte de stockage Azure vers Azure Data Lake Storage :

1. Ajoutez une autre activité **Copy Data** (Copier les données) dans le menu d'activité de gauche pour migrer les données vers le Data Lake Storage ; les deux tâches de copie s'exécuteront en parallèle :

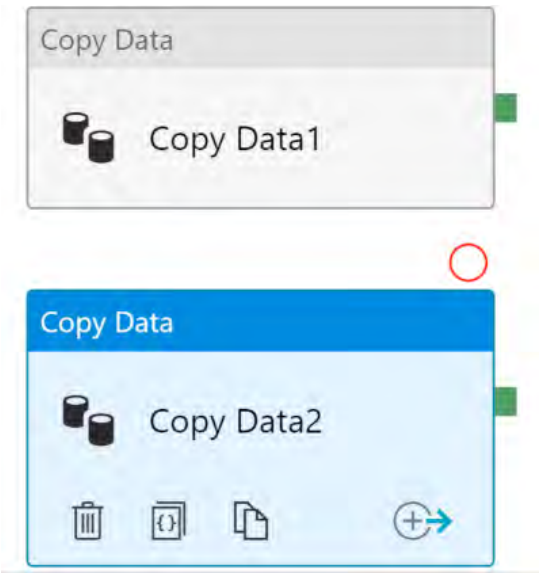


Figure 9.18 : activités de copie de données

   La configuration de la source est le compte de stockage Blob Azure qui contient le fichier `product.csv`.

   La configuration de récepteur ciblera le compte de stockage Data Lake Gen2.

2. Le reste de la configuration peut demeurer avec les paramètres par défaut pour la deuxième activité de données de copie.

Une fois le pipeline créé, il peut être publié dans un référentiel de contrôle de version tel que GitHub.

Intéressons-nous désormais à la création d'une solution à l'aide de Databricks et de Spark.

Création d'une solution à l'aide de Databricks

Databricks est une plateforme permettant d'utiliser Spark en tant que service. Nous n'avons pas besoin de configurer des nœuds maîtres ou de travail sur les machines virtuelles. Databricks nous fournit un environnement géré composé de nœuds maître et de travail, gérés par la plateforme. Nous devons suivre les étapes et la logique appliquées au traitement des données, et le reste est pris en charge par la plateforme Databricks.

Dans cette section, nous allons parcourir les étapes de création d'une solution à l'aide de Databricks. Nous allons télécharger des exemples de données à analyser.

L'exemple de fichier CSV a été téléchargé sur https://ourworldindata.org/coronavirus-source-data, bien qu'il soit également fourni avec le code de ce livre. L'URL mentionnée précédemment contiendra davantage de données à jour. Toutefois, le format a peut-être changé, et il est recommandé d'utiliser le fichier disponible avec les exemples de code de ce livre :

1. La première étape de la création d'une solution Databricks consiste à la configurer à l'aide du portail Azure. Une référence d'évaluation de 14 jours est disponible avec deux autres références (standard et Premium). La référence Premium applique le contrôle d'accès basé sur les rôles Azure (RBAC) au niveau des blocs-notes, des clusters, des tâches et des tables :

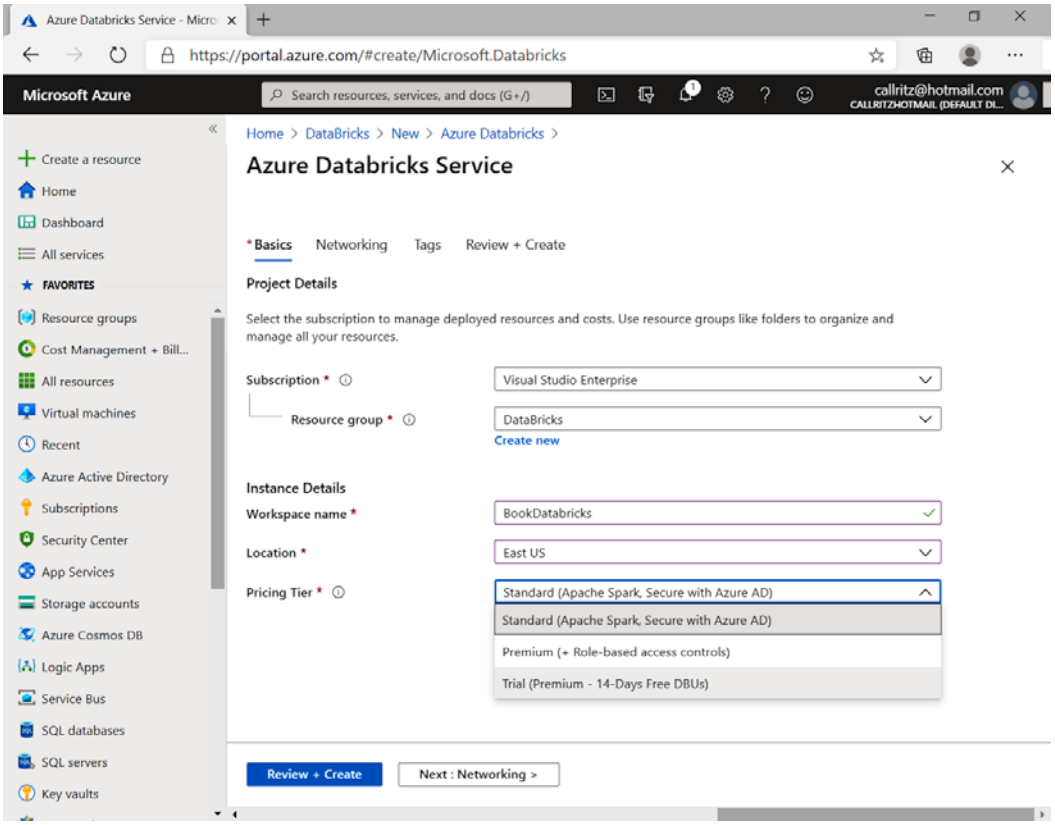


Figure 9.19 : portail Azure : service Databricks

2. Une fois l'espace de travail Databricks configuré, cliquez sur le bouton **Launch workspace** (Lancer l'espace de travail) de travail dans le volet **Overview** (Présentation). Une nouvelle fenêtre de navigateur s'ouvre alors et vous invite à vous connecter au portail Databricks.

3. Dans le portail Databricks, sélectionnez **Clusters** dans le menu de gauche et créez un cluster, comme illustré à la *Figure* 9.20 :

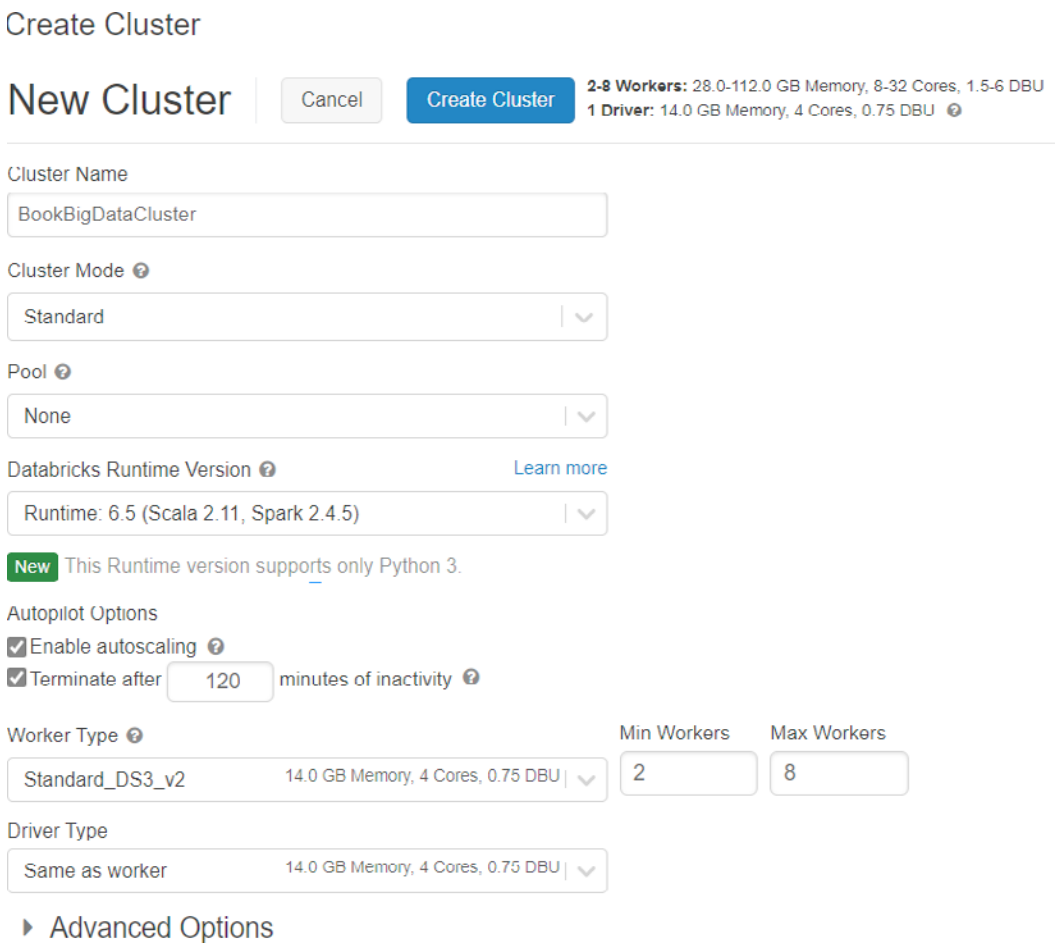


Figure 9.20 : Création d'un cluster

4. Indiquez le nom, la version d'exécution du runtime Databricks, le nombre de types de travailleurs, la configuration de la taille de machine virtuelle et la configuration de serveur de type pilote.

5. La création du cluster peut prendre quelques minutes. Après la création du cluster, cliquez sur **Home** (Accueil), sélectionnez un utilisateur dans son menu contextuel et créez un bloc-notes :

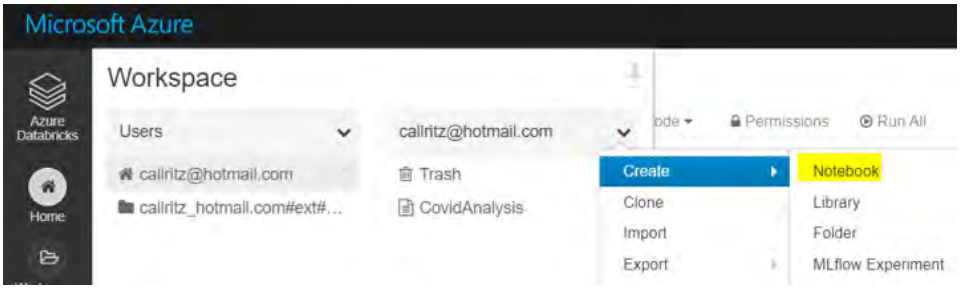


Figure 9.21 : sélection d'une nouveau bloc-notes

6. Donnez un nom au bloc-notes, comme illustré ci-après :

Figure 9.22 : création d'une bloc-notes

7. Créez un compte de stockage, comme illustré ci-après. Celui-ci servira à stocker les données COVID brutes au format CSV :

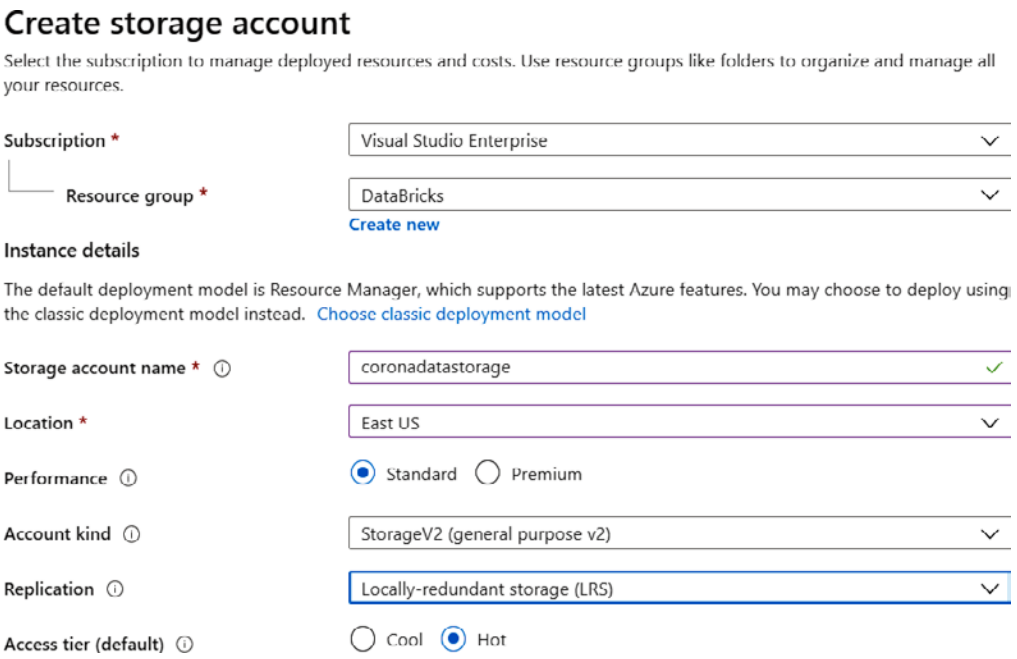


Figure 9.23 : création d'un compte de stockage

8. Créez un conteneur pour stocker le fichier CSV, comme illustré ci-après :

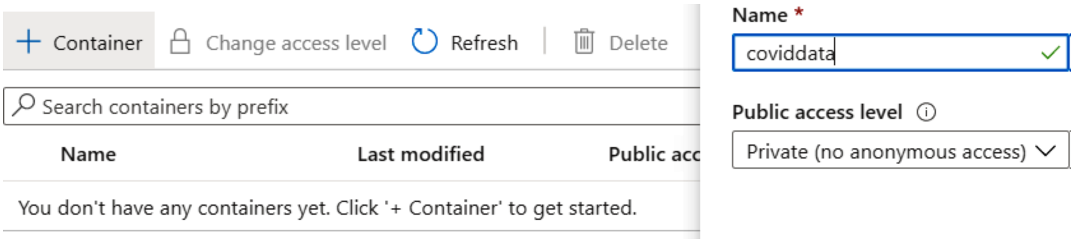


Figure 9.24 : création d'un conteneur

9. Téléchargez le fichier **`owid-covid-data.csv`** dans ce conteneur.

Une fois que vous avez terminé les étapes précédentes, la tâche suivante consiste à charger les données.

Chargement des données

La deuxième grande phase consiste à charger les données COVID dans l'espace de travail Databricks. Cette opération peut être accomplie de deux façons :

- Montez le conteneur de stockage Azure dans Databricks, puis chargez les fichiers disponibles dans le support.
- Chargez les données directement à partir du compte de stockage. Cette approche a été utilisée dans l'exemple suivant.

Les étapes suivantes doivent être effectuées pour charger et analyser les données à l'aide de Databricks :

1. Vous devez commencer par vous connecter et accéder au compte de stockage. La clé du compte de stockage est nécessaire et elle est stockée dans la configuration Spark. Notez que la clé ici est **`"fs.azure.account.key.coronadatastorage.blob.core.windows.net"`** et que la valeur est la clé associée :

```
spark.conf.set("fs.azure.account.key.coronadatastorage.blob.core.windows.
net","xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx==")
```

2. La clé du compte de stockage Azure peut être récupérée en accédant aux paramètres et à la propriété **Access Keys** (Clés d'accès) du compte de stockage dans le portail.

   L'étape suivante consiste à charger le fichier et à lire les données contenues dans le fichier CSV. Le schéma doit être déduit du fichier lui-même au lieu d'être fourni explicitement. Il existe également une ligne d'en-tête, représentée à l'aide de l'option dans la commande suivante.

   Le fichier est référencé à l'aide du format suivant : **`wasbs://{{container}}@{{storage account name}}.blob.core.windows.net/{{filename}}`**.

3. La méthode **read** de l'objet **SparkSession** fournit des méthodes pour lire les fichiers. Pour lire des fichiers CSV, la méthode **csv** doit être utilisée avec les paramètres requis, tels que le chemin d'accès au fichier CSV. D'autres paramètres optionnels peuvent être fournis pour personnaliser le processus de lecture des fichiers de données. Plusieurs types de formats de fichiers, tels que JSON, **Optimized Row Columnar** (**ORC**) et Parquet, ainsi que les bases de données relationnelles telles que SQL Server et MySQL, les banques de données NoSQL comme Cassandra et MongoDB, ainsi que les plateformes Big Data telles qu'Apache Hive, peuvent être utilisés dans Spark. Jetons un coup d'œil à la commande suivante pour comprendre la mise en œuvre de Spark DataFrames :

   ```
   coviddata = spark.read.format("csv").option("inferSchema", "true").
   option("header", "true").load("wasbs://coviddata@coronadatastorage.blob.core.
   windows.net/owid-covid-data.csv")
   ```

   L'utilisation de cette commande crée un objet du type DataFrame dans Spark. Spark fournit des objets **Resilient Distributed Dataset** (**RDD**) pour manipuler les données et travailler avec elles. Les RDD sont des objets de faible niveau et tout code écrit pour travailler avec eux est susceptible de ne pas être optimisé. Les DataFrames sont des constructions de niveau supérieur par rapport au RDD et fournissent une optimisation pour accéder et travailler avec les RDD. Mieux vaut travailler avec des DataFrames et non des RDD.

   Les DataFrames fournissent des données au format ligne/colonne, qui facilite la visualisation et le travail avec les données. Les Spark DataFrames sont similaires aux pandas DataFrames, à la différence près qu'il s'agit d'implémentations différentes.

4. La commande suivante affiche les données dans un DataFrame. Elle affiche toutes les lignes et les colonnes disponibles dans le DataFrame :

   ```
   coviddata.show()
   ```

   Vous devriez obtenir une sortie similaire au contenu de la *Figure* 9.25 :

```
|iso_code|location|                 date|total_cases|new_cases|total_deaths|new_deaths|total_cases_per_million|new_cases_per_million|total_deaths_per_million|new_deaths_per_million|
total_tests|new_tests|total_tests_per_thousand|new_tests_per_thousand|new_tests_smoothed|new_tests_smoothed_per_thousand|tests_units|stringency_index|population|population_density
|median_age|aged_65_older|aged_70_older|gdp_per_capita|extreme_poverty|cvd_death_rate|diabetes_prevalence|female_smokers|male_smokers|handwashing_facilities|hospital_beds_per_100k
|
+--------+--------+-------------------+-----------+---------+------------+----------+-----------------------+---------------------+------------------------+----------------------+
----------+---------+------------------------+----------------------+------------------+-------------------------------+-----------+----------------+----------+------------------
+----------+-------------+-------------+--------------+---------------+--------------+-------------------+--------------+------------+----------------------+----------------------
+
|     ABW|   Aruba|2020-03-13 00:00:00|          2|        2|           0|         0|                 18.733|               18.733|                     0.0|                   0.0|
null|     null|                    null|                  null|              null|                           null|       null|             0.0|  106766.0|             584.8|
41.2|       13.085|        7.452|     35973.781|           null|          null|              11.62|          null|        null|                  null|                  null|
|     ABW|   Aruba|2020-03-20 00:00:00|          4|        2|           0|         0|                 37.465|               18.733|                     0.0|                   0.0|
null|     null|                    null|                  null|              null|                           null|       null|           30.56|  106766.0|             584.8|
41.2|       13.085|        7.452|     35973.781|           null|          null|              11.62|          null|        null|                  null|                  null|
```

Figure 9.25 : les données brutes d'un DataFrame

5. Le schéma des données chargées est déduit par Spark et peut être vérifié à l'aide de la commande suivante :

   ```
   coviddata.printSchema()
   ```

   Nous devrions obtenir une sortie semblable à celle-ci :

```
root
 |-- iso_code: string (nullable = true)
 |-- location: string (nullable = true)
 |-- date: timestamp (nullable = true)
 |-- total_cases: integer (nullable = true)
 |-- new_cases: integer (nullable = true)
 |-- total_deaths: integer (nullable = true)
 |-- new_deaths: integer (nullable = true)
 |-- total_cases_per_million: double (nullable = true)
 |-- new_cases_per_million: double (nullable = true)
 |-- total_deaths_per_million: double (nullable = true)
 |-- new_deaths_per_million: double (nullable = true)
```

Figure 9.26 : obtention du schéma du DataFrame pour chaque colonne

6. Pour compter le nombre de lignes du fichier CSV, la commande suivante peut être utilisée, et sa sortie indique qu'il y a 19 288 lignes dans le fichier :

```
coviddata.count()
```

```
▸ (1) Spark Jobs
Out[7]: 19288
```

Figure 9.27 : recherche du nombre d'enregistrements dans un DataFrame

7. Le DataFrame d'origine contient plus de 30 colonnes. Nous pouvons également sélectionner un sous-ensemble des colonnes disponibles et travailler avec elles directement, comme illustré ci-après :

```
CovidDataSmallSet = coviddata.select("location","date", "new_cases", "new_deaths")
CovidDataSmallSet.show()
```

Le résultat du code est illustré à la *Figure 9.28* :

```
+--------+-------------------+---------+----------+
|location|               date|new_cases|new_deaths|
+--------+-------------------+---------+----------+
|   Aruba|2020-03-13 00:00:00|        2|         0|
|   Aruba|2020-03-20 00:00:00|        2|         0|
|   Aruba|2020-03-24 00:00:00|        8|         0|
|   Aruba|2020-03-25 00:00:00|        5|         0|
|   Aruba|2020-03-26 00:00:00|        2|         0|
|   Aruba|2020-03-27 00:00:00|        9|         0|
|   Aruba|2020-03-28 00:00:00|        0|         0|
|   Aruba|2020-03-29 00:00:00|        0|         0|
|   Aruba|2020-03-30 00:00:00|       22|         0|
|   Aruba|2020-04-01 00:00:00|        5|         0|
|   Aruba|2020-04-02 00:00:00|        0|         0|
```

Figure 9.28 : sélection de quelques colonnes parmi l'ensemble des colonnes

8. Il est également possible de filtrer les données à l'aide de la méthode **filter**, comme illustré ci-après :

```
CovidDataSmallSet.filter(" location == 'United States' ").show()
```

9. Il est également possible d'ajouter plusieurs conditions à l'aide des opérateurs AND (**&**) ou OR (**|**) :

```
CovidDataSmallSet.filter((CovidDataSmallSet.location == 'United States') |
(CovidDataSmallSet.location == 'Aruba')).show()
```

10. Pour connaître le nombre de lignes et d'autres données statistiques, tels que l'écart moyen, maximal, minimal et standard, la méthode **describe** peut être utilisée :

```
CovidDataSmallSet.describe().show()
```

En utilisant la commande précédente, vous obtiendrez une sortie similaire à celle-ci :

```
+-------+-----------+-----------------+-----------------+
|summary|   location|        new_cases|       new_deaths|
+-------+-----------+-----------------+-----------------+
|  count|      19288|            19288|            19288|
|   mean|       null|536.6524263790958|35.05174201576109|
| stddev|       null|4828.611755359697|335.3488977234115|
|    min|Afghanistan|            -2461|                0|
|    max|   Zimbabwe|           107909|            10520|
+-------+-----------+-----------------+-----------------+
```

Figure 9.29 : affichage des statistiques de chaque colonne à l'aide de la méthode describe

11. Il est également possible de déterminer le pourcentage de données nulles ou vides dans les colonnes spécifiées. Voici quelques exemples :

```
from pyspark.sql.functions import  col
(coviddata.where(col("diabetes_prevalence").isNull()).count() * 100)/coviddata.
count()
```

La sortie indique **5.998548320199087**, ce qui signifie que 95 % des données sont nulles. Nous devons supprimer ces colonnes de l'analyse de données. De même, l'exécution de la même commande au niveau de la colonne **total_tests_per_thousand** renvoie la sortie **73.62090418913314**, qui est bien meilleure que la colonne précédente.

12. La commande suivante permet de supprimer certaines des colonnes du DataFrame :

```
coviddatanew=coviddata.drop("iso_code").drop("total_tests").drop("total_tests").
drop("new_tests").drop("total_tests_per_thousand").drop("new_tests_per_thousand").
drop("new_tests_smoothed").drop("new_tests_smoothed_per_thousand ")
```

13. Un regroupement des données est parfois nécessaire. Dans de tels scénarios, vous pouvez effectuer le regroupement des données, comme illustré ici :

```
coviddatanew = coviddata.groupBy('location').agg({'date': 'max'})
```

Cette commande affichera les données de l'instruction **groupBy** :

location	max(date)
Chad	2020-05-23 00:00:00
Anguilla	2020-05-23 00:00:00
Paraguay	2020-05-23 00:00:00
Russia	2020-05-23 00:00:00
International	2020-03-10 00:00:00
Yemen	2020-05-23 00:00:00
World	2020-05-23 00:00:00
Senegal	2020-05-23 00:00:00
Sweden	2020-05-23 00:00:00
Guyana	2020-05-23 00:00:00
Eritrea	2020-05-23 00:00:00
Jersey	2020-05-23 00:00:00
Philippines	2020-05-23 00:00:00

Figure 9.30 : données provenant de l'instruction groupby

14. Comme vous pouvez le voir dans la colonne **max (date)**, les dates sont la plupart du temps identiques pour tous les pays, nous pouvons utiliser cette valeur pour filtrer les enregistrements et obtenir une seule ligne pour chaque pays représentant la date maximale :

```
coviddatauniquecountry = coviddata.filter("date='2020-05-23 00:00:00'")
coviddatauniquecountry.show()
```

15. Si nous comptabilisons les enregistrements du nouveau DataFrame, nous obtenons **209**.

Nous pouvons enregistrer le nouveau DataFrame dans un autre fichier CSV, qui peut être nécessaire pour d'autres processeurs de données :

```
coviddatauniquecountry.rdd.saveAsTextFile("dbfs:/mnt/coronadatastorage/
uniquecountry.csv")
```

Nous pouvons vérifier le fichier nouvellement créé à l'aide de la commande suivante :

```
%fs ls /mnt/coronadatastorage/
```

Le chemin monté sera affiché, comme illustré à la *Figure 9.31* :

Figure 9.31 : chemin monté dans les nœuds Spark

16. Il est également possible d'ajouter les données dans le catalogue Databricks à l'aide de la méthode **`createTempView`** ou **`createOrReplaceTempView`** dans le catalogue Databricks. Le fait de placer des données dans le catalogue les rend disponibles dans un contexte donné. Pour ajouter des données dans le catalogue, la méthode **`createTempView`** ou **`createOrReplaceTempView`** du DataFrame peut être utilisée, en fournissant une nouvelle vue pour la table dans le catalogue :

```
coviddatauniquecountry.createOrReplaceTempView("corona")
```

17. Une fois que la table est dans le catalogue, elle est accessible à partir de votre session SQL, comme illustré ci-après :

```
spark.sql("select * from corona").show()
```

Les données de l'instruction SQL s'afficheront comme illustré à la *Figure 9.32* :

```
+--------------------+-------------------+-----------+
|            location|               date|total_cases|
+--------------------+-------------------+-----------+
|               Aruba|2020-05-23 00:00:00|        101|
|         Afghanistan|2020-05-23 00:00:00|       9216|
|              Angola|2020-05-23 00:00:00|         60|
|            Anguilla|2020-05-23 00:00:00|          3|
|             Albania|2020-05-23 00:00:00|        981|
|             Andorra|2020-05-23 00:00:00|        762|
|United Arab Emirates|2020-05-23 00:00:00|      27892|
|           Argentina|2020-05-23 00:00:00|      10636|
|             Armenia|2020-05-23 00:00:00|       5928|
| Antigua and Barbuda|2020-05-23 00:00:00|         25|
|           Australia|2020-05-23 00:00:00|       7095|
|             Austria|2020-05-23 00:00:00|      16361|
|          Azerbaijan|2020-05-23 00:00:00|       3855|
|             Burundi|2020-05-23 00:00:00|         42|
|             Belgium|2020-05-23 00:00:00|      56511|
|               Benin|2020-05-23 00:00:00|        135|
|Bonaire Sint Eust...|2020-05-23 00:00:00|          6|
|        Burkina Faso|2020-05-23 00:00:00|        814|
|          Bangladesh|2020-05-23 00:00:00|      30205|
|            Bulgaria|2020-05-23 00:00:00|       2408|
+--------------------+-------------------+-----------+
```

Figure 9.32 : données provenant de l'instruction SQL

18. Il est possible d'effectuer une requête SQL supplémentaire sur la table, comme illustré ci-après :

```
spark.sql("select * from corona where location in ('India','Angola') order
by location").show()
```

Nous avons couvert quelques-unes des possibilités de Databricks. Il existe de nombreux autres services et fonctions, mais ceux-ci n'ont pas pu être évoqués dans un seul chapitre. Pour en savoir plus à ce sujet, consultez la page https://azure.microsoft.com/services/databricks.

Résumé

Ce chapitre traitait du service Azure Data Factory, qui est chargé de fournir des services ETL dans Azure. Étant donné qu'il s'agit d'une plateforme en tant que service, il offre une évolutivité illimitée, une haute disponibilité et des pipelines simples à configurer. Son intégration avec Azure DevOps et GitHub est également transparente. Nous avons également exploré les fonctions et les avantages de l'utilisation du stockage Azure Data Lake Gen2 pour stocker tout type de Big Data. Il s'agit d'un magasin de données hiérarchique rentable, hautement évolutif et hiérarchisé pour gérer les Big Data, qui est compatible avec Azure HDInsight, Databricks et l'écosystème Hadoop.

Nous n'avons pas eu le temps d'aborder en détail tous les sujets mentionnés dans ce chapitre. Nous avons surtout évoqué les possibilités dans Azure, en particulier avec Databricks et Spark. Plusieurs technologies Azure sont liées au Big Data, notamment HDInsight, Hadoop, Spark et son écosystème associé, ainsi que Databricks, qui est une plateforme en tant qu'environnement de service pour Spark avec des fonctionnalités ajoutées. Le chapitre suivant porte sur les fonctionnalités de calcul Serverless dans Azure.

10

Le Serverless dans Azure - Travailler avec Azure Functions

Dans le chapitre précédent, vous avez découvert différentes solutions de Big Data disponibles sur Azure. Dans ce chapitre, vous découvrirez comment la technologie Serverless peut vous aider à traiter une grande quantité de données.

Le Serverless est l'une des expressions les plus tendances dans le monde des technologies de nos jours et tout le monde veut prendre le train en marche. Le Serverless présente de nombreux avantages du point de vue de l'informatique en général, des processus de développement logiciel, des infrastructures et de la mise en place technique. L'industrie est en pleine effervescence : d'une part concernant tout ce qui touche **l'infrastructure en tant que service (IaaS)** et d'autre part, concernant le Serverless. Entre ces deux services, nous trouvons les **plateformes en tant que service (PaaS)** et les conteneurs. J'ai rencontré beaucoup de développeurs, et j'ai l'impression que certains confondent les concepts des solutions IaaS, PaaS, les conteneurs et l'informatique Serverless. En outre, il existe une grande confusion concernant les cas d'utilisation, la mise en application, l'architecture et l'implémentation des solutions Serverless. Le Serverless est un nouveau paradigme qui révolutionne non seulement la technologie mais également la culture et les processus en place dans les organisations.

Ce chapitre commence par la présentation du Serverless, avant d'aborder les sujets suivants :

- Fonctions en tant que service
- Azure Functions
- Azure Durable Functions
- Azure Event Grid

Serverless

Serverless désigne un modèle de déploiement dans le cadre duquel les utilisateurs sont uniquement responsables de leur code d'application et de leur configuration. Grâce à l'informatique Serverless, les clients n'ont pas à apporter leurs propres plateforme et infrastructure sous-jacentes. Ils peuvent donc se concentrer sur la résolution de leurs problèmes d'entreprise.

Le Serverless ne signifie pas que les serveurs sont inexistants. le calcul, le stockage et les réseaux sont toujours nécessaires pour exécuter le code et la configuration. Toutefois, les clients n'ont aucune visibilité sur ces calculs, ces stockages et ces réseaux. Ils ne se soucient pas de la plateforme et de l'infrastructure sous-jacentes. Ils n'ont pas besoin de gérer ni de surveiller l'infrastructure et la plateforme. Le Serverless fournit un environnement capable de se redimensionner, horizontalement et verticalement, automatiquement, sans même que les clients ne s'en rendent compte. Toutes les opérations liées aux plateformes et aux infrastructures s'effectuent en arrière-plan, et sont exécutée par le fournisseur de Cloud. Les clients bénéficient de **contrats de niveau de service (SLA)** qui garantissent les performances, et Azure veille à ce que ces SLA soient respectés, quelle que soit la demande.

Les clients n'ont qu'à intégrer leur code ; le fournisseur de services Cloud est tenu de fournir l'infrastructure et la plateforme nécessaire pour exécuter le code. Découvrons en détail les différents avantages d'Azure Functions.

Les avantages d'Azure Functions

L'informatique Serverless est un paradigme relativement récent qui permet aux organisations de convertir les grandes fonctions en fonctions à la demande, pouvant être invoquées et exécutées via des déclencheurs automatisés et des tâches planifiées. Cela est également connu sous le terme **Fonctions en tant que service (FaaS)**, grâce auxquelles les organisations peuvent se concentrer sur leurs propres enjeux plutôt que sur l'infrastructure et la plateforme sous-jacentes. Une FaaS permet également de déléguer l'architecture d'une solution, en mettant à disposition des fonctions réutilisables plus modestes, desquelles découle un retour sur investissement plus intéressant.

De nombreuses plateformes informatiques Serverless sont disponibles. Voici les plus importantes :

- Azure Functions
- AWS Lambda
- IBM OpenWhisk
- Iron.io
- Fonctions Google Cloud

En effet, de nouvelles plateformes/infrastructures sont introduites tous les jours, et il devient de plus en plus difficile pour les entreprises de décider quelle serait la solution la plus adaptée à leurs besoins. Azure offre un environnement Serverless enrichi, dénommé Azure Functions et voici quelques-unes des fonctions qu'il prend en charge :

- Nombreuses façons d'invoquer une fonction : manuelle, planifiée ou basée sur un événement
- Nombreux types de support de liaison.
- Possibilité d'exécuter des fonctions de manière synchrone et asynchrone.
- Possibilité d'exécuter des fonctions basées sur plusieurs types de déclencheurs.
- Possibilité d'exécuter des fonctions de longue et de courte durée. Toutefois, les fonctions volumineuses et de longue durée ne sont pas recommandées, car elles peuvent entraîner des délais d'attente inattendus.
- Possibilité d'utiliser des fonctions proxy pour différentes architectures de fonctions.
- Plusieurs modèles d'utilisation, y compris la consommation, ainsi que le modèle App Service.
- Possibilité de créer des fonctions à l'aide de plusieurs langages tels que JavaScript, Python et C#.
- Autorisation basée sur OAuth.
- L'extension de Durable Functions permet d'écrire des fonctions avec état.
- Plusieurs options d'authentification, notamment Azure AD, Facebook, Twitter et d'autres fournisseurs d'identité.
- Possibilité de configurer facilement des paramètres entrants et sortants.
- Intégration de Visual Studio pour la création d'Azure Functions.
- Parallélisme massif.

Jetons un coup d'œil aux FaaS et à leurs rôles dans l'architecture Serverless.

FaaS

Azure fournit des FaaS. Il s'agit d'implémentations Serverless d'Azure. Avec Azure Functions, le code peut être écrit dans n'importe quel langage, selon le choix de l'utilisateur, et Azure Functions fournira le runtime nécessaire à son exécution. Une plateforme appropriée sera fournie aux utilisateurs selon le langage choisi, afin de leur permettre d'y saisir leur propre code. Les fonctions sont des unités de déploiement qui peuvent être redimensionnées automatiquement. Concernant les fonctions, les utilisateurs ne peuvent pas voir les machines virtuelles et la plateforme sous-jacentes, mais Azure Functions fournit une petite fenêtre pour les voir au niveau de la **console Kudu**.

Il existe deux principaux composants dans Azure Functions :

- Le runtime d'Azure Functions
- Déclencheurs et liaisons dans Azure Functions

Examinons ces composants de plus près.

Le runtime d'Azure Functions

L'élément central d'Azure Functions est son runtime Azure. Azure WebJobs constitue le précurseur d'Azure Functions. Le code d'Azure WebJobs constitue également l'élément central d'Azure Functions. Des fonctionnalités supplémentaires ainsi que des extensions Azure WebJobs ont permis de créer Azure Functions. Le runtime d'Azure Functions est la magie qui permet à ce service de fonctionner. Azure Functions est hébergé au sein d'Azure App Services. Azure App Services charge le runtime d'Azure et attend qu'un événement externe se produise ou qu'une activité manuelle soit réalisée. Dès qu'un événement se produit ou qu'une requête lui parvient, le système App Service charge la charge utile entrante, lit le fichier **`function.json`** de Function à la recherche de liaisons et de déclencheurs, cartographie les données entrantes par rapport aux paramètres entrants et invoque la fonction en utilisant les valeurs de paramètres. Une fois que la fonction est exécutée, la valeur est à nouveau transmise au runtime Azure Function via un paramètre sortant défini en tant que liaison dans le fichier **`function.json`**. Le runtime de la fonction retourne les valeurs à l'appelant. Le runtime d'Azure Functions représente le ciment qui assure le bon déroulement de toutes les fonctions.

La version actuelle du runtime d'Azure est ~ 3. Elle est basée sur l'infrastructure .NET Core 3.1. Auparavant, la version ~ 2 était basée sur l'infrastructure .NET Core 2.2. La première version, ~ 1, était basée sur l'infrastructure .NET 4.7.

Des modifications substantielles ont été apportées entre la version 1 et la version 2 en raison des changements de l'infrastructure sous-jacente elle-même. Toutefois, très peu de modifications majeures ont été apportées entre la version 2 et la version 3 et la plupart des fonctions écrites dans la version 2 continueront à s'exécuter sur la version 3. Il est néanmoins recommandé d'effectuer des tests adéquats après la migration de la version 2 à la version 3. Par ailleurs des modifications importantes ont été apportées entre la version 1 et la version 2 concernant les déclencheurs et les liaisons. Les déclencheurs et les liaisons sont désormais disponibles en tant qu'extensions, chacun appartenant à un ensemble différent dans les versions 2 et 3.

Déclencheurs et liaisons dans Azure Functions

Si le runtime constitue le cerveau d'Azure Functions, les liaisons et déclencheurs des fonctions en sont le cœur. Azure Functions favorise un couplage souple et une haute cohésion entre les services à l'aide de déclencheurs et de liaisons. Les applications écrites ciblant des environnements non Serverless implémentent un code à l'aide de la syntaxe impérative pour des paramètres entrants et sortants et des valeurs de retour. Azure Functions utilise un mécanisme déclaratif pour appeler des fonctions à l'aide de déclencheurs et configure le flux de données à l'aide de liaisons.

Les liaisons font référence au processus de création d'un lien entre les données entrantes et la fonction Azure, en mappant les types de données. La connexion peut être dans une seule direction, depuis le runtime à Azure Functions et vice versa ou bien elle peut être multidirectionnelle, la même liaison pouvant transmettre des données entre le runtime d'Azure et Azure Functions dans les deux sens. Azure Functions utilise une méthode déclarative pour définir des liaisons.

Les déclencheurs sont un type spécial de liaison grâce auxquels les fonctions peuvent être appelées sur la base d'événements externes. Outre l'appel d'une fonction, ils transfèrent également les données entrantes, la charge utile et les métadonnées à la fonction.

Les liaisons sont définies dans le fichier **`function.json`**, comme suit :

```
{
  "bindings": [
    {
      "name": "checkOut",
      "type": "queueTrigger",
      "direction": "in",
      "queueName": "checkout-items",
      "connection": "AzureWebJobsDashboard"
    },
    {
      "name": "Orders",
      "type": "table",
      "direction": "out",
      "tableName": "OrderDetails",
      "connection": "<<Connection to table storage account>>"
   }
  ],
  "disabled": false
}
```

Dans cet exemple, un déclencheur est déclaré, lequel appelle la fonction chaque fois qu'un nouvel élément intègre la file d'attente du stockage. Le type est **`queueTrigger`**, la direction est entrante, **`queueName`** représente les **`objets de sortie`** et les détails sur la connexion au compte de stockage cible et le nom de la table sont également indiqués. Toutes ces valeurs sont importantes pour assurer le fonctionnement de cette liaison. Le nom **`checkOut`** peut être utilisé dans le code des fonctions en tant que variable.

De même, une liaison pour la valeur de retour est déclarée. Ici, la valeur de retour est dénommée **`Ordres`** et les données représentent le résultat d'Azure Functions. La liaison écrit les données de retour dans le stockage de table Azure à l'aide de la chaîne de connexion fournie.

Les liaisons et les déclencheurs peuvent être modifiés et créés à l'aide de l'onglet **Integrate** (Intégrer) dans Azure Functions. Le fichier **`function.json`** est mis à jour en arrière-plan. Le déclencheur **`checkOut`** est déclaré, comme illustré ici :

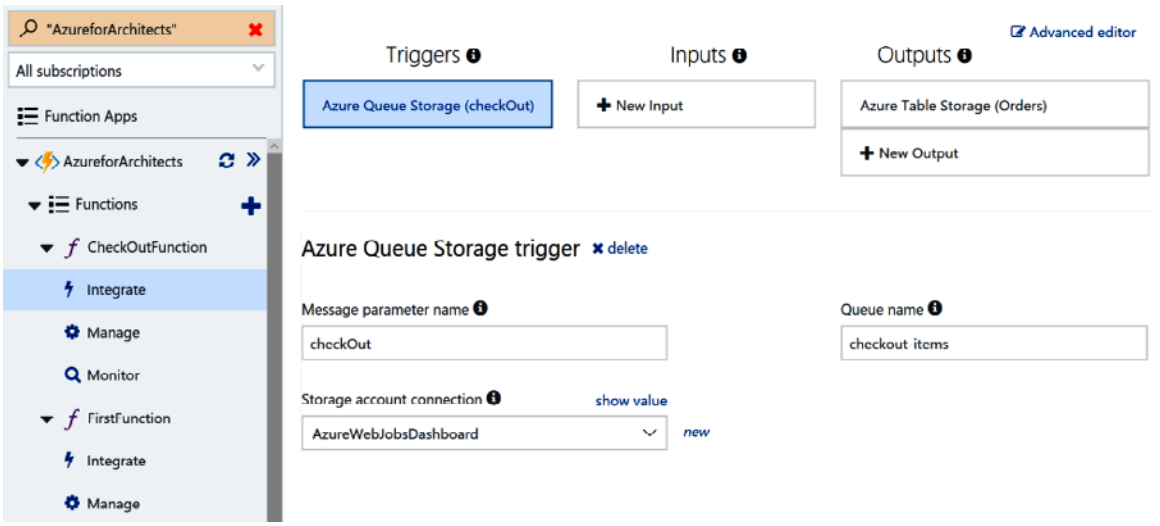


Figure 10.1 : la section Déclencheurs de l'onglet Intégrer

La sortie **Ordres** est illustrée ci-après :

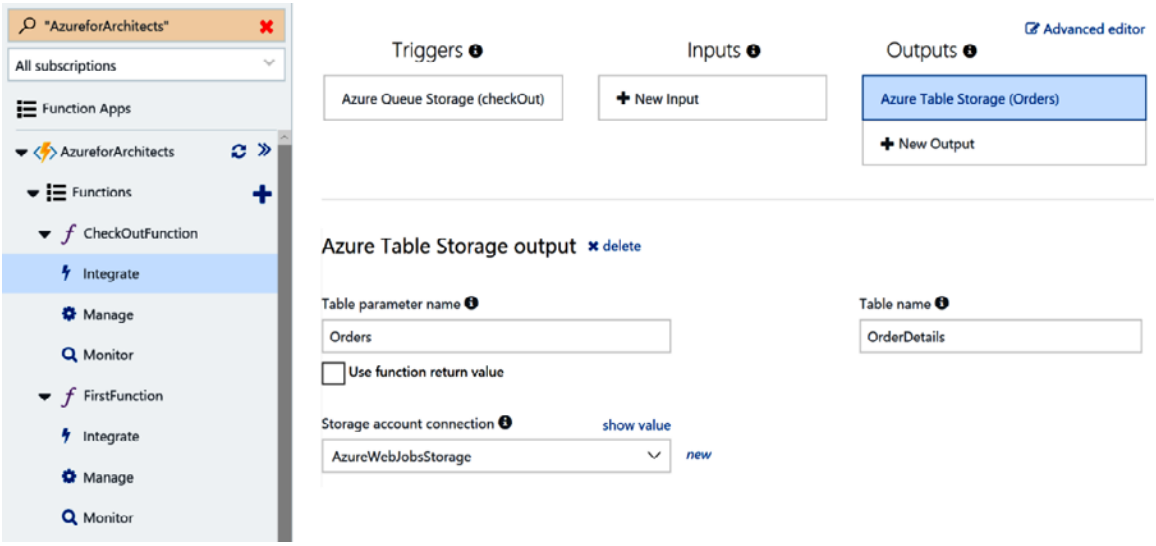


Figure 10.2 : ajout de détails de sortie du compte de stockage

Les auteurs d'Azure Functions n'ont pas besoin d'écrire du code pour obtenir des données provenant de sources multiples. Ils décident simplement du type de données attendu du runtime Azure. Ceci est illustré dans le prochain segment de code. Notez que la sortie est disponible sous forme de chaîne de la fonction. Plusieurs types de données peuvent être utilisés comme liaison pour les fonctions. Par exemple, une liaison de file d'attente peut fournir ce qui suit :

- Un ancien objet CLR (Common Language Runtime) ordinaire (POCO)
- Une chaîne
- Un octet
- **`CloudQueueMessage`**

L'auteur de la fonction peut utiliser l'un de ces types de données et le runtime Azure Function s'assurera qu'un objet en tant que paramètre approprié est envoyé à la fonction. Voici un extrait de code permettant d'accepter les données de chaîne et le runtime Functions encapsulera les données entrantes dans un type de données **`string`** avant d'appeler la fonction. Si le runtime n'est pas en mesure de convertir les données entrantes en une chaîne, il génère une exception :

```
using System;
public static void Run(string checkOut, TraceWriter log)
{
    log.Info($"C# Queue trigger function processed: { checkOut }");
}
```

Il est également important de savoir que dans la *Figure 10.2*, les noms du compte de stockage sont **`AzureWebJobsStorage`** et **`AzureWebJobsDashboard`**. Ces deux éléments sont des clés définies dans la section **`appSettings`** et contiennent des chaînes de connexion de compte de stockage. Ces comptes de stockage sont utilisés en interne par Azure Functions pour maintenir son état et l'état de l'exécution de la fonction.

Pour en savoir plus sur les liaisons et les déclencheurs Azure, accédez à l'adresse suivante https://docs.microsoft.com/azure/azure-functions/functions-bindings-storage-queue.

Maintenant que nous avons abordé en détail les liaisons et les déclencheurs Azure, intéressons-nous aux différentes options de configuration proposées par Azure Functions.

Configuration d'Azure Functions

Azure Functions offre des options de configuration à plusieurs niveaux. Celles-ci permettent de configurer les éléments suivants :

- La plateforme elle-même
- Functions App Services

Ces paramètres affectent toutes les fonctions en leur sein. Pour en savoir plus sur ces paramètres, accédez à https://docs.microsoft.com/azure/azure-functions/functions-how-to-use-azure-function-app-settings.

Configuration de la plateforme

Azure Functions est hébergée au sein d'Azure App Services, par conséquent elle bénéficie de toutes ses fonctionnalités. Les journaux de diagnostic et de suivi peuvent être aisément configurés à l'aide des fonctions de la plateforme. En outre, App Service fournit des options pour attribuer des certificats SSL, en utilisant un domaine personnalisé, l'authentification et l'autorisation dans le cadre de ses fonctions de mise en réseau.

Bien que les clients n'aient pas besoin de se préoccuper de l'infrastructure, du système d'exploitation, du système de fichiers ou de la plateforme qui exécutent les fonctions, Azure Functions fournit l'outillage nécessaire afin d'accéder au système sous-jacent si de petites modifications sont nécessaires. La console du portail et la console Kudu sont des outils utilisés à cette fin. Elles fournissent un éditeur enrichi qui permet de créer des fonctions Azure et de modifier leur configuration.

Azure Functions, tout comme App Service, permet de stocker des informations de configuration au sein de la section `web.config` des paramètres de l'application, qui peut être consultée à la demande. Certaines des fonctions de la plateforme des applications de fonction sont illustrées à la *Figure 10.3* :

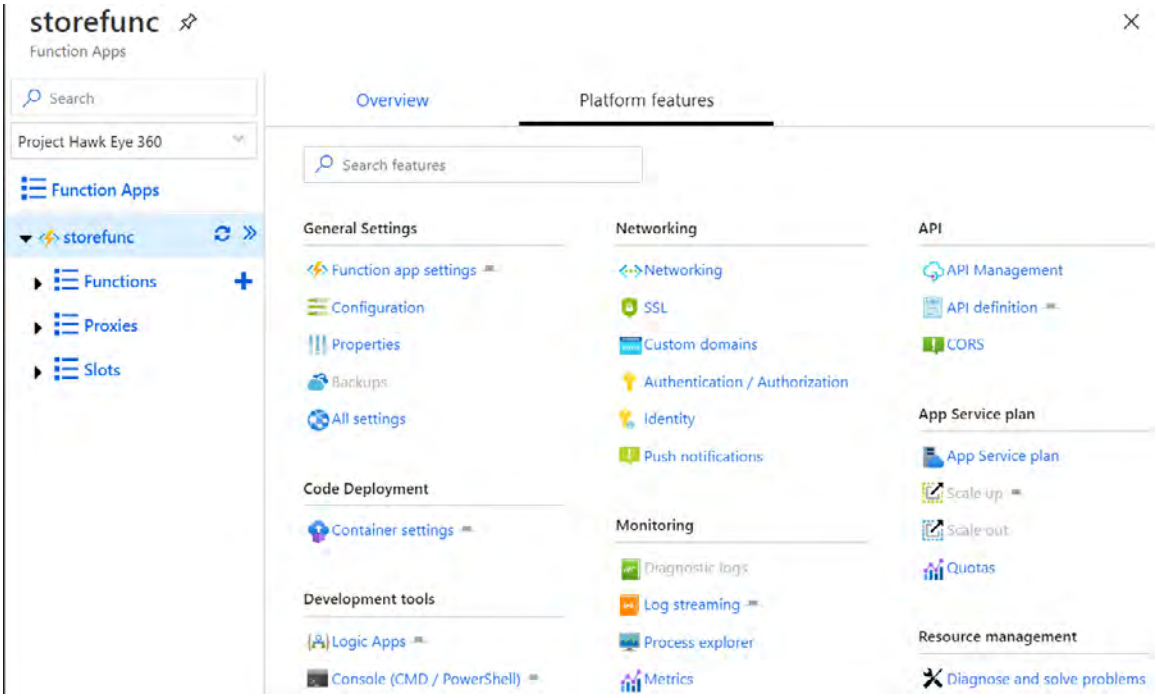


Figure 10.3 : fonctions de la plateforme d'une application de fonction

Ces fonctions de plateforme peuvent être utilisées pour configurer l'authentification, les domaines personnalisés, SSL, etc. En outre, l'onglet **Platform Features** (Fonctions de la plateforme) fournit une vue d'ensemble des outils de développement qui peuvent être utilisés avec l'application de fonction. La section suivante porte sur les paramètres de l'application de fonction disponibles dans les fonctionnalités de la plateforme.

Paramètres de la fonction App Service

Ces paramètres ont un impact sur toutes les fonctions. C'est ici que les paramètres d'application peuvent être gérés. Les proxys d'Azure Functions peuvent être activés et désactivés. Nous allons discuter des proxys plus loin dans ce chapitre. Ceux-ci permettent également de changer le mode d'édition d'une application de la fonction et d'effectuer le déploiement sur des emplacements :

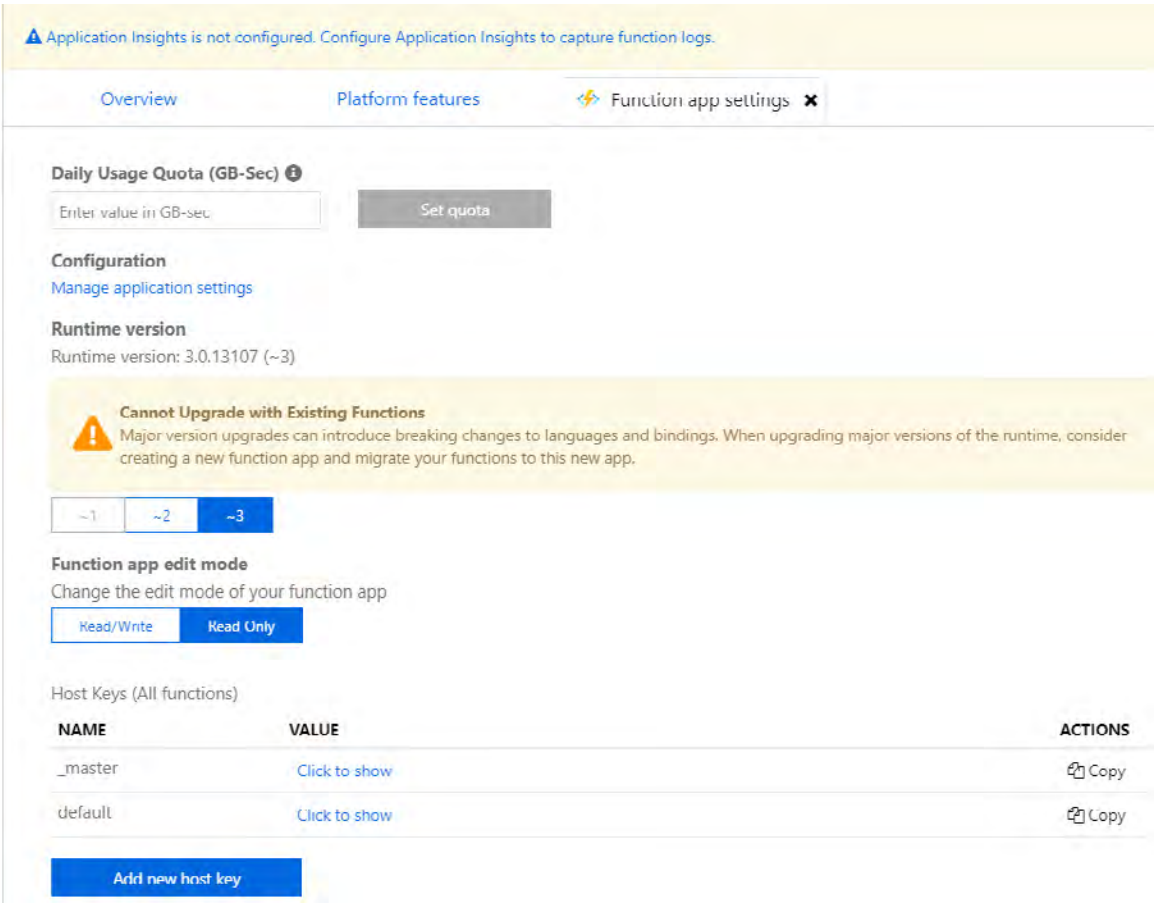


Figure 10.4 : paramètres de l'application de fonction

Le budget joue un rôle essentiel dans la réussite de tout projet. Découvrons les différents plans proposés pour Azure Functions.

Plans tarifaires d'Azure Functions

Azure Functions repose sur Azure App Service et offre un modèle de coûts avantageux aux utilisateurs. Il existe trois modèles de coût.

Un plan de consommation

Celui-ci se base sur la consommation par seconde réelle et l'exécution des fonctions. Ce plan calcule les coûts en fonction de l'utilisation des ressources de calcul durant la consommation effective et l'exécution de la fonction. Si une fonction n'est pas exécutée, aucuns frais ne lui est associé. Toutefois, un tel plan ne nuit en aucun cas aux performances. Azure Functions procédera à un dimensionnement automatique à la demande afin de maintenir des niveaux de performance minimaux. Une fonction dispose de 10 minutes pour mener à bien son exécution.

L'un des principaux inconvénients de ce plan réside dans le fait que si aucune fonction n'est consommée pendant quelques secondes, la fonction risque de se refroidir et la réponse à la prochaine requête est susceptible de présenter un léger retard, en raison de l'inactivité de la fonction. Ce phénomène est appelé démarrage à froid. Toutefois, des solutions de contournement peuvent maintenir les fonctions chaudes même en l'absence de demandes légitimes. Pour ce faire, vous pouvez écrire une fonction planifiée qui continue d'invoquer la fonction cible pour la garder active.

Un plan Premium

Il s'agit d'un plan relativement nouveau qui offre de nombreux avantages par rapport à App Service et à un plan de consommation. Ce plan n'implique aucun démarrage à froid pour Azure Functions. Les fonctions peuvent être associées à un réseau privé et les clients peuvent choisir leurs propres tailles de machines virtuelles pour exécuter des fonctions. Il fournit de nombreuses installations prêtes à l'emploi qui n'étaient pas proposées auparavant avec les deux autres types de plans.

Un plan App Service

Ce plan fournit des machines virtuelles entièrement dédiées en arrière-plan des fonctions, par conséquent le coût est directement proportionnel à celui de la machine virtuelle et à sa taille. Dans le cadre de ce plan, des frais fixes sont applicables même si les fonctions ne sont pas invoquées. Le code de fonction peut s'exécuter aussi longtemps que nécessaire. Bien qu'il n'y ait aucune restriction de temps, la limite par défaut est définie à 30 minutes. Cette valeur peut être modifiée dans le fichier **`hosts.json`**. Dans le cadre du plan App Service, le runtime de la fonction devient inactif s'il n'est pas utilisé durant quelques minutes et peut être activé uniquement à l'aide d'un déclencheur HTTP. Il existe un paramètre **Toujours activé**, qui peut être utilisé pour éviter que le runtime de la fonction ne devienne inactif. Le dimensionnement se fait manuellement, ou selon des paramètres de dimensionnement automatique.

Parallèlement à l'option de tarification flexible, Azure propose également différentes options d'hébergement pour le déploiement d'architecture.

Hôtes de destination Azure Functions

Le runtime Azure Functions peut être hébergé sur Windows, ainsi que sur des hôtes Linux. Les fonctions PowerShell Core, node. js, Java, Python et .NET Core peuvent s'exécuter à la fois sur Windows et sur les systèmes d'exploitation Linux. Il est important d'identifier le type de système d'exploitation sous-jacent requis pour les fonctions, car ce paramètre de configuration est lié à l'application de fonction et finalement à toutes les fonctions qui y sont contenues.

Il est également possible d'exécuter les fonctions au sein de conteneurs Docker. En effet, Azure fournit des images Docker qui disposent d'un runtime de fonction pré-intégré et les fonctions peuvent être hébergées à l'aide de ces images. Les images Docker peuvent désormais être utilisées pour créer des conteneurs dans des pods Kubernetes et hébergés sur Azure Kubernetes Service, Azure Container Instances ou des clusters Kubernetes non gérés. Ces images peuvent être stockées dans Docker Hub, Azure Container Registry ou tout autre référentiel d'images global et privé.

Pour vous aider à mieux comprendre ces notions, étudions certains des cas d'utilisation les plus importants d'Azure Functions.

Cas d'utilisation des fonctions Azure Functions

Azure Functions s'applique à de nombreuses mises en œuvre. Examinons quelques-uns de ces cas d'utilisation.

Mise en œuvre de microservices

Azure Functions permet de diviser des applications à grande échelle en des unités de code fonctionnelles plus modestes. Chaque unité est traitée indépendamment des autres et suit son propre cycle de vie. Chacune de ces unités de code s'assortit de ses propres exigences en termes de calcul, de matériel et de suivi. Chaque fonction peut être connectée aux autres. Ces unités sont connectées ensemble par des orchestrateurs afin de créer une fonction complète. Si nous prenons l'exemple d'une application de commerce électronique, il peut y avoir des fonctions individuelles (unités de code), chacune responsable de catalogues de listes, de recommandations, de catégories, de sous-catégories, des paniers d'achat, du règlement, des types de paiement, des passerelles de paiement, des adresses de livraison, des adresses de facturation, des taxes, des frais d'expédition, des annulations, des retours, des e-mails, des messages SMS et ainsi de suite. Certaines de ces fonctions sont regroupées afin de créer des cas d'utilisation pour des applications de commerce électronique, par exemple la navigation à travers les produits, le flux de règlement, etc.

Intégration entre plusieurs points de terminaison

Azure Functions peut créer la fonctionnalité globale de l'application en intégrant plusieurs fonctions. Cette intégration peut se fonder sur le déclenchement d'événements, ou s'appuyer sur la publication d'éléments (« Push»). Elle permet de décomposer les grandes applications monolithiques en petits composants.

Traitement des données

Azure Functions peut être utilisé pour traiter les données entrantes par lot. Cela permet de traiter les données se présentant sous divers formats, tels que XML, CSV, JSON et texte. Cette solution peut également exécuter la conversion, l'enrichissement, le nettoyage et le filtrage des algorithmes. En effet, plusieurs fonctions peuvent être utilisées, chacune prenant en charge la conversion, l'enrichissement, le nettoyage ou le filtrage. Azure Functions permet également d'incorporer des Cognitives services avancés tels que la **reconnaissance optique des caractères (OCR)**, la vision par ordinateur, la manipulation d'images et la conversion. C'est idéal si vous souhaitez traiter les réponses de l'API et les convertir.

Intégration des applications héritées

Azure Functions permet de faciliter l'intégration des applications héritées avec des protocoles plus récents et des applications modernes. Il est possible que les applications héritées ne prennent pas en charge les protocoles et les formats standard de l'industrie. Azure Functions peut agir comme un proxy pour ces applications héritées, accepter les requêtes des utilisateurs ou d'autres applications, convertir les données dans un format compris par une application héritée et communiquer avec cette dernière via des protocoles qu'elle comprend. Cela ouvre la porte à une multitude de possibilités pour intégrer des applications anciennes et héritées au portefeuille général.

Tâches planifiées

Azure Functions peut être utilisé pour exécuter continuellement ou périodiquement certaines fonctions de l'application. Ces fonctions d'application peuvent varier, il peut s'agit de sauvegardes périodiques, de restaurations, d'exécution de tâches par lot, de l'exportation et de l'importation de données, d'envoi d'e-mails groupés, etc.

Passerelles de communication

Azure Functions peut être utilisé dans des passerelles de communication, par exemple des hubs de notification, des services SMS, le courrier électronique, etc. Vous pouvez par exemple utiliser Azure Functions pour envoyer une notification push aux appareils Android et iOS à l'aide d'Azure Notification Hubs.

Les fonctions Azure sont disponibles dans différents types, qui doivent être sélectionnés en fonction de leur relation avec l'optimisation des charges de travail. Examinons-les de plus près.

Types de fonctions Azure Functions

Azure Functions possède des fonctions qui peuvent se classer en trois types différents :

- **Fonctions à la demande :** il s'agit des fonctions qui sont exécutées lorsqu'elles sont explicitement appelées ou invoquées. Il s'agit par exemple des webhooks et des fonctions basées sur le protocole HTTP.
- **Fonctions planifiées :** ces dernières s'apparentent à des tâches de minuterie et exécutent des fonctions à des intervalles fixes.
- **Fonctions basées sur les événements :** ces fonctions sont exécutées en fonction d'événements externes. Par exemple, le chargement d'un nouveau fichier dans le stockage blob Azure génère un événement qui peut enclencher l'exécution des fonctions Azure.

La section suivante porte sur la création d'une fonction pilotée par les événements qui sera connectée à un compte de stockage Azure.

Création d'une fonction pilotée par les événements

Dans cet exemple, une fonction Azure sera créée et connectée à un compte de stockage Azure. Ce compte de stockage dispose d'un conteneur qui détient tous les fichiers Blob. Le nom du compte de stockage est **incomingfiles** et le conteneur est **ordres**, comme illustré à la *Figure* 10.5 :

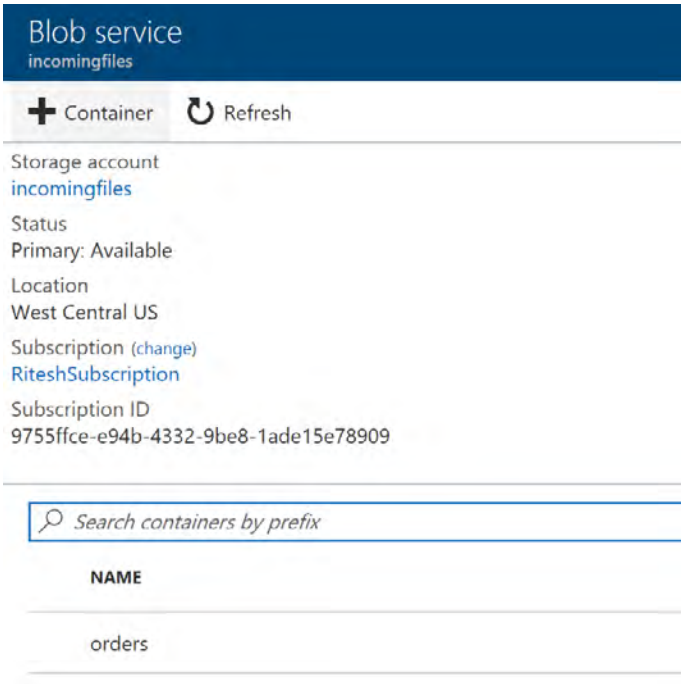


Figure 10.5 : détails du compte de stockage

Pour créer une fonction Azure dans le portail Azure, procédez comme suit :

1. Cliquez sur le bouton **+** à côté du menu **Functions** (Fonctions) sur la gauche.
2. Sélectionnez **in-Portal** sur l'écran qui s'affiche, puis cliquez sur le bouton **Continue** (Continuer).
3. Sélectionnez le **déclencheur de stockage Blob Azure**, comme illustré à la *Figure 10.6* :

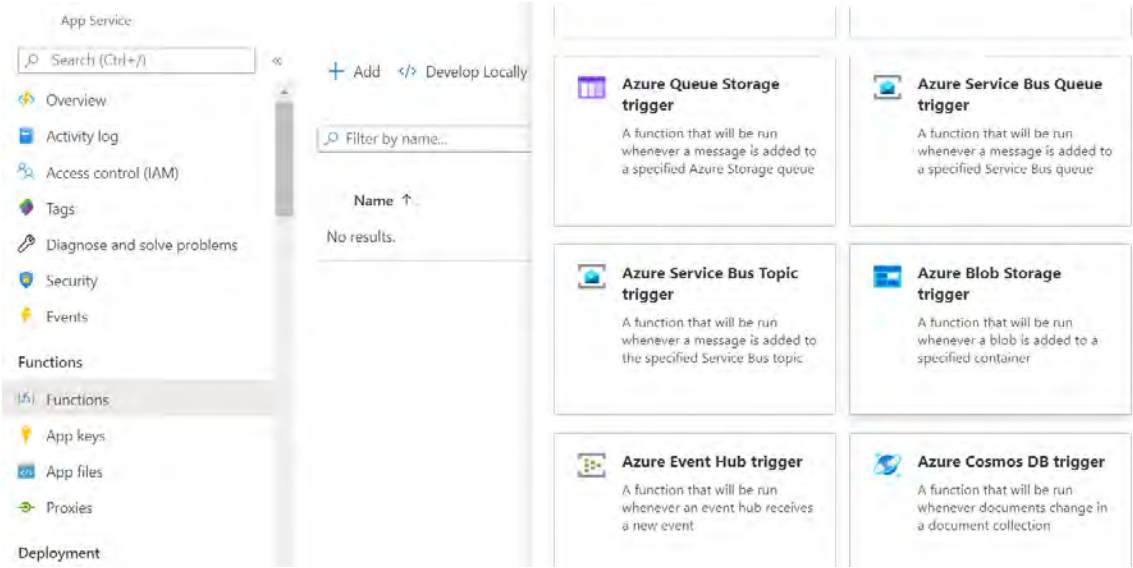


Figure 10.6 : sélection du déclencheur de stockage Blob Azure

Cette fonction Azure ne dispose pas d'informations de connectivité au compte de stockage à l'heure actuelle. Les fonctions Azure ont besoin d'informations de connexion au compte de stockage, lesquelles sont disponibles dans l'onglet **Access keys** (Clés d'accès) du compte de stockage. Ces mêmes informations peuvent être obtenues dans l'environnement d'édition de la fonction Azure. En effet, il est possible de créer un compte de stockage à partir du même environnement d'édition.

Le déclencheur de stockage Blob Azure peut être ajouté à l'aide du bouton **New** (Nouveau) situé en regard du type **Entrée de connexion au compte stockage**. Celui-ci permet de sélectionner un compte de stockage existant ou d'en créer un nouveau. J'ai quelques comptes de stockage, je vais donc les réutiliser, mais vous devez créer un compte de stockage Azure distinct. La sélection d'un compte de stockage mettra à jour les paramètres de la section **appSettings**, en intégrant la chaîne de connexion ajoutée.

Veillez à ce qu'un conteneur existe déjà au sein du service blob du compte de stockage Azure cible. L'entrée du chemin d'accès désigne le chemin d'accès au conteneur. Dans ce cas, le conteneur **ordres** existe déjà dans le compte de stockage. Le bouton **Create** (Créer) indiqué ici met en service la nouvelle fonction de contrôle du conteneur du compte de stockage :

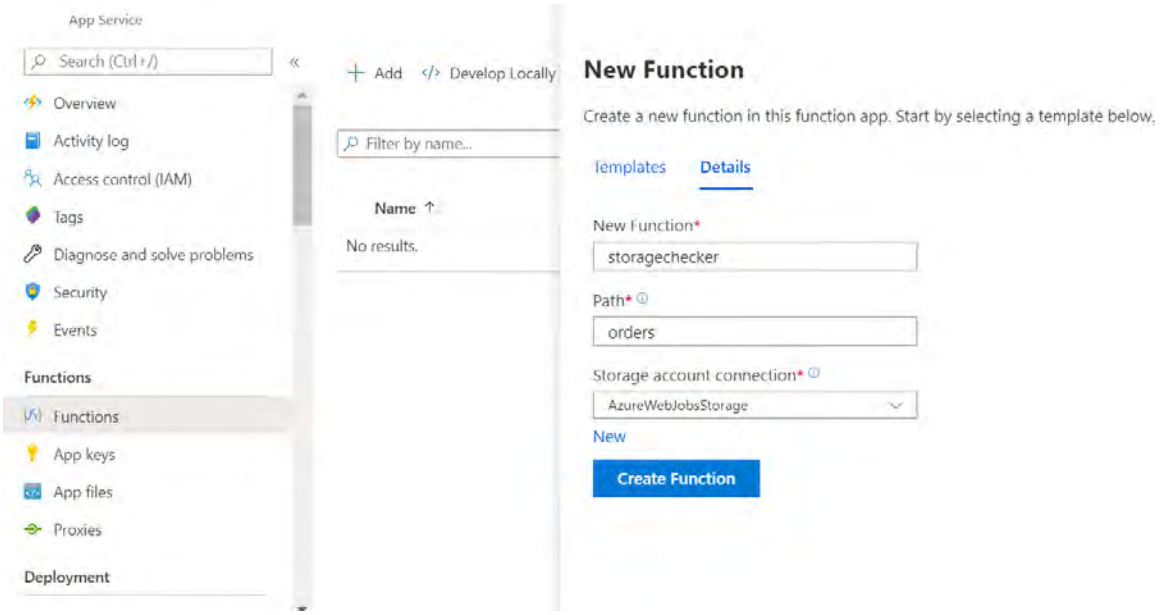


Figure 10.7 : création d'une fonction de surveillance d'un conteneur de compte de stockage

Le code de la fonction **`storagerelatedfunctions`** se présente comme suit :

```
public static void Run(Stream myBlob, TraceWriter log)
{
    log.Info($"C# Blob trigger function Processed blob\n  \n Size {myBlob.
Length} Bytes");
}
```

Les liaisons sont les suivantes :

```
{
  "bindings": [
    {
      "name": "myBlob",
      "type": "blobTrigger",
      "direction": "in",
      "path": "orders",
      "connection": "azureforarchitead2b_STORAGE"
    }
  ],
  "disabled": false
}
```

Désormais, le chargement d'un fichier blob dans le conteneur **ordres** devrait déclencher la fonction :

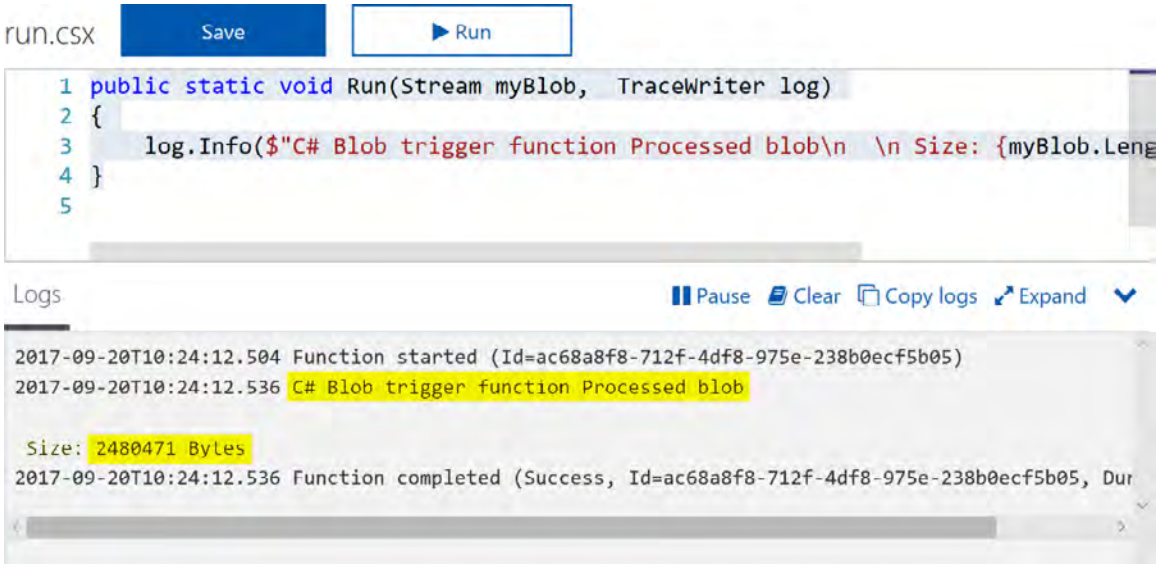


Figure 10.8 : blob traité de la fonction de déclenchement blob C#

La section suivante porte sur les proxys de fonction Azure, lesquels vous permettront de gérer efficacement les demandes et les réponses de vos API.

Proxys d'Azure Function

Les proxys de fonction Azure ont récemment été ajouté à Azure Functions. Les proxys de fonction masquent les détails d'Azure Functions et exposent des points de terminaison complètement différents aux clients. Les proxys de fonction peuvent recevoir des requêtes sur des points de terminaison, modifier le contenu, le corps, les en-têtes et l'URL de la demande en modifiant les valeurs et les enrichir avec des données supplémentaires et les transmettre en interne à Azure Functions. Une fois qu'ils obtiennent une réponse de la part de ces fonctions, ils peuvent à nouveau convertir, substituer et augmenter la réponse et la renvoyer au client.

Ils permettent également d'invoquer différentes fonctions pour les opérations CRUD (créer, lire, supprimer et mettre à jour) à l'aide de différents en-têtes, en scindant les grandes fonctions en des fonctions plus petites. Ils fournissent un niveau de sécurité en n'exposant pas le point de terminaison de la fonction d'origine et contribuent également à modifier l'implémentation de la fonction interne et des points de terminaison sans affecter l'appelant. Les proxys de fonction fournissent aux clients une seule fonction URL, puis invoquent plusieurs fonctions Azure en arrière-plan pour finaliser les charges de travail. Pour en savoir plus sur les proxys Azure Functions, accédez à https://docs.microsoft.com/azure/azure-functions/functions-proxies.

Nous allons découvrir Durable Functions dans la section suivante.

Durable Functions

Durable Functions est l'une des toutes dernières fonctions d'Azure Functions. Il permet aux architectes d'écrire des charges de travail avec état dans une fonction Orchestrator, qui est un nouveau type de fonction. En tant que développeur, vous pouvez choisir de le coder ou d'utiliser n'importe quelle forme d'IDE. Voici les avantages de Durable Functions :

- La sortie de la fonction peut être enregistrée dans des variables locales et vous pouvez appeler d'autres fonctions de manière synchrone et asynchrone.
- L'état est préservé pour vous.

Voici le mécanisme de base permettant d'appeler Durable Functions :

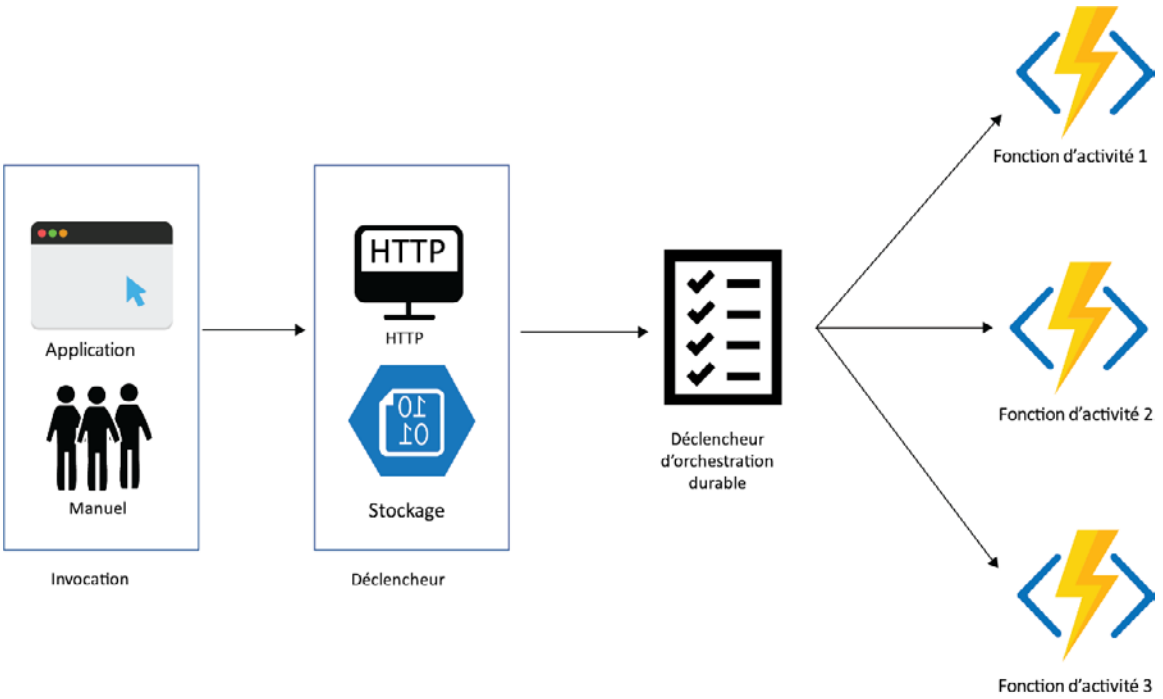


Figure 10.9 : mécanisme permettant d'appeler Durable Functions

Les fonctions Durable Functions peuvent être appelées par tout déclencheur fourni par Azure Functions. Ces déclencheurs incluent HTTP, le stockage blob, le stockage de table, la file d'attente Service Bus, etc. Ils peuvent être déclenchés manuellement par une personne ayant accès à ces derniers, ou par une application. La *Figure 10.9* montre quelques déclencheurs à titre d'exemple. Ils sont également considérés comme des fonctions Durable Functions de départ. Les fonctions Durable Functions de départ appellent le **déclencheur d'orchestration durable**, qui contient la logique principale pour l'orchestration et orchestre l'invocation des fonctions d'activité.

Le code écrit au sein de l'orchestrateur durable doit être déterministe. Cela signifie que quel que soit le nombre de fois où le code est exécuté, les valeurs retournées doivent rester les mêmes. La fonction Orchestrator est une fonction longue par nature. Cela signifie qu'elle peut être hydraté, sérialisée par l'état et qu'elle se met en veille une fois qu'elle a appelé une fonction d'activité durable. En effet, elle ignore la durée d'exécution de la fonction d'activité durable et ne veut pas l'attendre. Lorsque la fonction d'activité durable termine son exécution, la fonction Orchestrator est à nouveau exécutée. L'exécution de la fonction commence à partir du haut et s'exécute jusqu'à ce qu'elle appelle une autre fonction d'activité durable ou termine l'exécution de la fonction. Elle doit réexécuter les lignes de code qu'elle a déjà exécutées précédemment et doit obtenir les mêmes résultats qu'auparavant. Sachez que le code écrit au sein de l'orchestrateur durable doit être déterministe. Cela signifie que quel que soit le nombre de fois où le code est exécuté, les valeurs retournées doivent rester les mêmes.

Essayons de comprendre cela à l'aide d'un exemple. Si nous utilisons une classe générale .NET Core datetime et que nous renvoyons la date/heure actuelle, une nouvelle valeur sera générée chaque fois que nous exécuterons la fonction. L'objet de contexte de Durable Functions indique **`CurrentUtcDateTime`** , qui retournera la valeur date/heure qu'elle a renvoyée la première fois lors de la nouvelle exécution.

Ces fonctions d'orchestration peuvent également attendre des événements externes et activer des scénarios liés au transfert humain. Ce concept sera expliqué plus loin dans cette section.

Ces fonctions d'activité peuvent être appelées avec ou sans mécanisme de nouvelle tentative. Les fonctions Durable Functions peuvent aider à résoudre de nombreux défis et fournir des fonctions pour écrire des fonctions qui permettent d'effectuer les opérations suivantes :

- Exécution de fonctions longues
- Maintien de l'état
- Exécution des fonctions enfants en parallèle ou en séquence
- Récupération aisée après l'échec
- Orchestration de l'exécution des fonctions dans une charge de travail

Maintenant que vous connaissez le fonctionnement interne d'une fonction durable, découvrons comment créer une fonction durable dans Visual Studio.

Procédure de création d'une fonction durable à l'aide de Visual Studio

Voici la procédure de création d'une fonction durable :

1. Accédez au portail Azure et cliquez sur **Resource groups** (Groupes de ressource) dans le menu de gauche.
2. Cliquez sur le bouton **+Add** (+Ajouter) dans le menu supérieur pour créer un groupe de ressources.
3. Fournissez les informations du groupe de ressources dans le formulaire résultant et cliquez sur le bouton **Create** (Créer), comme illustré ici :

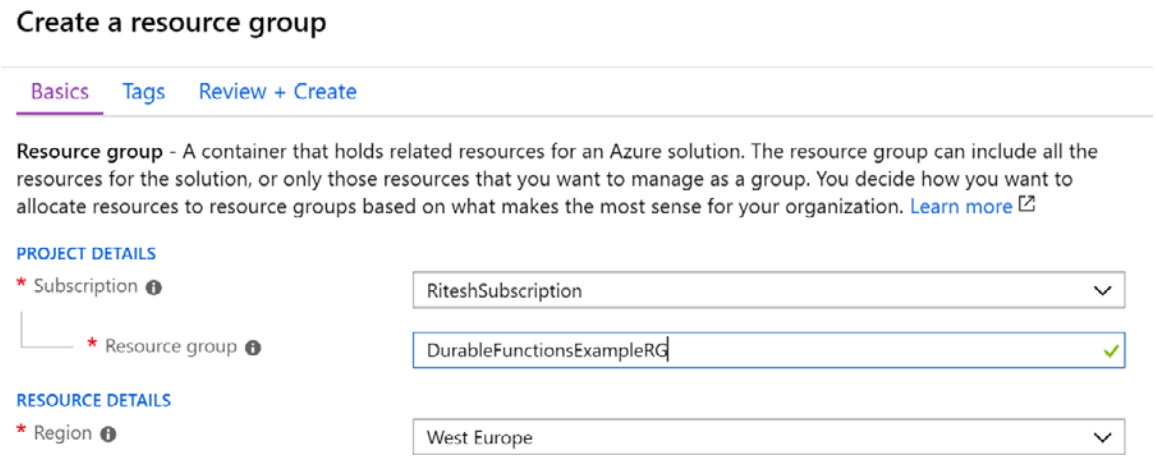


Figure 10.10 : création d'un groupe de ressources

4. Accédez au groupe de ressources nouvellement créé et ajoutez une nouvelle application de fonction en cliquant sur le bouton **+Add** (+Ajouter) dans le menu supérieur et recherchez l'`application de fonction` dans la zone de recherche résultante.
5. Sélectionnez **Function App** (Application de fonction) et cliquez sur le bouton **Create** (Créer). Remplissez le formulaire d'application de fonction résultant et cliquez sur le bouton **Create** (Créer) Vous pouvez également réutiliser l'application de fonction créée précédemment.
6. Une fois l'application de fonction créée, accédons à notre environnement de développement local, sur lequel est installé Visual Studio 2019. Nous allons commencer avec Visual Studio et créer un projet de type `Azure Functions` , lui fournir un nom et sélectionner **Azure Functions v3** (**.NET core**) pour **Function runtime** (Runtime de fonction).
7. Une fois le projet créé, nous devons ajouter le package NuGet `DurableTask` au projet pour travailler avec Durable Functions. La version utilisée au moment de la rédaction de ce chapitre est la version 2.2.2 :

Figure 10.11 : ajout d'un package DurableTask NuGet

8. Nous pouvons désormais coder nos fonctions durables au sein de Visual Studio. Ajoutez une nouvelle fonction, nommez-la et sélectionnez le type de déclencheur **Durable Functions Orchestration** (Orchestration Durable Functions) :

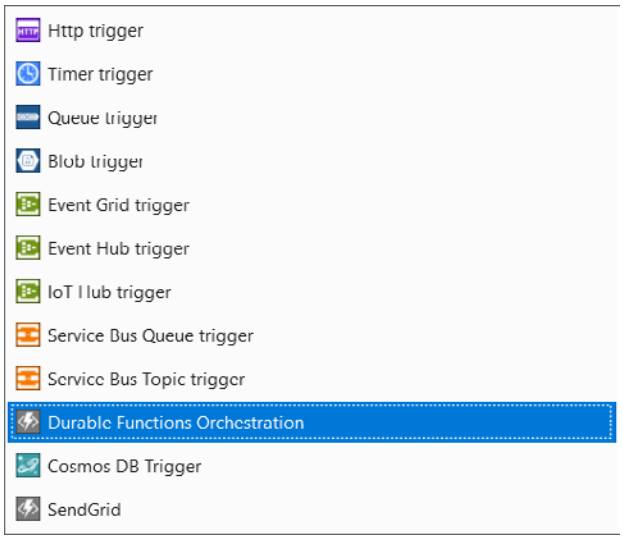


Figure 10.12 : sélection d'un déclencheur d'orchestration Durable Functions

9. Visual Studio génère le code standard pour Durable Functions, et nous allons l'utiliser pour en savoir plus sur Durable Functions. Les activités Durable Functions sont des fonctions appelées par la fonction Orchestrator principale. Il y a généralement une fonction Orchestrator principale et plusieurs activités Durable Functions. Une fois l'extension installée, fournissez un nom pour la fonction et écrivez le code qui effectue une tâche utile, comme l'envoi d'un e-mail ou d'un SMS, la connexion à des systèmes externes et l'exécution de la logique, ou l'exécution de services à l'aide de leurs points de terminaison, tels que Cognitive Services.

   Visual Studio génère trois ensembles de fonctions dans une seule ligne de code :

   - **HttpStart** : il s'agit de la fonction de démarrage. Elle est responsable du démarrage de l'orchestration de la fonction durable. Le code généré se compose d'une fonction de démarrage de déclencheur HTTP. Il peut toutefois s'agir de n'importe quelle fonction basée sur un déclencheur, telle que **`BlobTrigger`**, une file d'attente **`ServiceBus`** ou une fonction basée sur un déclencheur.
   - **RunOrchestrator** : il s'agit de la principale fonction d'orchestration durable. Elle est responsable de l'acceptation des paramètres de la fonction de démarrage et, à son tour, appelle plusieurs fonctions de tâche durables. Chaque fonction de tâche durable est responsable d'une fonctionnalité et ces tâches durables peuvent être appelées en parallèle ou en séquence selon les besoins.

- **SayHello** : il s'agit de la fonction de tâche durable qui est appelée à partir de l'orchestrateur de fonction durable pour effectuer une tâche particulière.

10. Le code de la fonction de démarrage (**`HttpStart`**) s'affiche ensuite. Cette fonction est associée à un déclencheur de type HTTP et accepte une liaison supplémentaire de type **`DurableClient`**. Cet objet **`DurableClient`** permet d'invoquer la fonction Orchestrator :

```
[FunctionName("Function1_HttpStart")]
0 references
public static async Task<HttpResponseMessage> HttpStart(
    [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post")] HttpRequestMessage req,
    [DurableClient] IDurableOrchestrationClient starter,
    ILogger log)
{
    // Function input comes from the request content.
    string instanceId = await starter.StartNewAsync("Function1", null);

    log.LogInformation($"Started orchestration with ID = '{instanceId}'.");

    return starter.CreateCheckStatusResponse(req, instanceId);
}
```

Figure 10.13 : code de la fonction de démarrage

11. Le code de la fonction Orchestrator (**`RunOrchestrator`**) s'affiche ensuite. Cette fonction est associée à un déclencheur de type **`OrchestrationTrigger`** et accepte un paramètre de type **`IDurableOrchestrationContext`**. Cet objet de contexte permet d'invoquer des tâches durables :

```
[FunctionName("Orchestrator")]
0 references
public static async Task<List<string>> RunOrchestrator(
    [OrchestrationTrigger] IDurableOrchestrationContext context)
{
    var outputs = new List<string>();

    // Replace "hello" with the name of your Durable Activity Function.
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "Tokyo"));
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "Seattle"));
    outputs.Add(await context.CallActivityAsync<string>("Function1_Hello", "London"));

    // returns ["Hello Tokyo!", "Hello Seattle!", "Hello London!"]
    return outputs;
}
```

Figure 10.14 : code pour la fonction de déclencheur d'orchestration

12. Le code de la fonction de tâche durable (**`HelloFunction`**) s'affiche ensuite. Cette fonction est associée à un déclencheur de type **`ActivityTrigger`** et accepte un paramètre qui peut être de n'importe quel type nécessaire pour exécuter sa fonctionnalité. Il présente une valeur de retour de type **`string`** et la fonction est responsable du renvoi d'une valeur de chaîne vers la fonction d'orchestration :

```
[FunctionName("HelloFunction")]
0 references
public static string SayHello([ActivityTrigger] string name, ILogger log)
{
    log.LogInformation($"Saying hello to {name}.");
    return $"Hello {name}!";
}
```

Figure 10.15 : code pour la fonction de tâche durable

Ensuite, nous pouvons exécuter la fonction localement, qui va démarrer un émulateur de stockage si aucun d'entre eux n'est démarré, et fournira une URL pour la fonction de déclencheur HTTP :

```
C:\Users\Ritesh\AppData\Local\AzureFunctionsTools\Releases\3.6.0\cli_x64\func.exe
[05 06 2020 07:20:20] {
[05-06-2020 07:20:20]   "DynamicThrottlesEnabled": false,
[05-06-2020 07:20:20]   "MaxConcurrentRequests": -1,
[05-06-2020 07:20:20]   "MaxOutstandingRequests": -1,
[05-06-2020 07:20:20]   "RoutePrefix": "api"
[05-06-2020 07:20:20] }
[05-06-2020 07:20:20] Starting JobHost
[05-06-2020 07:20:20] Starting Host (HostId=riteshpc-14923052, InstanceId=c2b4b895-e8e8-4f2d-9508-d4e4b25f3760, Version=
3.0.13353.0, ProcessId=11308, AppDomainId=1, InDebugMode=False, InDiagnosticMode=False, FunctionsExtensionVersion=(null)
)
[05 06 2020 07:20:20] Loading functions metadata
[05-06-2020 07:20:20] 3 functions loaded
[05-06-2020 07:20:20] Generating 3 job function(s)
[05-06-2020 07:20:20] Found the following functions:
[05-06-2020 07:20:20] FunctionApp7.DurbleFunctions.SayHello
[05-06-2020 07:20:20] FunctionApp7.DurbleFunctions.HttpStart
[05-06-2020 07:20:20] FunctionApp7.DurbleFunctions.RunOrchestrator
[05-06-2020 07:20:20]
[05-06-2020 07:20:20] Initializing function HTTP routes
[05 06 2020 07:20:20] Mapped function route 'api/HttpStart' [get,post] to 'HttpStart'
[05 06 2020 07:20:20]
[05-06-2020 07:20:20] Host initialized (360ms)
[05-06-2020 07:20:20] Starting task hub worker. InstanceId: . Function: . HubName: TestHubName. AppName: . SlotName: . E
xtensionVersion: 2.2.2. SequenceNumber: 0.

Http Functions:

        HttpStart: [GET,POST] http://localhost:7071/api/HttpStart
```

Figure 10.16 : démarrage de l'émulateur de stockage

Nous allons invoquer cette URL à l'aide d'un outil connu sous le nom de **Postman** (qui peut être téléchargé à l'adresse https://www.getpostman.com/). Il nous suffit de copier l'URL et de l'exécuter dans Postman. Cette activité est illustrée à la *Figure 10.17* :

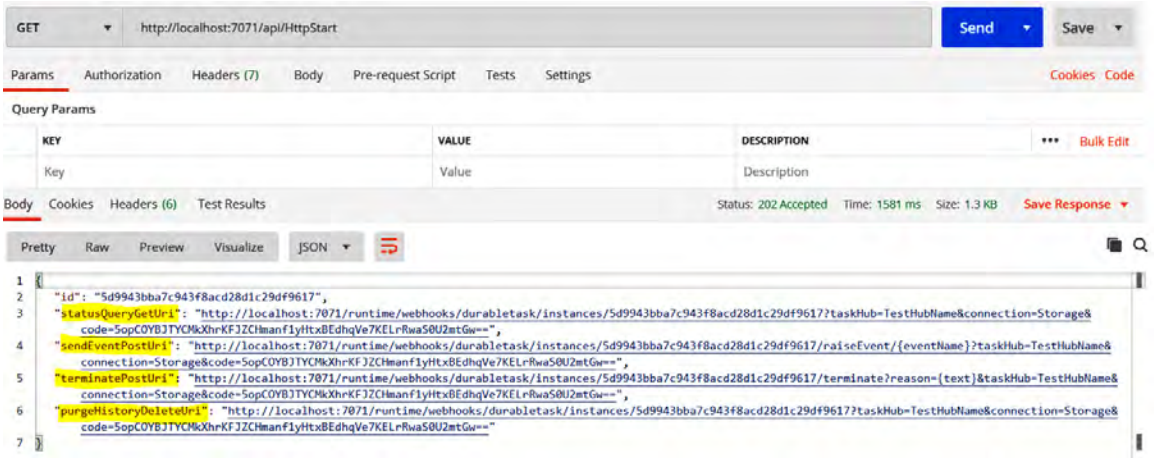


Figure 10.17 : appel d'URL à l'aide de Postman

Notez que cinq URL sont générées lorsque vous démarrez l'orchestrateur :

- L'URL **`statusQueryGetUri`** est utilisée pour rechercher l'état actuel de l'orchestrateur. Cliquez sur cette URL sur Postman pour ouvrir un nouvel onglet et si vous exécutez cette requête, l'état de la charge de travail s'affiche :

Figure 10.18 : état actuel de l'orchestrateur

- L'URL **`terminatePostUri`** est utilisée pour interrompre une fonction Orchestrator déjà en cours d'exécution.
- L'URL **`sendEventPostUri`** est utilisée pour publier un événement sur une fonction durable suspendue. Les fonctions durables peuvent être suspendues si elles attendent un événement externe. Cette URL est utilisée dans ces cas.

- L'URL **`purgeHistoryDeleteUri`** est utilisée pour supprimer l'historique géré par Durable Functions pour un appel particulier à partir de son compte de stockage de table.

Maintenant que vous savez comment travailler avec Durable Functions à l'aide de Visual Studio, découvrons un autre aspect d'Azure Functions : enchaîner les fonctions ensemble.

Création d'une architecture connectée avec des fonctions

Une architecture connectée avec des fonctions fait référence à la création de fonctions multiples, dans le cadre desquelles la sortie d'une fonction déclenche une autre fonction et fournit des données à la fonction suivante afin qu'elle exécute sa logique. Dans cette section, nous allons poursuivre le scénario précédent du compte de stockage. Dans ce cas, la sortie de la fonction se déclenche à l'aide de fichiers de stockage blob Azure, et va écrire la taille de fichier dans Azure Cosmos DB.

La configuration de Cosmos DB est illustrée ci-après. Par défaut, aucune collection n'est créée dans Cosmos DB.

Une collection sera créée automatiquement durant la création d'une fonction qui se déclenchera lorsque Cosmos DB obtiendra des données.

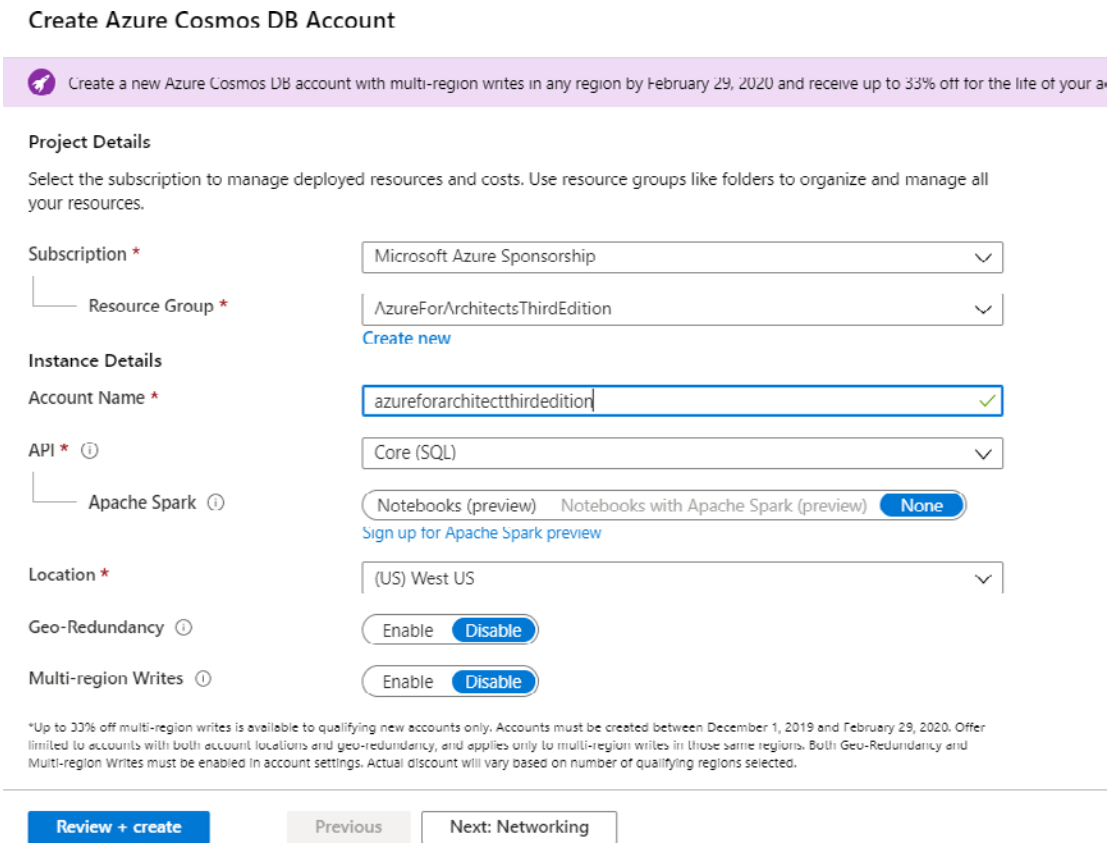


Figure 10.19 : création d'un compte Azure Cosmos DB

Voici la procédure à suivre pour récupérer les données de la fonction suivante à partir de la sortie d'une fonction.

1. Créez une base de données **`testdb`**, dans Cosmos DB et une collection dénommée **`testcollection`** en son sein. Vous aurez besoin du nom de la base de données et de la collection lors de la configuration des fonctions Azure :

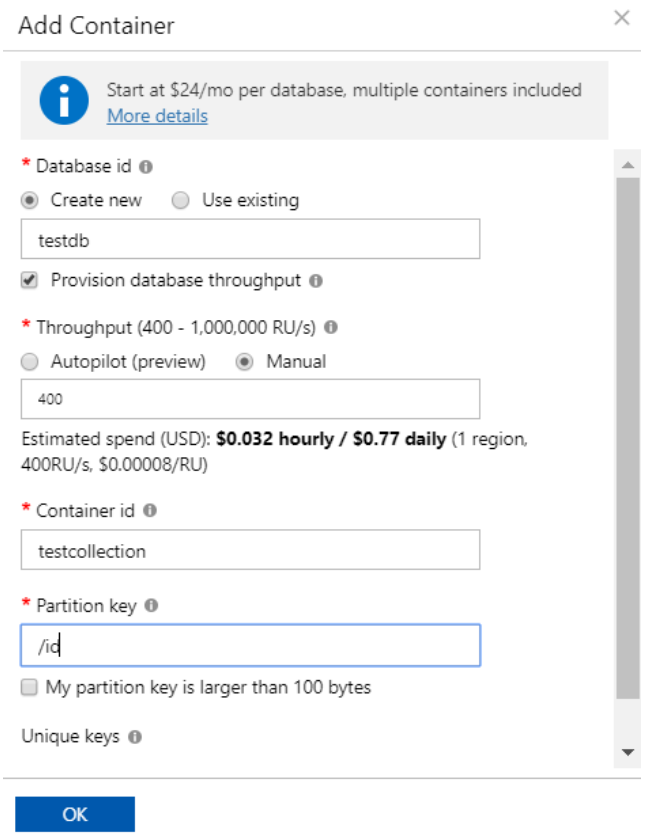


Figure 10.20 : ajout d'un conteneur

2. Créez une fonction qui présentera un déclencheur de stockage blob et une liaison de sortie CosmosDB. La valeur retournée par la fonction sera la taille des données du fichier téléchargé. Cette valeur retournée est écrite dans Cosmos DB. La liaison de sortie sera écrite dans la collection Cosmos DB. Accédez à l'onglet **Integrate** (Intégrer) et cliquez sur le bouton **New Output** (Nouvelle sortie) en dessous de l'étiquette **Outputs** (Sorties) et sélectionnez **Azure Cosmos DB** :

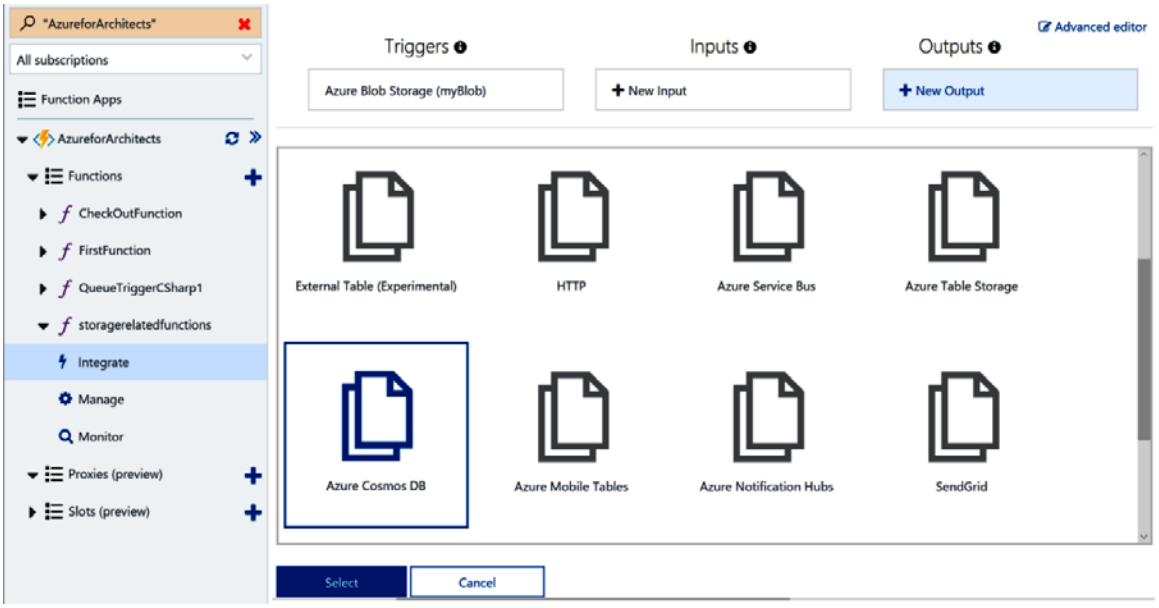


Figure 10.21 : liaison de sortie vers Azure Cosmos DB

3. Indiquez les noms appropriés pour la base de données et la collection (cochez la case afin de créer la collection si celle-ci n'existe pas), cliquez sur le bouton **New** (Nouveau) pour sélectionner la base de données Azure Cosmos DB que vous venez de créer et laissez le nom du paramètre **outputDocument** :

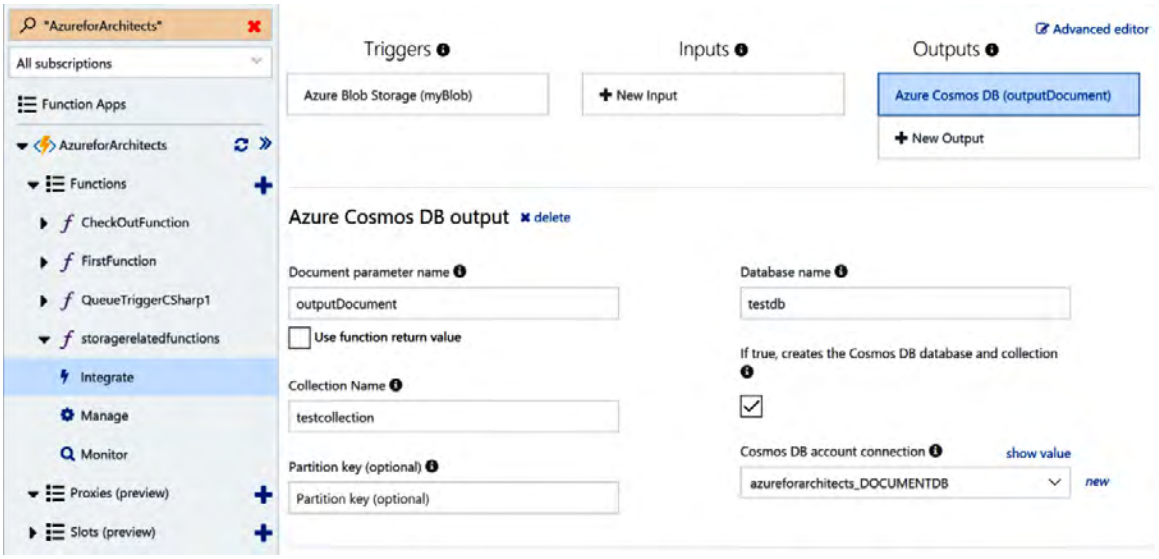


Figure 10.22 : base de données Azure Cosmos DB nouvellement créée

4. Modifiez la fonction comme illustré dans la *Figure* 10.23 :

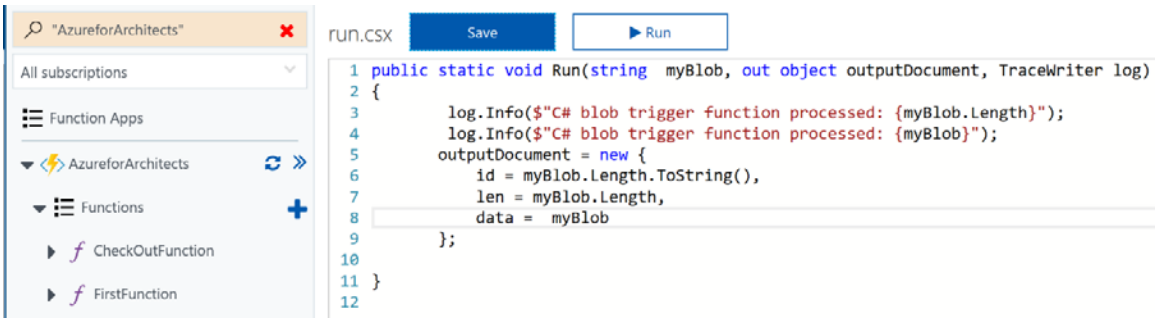


Figure 10.23 : modification de la fonction

5. Désormais, le chargement d'un nouveau fichier dans la collection de commandes du compte de stockage Azure exécutera une fonction qui sera écrite dans la collection Azure Cosmos DB. Une autre fonction peut être écrite dans le compte Azure Cosmos DB nouvellement créé en tant que déclencheur de liaison. Celle-ci fournira la taille des fichiers, et la fonction pourra agir en conséquence. Ceci est illustré ci-après :

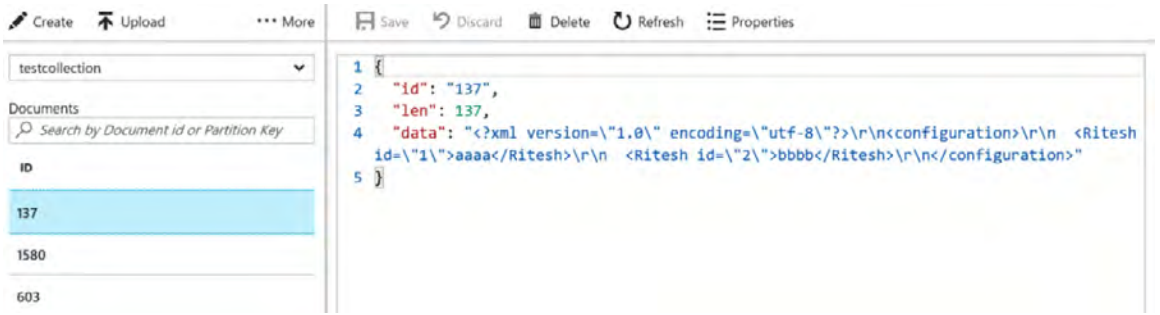


Figure 10.24 : écriture d'une fonction de liaison de déclencheurs

Cette section porte sur la façon dont la sortie d'une fonction peut être utilisée pour récupérer des données pour la fonction suivante. Dans la section suivante, vous découvrirez comment activer la création d'événement Serverless à l'aide d'Azure Event Grid.

Azure Event Grid

Azure Event Grid est un service relativement nouveau. Il a également été désigné sous le terme de plateforme de création d'événement sans serveur. Il permet de créer des applications sur la base d'événements (également connu sous le terme de **conception événementielle**). Il est important de comprendre ce que sont les événements et comment nous les avons traités avant Event Grid. Un événement est quelque chose qui s'est produit, c'est-à-dire, une activité qui a changé l'état d'un sujet. Lorsqu'un sujet subit un changement dans son état, il soulève généralement un événement.

Les événements suivent généralement le modèle de publication/abonnement (également connu sous le nom de **pub/sous-modèle**), dans lequel un sujet déclenche un événement en raison de son changement d'état, et cet événement peut ensuite être souscrit par plusieurs parties intéressées, également connues sous le nom **d'abonnés**. L'objectif de l'événement est d'informer les abonnés de ces changements et de leur fournir des données dans le cadre de son contexte. Les abonnés peuvent prendre toutes les mesures qu'ils jugent nécessaires, ce qui varie d'un abonné à un autre.

Avant Event Grid, il n'y avait pas de service qui pourrait être décrit comme une plateforme d'événements en temps réel. Il y avait des services distincts, et chacun fournissait son propre mécanisme pour gérer les événements.

Par exemple, Log Analytics, également connu sous le nom **d'Operations Management Suite** (**OMS**), fournit une infrastructure pour capturer les journaux d'environnement et la télémétrie sur lesquels des alertes peuvent être générées. Ces alertes peuvent être utilisées pour exécuter un runbook, un webhook ou une fonction. Ces événements sont quasiment en temps réel, mais ils ne sont pas complètement en temps réel. En outre, capturer chaque journal individuel pour prendre des mesures en conséquence est un travail laborieux. De même, il existe Application Insights, qui fournit des fonctions similaires à Log Analytics mais pour les applications.

Il existe d'autres journaux, tels que les journaux d'activité et les journaux de diagnostic, mais encore une fois, ils reposent sur des principes similaires que d'autres fonctions liées au journal. Les solutions sont déployées sur plusieurs groupes de ressources dans plusieurs régions et les événements déclenchés à partir de n'importe lequel d'entre eux doivent être disponibles pour les ressources qui sont déployées ailleurs.

Event Grid supprime toutes ces barrières et, par conséquent, les événements peuvent être générés par la plupart des ressources (ils deviennent de plus en plus disponibles), et même des événements personnalisés peuvent être générés. Ces événements peuvent ensuite être souscrits par n'importe quelle ressource, dans n'importe quelle région et dans n'importe quel groupe de ressources au sein de l'abonnement.

Event Grid est déjà défini dans le cadre de l'infrastructure Azure, ainsi que des datacenters et des réseaux. Les événements soulevés dans une région peuvent aisément être souscrits par des ressources dans d'autres régions, et étant donné que ces réseaux sont connectés, ce système est extrêmement efficace pour la livraison d'événements aux abonnés.

Event Grid

Event Grid vous permet de créer des applications avec une architecture basée sur les événements. Il existe des éditeurs d'événements et des consommateurs d'événements ; cependant, il peut y avoir plusieurs abonnés pour le même événement.

L'éditeur d'un événement peut être une ressource Azure, telle que le stockage Blob, les hubs **Internet des objets** (**IoT**) et bien d'autres encore. Ces éditeurs sont également appelés sources d'événements. Ces éditeurs utilisent des rubriques Azure prêtes à l'emploi pour envoyer leurs événements à Event Grid. Il n'est pas nécessaire de configurer la ressource ou la rubrique. Les événements déclenchés par les ressources Azure utilisent déjà des rubriques en interne pour envoyer leurs événements à Event Grid. Une fois que l'événement atteint la grille, il peut être consommé par les abonnés.

Les abonnés, ou les consommateurs, sont des ressources qui sont intéressées par les événements et qui souhaitent exécuter une action sur la base de ces événements. Ces abonnés fournissent un gestionnaire d'événements lorsqu'ils s'abonnent à la rubrique. Les gestionnaires d'événements peuvent être des fonctions Azure Functions, des webhooks personnalisés, des applications logiques ou d'autres ressources. Les sources d'événements et les abonnés qui exécutent des gestionnaires d'événements sont indiqués dans la *Figure 10.25* :

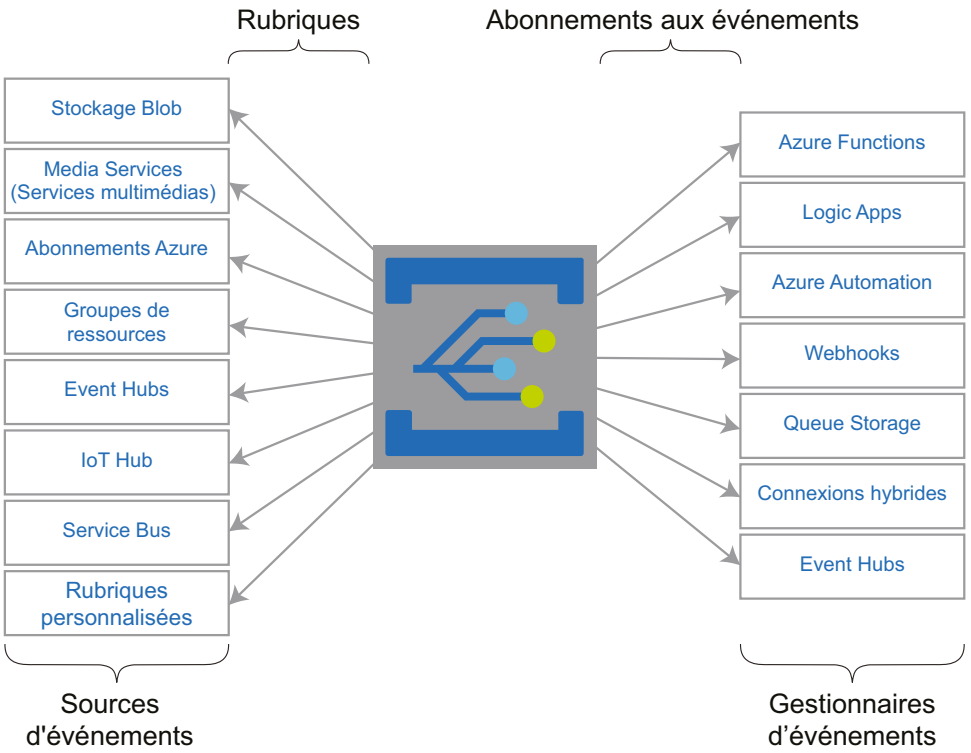


Figure 10.25 : l'architecture Event Grid

Lorsqu'un événement atteint une rubrique, plusieurs gestionnaires d'événements peuvent être exécutés simultanément, chacun entreprenant sa propre action.

Il est également possible de déclencher un événement personnalisé et d'envoyer une rubrique personnalisée à Event Grid. Event Grid fournit des fonctions pour la création de rubriques personnalisées, et ces rubriques sont automatiquement attachées à Event Grid. Ces rubriques connaissent le stockage Event Grid et y envoient automatiquement leurs messages. Les rubriques personnalisées ont deux propriétés importantes, comme suit :

- **Un point de terminaison** : il s'agit du point de terminaison de la rubrique. Les éditeurs et les sources d'événements utilisent ce point de terminaison pour envoyer et publier leurs événements dans Event Grid. En d'autres termes, les rubriques sont reconnues à l'aide de leurs points de terminaison.

- **Clés** : les rubriques personnalisées fournissent des clés. Ces clés sécurisent la consommation du point de terminaison. Seuls les éditeurs disposant de ces clés peuvent envoyer et publier leurs messages dans Event Grid.

Chaque événement a un type d'événement et il est reconnu par celui-ci. Par exemple, le stockage Blob fournit des types d'événements tels que **`blobAdded`** et **`blobDeleted`**. Les rubriques personnalisées peuvent être utilisées pour envoyer un événement défini sur mesure, tel qu'un événement personnalisé du type **`KeyVaultSecretExpired`**.

D'autre part, les abonnés ont la possibilité d'accepter tous les messages ou d'obtenir uniquement des événements basés sur des filtres. Ces filtres peuvent être basés sur le type d'événement ou d'autres propriétés dans la charge utile d'événement.

Chaque événement possède au moins les cinq propriétés suivantes :

- **`id`** : il s'agit de l'identificateur unique de l'événement.
- **`eventType`** : il s'agit du type d'événement.
- **`eventTime`** : il s'agit de la date et de l'heure auxquelles l'événement a été déclenché.
- **`subject`** : il s'agit d'une brève description de l'événement.
- **`data`** : il s'agit d'un objet dictionnaire qui contient des données spécifiques aux ressources ou des données personnalisées (pour les rubriques personnalisées).

Actuellement, les fonctions d'Event Grid ne sont pas disponibles avec toutes les ressources ; toutefois, Azure ajoute continuellement des ressources avec la fonction Event Grid.

Pour en savoir plus sur les ressources qui peuvent déclencher des événements liés à Event Grid et les gestionnaires qui peuvent gérer ces événements, accédez à https://docs.microsoft.com/azure/event-grid/overview.

Événements de ressources

Dans cette section, les étapes suivantes permettent de créer une solution dans laquelle les événements déclenchés par le stockage Blob sont publiés dans Event Grid et finalement acheminés vers une fonction Azure :

1. Connectez-vous au portail Azure à l'aide de vos identifiants et créez un compte de stockage dans un groupe de ressources nouveau ou existant. Le compte de stockage doit être soit **StorageV2**, soit **Stockage Blob**. Comme illustré dans la *Figure 10.26*, Event Grid ne fonctionnera pas avec **StorageV1** :

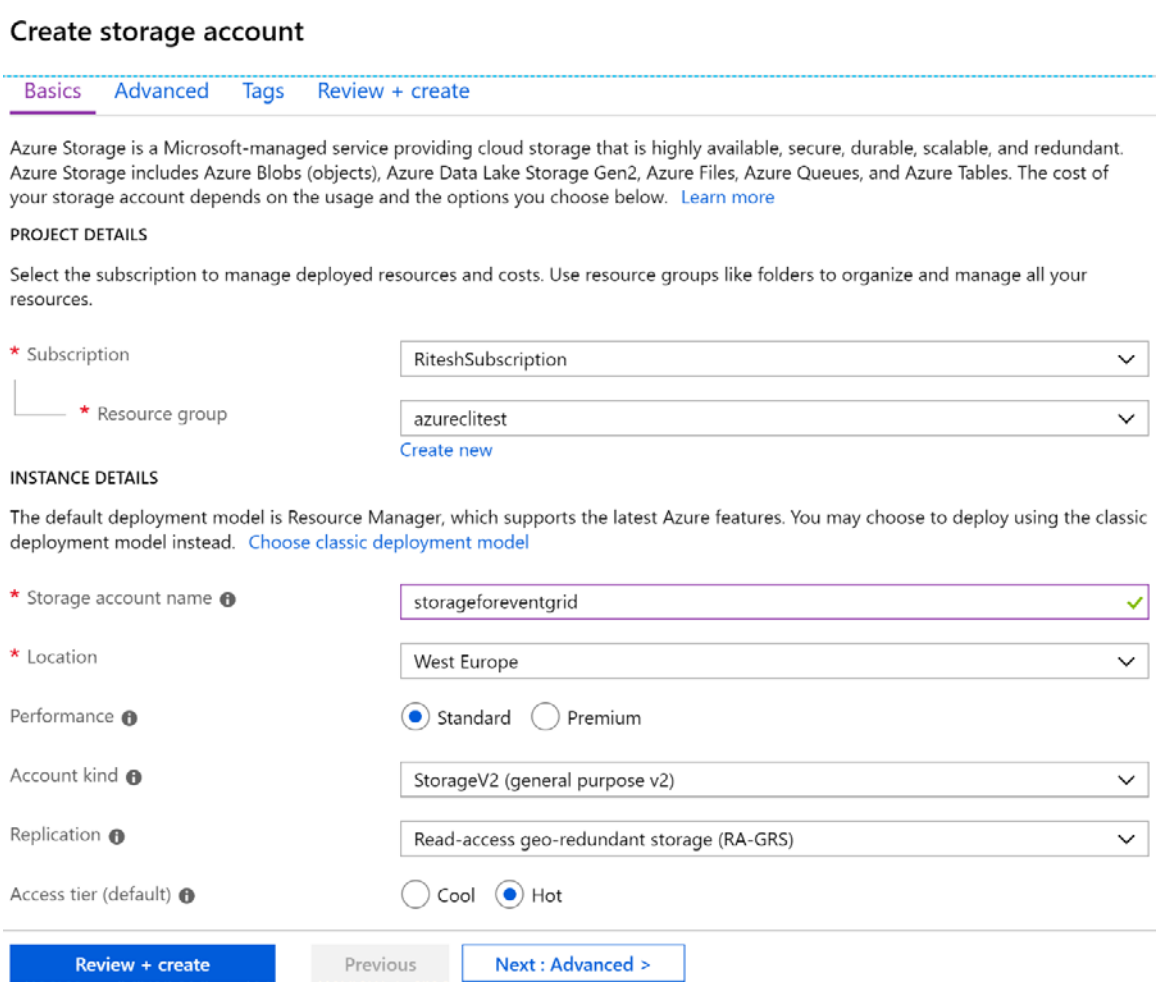


Figure 10.26 : création d'un compte de stockage

2. Créez une application de fonction ou réutilisez une application de fonction existante pour créer une fonction Azure. La fonction Azure sera hébergée dans l'application Functions.

3. Créez une fonction à l'aide du modèle de **déclencheur Azure Event Grid**. Installez l'extension **Microsoft.Azure.WebJobs.Extensions.EventGrid** si elle n'est pas déjà installée, comme illustré à la *Figure 10.27* :

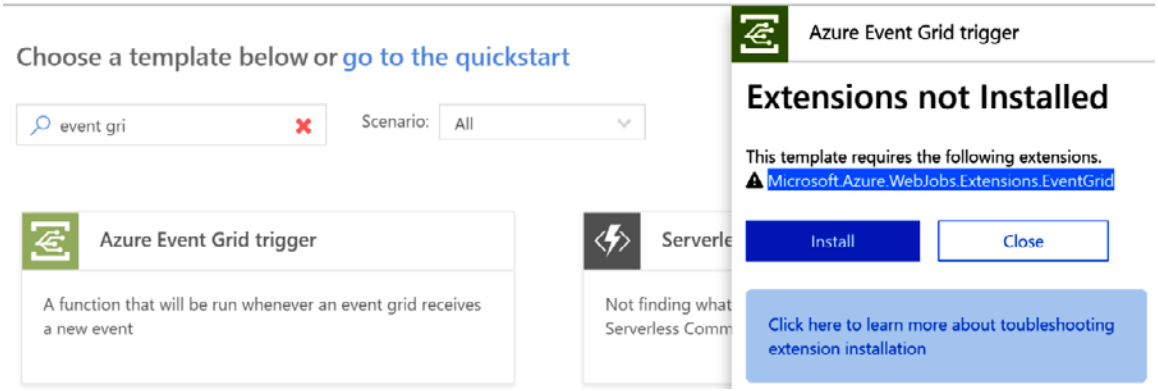


Figure 10.27 : installation des extensions pour un déclencheur Azure Event Grid

4. Nommez la fonction **`StorageEventHandler`** et créez-la. Le code généré par défaut suivant sera utilisé comme gestionnaire d'événements :

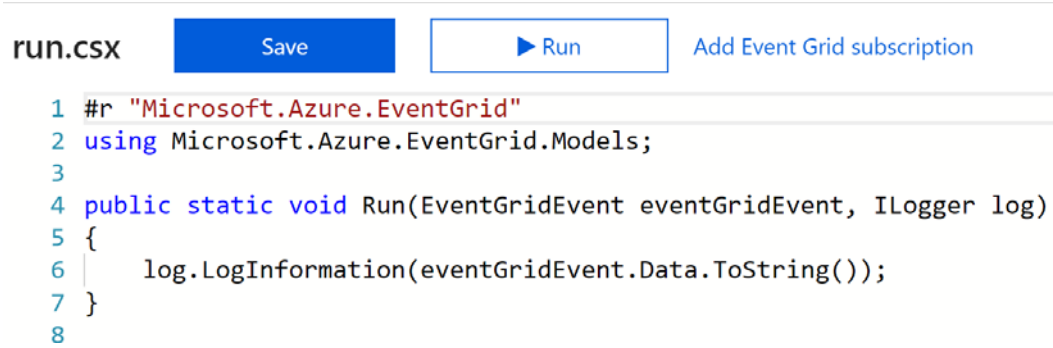


Figure 10.28 : code appliqué au gestionnaire d'événements

L'abonnement aux événements de stockage peut être configuré à partir de **l'interface utilisateur** (IU) Azure Functions en cliquant sur **Add Event Grid subscription** (Ajouter un abonnement Event Grid), ou à partir du compte de stockage lui-même.

5. Cliquez sur le lien **Add Event Grid subscription** (Ajouter un abonnement Event Grid) dans l'interface utilisateur Azure Functions pour ajouter un abonnement aux événements déclenchés par le compte de stockage créé à l'étape précédente. Fournissez un nom significatif pour l'abonnement, puis choisissez **Event Schema** (Schéma d'événement), suivi de **Event Grid Schema** (Schéma Event Grid). Définissez **Topic Types** (Types de rubrique) en tant que **Storage Accounts** (Comptes de stockage), définissez un **Subscription** (Abonnement) approprié et le groupe de ressources contenant le compte de stockage :

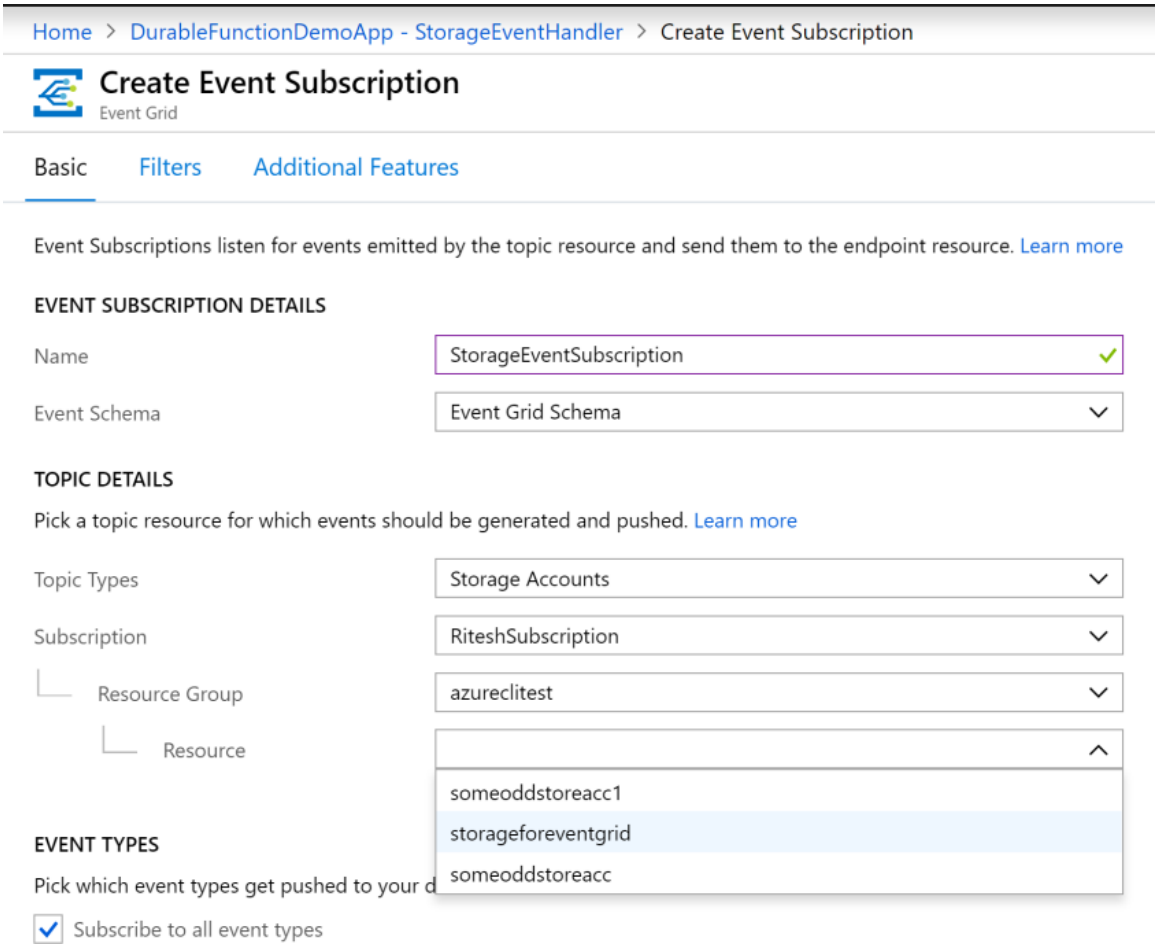


Figure 10.29 : création d'un abonnement Event Grid

Vérifiez que la case **Subscribe to all event types** (S'abonner à tous les types d'événements) est cochée et cliquez sur le bouton **Create** (Créer), celui-ci doit être activé dès qu'un compte de stockage est sélectionné.

6. Si nous accédons maintenant au compte de stockage dans le portail Azure et que nous cliquons sur le lien **Events** (Événements) dans le menu de gauche, l'abonnement au compte de stockage doit être visible :

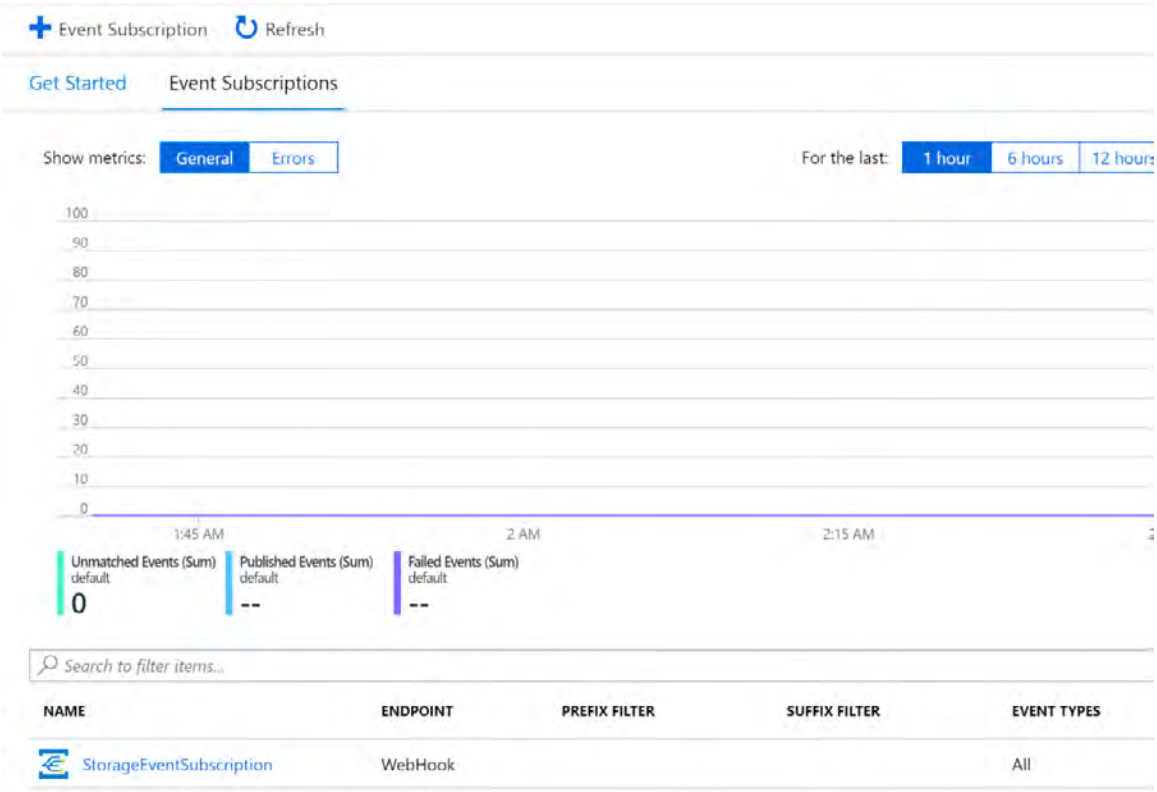


Figure 10.30 : liste des abonnements aux événements

7. Chargez un fichier dans le stockage Blob après avoir créé un conteneur et la fonction Azure doit être exécutée. L'action de chargement déclenchera un nouvel événement de type **blobAdded** et l'enverra à la rubrique Event Grid pour les comptes de stockage. Comme illustré à la *Figure* 10.31, l'abonnement est déjà défini pour obtenir tous les événements de cette rubrique, et la fonction est exécutée dans le cadre du gestionnaire d'événements :

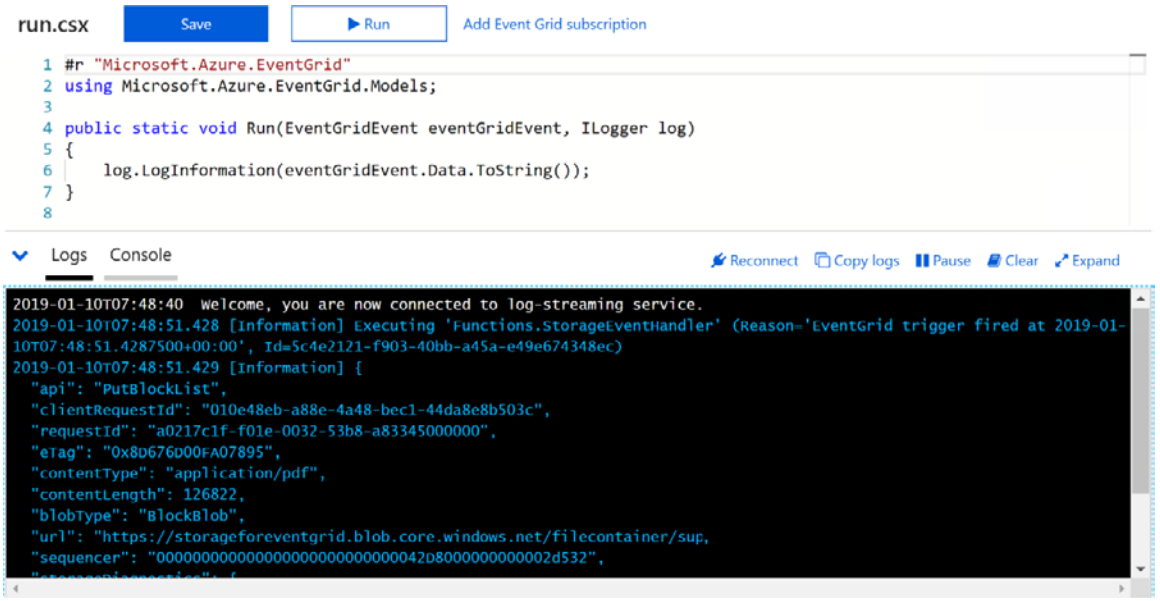


Figure 10.31 : déclenchement d'un nouvel événement

Dans cette section, vous avez appris comment les événements déclenchés par le stockage blob sont acheminés vers une fonction Azure. Dans la section suivante, vous apprendrez à tirer parti des événements personnalisés.

Événements personnalisés

Dans cet exemple, au lieu d'utiliser des ressources existantes pour générer des événements, des événements personnalisés seront utilisés. Nous utiliserons PowerShell pour créer cette solution et réutiliser la même fonction Azure qui a été créée dans le dernier exercice en tant que gestionnaire :

1. Connectez-vous à l'abonnement Azure de votre choix via **Login-AzAccount** et la **cmdlet de commande Set-AzContext**.

2. L'étape suivante consiste à créer une nouvelle rubrique Event Grid dans Azure dans un groupe de ressources. La cmdlet de commande **New-AzEventGridTopic** est utilisée pour créer une rubrique :

```
New-AzEventGridTopic -ResourceGroupName CustomEventGridDemo -Name
"KeyVaultAssetsExpiry" -Location "West Europe"
```

3. Une fois la rubrique créée, son URL de point de terminaison et sa clé doivent être récupérées car elles sont nécessaires pour envoyer et publier l'événement. Les cmdlets de commande **Get-AzEventGridTopic** et **Get-AzEventGridTopicKey** sont utilisées pour récupérer ces valeurs. Notez que **Key1** est récupéré pour connecter le point de terminaison :

```
$topicEndpoint = (Get-AzEventGridTopic -ResourceGroupName containers -Name
KeyVaultAssetsExpiry).Endpoint

$keys = (Get-AzEventGridTopicKey -ResourceGroupName containers -Name
KeyVaultAssetsExpiry).Key1
```

4. Une nouvelle table de hachage est créée avec les cinq propriétés d'événement Event Grid essentielles. Une nouvelle propriété **id** est générée pour l'identifiant, la propriété **subject** est définie sur **Asset Expiry** (Expiration de l'actif) **Key vault**, **eventType** est défini sur **Certificate Expiry** (Expiration du certificat), **eventTime** est défini sur l'heure actuelle et les **données** contiennent des informations sur le certificat :

```
$eventgridDataMessage = @{
id = [System.guid]::NewGuid()
subject = "Key Vault Asset Expiry"
eventType = "Certificate Expiry"
eventTime = [System.DateTime]::UtcNow
data = @{
CertificateThumbprint = "sdfervdserwetsgfhgdg"
ExpiryDate = "1/1/2019"
Createdon = "1/1/2018"
}
}
```

5. Étant donné que les données Event Grid doivent être publiées sous la forme d'un tableau JSON, la charge utile est convertie dans le tableau JSON. Les crochets **« [»,«] »** représentent un tableau JSON :

```
$finalBody = "[" + $(ConvertTo-Json $eventgridDataMessage) + "]"
```

6. L'événement sera publié à l'aide du protocole HTTP et les informations d'en-tête appropriées devront être ajoutées à la demande. La demande est envoyée à l'aide du type de contenu application/JSON et la clé appartenant à la rubrique est assignée à l'en-tête **aeg-sas-key**. Il est obligatoire de nommer l'en-tête et l'ensemble de clés sur **aeg-sas-key** :

```
$header = @{
"contentType" = "application/json"
"aeg-sas-key" = $keys}
```

7. Un nouvel abonnement est créé pour la rubrique personnalisée avec un nom, le groupe de ressources contenant la rubrique, le nom de la rubrique, le point de terminaison webhook et le point de terminaison réel qui agit en tant que gestionnaire d'événements. Le gestionnaire d'événements dans ce cas est la fonction Azure :

```
New-AzEventGridSubscription -TopicName KeyVaultAssetsExpiry
-EventSubscriptionName "customtopicsubscriptionautocar" -ResourceGroupName
CustomEventGridDemo -EndpointType webhook '
-Endpoint "https://durablefunctiondemoapp.
azurewebsites.net/runtime/webhooks/
EventGrid?functionName=StorageEventHandler&code=0aSw6sxvtFmafXHvt7iOw/
Dsb8o1M9RKKagzVchTUkwe9EIkzl4mCg=='
-Verbose
```

L'URL de la fonction Azure est disponible à partir de l'onglet **Integrate** (Intégrer), comme illustré à la *Figure 10.31* :

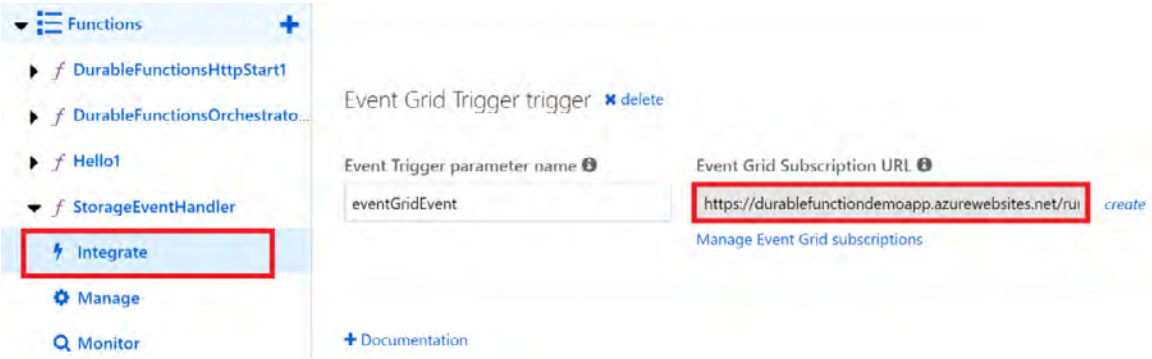


Figure 10.32 : URL de l'abonnement Event Grid dans l'onglet Intégrer

8. À présent, l'abonné (gestionnaire d'événements) et l'éditeur ont été tous deux configurés. L'étape suivante consiste à envoyer et à publier un événement dans la rubrique personnalisée. Les données d'événement ont déjà été créées à l'étape précédente et, grâce à la cmdlet de commande **`Invoke-WebRequest`**, la demande est envoyée au point de terminaison avec le corps et l'en-tête :

```
Invoke-WebRequest -Uri $topicEndpoint -Body $finalBody -Headers $header
-Method Post
```

L'appel d'API déclenchera l'événement et Event Grid signalera le point de terminaison que nous avons configuré, qui est l'application de fonction. Ce chapitre se conclu par cette activité.

Résumé

L'évolution des fonctions à partir des méthodes traditionnelles a conduit à la conception d'une architecture sans serveur autonome, associée librement et évoluant de manière indépendante, qui n'était qu'un simple concept auparavant. Les fonctions sont une unité de déploiement qui fournissent un environnement où les utilisateurs n'ont aucun besoin de gérer l'environnement. Il leur suffit d'écrire du code pour la fonction. Azure fournit une plateforme mature pour les fonctions d'hébergement et les intègre en toute transparence en fonction d'événements ou à la demande. Quasiment toutes les ressources d'Azure peuvent participer à une architecture composée de fonctions Azure. L'avenir sera fait de fonctions, car les organisations chercheront de plus en plus à se défaire de la gestion des infrastructures et des plateformes. Elles souhaiteront confier ces tâches aux fournisseurs Cloud. Azure Functions est une caractéristique essentielle que chaque architecte travaillant avec Azure doit apprendre à maîtriser.

Dans ce chapitre, nous avons découvert en détail Azure Functions, les fonctions en tant que service, Durable Functions et Event Grid. Le chapitre suivant portera sur Azure Logic Apps et nous créerons une solution complète end-to-end combinant plusieurs services Serverless, ainsi que d'autres services Azure, tels qu'Azure Key Vault et Azure Automation.

11

Solutions Azure utilisant Azure Logic Apps, Event Grid et Functions

Ce chapitre est une continuité du chapitre précédent et présentera plus en détail les services Serverless disponibles dans Azure. Dans le chapitre précédent, vous avez découvert en détail Azure Functions, les fonctions en tant que service, durable Functions et Event Grid. Ce chapitre se concentrera sur Logic Apps, puis sur la création d'une solution Serverless complète end-to-end qui combine plusieurs services Serverless et d'autres types de services, tels que Key Vault et Azure Automation.

Dans ce chapitre, nous découvrirons de plus près les services Azure en abordant les sujets suivants :

- Azure Logic Apps
- Créer une solution end-to-end à l'aide des technologies Serverless

Azure Logic Apps

Logic Apps est l'offre de charge de travail Serverless d'Azure. Elle dispose de toutes les fonctions des technologies Serverless, telles que le CCR basé sur la consommation et une évolutivité illimitée. Logic Apps permet de créer facilement un processus métier et une solution de charge de travail à l'aide du portail Azure. Cette solution fournit une interface utilisateur glisser-déposer pour créer et configurer les charges de travail.

Logic Apps est le moyen privilégié d'intégrer des services et des données et de créer des projets d'entreprise et un flux de logique complet. Il existe plusieurs concepts importants qui doivent être compris avant de créer une application logique.

Activités

Une activité est une seule unité de travail. Les exemples d'activités incluent la conversion de XML en JSON, la lecture d'objets Blob à partir du stockage Azure et l'écriture dans une collection de documents Cosmos DB. Logic Apps fournit une définition de charge de travail consistant en plusieurs activités corrélées dans une séquence. Il existe deux types d'activité dans Logic Apps :

- **Déclencheur :** un déclencheur se réfère à l'initiation d'une activité. Toutes les applications logiques ont un déclencheur unique qui forme la première activité. C'est le déclencheur qui crée une instance de l'application logique et démarre l'exécution. Les exemples de déclencheurs sont l'arrivée des messages Event Grid, un e-mail, une demande HTTP ou une planification.
- **Actions :** toute activité qui n'est pas un déclencheur est une activité d'étape et chacune est responsable d'effectuer une tâche. Les étapes sont connectées les unes aux autres dans une charge de travail. Une action sera effectuée au cours de chaque étape et devra être terminée avant de passer à l'étape suivante.

Connecteurs

Les connecteurs sont des ressources Azure qui permettent de connecter une application logique à des services externes. Ces services peuvent être dans le Cloud ou sur site. Par exemple, il existe un connecteur permettant de connecter des applications logiques à Event Grid. De même, il existe un autre connecteur pour se connecter à Office 365 Exchange. Presque tous les types de connecteurs sont disponibles dans Logic Apps et ils peuvent être utilisés pour se connecter aux services. Les connecteurs contiennent des informations de connexion et une logique pour se connecter à des services externes à l'aide de ces informations de connexion.

La liste complète des connecteurs est disponible à l'adresse https://docs.microsoft.com/connectors.

Maintenant que vous connaissez les connecteurs, vous devez comprendre comment les aligner au cours d'une procédure pas à pas pour garantir le bon fonctionnement de la charge de travail. La section suivante porte sur le fonctionnement d'une application logique.

Fonctionnement d'une application logique

Nous allons créer une charge de travail Logic Apps qui se déclenche lorsque l'un des comptes de messagerie reçoit un e-mail. Elle répond à l'expéditeur avec un e-mail par défaut et effectue une analyse des sentiments sur le contenu de l'e-mail. Pour l'analyse des sentiments, la ressource Analyse de texte des Cognitive Services doit être provisionnée avant de créer l'application logique :

1. Accédez au portail Azure, connectez-vous à votre compte et créez une ressource **Analyse de texte** dans un groupe de ressources. L'analyse de texte fait partie des Cognitive Services et comporte des fonctionnalités telles que l'analyse des sentiments, l'extraction des phrases clés et la détection de langue. Ce service est disponible dans le portail Azure , comme illustré à la *Figure 11.1* :

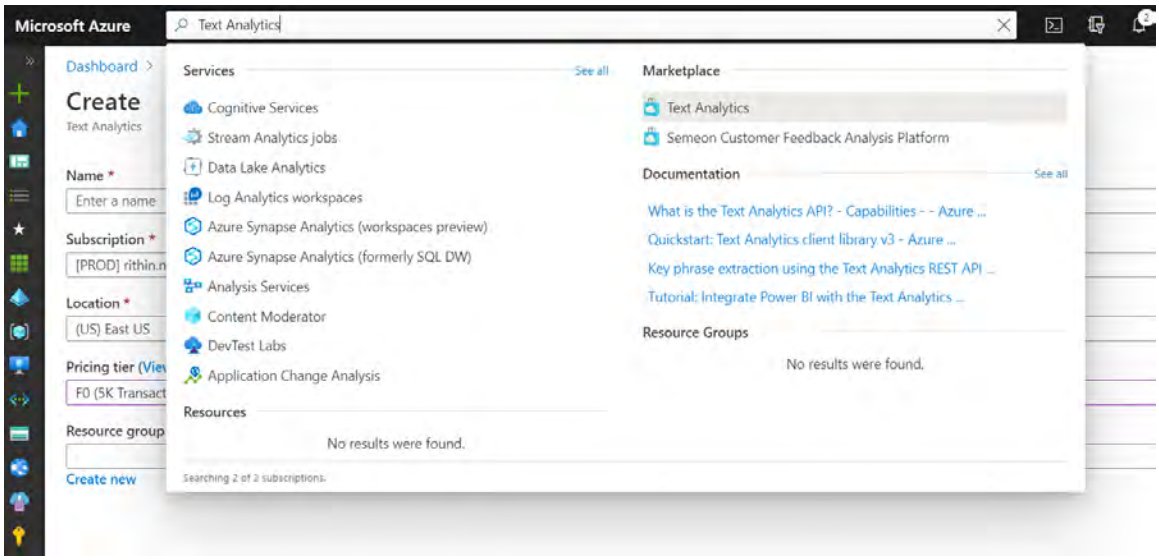


Figure 11.1 : navigation vers le service d'analyse de texte dans le portail Azure

2. Renseignez les champs **Name** (Nom), **Location** (Emplacement), **Subscription** (Abonnement), **Resource group** (Groupe de ressources) et **Pricing tier** (Niveau tarifaire). Nous allons utiliser le niveau gratuit (niveau F0) de ce service pour cette démonstration.

3. Une fois la ressource provisionnée, accédez à la page **Overview** (Présentation) et copiez l'URL du point de terminaison. Stockez-la dans un emplacement temporaire. Cette valeur sera requise lors de la configuration de l'application logique.
4. Naviguez jusqu'à la page **Keys** (Clés), copiez la valeur depuis **Key 1** et stockez-la dans un emplacement temporaire. Cette valeur sera nécessaire lors de la configuration de l'application logique.
5. La prochaine étape consiste à créer une application logique. Pour créer une application logique, accédez au groupe de ressources dans le portail Azure dans lequel l'application logique doit être créée. Recherchez Logic App et créez-la en renseignant les champs **Name** (Nom), **Location** (Emplacement), **Resource group** (Groupe de ressources) et **Subscription** (Abonnement).
6. Une fois l'application logique créée, naviguez jusqu'à la ressource, cliquez sur le **concepteur Logic App** dans le menu de gauche, puis sélectionnez le modèle **When a new email is received in Outlook.com** (Quand un nouvel e-mail est reçu dans Outlook.com) pour créer une charge de travail. Le modèle fournit un bon point de départ pour ajouter des déclencheurs et des activités réutilisables. Cela permettra d'ajouter un déclencheur Outlook Office 365 automatiquement à la charge de travail.
7. Cliquez sur le bouton **Sign in** (Connexion) sur le déclencheur ; une nouvelle fenêtre d'Internet Explorer s'ouvrira. Connectez-vous ensuite à votre compte. Une fois la connexion réussie, un nouveau connecteur de messagerie Office 365 sera créé, contenant les informations de connexion au compte.
8. Cliquez sur le bouton **Continue** (Continuer) et configurez le déclencheur avec une fréquence de sondage de 3 minutes, comme illustré à la *Figure 11.2* :

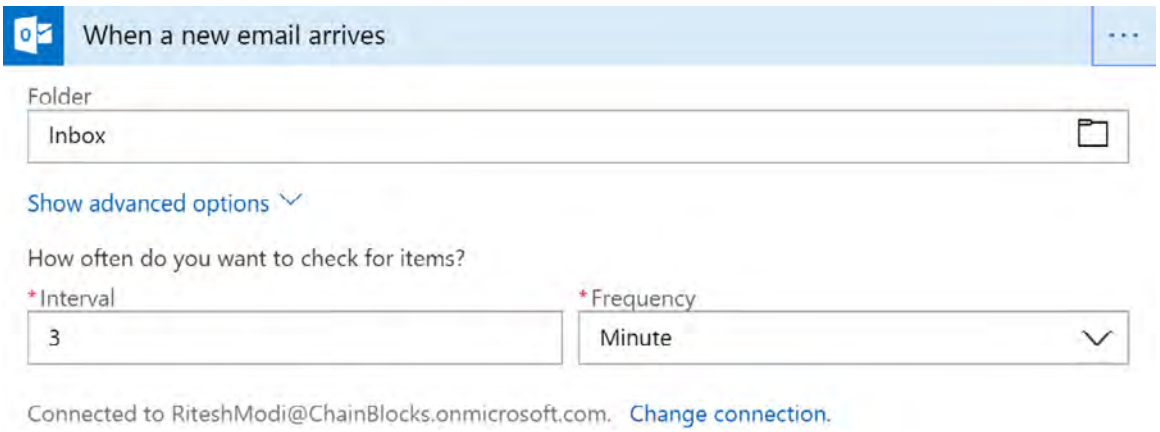


Figure 11,2 : configuration du déclencheur avec une fréquence de sondage de 3 minutes

9. Cliquez sur **Next step** (Étape suivante) pour ajouter une autre action et saisissez le mot-clé **`variable`** dans la barre de recherche. Ensuite, sélectionnez l'action **Initialize variable** (Initialiser la variable), comme illustré à la *Figure 11.3* :

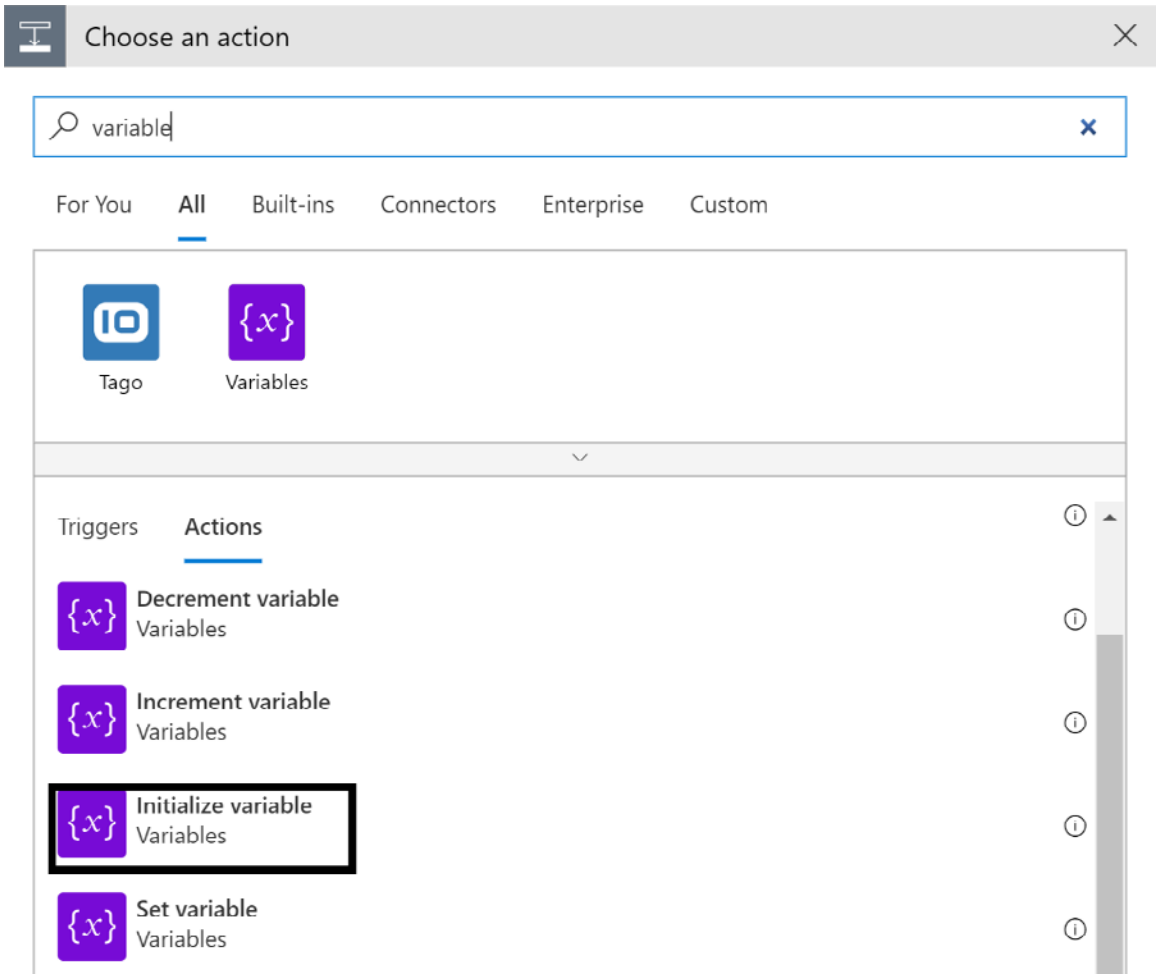


Figure 11.3 : ajout de l'action Initialiser la variable

10. Ensuite, configurez l'action de variable. Lorsque vous cliquez sur l'encadré **Value** (Valeur), une fenêtre contextuelle apparaît et affiche le **contenu Dynamique** et **Expression**. Le contenu dynamique fait référence aux propriétés disponibles pour l'action en cours, qui sont renseignées avec les valeurs d'exécution des actions et déclencheurs précédents. Les variables aident à conserver les charges de travail génériques. Dans cette fenêtre, sélectionnez **Body** (Corps) à partir de **Dynamic content** (Contenu dynamique) :

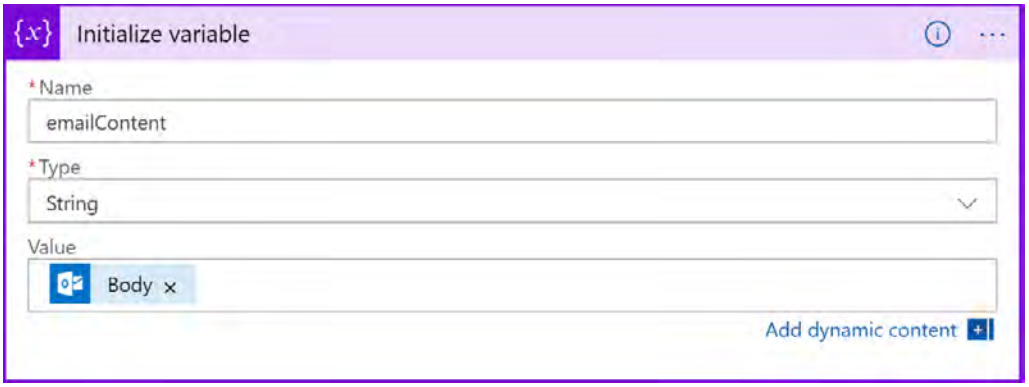


Figure 11.4 : configuration de l'action de variable

11. Ajoutez une autre action en cliquant sur **Add step** (Ajouter une étape), puis en saisissant `outlook` dans la barre de recherche et en sélectionnant l'action **Reply to email** (Répondre à l'e-mail) :

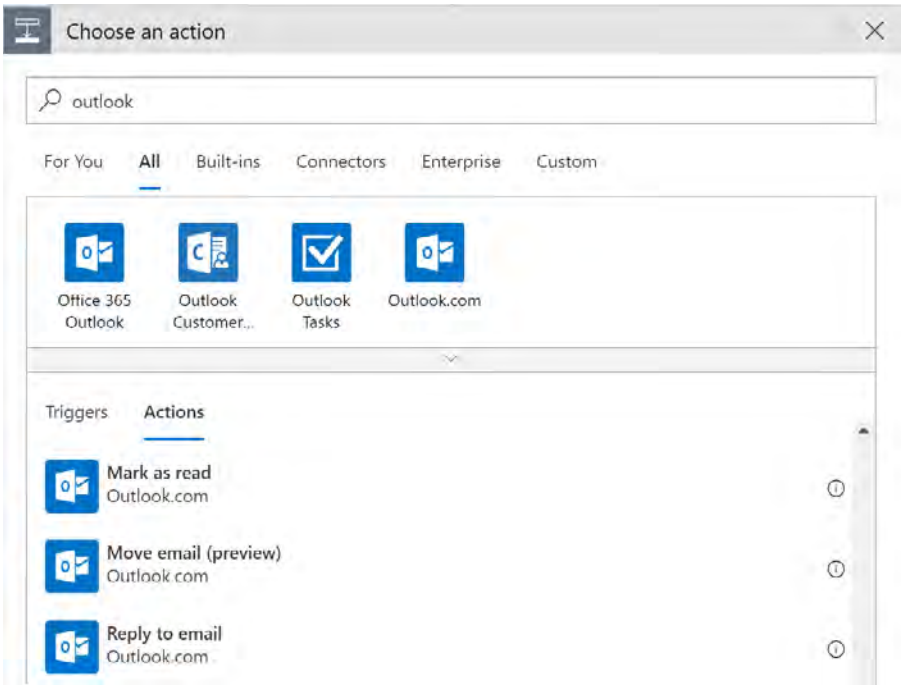


Figure 11.5 : ajout de l'action Répondre à l'e-mail

12. Configurez la nouvelle action. Assurez-vous que **Message Id** (ID de message) est défini avec le contenu dynamique, **Message Id** (ID de message), puis saisissez la réponse que vous souhaitez envoyer au destinataire dans la zone **Comment** (Commentaire) :

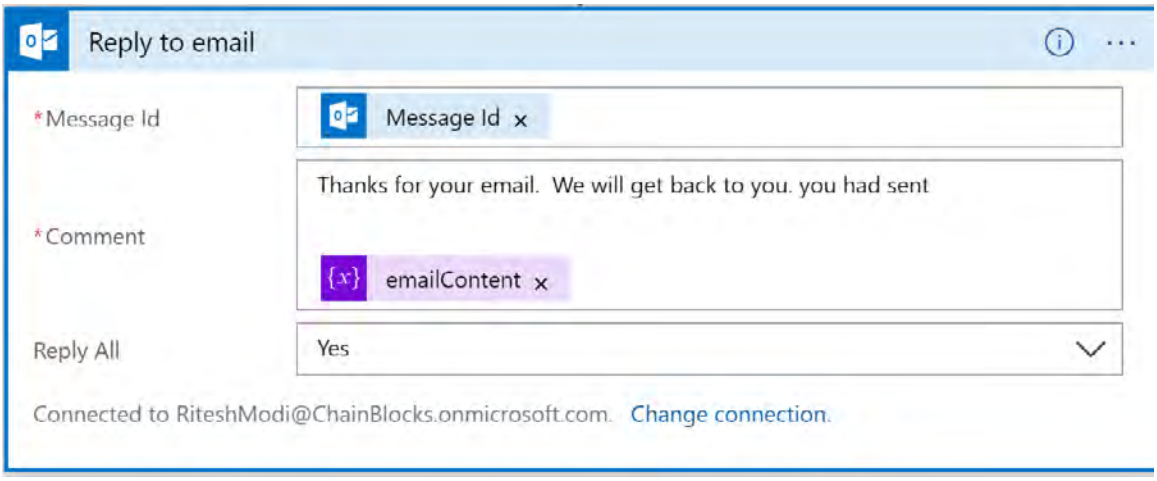


Figure 11.6 : configuration de l'action Répondre à l'e-mail

13. Ajoutez une autre action, saisissez `text analytics` (analyse de texte) dans la barre de recherche, puis sélectionnez **Detect Sentiment (preview)** (Détecter le sentiment [aperçu]) :

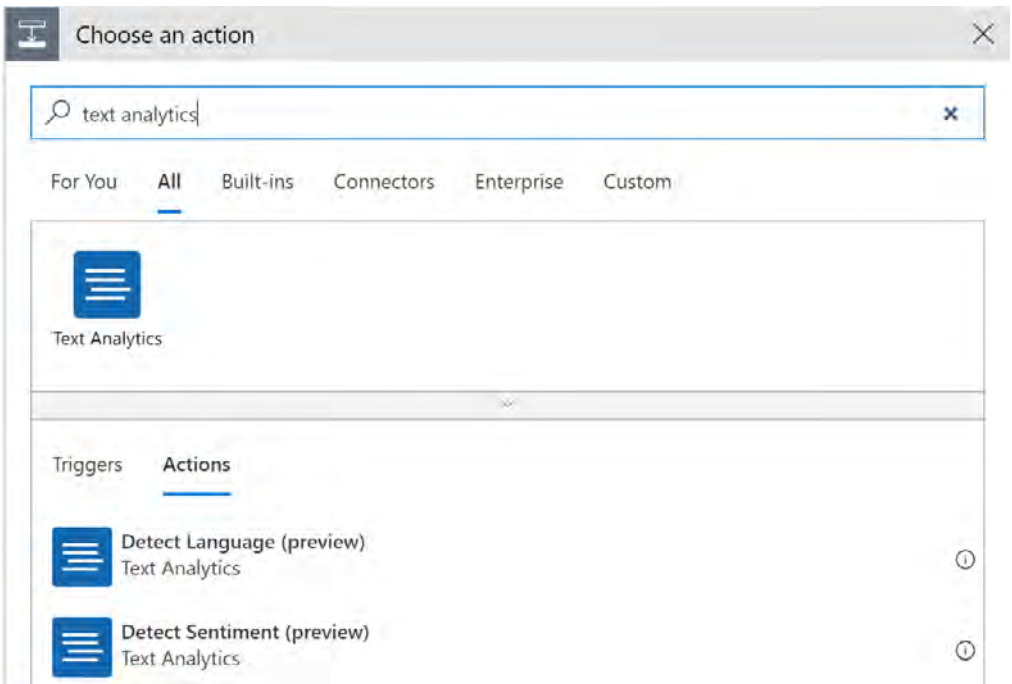


Figure 11.7 : ajout de l'action Détecter le sentiment (aperçu).

14. Configurez l'action de sentiment, comme illustré à la *Figure 11.8* : les valeurs de point de terminaison et de clé doivent être utilisées ici. Cliquez désormais sur le bouton **Create** (Créer), comme illustré à la *Figure 11.8* :

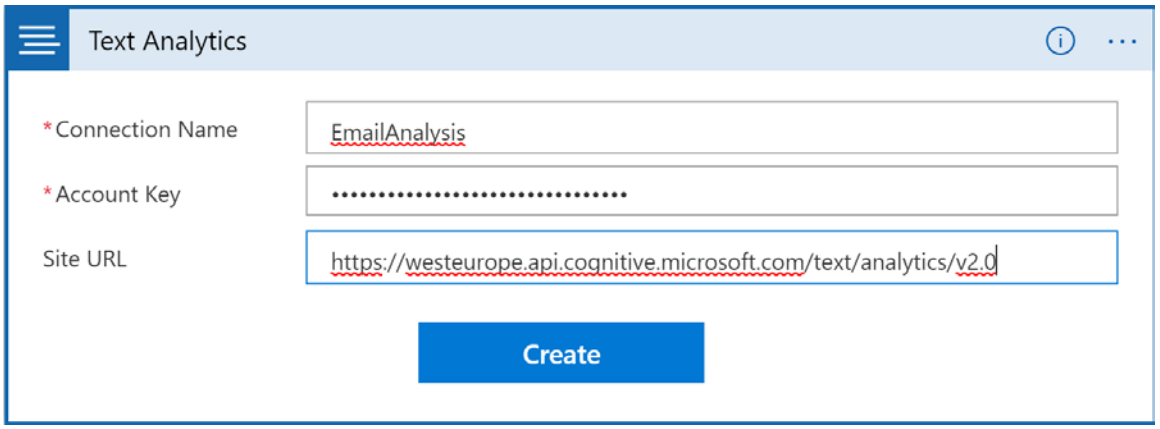


Figure 11.8 : configuration de l'action Détecter le sentiment (aperçu).

15. Fournissez le texte à l'action en ajoutant du contenu dynamique et en sélectionnant la variable précédemment créée, **emailContent**. Ensuite, cliquez sur **Show advanced options** (Afficher les options avancées) et sélectionnez **fr pour** la langue :

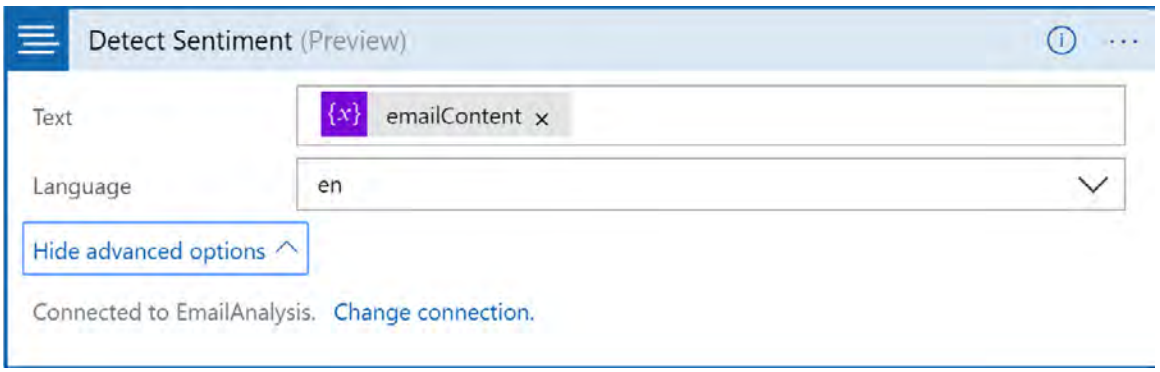


Figure 11.9 : sélection de la langue pour l'action de sentiment

16. Ensuite, ajoutez une nouvelle action en sélectionnant **Outlook**, puis sélectionnez **Send an email** (Envoyer un e-mail). Cette action envoie au destinataire original le contenu de l'e-mail avec le score de sentiment indiqué dans l'objet. Elle doit être configurée comme illustré à la *Figure 11.10*. Si le score n'est pas visible dans la fenêtre de contenu dynamique, cliquez sur **See more** (Voir plus) en regard de celle-ci :

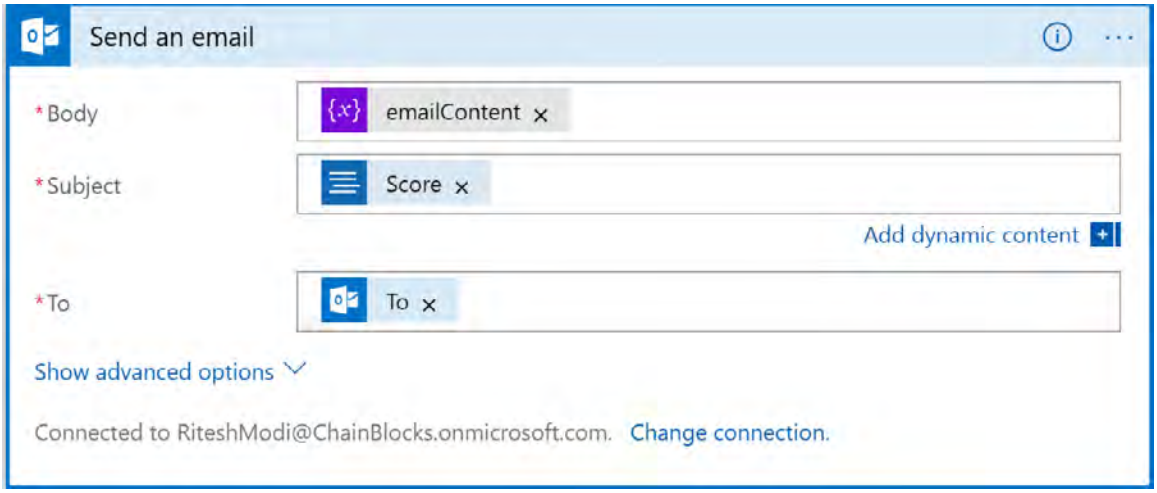


Figure 11.10 : ajout de l'action Envoyer un e-mail

17. Enregistrez l'application logique, revenez à la page de présentation, puis cliquez sur **Exécuter le déclencheur**. Le déclencheur vérifiera les nouveaux e-mails toutes les trois minutes, répondra aux expéditeurs, effectuera l'analyse des sentiments et enverra un e-mail au destinataire d'origine. Un exemple d'e-mail avec des connotations négatives est envoyé à l'ID de courrier électronique donné :

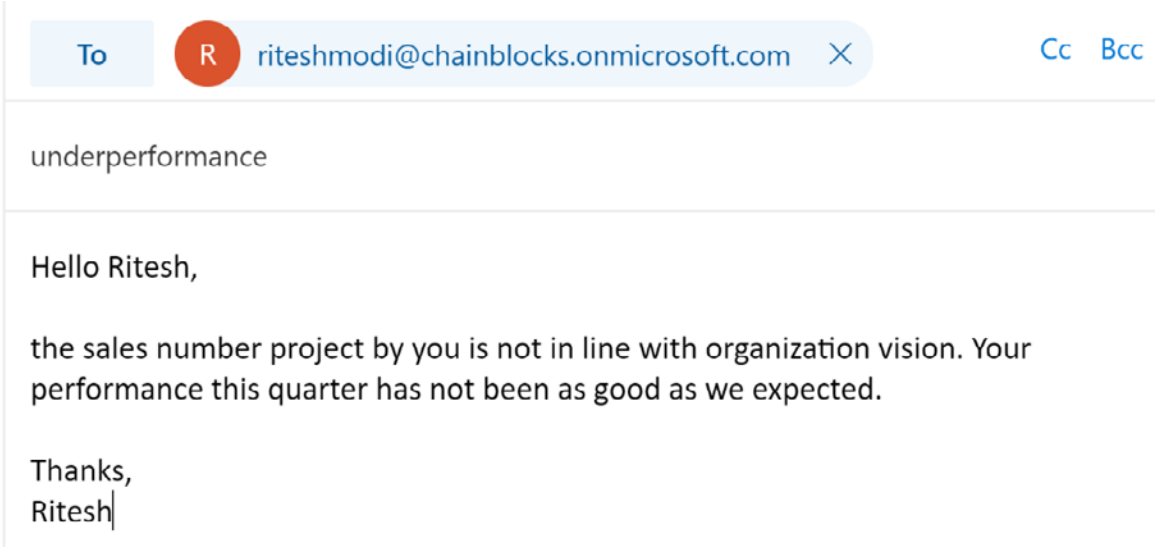


Figure 11.11 : exemple d'e-mail

18. Après quelques secondes, l'application logique s'exécute et l'expéditeur obtient la réponse suivante :

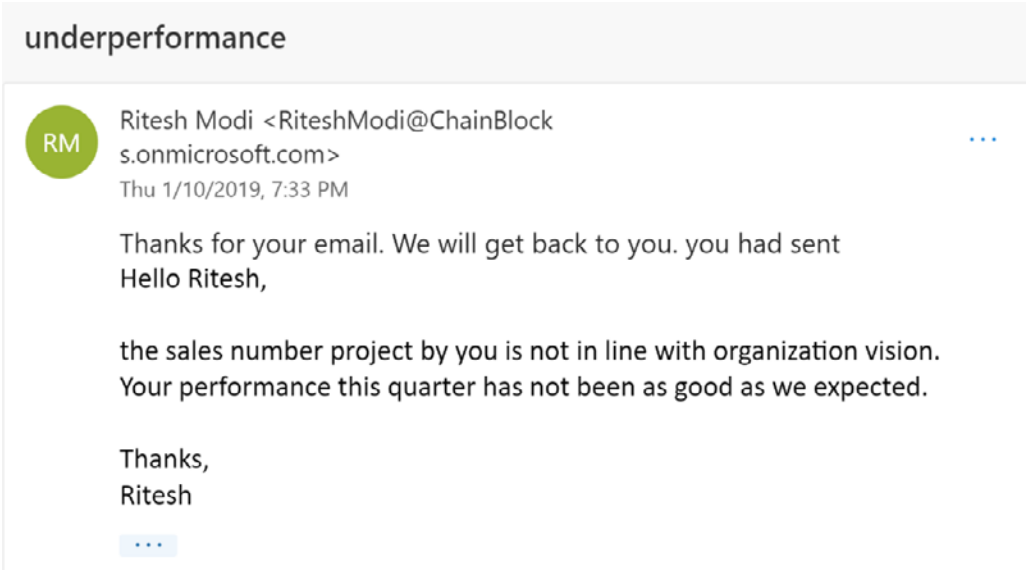


Figure 11.12 : e-mail de réponse à l'expéditeur d'origine

19. Le destinataire d'origine obtient un e-mail avec le score de sentiment et le texte de l'e-mail d'origine, comme illustré à la *Figure 11.13* :

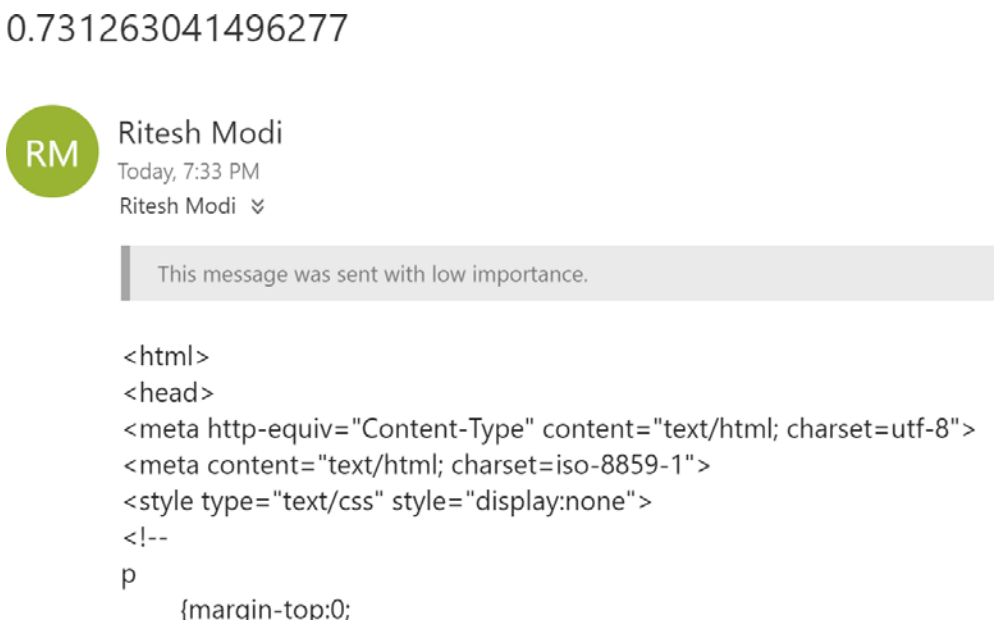


Figure 11.13 : vue HTML du message électronique

L'activité nous a permis de comprendre le fonctionnement d'une application logique. L'application a été déclenchée lorsqu'un e-mail a été reçu dans la boîte de réception de l'utilisateur et le processus a suivi la séquence des étapes indiquées dans l'application logique. Dans la section suivante, vous allez apprendre à créer une solution end-to-end à l'aide de technologies Serverless.

Créer une solution end-to-end à l'aide des technologies Serverless

Dans cette section, nous allons créer une solution end-to-end comprenant les technologies Serverless dont nous avons discuté dans les sections précédentes. L'exemple suivant illustre une mise en œuvre intelligente des charges de travail afin d'éviter les frais de gestion. Dans l'activité suivante, nous allons créer une charge de travail pour avertir les utilisateurs lorsque les clés, les secrets et les certificats seront stockés dans Azure Key Vault. Nous allons étudier cette problématique, puis trouver, concevoir et mettre en œuvre une solution.

Problématique

Nous allons maintenant résoudre un problème rencontré par des utilisateurs et des organisations, qui ne sont pas informés de l'expiration des secrets dans leur coffre de clés ; or les applications cessent de fonctionner quand ceux-ci expirent. Les utilisateurs se plaignent qu'Azure ne fournit pas l'infrastructure pour surveiller les secrets, les clés et les certificats de coffre de clés.

Solution

La solution à ce problème consiste à combiner plusieurs services Azure et à les intégrer afin que les utilisateurs puissent être informés de manière proactive de l'expiration des secrets. La solution enverra des notifications à l'aide de deux canaux : e-mail et SMS.

Les services Azure utilisés pour créer cette solution sont les suivants :

- Azure Key Vault
- **Azure Active Directory** (**Azure AD**)
- Azure Event Grid
- Azure Automation
- Logic Apps
- Azure Functions
- SendGrid
- Twilio SMS

Maintenant que vous connaissez les services qui seront utilisés dans le cadre de la solution, créons une architecture pour cette solution.

Architecture

Dans la section précédente, nous avons exploré la liste des services qui seront utilisés dans la solution. Pour mettre en œuvre la solution, les services doivent être organisés dans le bon ordre. L'architecture nous aidera à développer la charge de travail, en avançant dans la conception de la solution.

L'architecture de la solution comprend plusieurs services, comme illustré à la *Figure 11.14* :

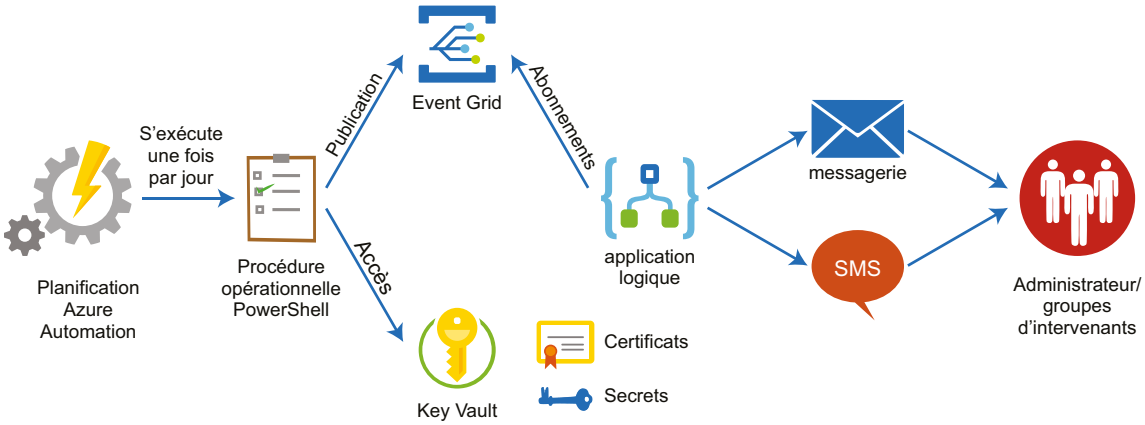


Figure 11.14 : architecture de solution

Examinons chacun de ces services pour comprendre leurs rôles et les fonctions qu'ils fournissent dans la solution globale.

Azure Automation

Azure Automation fournit des procédures opérationnelles, lesquelles peuvent exécuter la logique à l'aide de PowerShell, Python et d'autres langages de script. Les scripts peuvent être exécutés localement ou dans le Cloud, et fournissent une infrastructure et des installations riches pour créer des scripts. Ces types de scripts sont appelés des **procédures opérationnelles**. En règle générale, les procédures opérationnelles implémentent un scénario tel que l'arrêt ou le démarrage d'une machine virtuelle, ou la création et la configuration de comptes de stockage. Il est assez facile de se connecter à l'environnement Azure à partir de procédures opérationnelles à l'aide d'actifs tels que des variables, des certificats et des connexions.

Dans la solution actuelle, nous souhaitons nous connecter à Azure Key Vault, lire tous les secrets et clés stockés dans ce dernier et récupérer leurs dates d'expiration. Ces dates d'expiration doivent être comparées à la date d'aujourd'hui et, si la date d'expiration est dans un mois, la procédure opérationnelle doit déclencher un événement personnalisé sur Event Grid à l'aide d'une rubrique personnalisée Event Grid.

Une procédure opérationnelle Azure Automation à l'aide d'un script PowerShell sera implémentée pour y parvenir. En plus de la procédure opérationnelle, un planificateur sera également créé, lequel exécutera la procédure opérationnelle une fois par jour à 00 : 00 (minuit).

Une rubrique Azure Event Grid personnalisée

Une fois que la procédure opérationnelle identifie qu'un secret ou une clé va expirer dans un mois, elle lèvera un nouvel événement personnalisé et le publiera dans la rubrique personnalisée créée spécifiquement à cet effet. Encore une fois, nous allons entrer dans les détails de la mise en œuvre dans la section suivante.

Azure Logic Apps

Une application logique est un service Serverless qui fournit des fonctions de charge de travail. Notre application logique sera configurée pour être déclenchée au fur et à mesure qu'un événement est publié sur la rubrique Event Grid personnalisée. Une fois celle-ci déclenchée, elle appellera la charge de travail et exécutera toutes les activités qu'elle contient, une à une. En général, il existe plusieurs activités, mais dans cet exemple, nous appellerons une fonction Azure qui enverra à la fois des e-mails et des messages SMS. Dans une implémentation complète, ces fonctions de notification doivent être implémentées séparément dans des fonctions Azure Functions distinctes.

Azure Functions

Azure Functions est utilisé pour informer les utilisateurs et les parties intéressées de l'expiration des secrets et des clés à l'aide d'e-mails et de SMS. SendGrid est utilisé pour envoyer des e-mails, tandis que Twilio est utilisé pour envoyer des messages SMS à partir d'Azure Functions.

Dans la section suivante, nous allons examiner les conditions préalables à la mise en œuvre de la solution.

Prérequis

Vous aurez besoin d'un abonnement Azure avec des droits de contributeur au minimum. Étant donné que nous déployons uniquement des services vers Azure et qu'aucun service externe n'est déployé, l'abonnement est la seule condition préalable. Poursuivons et mettons en œuvre la solution.

Mise en œuvre

Un coffre de clés devrait déjà exister. Dans le cas contraire, il doit être créé.

Cette étape doit être effectuée si une nouvelle instance Azure Key Vault doit être mise en service. Plusieurs méthodes permettent de mettre des ressources en service dans Azure. Parmi celles-ci figurent notamment Azure PowerShell et Azure CLI. Azure CLI est une interface de ligne de commande qui fonctionne sur plusieurs plateformes. La première tâche sera de mettre en service un coffre de clés dans Azure. Nous utiliserons Azure PowerShell pour provisionner le coffre de clés dans cette mise en œuvre.

Avant qu'Azure PowerShell ne puisse être utilisé pour créer un coffre de clés, il est important de se connecter à Azure afin que les commandes suivantes puissent être exécutées pour créer un coffre de clés.

Étape 1 : mise en service d'une instance Azure Key Vault

La première étape consiste à préparer l'environnement pour l'échantillon. Cela implique de vous connecter au portail Azure, de sélectionner un abonnement approprié, puis de créer un groupe de ressources Azure et une ressource Azure Key Vault :

1. Exécutez la commande **`Connect-AzAccount`** pour vous connecter à Azure. Vous serez invité à saisir les informations d'identification dans une nouvelle fenêtre.

2. Après une connexion réussie, si plusieurs abonnements sont disponibles pour l'ID de connexion fourni, ils seront tous répertoriés. Il est important de sélectionner un abonnement approprié ; cela peut être effectué en exécutant la cmdlet de commande **`Set-AzContext`** :

   ```
   Set-AzContext -SubscriptionId xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx
   ```

3. Créez un nouveau groupe de ressources dans votre emplacement préféré. Dans ce cas, le nom du groupe de ressources est **`IntegrationDemo`** et il est créé dans la région **`Europe de l'Ouest`** :

   ```
   New-AzResourceGroup -Name IntegrationDemo -Location "West Europe" -Verbose
   ```

4. Créez une ressource Azure Key Vault ; le nom du coffre-fort dans ce cas est **`keyvaultbook`** et il est activé pour le déploiement, le déploiement de modèle, le chiffrement de disque, la suppression logicielle et la protection de purge :

   ```
   New-AzKeyVault -Name keyvaultbook -ResourceGroupName
   IntegrationDemo -Location "West Europe" -EnabledForDeployment
   -EnabledForTemplateDeployment -EnabledForDiskEncryption
   -EnablePurgeProtection -Sku Standard - Verbose
   ```

Veuillez noter que le nom du coffre de clés doit être unique. Il se peut que vous ne puissiez pas utiliser le même nom pour deux coffres de clés. La commande précédente, lorsqu'elle est exécutée avec succès, créera une ressource Azure Key Vault. L'étape suivante consiste à fournir l'accès à un principal de service sur le coffre de clés.

Étape 2 : création d'un principal de service

Au lieu d'utiliser un compte individuel pour se connecter à Azure, Azure fournit des principaux de service, qui sont, en substance, des comptes de service pouvant être utilisés pour se connecter à Azure Resource Manager et exécuter des activités. L'ajout d'un utilisateur au répertoire/locataire Azure les rend disponibles partout, y compris dans tous les groupes de ressources et les ressources, en raison de la nature de l'héritage de sécurité dans Azure. L'accès aux groupes de ressources doit être explicitement révoqué pour les utilisateurs qui ne sont pas autorisés à y accéder. Les principaux de service aident à affecter un accès granulaire et le contrôle aux groupes de ressources, aux ressources et, si nécessaire, l'accès à la portée de l'abonnement. Des autorisations granulaires peuvent également être assignées, telles que les autorisations de lecteur, de contributeur ou de propriétaire.

En bref, les principaux de service doivent être le mécanisme privilégié pour consommer des services Azure. Ils peuvent être configurés soit avec un mot de passe, soit avec une clé de certificat. Les principaux de service peuvent être créés à l'aide de la commande **New-AzAdServicePrinicipal**, comme illustré ici :

```
$sp = New-AzADServicePrincipal -DisplayName "keyvault-book" -Scope "/
subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" -Role Owner -StartDate
([datetime]::Now) -EndDate $([datetime]::now.AddYears(1)) -Verbose
```

Les valeurs de configuration importantes sont la portée et le rôle. La portée détermine la zone d'accès pour l'application de service ; elle est actuellement indiquée au niveau de l'abonnement. Les valeurs valides pour la portée sont les suivantes :

```
/subscriptions/{subscriptionId}
/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}
/subscriptions/{subscriptionId}/resourcegroups/{resourceGroupName}/providers/
{resourceProviderNamespace}/{resourceType}/{resourceName}
/subscriptions/{subscriptionId}/resourcegroups/{resourceGroupName}/
providers/{resourceProviderNamespace}/{parentResourcePath}/{resourceType}/
{resourceName}
```

Le rôle fournit des autorisations à la portée affectée. Les valeurs valides sont les suivantes :

- Propriétaire
- Contributeur
- Lecteur
- Autorisations spécifiques aux ressources

Dans la commande précédente, les autorisations de propriétaire ont été fournies au principal de service nouvellement créé.

Nous pouvons également utiliser des certificats si nécessaire. Pour plus de simplicité, nous allons continuer avec le mot de passe.

Avec le principal de service que nous avons créé, le secret sera masqué. Pour connaître le secret, essayez les commandes suivantes :

```
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($sp.
Secret)
$UnsecureSecret = [System.Runtime.
InteropServices.Marshal]::PtrToStringAuto($BSTR)
```

$UnsecureSecret aura votre clé secrète.

Une application Application Directory sera créée en même temps que le principal de service. L'application agit comme la représentation globale de notre application dans les répertoires et le principal est assimilé à la représentation locale de l'application. Nous pouvons créer plusieurs principaux à l'aide de la même application dans un répertoire différent. Nous pouvons obtenir les détails de l'application créée à l'aide de la commande **`Get-AzAdApplication`**. Nous allons enregistrer la sortie de cette commande sur une variable, **`$app`**, car nous en aurons besoin plus tard :

```
$app = Get-AzAdApplication -DisplayName $sp.DisplayName
```

Nous avons maintenant créé un principal de service à l'aide d'un secret ; une autre méthode sécurisée pour en créer un consiste à utiliser des certificats. La section suivante porte sur la création de principaux de service à l'aide de certificats.

Étape 3 : création d'un principal de service à l'aide de certificats

Pour créer un principal de service à l'aide de certificats, les étapes suivantes doivent être exécutées :

1. **Créer un certificat auto-signé ou acheter un certificat** : un certificat auto-signé est utilisé pour créer cet exemple d'application end-to-end. Pour les déploiements de la vie réelle, un certificat valide doit être acheté auprès d'une autorité de certification.

   Pour créer un certificat auto-signé, la commande suivante peut être exécutée. Le certificat auto-signé est exportable et stocké dans un dossier personnel sur l'ordinateur local ; il comporte également une date d'expiration :

   ```
   $currentDate = Get-Date
   $expiryDate = $currentDate.AddYears(1)
   $finalDate = $expiryDate.AddYears(1)
   $servicePrincipalName = "https://automation.book.com"
   $automationCertificate = New-SelfSignedCertificate -DnsName
   $servicePrincipalName -KeyExportPolicy Exportable -Provider "Microsoft
   Enhanced RSA and AES Cryptographic Provider" -NotAfter $finalDate
   -CertStoreLocation "Cert:\LocalMachine\My"
   ```

2. **Exportez le certificat nouvellement créé** : le nouveau certificat doit être exporté vers le système de fichiers afin qu'ultérieurement, il puisse être téléchargé vers d'autres destinations, telles qu'Azure AD, pour créer un principal de service.

   Les commandes utilisées pour exporter le certificat vers le système de fichiers local sont affichées ensuite. Veuillez noter que ce certificat comporte des clés publiques et privées ; par conséquent, lorsqu'il est exporté, il doit être protégé à l'aide d'un mot de passe qui doit être une chaîne sécurisée :

```
$securepfxpwd = ConvertTo-SecureString -String 'password' -AsPlainText
-Force # Password for the private key PFX certificate
$cert1 = Get-Item -Path Cert:\LocalMachine\My\$($automationCertificate.
Thumbprint)
Export-PfxCertificate -Password $securepfxpwd -FilePath " C:\
azureautomation.pfx" -Cert $cert1
```

La cmdlet de commande **`Get-Item`** lit le certificat à partir du magasin de certificats et le stocke dans la variable **`$cert1`**. La cmdlet de commande **`Export-PfxCertificate`** exporte le certificat dans le magasin de certificats vers le système de fichiers. Dans ce cas, elle se trouve dans le dossier **`C:\book`**.

3. **Lire le contenu du fichier PFX nouvellement généré** : un objet de **`X509Certificate2`** est créé pour contenir le certificat en mémoire, et les données sont converties en une chaîne base64 à l'aide de la fonction **`System.Convert`** :

```
$newCert = New-Object System.Security.Cryptography.X509Certificates.
X509Certificate2 -ArgumentList "C:\azureautomation.pfx", $securepfxpwd
$newcertdata = [System.Convert]::ToBase64String($newCert.GetRawCertData())
```

Nous allons utiliser ce même principal pour nous connecter à Azure à partir du compte Azure Automation. Il est important que l'ID d'application, l'ID de locataire, l'ID d'abonnement et les valeurs d'empreinte de certificat soient stockés dans un emplacement temporaire afin qu'ils puissent être utilisés pour configurer les ressources suivantes :

```
$adAppName = "azure-automation-sp"
$ServicePrincipal = New-AzADServicePrincipal -DisplayName $adAppName
-CertValue $newcertdata -StartDate $newCert.NotBefore -EndDate
$newCert NotAfter
Sleep 10
New-AzRoleAssignment -ServicePrincipalName $ServicePrincipal.
ApplicationId -RoleDefinitionName Owner -Scope /subscriptions/xxxxx-
xxxxxxx-xxxxxx-xxxxxxx
```

Notre principal de service est prêt. Le coffre de clés que nous avons créé ne dispose pas d'un ensemble de stratégies d'accès, ce qui signifie qu'aucun utilisateur ni aucune application ne sera en mesure d'accéder au coffre. Dans l'étape suivante, nous accorderons des autorisations à l'application Application Directory que nous avons créée pour accéder au coffre de clés.

Étape 4 : création d'une stratégie de coffre de clés

À ce stade, nous avons créé le principal de service et le coffre de clés. Toutefois, le principal de service n'a toujours pas accès au coffre de clés. Ce principal de service sera utilisé pour interroger et répertorier tous les secrets, les clés et les certificats du coffre de clés, et il devrait avoir les autorisations nécessaires pour ce faire.

Pour fournir l'autorisation au principal de service nouvellement créé d'accéder au coffre de clés, nous allons revenir à la console Azure PowerShell et exécuter la commande suivante :

```
Set-AzKeyVaultAccessPolicy -VaultName keyvaultbook -ResourceGroupName
IntegrationDemo -ObjectId $ServicePrincipal.Id -PermissionsToKeys get,list,create
-PermissionsToCertificates get,list,import -PermissionsToSecrets get,list -Verbose
```

En vous référant au bloc de commande précédent, observez les points suivants :

- **`Set-AzureRmKeyVaultAccessPolicy`** fournit des autorisations d'accès aux utilisateurs, aux groupes et aux principaux de service. Il accepte le nom du coffre de clés et l'ID d'objet du principal du service. Cet objet est différent de l'ID d'application. La sortie du principal de service contient une propriété **`Id`**, comme illustré ici :

```
PS C:\windows\system32> $ServicePrincipal


ServicePrincipalNames : {f48850c9-580a-41d4-a062-77cd623e519e, http://azure-automation-sp}
ApplicationId         : f48850c9-580a-41d4-a062-77cd623e519e
ObjectType            : ServicePrincipal
DisplayName           : azure-automation-sp
Id                    : c95cdd04-5906-4bd7-ab5a-d4e617e98946
Type                  :
```

Figure 11.15 : recherche de l'ID d'objet du principal de service

- **`PermissionsToKeys`** fournit l'accès aux clés dans le coffre de clés et les autorisations **`get`**, **`list`** et **`create`** sont fournies à ce principal de service. Aucune autorisation d'écriture ou de mise à jour n'est fournie à ce principal.
- **`PermissionsToSecrets`** fournit l'accès aux secrets dans le coffre de clés et les autorisations **`get`** et **`list`** sont fournies à ce principal de service. Aucune autorisation d'écriture ou de mise à jour n'est fournie à ce principal.
- **`PermissionsToCertificates`** fournit l'accès aux secrets dans le coffre de clés et les autorisations **`get`**, **`import`** et **`list`** sont fournies à ce principal de service. Aucune autorisation d'écriture ou de mise à jour n'est fournie à ce principal.

À ce stade, nous avons configuré le principal de service à utiliser avec le coffre de clés Azure. La prochaine partie de la solution consiste à créer un compte Automation.

Étape 5 : création d'un compte Azure Automation

Comme avant, nous allons utiliser Azure PowerShell pour créer un compte Azure Automation dans un groupe de ressources. Avant de créer un groupe de ressources et un compte Automation, une connexion à Azure doit être établie. Toutefois, cette fois, vous allez utiliser les informations d'identification pour le principal de service afin de vous connecter à Azure. Procédez comme suit :

1. La commande permettant de se connecter à Azure à l'aide de l'application de service est la suivante. La valeur est extraite des variables que nous avons initialisées au cours des étapes précédentes :

```
Login-AzAccount -ServicePrincipal -CertificateThumbprint $newCert.
Thumbprint -ApplicationId $ServicePrincipal.ApplicationId -Tenant "xxxx-
xxxxxx-xxxxx-xxxxx"
```

2. Assurez-vous d'avoir accès en vérifiant la commande **`Get-AzContext`**, comme illustré ici. Notez l'ID d'abonnement car il sera nécessaire dans les commandes suivantes :

```
Get-AzContext
```

3. Après la connexion à Azure, une nouvelle ressource contenant les ressources de la solution et un nouveau compte Azure Automation doivent être créés. Nommez le groupe de ressources **`VaultMonitoring`** et créez-le dans la région **`Europe de l'Ouest`**. Vous allez également créer le reste des ressources dans ce groupe de ressources :

```
$IntegrationResourceGroup = "VaultMonitoring"
$rgLocation = "West Europe"
$automationAccountName = "MonitoringKeyVault"
New-AzResourceGroup -name $IntegrationResourceGroup -Location $rgLocation
New-AzAutomationAccount -Name $automationAccountName -ResourceGroupName
$IntegrationResourceGroup -Location $rgLocation -Plan Free
```

4. Ensuite, créez trois variables d'automation. Les valeurs qui leur correspondent, c'est-à-dire l'ID d'abonnement, l'ID de locataire et l'ID d'application, doivent déjà être disponibles à l'aide des étapes précédentes :

```
New-AzAutomationVariable -Name "azuresubscriptionid"
-AutomationAccountName $automationAccountName -ResourceGroupName
$IntegrationResourceGroup -Value " xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx "
-Encrypted $true

New-AzAutomationVariable -Name "azuretenantid" -AutomationAccountName
$automationAccountName -ResourceGroupName $IntegrationResourceGroup -Value
" xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx " -Encrypted $true

New-AzutomationVariable -Name "azureappid" -AutomationAccountName
$automationAccountName -ResourceGroupName $IntegrationResourceGroup -Value
" xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx " -Encrypted $true
```

5. Maintenant, il est temps de charger un certificat qui sera utilisé pour se connecter à Azure à partir d'Azure Automation :

```
$securepfxpwd = ConvertTo-SecureString -String 'password' -AsPlainText
-Force # Password for the private key PFX certificate
New-AzAutomationCertificate -Name "AutomationCertifcate" -Path "C:\book\
azureautomation.pfx" -Password $securepfxpwd -AutomationAccountName
$automationAccountName -ResourceGroupName $IntegrationResourceGroup
```

6. L'étape suivante consiste à installer des modules PowerShell liés à Key Vault et à Event Grid dans le compte Azure Automation, car ces modules ne sont pas installés par défaut.

7. À partir du portail Azure, accédez au groupe de ressources **`VaultMonitoring`** déjà créé en cliquant sur l'icône **Resource Groups** (Groupes de ressources) dans le menu de gauche.

8. Cliquez sur le compte Azure Automation déjà provisionné, **MonitoringKeyVault**, puis cliquez sur **Modules** dans le menu de gauche. Le module Event Grid dépend du module **`Az.profile`** et nous devons donc l'installer avant le module Event Grid.

9. Cliquez sur **Browse Gallery** (Parcourir la galerie) dans le menu supérieur et saisissez **`Az.profile`** dans la zone de recherche, comme illustré à la *Figure 11.16* :

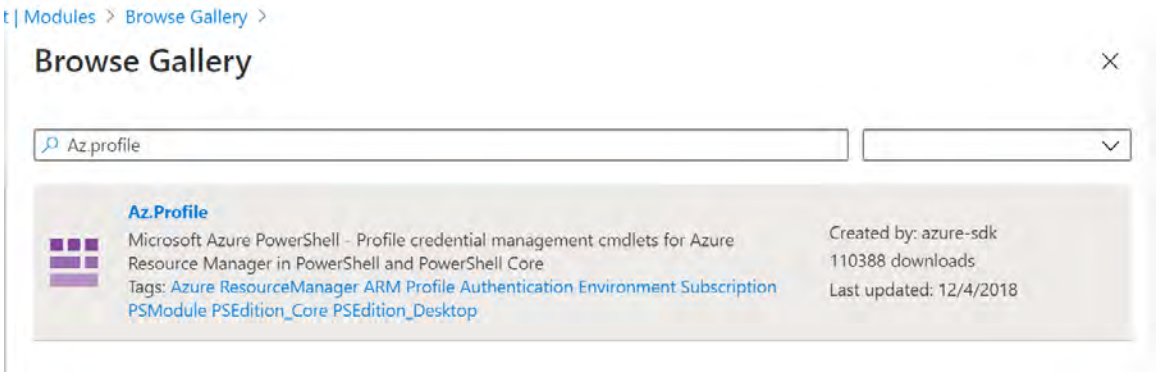


Figure 11.16 : le module Az.Profile dans la galerie du module

10. Dans les résultats de la recherche, sélectionnez **`Az.Profile`** et cliquez sur le bouton **Import** (Importer) dans le menu supérieur. Enfin, cliquez sur le bouton **OK**. Cette étape prend quelques secondes. Après quelques secondes, le module doit être installé.

11. L'état de l'installation peut être vérifié à partir de l'élément de menu **Module**. La *Figure 11.17* montre comment nous pouvons importer un module :

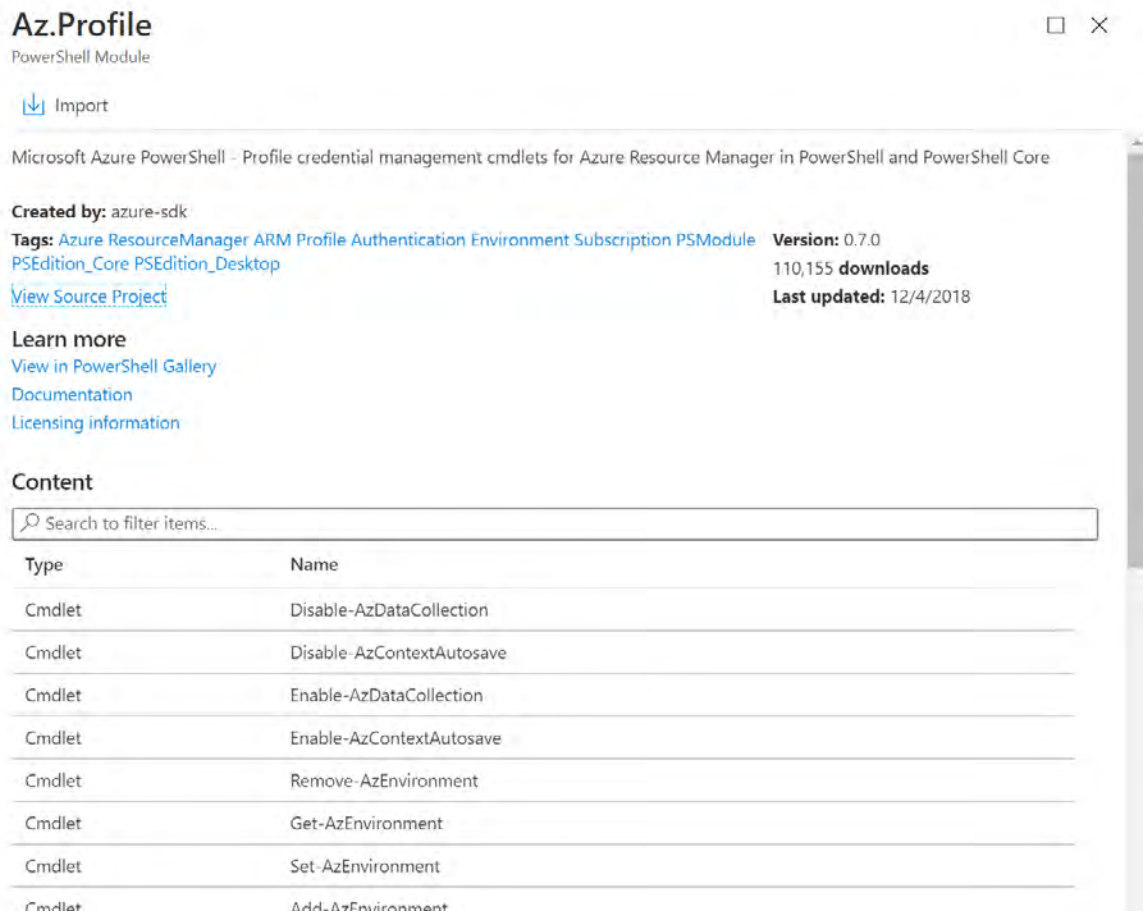


Figure 11,17 : état du module Az.Profile

12. Effectuez à nouveau les *étapes* 9, 10 et 11 afin d'importer et d'installer le module **`Az.EventGrid`**. Si vous êtes invité à installer des dépendances avant de continuer, suivez cette recommandation et commencez par installer les dépendances.
13. Effectuez à nouveau les *étapes* 9, 10 et 11 afin d'importer et d'installer le module **`Az.KeyVault`**. Si vous êtes invité à installer des dépendances avant de continuer, suivez cette recommandation et commencez par installer la dépendance.

Étant donné que nous avons importé les modules nécessaires, continuons en créant la rubrique Event Grid.

Étape 6 : création d'une rubrique Event Grid

Si vous vous souvenez de l'architecture que nous avons utilisée, une rubrique Event Grid est nécessaire. Créons-en une.

La commande utilisée pour créer une rubrique Event Grid à l'aide de PowerShell est la suivante :

```
New-AzEventGridTopic -ResourceGroupName VaultMonitoring -Name
azureforarchitects-topic -Location "West Europe"
```

Le processus de création d'une rubrique Event Grid à l'aide du portail Azure est le suivant :

1. À partir du portail Azure, accédez au groupe de ressources **`VaultMonitoring`** déjà créé en cliquant sur l'icône **Resource Groups** (Groupes de ressources) dans le menu de gauche.
2. Ensuite, cliquez sur le bouton **+Add** (+ Ajouter) et recherchez **`Event Grid Topic`** (Rubrique Event Grid) dans la zone de recherche. Sélectionnez-la et cliquez sur le bouton **Create** (Créer).
3. Renseignez les valeurs appropriées dans le formulaire résultant en fournissant un nom et en sélectionnant un abonnement, le groupe de ressources nouvellement créé, l'emplacement et le schéma d'événement.

Comme nous l'avons déjà mentionné, la rubrique Event Grid fournit un point de terminaison où la source enverra les données. Notre rubrique est prête, le moment est donc venu de préparer le compte Automation source.

Étape 7 : configuration de la procédure opérationnelle

Cette étape se concentrera sur la création d'un compte Azure Automation et des procédures opérationnelles PowerShell qui contiendront la logique principale de lecture des coffres de clés Azure et la récupération des secrets stockés dans ceux-ci. Voici les étapes requises pour la configuration d'Azure Automation :

1. **Création d'une procédure opérationnelle Azure Automation** : à partir du portail Azure, accédez au groupe de ressources **`Vaultmonitoring`** existant en cliquant sur l'icône **Resource Groups** (Groupes de ressources) dans le menu de gauche.
2. Cliquez sur le compte Azure Automation déjà provisionné, **`MonitoringKeyVault`**. Ensuite, cliquez sur **Runbooks** (Procédures opérationnelles) dans le menu de gauche, et cliquez sur **+Add a Runbook** (+ Ajouter un runbook) dans le menu supérieur.
3. Cliquez sur **Create a new Runbook** (Créer une procédure opérationnelle) et fournissez un nom. Nous allons appeler cette procédure opérationnelle **CheckExpiredAssets**, puis nous allons définir le **Runbook type** (Type de runbook) en tant que **PowerShell** :

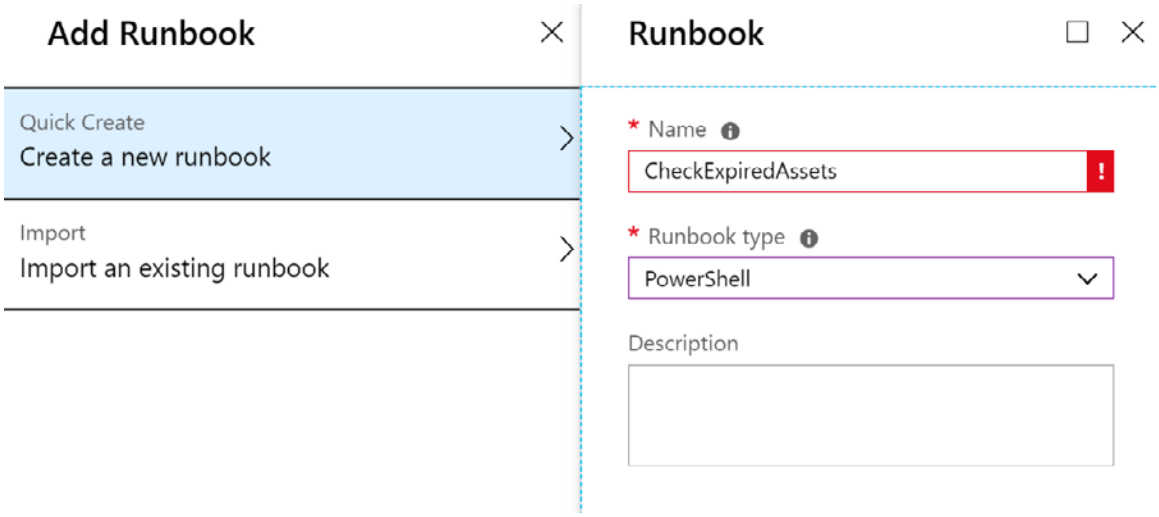


Figure 11.18 : Création d'une procédure opérationnelle

4. **Coder la procédure opérationnelle** : déclarez quelques variables pour contenir l'ID d'abonnement, l'ID de locataire, l'ID d'application et les informations d'empreinte de certificat. Ces valeurs doivent être stockées dans les variables Azure Automation et le certificat doit être chargé sur les certificats Automation. La clé utilisée pour le certificat chargé est **`AutomationCertifcate`**. Les valeurs sont récupérées à partir de ces magasins et sont affectées aux variables, comme illustré ci-après :

```
$subscriptionID = get-AutomationVariable "azuresubscriptionid"
$tenantID = get-AutomationVariable "azuretenantid"
$applicationId = get-AutomationVariable "azureappid"
$cert = get-AutomationCertificate "AutomationCertifcate"
$certThumbprint = ($cert.Thumbprint).ToString()
```

5. Le code suivant dans la procédure opérationnelle permet de se connecter à Azure à l'aide du principal de service avec les valeurs des variables déclarées précédemment. En outre, le code sélectionne un abonnement approprié. Le code s'affiche ensuite :

```
Login-AzAccount -ServicePrincipal -CertificateThumbprint $certThumbprint
-ApplicationId $applicationId -Tenant $tenantID
Set-AzContext -SubscriptionId $subscriptionID
```

   Dans la mesure où Azure Event Grid a été mis en service à l'*étape* 6 de cette section, son point de terminaison et ses clés sont récupérés à l'aide des cmdlets de commande **`Get-AzEventGridTopic`** et **`Get-AzEventGridTopicKey`**.

Azure Event Grid génère deux clés (une principale et une secondaire). La première référence de clé est la suivante :

```
$eventGridName = "ExpiredAssetsKeyVaultEvents"
$eventGridResourceGroup = "VaultMonitoring"
$topicEndpoint = (Get-AzEventGridTopic -ResourceGroupName
$eventGridResourceGroup -Name $eventGridName).Endpoint
$keys = (Get-AzEventGridTopicKey -ResourceGroupName
$eventGridResourceGroup -Name $eventGridName ).Key1
```

6. Ensuite, tous les coffres de clés qui ont été provisionnés dans l'abonnement sont récupérés à l'aide de l'itération. Pendant la boucle, tous les secrets sont récupérés à l'aide de la cmdlet de commande **`Get-AzKeyVaultSecret`**.

   La date d'expiration de chaque secret est comparée à la date actuelle, et si la différence est inférieure à un mois, elle génère un événement Event Grid et la publie à l'aide de la commande **`invoke-webrequest`**.

   Les mêmes étapes sont exécutées pour les certificats stockés dans le coffre de clés. La cmdlet de commande utilisée pour récupérer tous les certificats est **`Get-AzKeyVaultCertificate`**.

   L'événement qui est publié dans Event Grid doit figurer dans le tableau JSON. Le message généré est converti en JSON à l'aide de la cmdlet de commande **`ConvertTo-Json`**, puis converti en tableau en ajoutant **`[`** et **`]`** comme préfixe et suffixe.

   Afin de se connecter à Azure Event Grid et de publier l'événement, l'expéditeur doit fournir la clé dans son en-tête. La demande échouera si ces données sont manquantes dans la charge utile de la demande :

```
$keyvaults = Get-AzureRmKeyVault
foreach($vault in $keyvaults) {
$secrets = Get-AzureKeyVaultSecret -VaultName $vault.VaultName
foreach($secret in $secrets) {
if( ![string]::IsNullOrEmpty($secret.Expires) ) {
if($secret.Expires.AddMonths(-1) -lt [datetime]::Now)
{
$secretDataMessage = @{
id = [System.guid]::NewGuid()
subject = "Secret Expiry happening soon !!"
eventType = "Secret Expiry"
eventTime = [System.DateTime]::UtcNow
data = @{
"ExpiryDate" = $secret.Expires
```

```
"SecretName" = $secret.Name.ToString()
"VaultName" = $secret.VaultName.ToString()
"SecretCreationDate" = $secret.Created.ToString()
"IsSecretEnabled" = $secret.Enabled.ToString()
"SecretId" = $secret.Id.ToString()
}
}
...
Invoke-WebRequest -Uri $topicEndpoint -Body $finalBody -Headers $header
-Method Post -UseBasicParsing
}
}
Start-Sleep -Seconds 5
}
}
```

7. Publiez la procédure opérationnelle en cliquant sur le bouton **Publish** (Publier), comme illustré à la *Figure 11.19* :

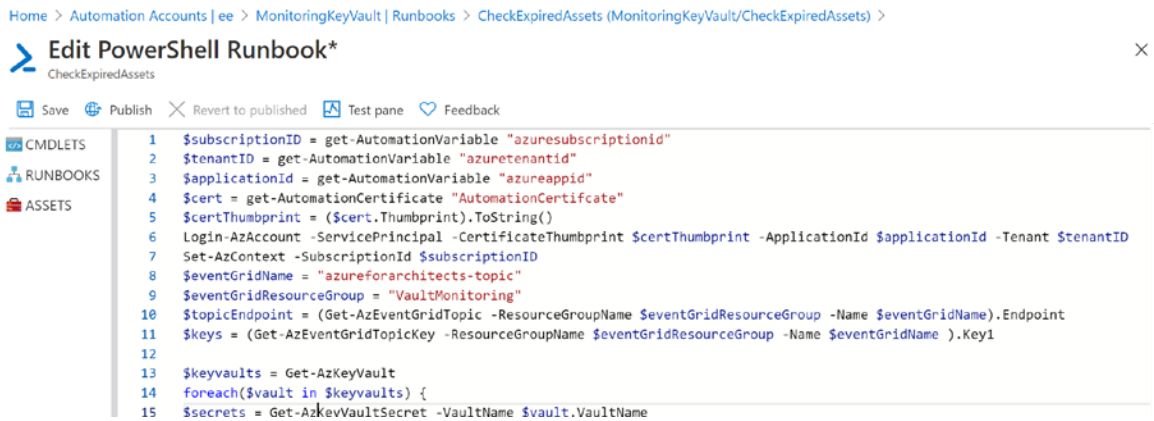


Figure 11.19 : publication de la procédure opérationnelle

8. **Planificateur** : créez une ressource de planificateur Azure Automation pour exécuter cette procédure opérationnelle une fois par jour à minuit. Ensuite, cliquez sur **Schedules** (Planifications) dans le menu de gauche d'Azure Automation et cliquez sur **+Add a schedule** (+ Ajouter une planification) dans le menu supérieur.
9. Fournissez les informations de planification dans le formulaire résultant.

Cela devrait finaliser la configuration du compte Azure Automation.

Étape 8 : travailler avec SendGrid

Dans cette étape, nous allons créer une ressource SendGrid. La ressource SendGrid est utilisée pour envoyer des e-mails à partir de l'application sans avoir besoin d'installer un serveur **SMTP (Simple Mail Transfer Protocol)**. Elle fournit une API REST et un **kit de développement logiciel (SDK)** C#, au moyen duquel il est assez facile d'envoyer des e-mails groupés. Dans la solution actuelle, Azure Functions sera utilisé pour appeler les API SendGrid pour envoyer des e-mails ; cette ressource doit donc être provisionnée. Cette ressource a un coût distinct et n'est pas couverte dans le cadre du coût Azure ; il existe un niveau gratuit disponible qui peut être utilisé pour l'envoi d'e-mails :

1. Une ressource **`SendGrid`** est créée comme n'importe quelle autre ressource Azure. Recherchez **`sendgrid`** pour obtenir la livraison **SendGrid Email Delivery** dans les résultats.
2. Sélectionnez la ressource et cliquez sur le bouton **Create** (Créer) pour ouvrir son formulaire de configuration.
3. Sélectionnez un niveau de tarification approprié.
4. Fournissez les coordonnées appropriées.
5. Cochez la case **Terms of use** (Conditions d'utilisation).
6. Complétez le formulaire et cliquez sur le bouton **Create** (Créer).
7. Une fois la ressource provisionnée, cliquez sur le bouton **Manage** (Gérer) dans le menu supérieur, ce qui ouvrira le site Web SendGrid. Le site Web peut exiger la configuration du courrier électronique. Ensuite, sélectionnez les **Clés API** dans la section **Settings** (Paramètres) et cliquez sur le bouton **Create API Key** (Créer une clé API) :

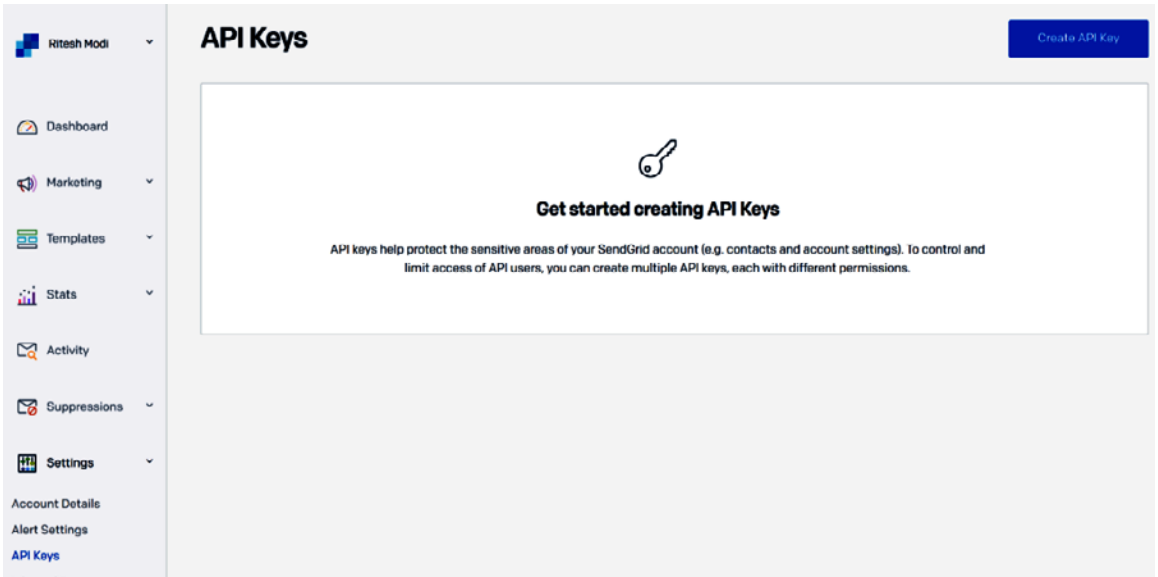


Figure 11.20 : création de clés d'API pour SendGrid

8. Dans la fenêtre résultante, sélectionnez **Full Access** (Accès complet) et cliquez sur le bouton **Create & View** (Créer et afficher). Cela créera les clés de la ressource SendGrid ; conservez une note de cette clé, car elle sera utilisée avec la configuration Azure Functions pour SendGrid :

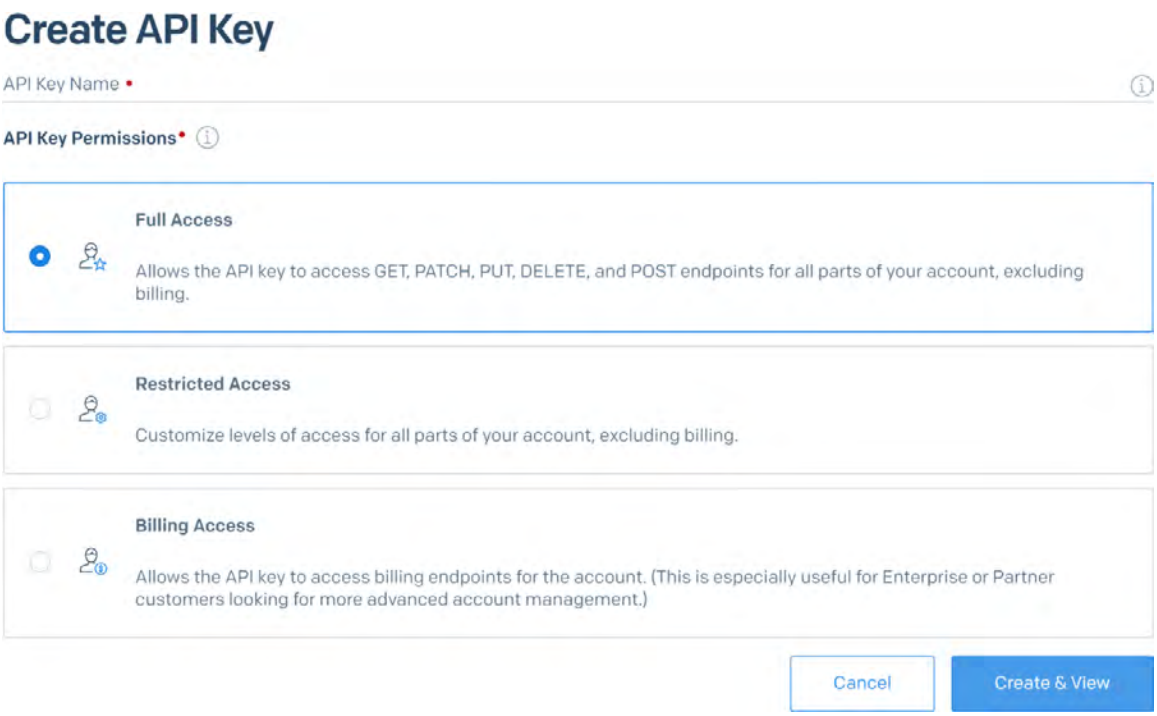


Figure 11.21 : configuration du niveau d'accès sur le portail SendGrid

Maintenant que nous avons configuré des niveaux d'accès de SendGrid, nous allons configurer un autre service tiers, appelé Twilio.

Étape 9 : prise en main de Twilio

Dans cette étape, nous allons créer un compte Twilio. Twilio est utilisé pour envoyer des messages SMS groupés. Pour créer un compte avec Twilio, accédez à twillio.com et créez un compte. Après avoir créé un compte, un numéro de mobile est généré, qui peut être utilisé pour envoyer des messages SMS aux destinataires :

Figure 11.22 : choix d'un numéro Twilio

Le compte Twilio fournit à la fois des clés de production et de test. Copiez la clé de test et le jeton dans un emplacement temporaire, tel que le bloc-notes, car ils seront requis ultérieurement dans Azure Functions :

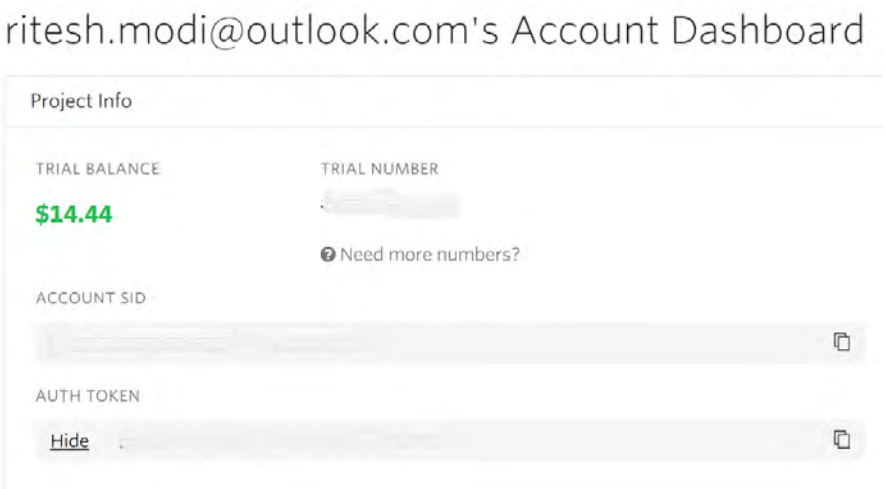


Figure 11.23 : configuration de Twilio

SendGrid et Twilio sont opérationnels pour le service de notification. Toutefois, nous avons besoin d'une fonction qui prend en charge l'événement et notifie les utilisateurs. C'est ici qu'une application de fonction entre en jeu. Dans la section suivante, nous allons créer une application de fonction qui permettra d'envoyer des SMS et des e-mails.

Étape 10 : configuration d'une application de fonction

Dans cette étape, nous allons créer une application de fonction responsable de l'envoi des e-mails et des notifications SMS. Le but de l'application de fonction dans la solution est d'envoyer des messages de notification aux utilisateurs concernant l'expiration des secrets dans le coffre de clés. Une seule fonction sera responsable de l'envoi des e-mails et des messages SMS ; notez que cela aurait pu être divisé en deux fonctions distinctes. La première étape consiste à créer une application de fonction et à héberger une fonction en son sein :

1. Comme nous l'avons fait précédemment, accédez à votre groupe de ressources, cliquez sur le bouton **+Add** (+ Ajouter) dans le menu supérieur, et recherchez la ressource **`application de fonction`**. Ensuite, cliquez sur le bouton **Create** (Créer) pour obtenir le formulaire **Function App** (Application de fonction).

2. Remplissez le formulaire **Function App** (Application de fonction) et cliquez sur le bouton **Create** (Créer). Le nom de l'application de fonction doit être unique dans Azure.

3. Une fois l'application de fonction mise en service, créez une fonction appelée **`SMSandEMailFunction`** en cliquant sur le bouton **+** en regard de l'élément **Functions** (Fonctions) dans le menu de gauche. Ensuite, sélectionnez **in-Portal** dans le tableau de bord central.

4. Sélectionnez **HTTP trigger** (Déclencheur HTTP) et nommez-le **`SMSandEMailFunction`**. Ensuite, cliquez sur le bouton **Create** (Créer) ; l'option **Authorization level** (Niveau d'autorisation) peut être n'importe quelle valeur.

5. Supprimez le code par défaut, remplacez-le par le code indiqué dans la liste suivante, puis cliquez sur le bouton **Save** (Enregistrer) dans le menu supérieur :

```
#r "SendGrid"
#r "Newtonsoft.Json"
#r "Twilio.Api"
using System.Net;
using System;
using SendGrid.Helpers.Mail;
using Microsoft.Azure.WebJobs.Host;
using Newtonsoft.Json;
using Twilio;
using System.Configuration;
public static HttpResponseMessage Run(HttpRequestMessage req, TraceWriter
log, out Mail message,out SMSMessage sms)
{
log.Info("C# HTTP trigger function processed a request.");
```

```
string alldata = req.Content.ReadAsStringAsync().GetAwaiter().GetResult();
message = new Mail();
var personalization = new Personalization();
personalization.AddBcc(new Email(ConfigurationManager.
AppSettings["bccStakeholdersEmail"]));
personalization.AddTo(new Email(ConfigurationManager.
AppSettings["toStakeholdersEmail"]));
var messageContent = new Content("text/html", alldata);
message.AddContent(messageContent);
message.AddPersonalization(personalization);
message.Subject = "Key Vault assets Expiring soon..";
message.From = new Email(ConfigurationManager.AppSettings["serviceEmail"]);
string msg = alldata;
sms = new SMSMessage();
sms.Body = msg;
sms.To = ConfigurationManager.AppSettings["adminPhone"];
sms.From = ConfigurationManager.AppSettings["servicePhone"];
return req.CreateResponse(HttpStatusCode.OK, "Hello ");
}
```

6. Cliquez sur le nom de l'application de fonction dans le menu de gauche et cliquez sur le lien **Application settings** (Paramètres d'application) dans la fenêtre principale :

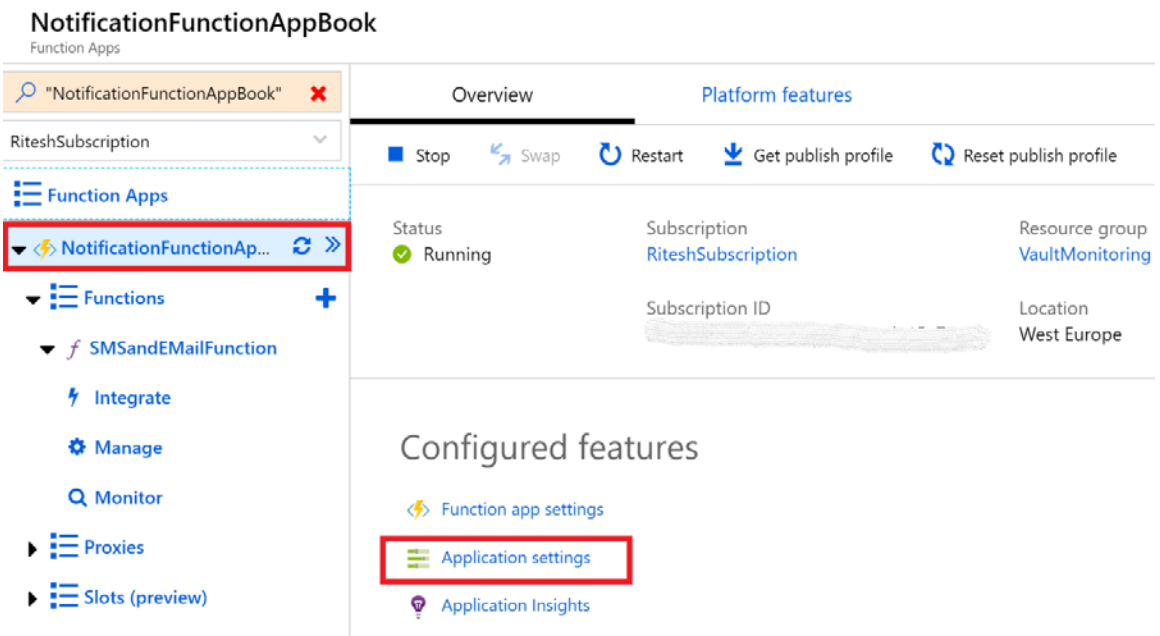


Figure 11.24 : accès aux paramètres d'application

7. Accédez à la section **Application settings** (Paramètres d'application), comme illustré à la *Figure 11.24* et ajoutez quelques entrées en cliquant sur **+ Add new setting** (+ Ajouter un nouveau paramètre) pour chaque entrée.

   Notez que les entrées sont sous la forme de paires clé-valeur, et les valeurs doivent être des valeurs réelles en temps réel. **`adminPhone`** et **`servicePhone`** devraient déjà être configurés sur le site Web Twilio. **`servicePhone`** est le numéro de téléphone généré par Twilio qui est utilisé pour l'envoi de messages SMS et **`adminPhone`** est le numéro de téléphone de l'administrateur à qui le SMS doit être envoyé.

   Notez également que Twilio s'attend à ce que le numéro de téléphone de destination soit dans un format particulier selon le pays (pour l'Inde le format est **`+91 xxxxx xxxxx`**). Notez les espaces et le code pays dans le numéro.

   Nous devons également ajouter les clés pour SendGrid et Twilio dans les paramètres de l'application. Ces paramètres sont mentionnés dans la liste suivante. Vous pouvez déjà avoir ces valeurs à portée de main en raison des activités exécutées dans les étapes précédentes :

   - La valeur de **`SendGridAPIKeyAsAppSetting`** est la clé de SendGrid.
   - **`TwilioAccountSid`** est l'identificateur système du compte Twilio. Cette valeur a déjà été copiée et stockée dans un emplacement temporaire au cours de l'*étape 9 : prise en main de Twilio*.
   - **`TwilioAuthToken`** est le jeton du compte Twilio. Cette valeur a déjà été copiée et stockée dans un emplacement temporaire lors d'une étape antérieure.

8. Enregistrez les paramètres en cliquant sur le bouton **Save** (Enregistrer) dans le menu supérieur :

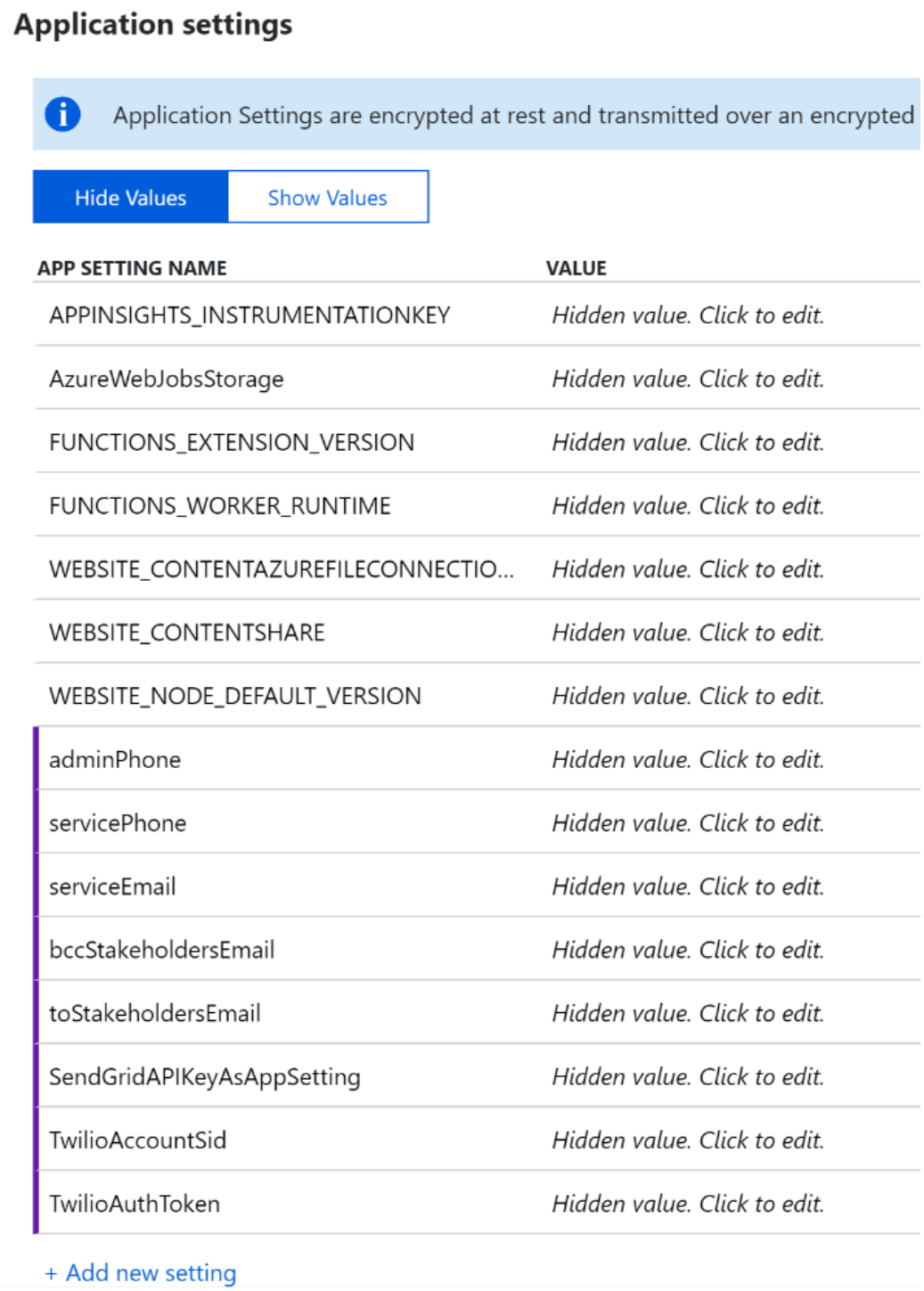


Figure 11.25 : configuration des paramètres d'application

9. Cliquez sur le lien **Integrate** (Intégrer) dans le menu de gauche juste en dessous du nom de la fonction, et cliquez sur **+ New Output** (+ Nouvelle sortie). Il s'agit d'ajouter une sortie pour le service SendGrid :

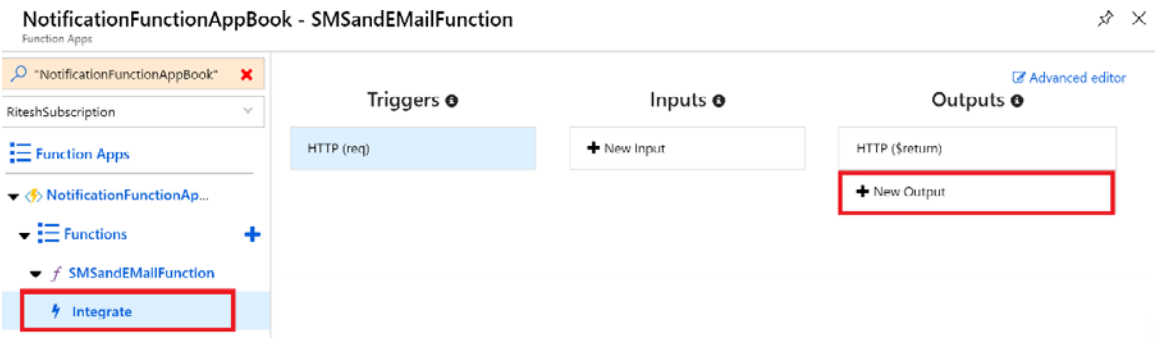


Figure 11.26 : ajout d'une sortie à l'application de fonction.

10. Ensuite, sélectionnez **SendGrid** ; vous devrez peut-être installer l'extension SendGrid. Installez l'extension, ce qui prendra quelques minutes :

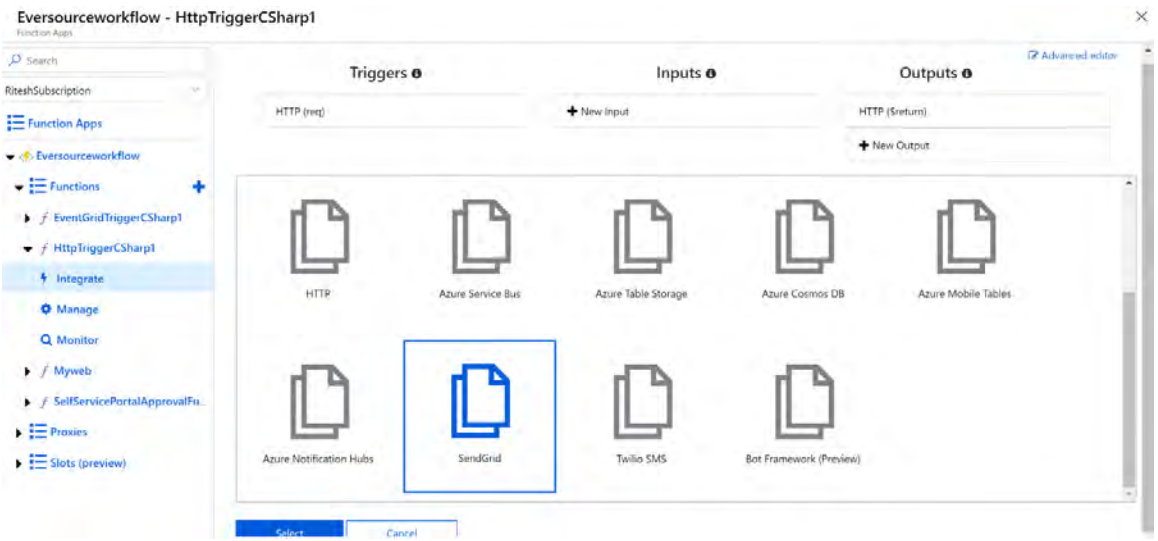


Figure 11.27 : configuration d'une application de fonction

11. Après l'installation de l'extension, le formulaire de configuration de sortie apparaît. Les éléments de configuration importants de ce formulaire sont le **Message parameter name** (nom de paramètre du message) et **SendGridAPIKeyAppSetting**. Conservez la valeur par défaut pour le **Message parameter name** (nom de paramètre du message) et cliquez sur la liste déroulante pour sélectionner **SendGridAPIKeyAsAppSetting** comme clé de configuration de l'application API. Cela a déjà été configuré dans une étape précédente dans la configuration des paramètres de l'application. Configurez le formulaire comme indiqué à la *Figure 11.28*, puis cliquez sur le bouton **Save** (Enregistrer) :

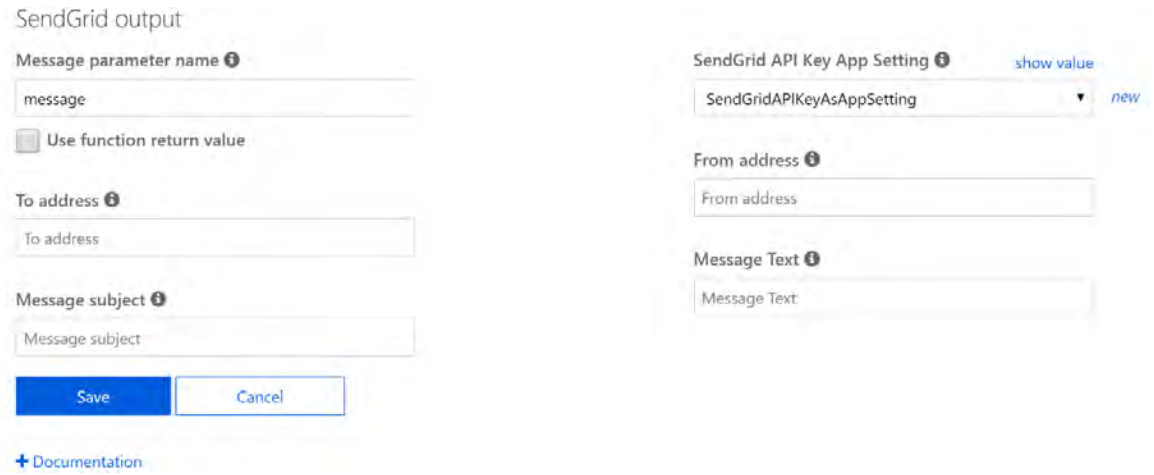


Figure 11.28 : configuration de SendGrid

12. Cliquez sur **+ New Output** (+ Nouvelle sortie) à nouveau pour ajouter une sortie pour le service Twilio.

13. Ensuite, sélectionnez **Twilio SMS**. Vous devrez peut-être installer l'extension Twilio SMS. Installez l'extension, ce qui prendra quelques minutes.

14. Après l'installation de l'extension, le formulaire de configuration de sortie apparaît. Les éléments de configuration importants de ce formulaire sont **Message parameter name** (nom du paramètre de message), **Account SID setting** (paramètre de compte SID) et **Auth Token setting** (paramètre Auth Token). Remplacez la valeur par défaut du **Message parameter name** (nom du paramètre de message) par `sms`. En effet, le paramètre `message` est déjà utilisé pour le paramètre de service SendGrid. Assurez-vous que la valeur du **Account SID setting** (paramètre SID du compte) est `TwilioAccountSid` et que la valeur du **Auth Token setting** (paramètre Auth Token) est **TwilioAuthToken**. Ces valeurs ont déjà été configurées dans une étape précédente dans la configuration des paramètres de l'application. Configurez le formulaire comme indiqué à la *Figure 11.29*, puis cliquez sur le bouton **Save** (Enregistrer) :

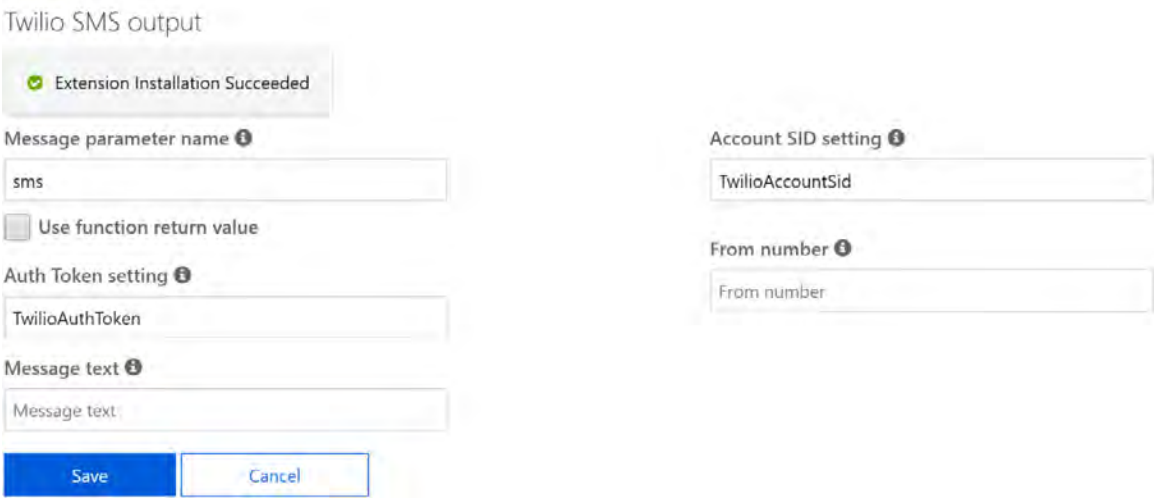


Figure 11.29 : configuration de la sortie Twilio SMS.

Nos comptes SendGrid et Twilio sont prêts. Le moment est venu d'utiliser les connecteurs et de les ajouter à l'application logique. Dans la partie suivante, nous allons créer l'application logique et utiliser des connecteurs pour travailler avec les ressources que nous avons créées jusqu'à présent.

Étape 11 : création d'une application logique

Dans cette étape, nous allons créer une charge de travail d'application logique. Nous avons créé une procédure opérationnelle Azure Automation qui interroge tous les secrets dans tous les coffres de clés et publie un événement au cas où elle détecte que l'un d'eux expire dans un mois. La charge de travail de l'application logique agit en tant qu'abonné à ces événements :

1. La première étape du menu **Logic App** est de créer une charge de travail d'application logique.
2. Remplissez le formulaire résultant après avoir cliqué sur le bouton **Create** (Créer). Nous mettons en service l'application logique dans le même groupe de ressources que les autres ressources pour cette solution.
3. Une fois que l'application logique est provisionnée, elle ouvre la fenêtre du concepteur. Sélectionnez **Blank Logic App** (Application logique vide) dans la section **Templates** (Modèles).
4. Dans la fenêtre résultante, ajoutez un déclencheur qui peut s'abonner aux événements Event Grid. Logic Apps fournit un déclencheur pour Event Grid, et vous pouvez effectuer une recherche pour voir s'il est disponible.

5. Ensuite, sélectionnez le déclencheur **When a resource event occurs (preview)** (Lorsqu'un événement de ressource se produit [aperçu]) :

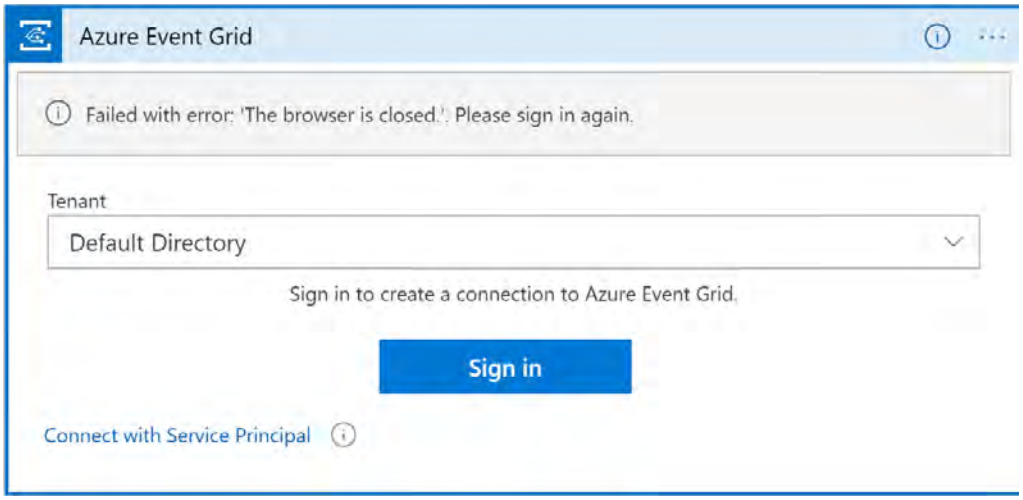


Figure 11.30 : sélection d'un déclencheur à l'aide d'Event Grid

6. Dans la fenêtre résultante, sélectionnez **Connect with Service Principal** (Connexion avec le principal du service).

   Fournissez les détails du principal de service, y compris l'ID d'application (**ID client**), l'ID de locataire et le mot de passe. Ce déclencheur n'accepte pas un principal de service qui s'authentifie avec le certificat ; il ne reçoit un principal de service qu'avec un mot de passe. Créez un principal de service à ce stade qui s'authentifie avec un mot de passe (les étapes de création d'un principal de service basé sur l'authentification par mot de passe a été abordée un peu plus tôt à l'*étape 2*) et utilisez les détails du principal de service nouvellement créé pour la configuration Azure Event Grid, comme indiqué à la *Figure 11.31* :

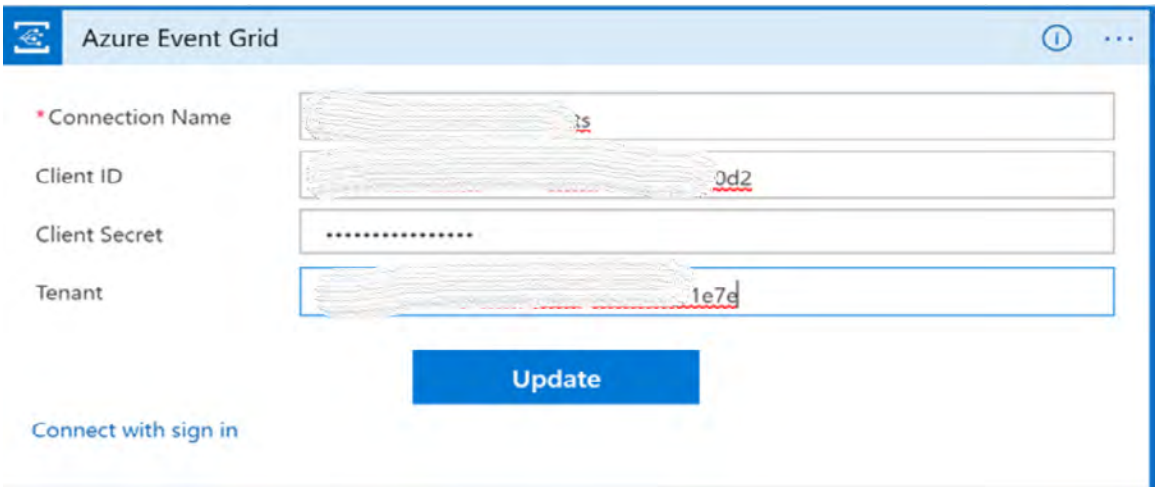


Figure 11.31 : saisie des détails du principal de service pour la connexion

7. Sélectionnez l'abonnement. Cela sera automatiquement rempli, selon la portée du principal de service. Sélectionnez **Microsoft.EventGrid.Topics** comme **Resource Type** (Type de ressource) et définissez le nom de la rubrique personnalisée comme **ExpiredAssetsKeyVaultEvents** :

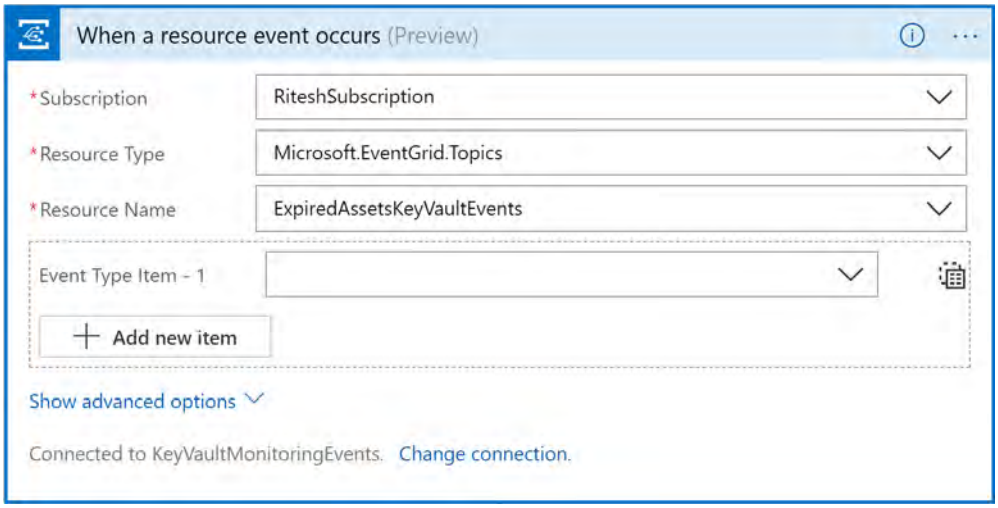


Figure 11.32 : saisie des détails sur le déclencheur Event Grid

8. L'étape précédente créera un connecteur, et les informations de connexion peuvent être modifiées en cliquant sur **Change connection** (Modifier la connexion).
9. La configuration finale du déclencheur Event Grid doit être similaire à la *Figure 11.33* :

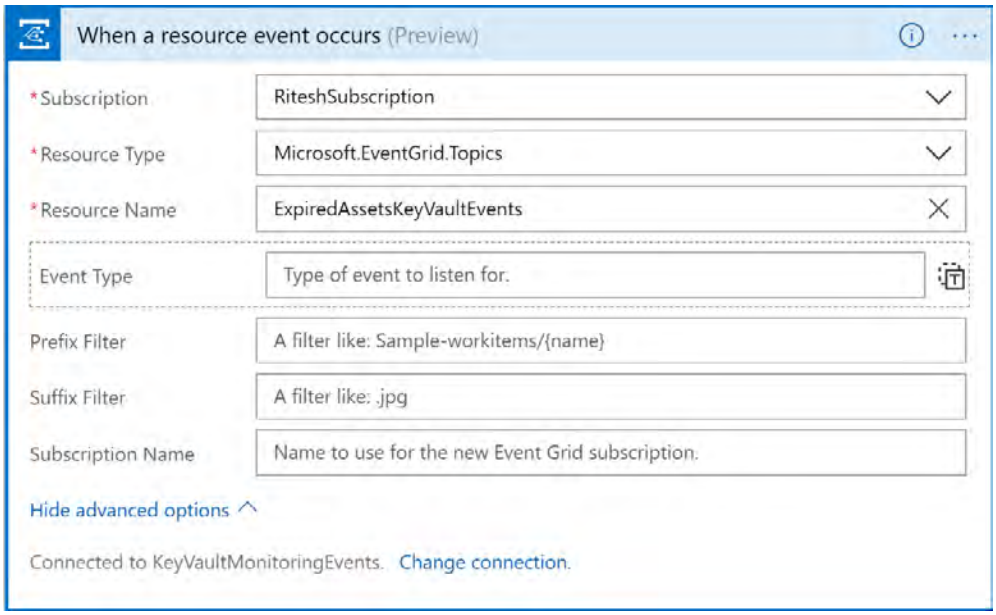


Figure 11.33 : présentation du déclencheur Event Grid

10. Ajoutez une nouvelle activité **Parse JSON** (Analyser JSON) après le déclencheur Event Grid ; cette activité nécessite le schéma JSON. En général, le schéma n'est pas disponible, mais cette activité permet de générer le schéma si un JSON valide lui est fourni :

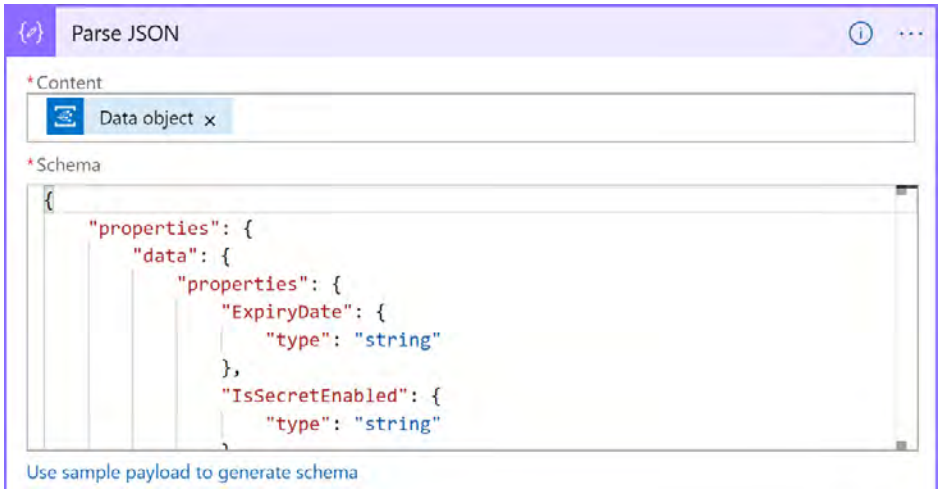


Figure 11.34 : Activité Analyser JSON

11. Cliquez sur **Use sample payload to generate schema** (Utiliser l'exemple de charge utile pour générer le schéma) et fournissez les données suivantes :

```
{
"ExpiryDate": "",
"SecretName": "",
"VaultName": "",
"SecretCreationDate": "",
"IsSecretEnabled": "",
"SecretId": ""
}
```

Une question peut se poser ici concernant la charge utile de l'échantillon. Comment calculez-vous à ce stade la charge utile générée par l'éditeur Event Grid ? La réponse à cela réside dans le fait que cet exemple de charge utile est exactement le même que celui utilisé dans l'élément de données de la procédure opérationnelle Azure Automation. Vous pouvez jeter un œil à cet extrait de code à nouveau :

```
data = @{
"ExpiryDate" = $certificate.Expires
"CertificateName" = $certificate.Name.ToString()
"VaultName" = $certificate.VaultName.ToString()
"CertificateCreationDate" = $certificate.Created.ToString()
"IsCertificateEnabled" = $certificate.Enabled.ToString()
"CertificateId" = $certificate.Id.ToString()
}
```

12. La zone de texte **Content** (Contenu) doit contenir du contenu dynamique sortant du déclencheur précédent, comme illustré à la *Figure* 11.35 :

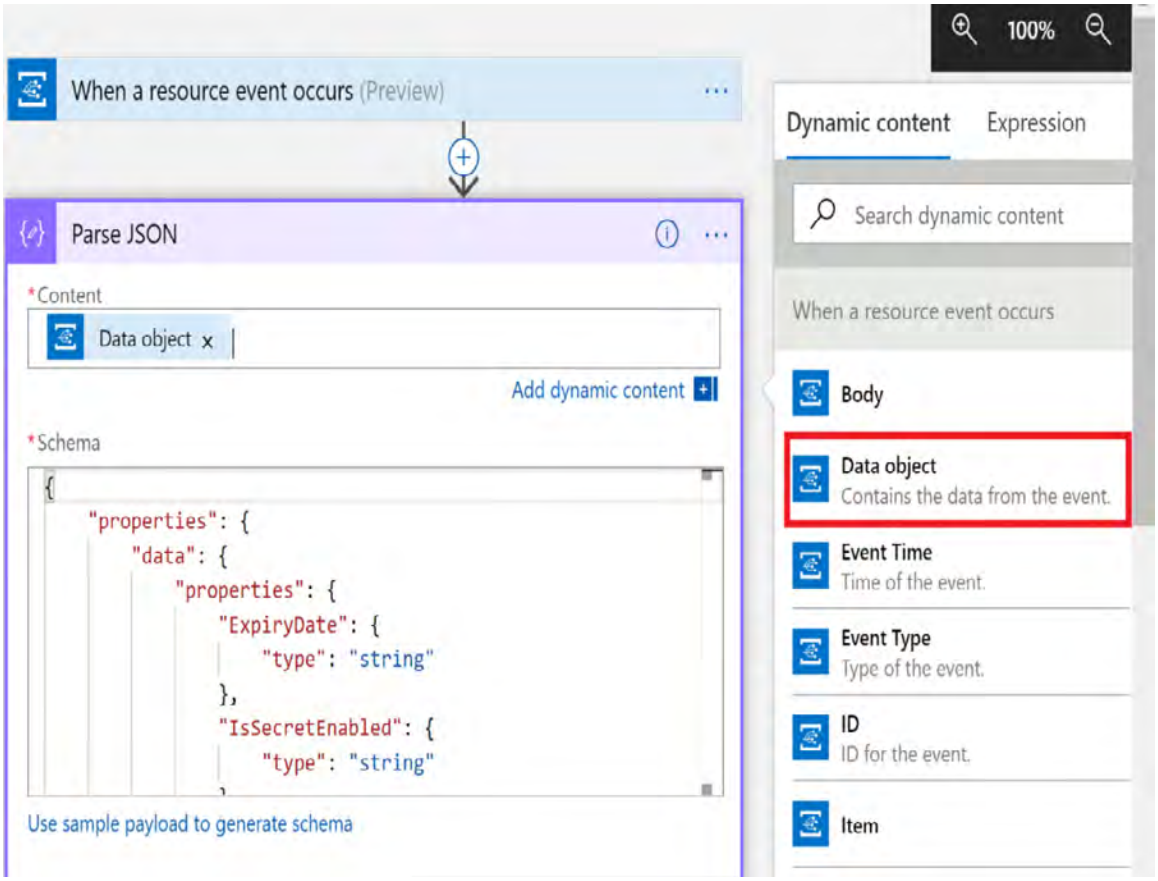


Figure 11.35 : fourniture d'un contenu dynamique à l'activité Analyser JSON

13. Ajoutez une autre action **Azure Functions** après **Parse JSON** (Analyser JSON), puis sélectionnez **Choose an Azure function** (Choisir une fonction Azure). Sélectionnez les applications de fonction Azure dénommées **NotificationFunctionAppBook** et **SMSAndEmailFunction**, qui ont été créées précédemment :

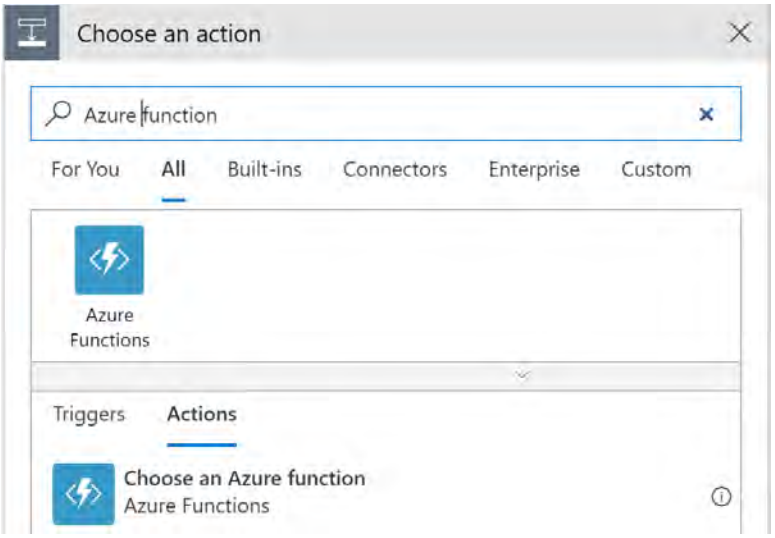


Figure 11.36 : ajout d'une action Azure Functions

14. Cliquez sur la zone de texte **Request Body** (Demander un corps) et remplissez-la avec la liste de codes suivante. Ceci permet de convertir les données en JSON avant de les envoyer à la fonction Azure :

```
{
"alldata" :
}
```

15. Placez le curseur après « : » dans le code précédent et cliquez sur **Add dynamic content | Body** (Ajouter un contenu dynamique | Corps) de l'activité précédente :

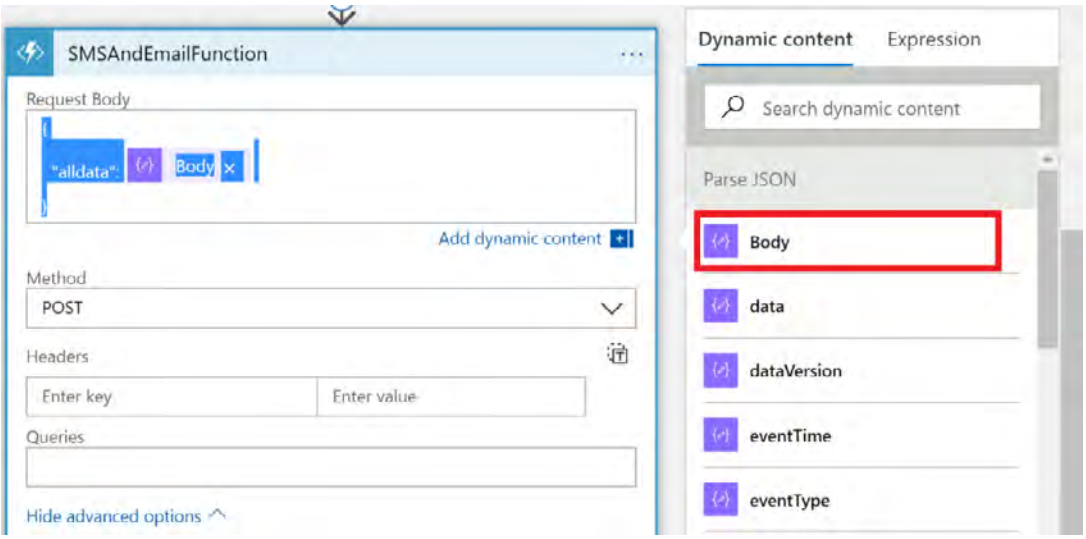


Figure 11.37 : conversion des données en JSON avant de les envoyer à une fonction Azure.

16. Enregistrez l'intégralité de l'application logique ; cette dernière doit ressembler à ceci :

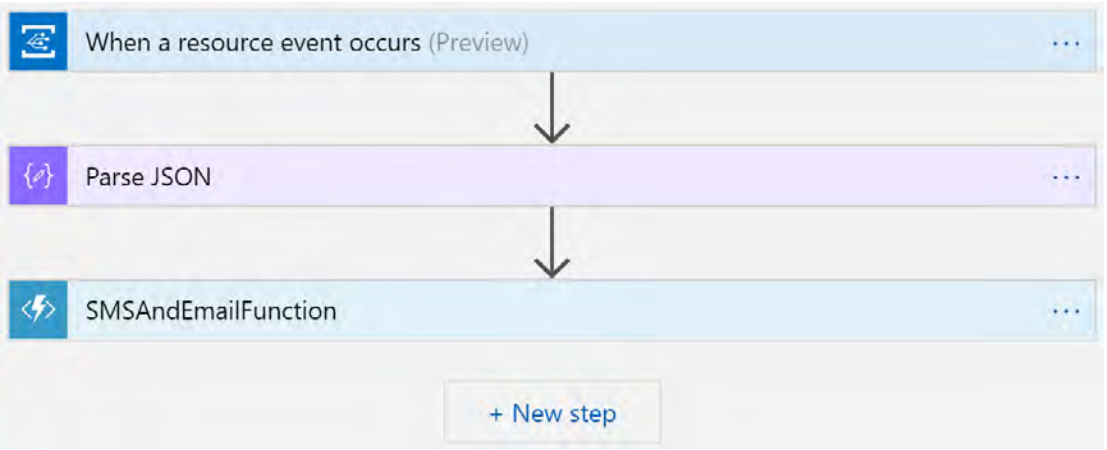


Figure 11.38 : charge de travail d'application logique

Une fois que vous avez enregistré l'application logique, votre solution est prête à être testée. Si vous n'avez pas de clés ou de secrets, ajoutez-les avec une date d'expiration afin de vérifier le fonctionnement de votre solution.

Tests

Chargez des secrets et des certificats qui ont des dates d'expiration dans le coffre de clés Azure et exécutez la procédure opérationnelle Azure Automation. Le runbook est planifié pour s'exécuter sur une planification. En outre, le runbook publiera des événements dans Event Grid. L'application logique doit être activée, elle choisira l'événement et enfin appellera la fonction Azure pour envoyer des notifications par e-mail et SMS.

L'e-mail devrait ressembler à ceci :

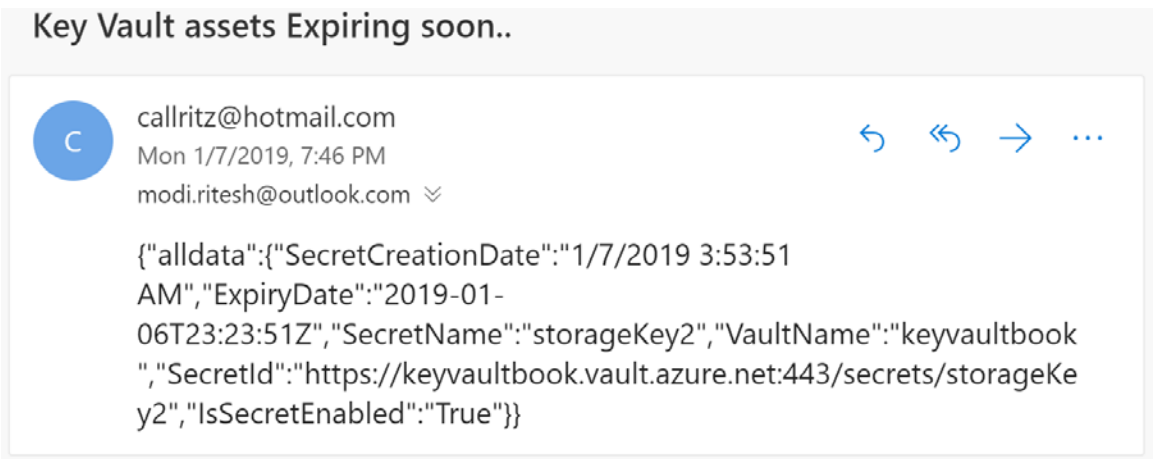


Figure 11.39 : e-mail reçu concernant les clés expirant

Au cours de cet exercice, nous avons identifié un problème, avant de concevoir et de mettre en œuvre une solution. C'est précisément ce que doit faire un architecte. Les clients auront des exigences spécifiques et vous devrez tenir compte de celles-ci lorsque vous concevrez des solutions. C'est sur cette note que nous concluons ce chapitre. Récapitulons brièvement les sujets abordés.

Résumé

Dans ce chapitre, nous avons découvert Logic Apps et conçu une solution complète end-to-end à l'aide de plusieurs services Azure. Le chapitre s'est fortement concentré sur la création d'une architecture qui intégrait plusieurs services Azure pour créer une solution end-to-end. Les services utilisés dans la solution étaient Azure Automation, Azure Logic Apps, Azure Event Grid, Azure Functions, SendGrid et Twilio. Ces services ont été implémentés via le portail Azure et PowerShell à l'aide des principaux de service en tant que comptes de service. Ce chapitre a également présenté un certain nombre de façons de créer des principaux de service avec l'authentification par mot de passe et par certificat.

De nombreuses méthodes permettent de résoudre un problème. Vous pouvez utiliser un déclencheur Outlook dans une application logique au lieu de SendGrid. Un seul problème présente de nombreuses solutions et celle que vous adopterez dépendra de l'approche appliquée. Plus vous connaîtrez les services et plus vous disposerez d'options pour résoudre les problèmes. Dans le chapitre suivant, vous découvrirez l'importance des événements dans Azure et l'architecture d'applications Azure.

12

Solutions d'événements de Big Data Azure

Les événements sont partout ! Toute activité ou tâche qui modifie l'état actuel d'un élément de travail génère un événement. En raison d'un manque d'infrastructure et de l'absence d'appareils bon marché, l'**Internet des objets** rencontrait un succès limité. Historiquement, les organisations utilisaient des environnements hébergés provenant des **fournisseurs d'accès Internet** (**FAI**) qui fonctionnaient simplement avec des systèmes de surveillance. Ces systèmes de surveillance signalaient des événements qui étaient rares et isolés entre eux.

Cependant, avec l'avènement du Cloud, les choses évoluent rapidement. Avec l'accroissement des déploiements dans le Cloud, en particulier des services de **plateforme en tant que service** (**PaaS**), les organisations n'ont plus besoin de beaucoup de contrôle sur le matériel et la plateforme, et un événement est désormais déclenché chaque fois que l'environnement connaît un changement. Avec l'émergence des événements de Cloud, l'IoT a pris une importance considérable et les événements commencent à occuper une place centrale.

La soudaine croissance rapide de la disponibilité des données est un autre phénomène récent. La rapidité, la variété et le volume des données ont enregistré un pic de croissance, ce qui a entraîné en conséquence une demande croissante pour des solutions de stockage et de traitement des données. Plusieurs solutions et plateformes ont émergé, telles que Hadoop, les lacs de données pour le stockage, les lacs de données pour l'analyse de données et les services de Machine Learning.

Outre le stockage et l'analyse de données, il existe également une demande pour des services capables d'ingérer des millions et des millions d'événements et de messages provenant de différentes sources. De plus, on constate un besoin en services capables de traiter des données temporelles, plutôt que de travailler sur l'ensemble de l'instantané des données. Par exemple, les données d'événement/d'IoT sont utilisées dans des applications qui prennent des décisions basées sur des données en temps réel ou quasi temps réel, par exemple des systèmes de gestion du trafic ou des systèmes qui surveillent la température.

Azure fournit une pléthore de services qui permettent de capturer et d'analyser les données en temps réel des capteurs. Dans ce chapitre, nous allons aborder quelques services d'événements dans Azure, dont voici la liste :

- Azure Event Hubs
- Azure Stream Analytics

Il existe d'autres services d'événements, tels qu'Azure Event Grid, qui ne sont pas abordés dans ce chapitre. Toutefois, ils sont largement couverts au *chapitre 10*, *Azure Integration Services avec Azure Functions* (*fonctions durables et fonctions de proxy*).

Présentation des événements

Les événements sont des constructions importantes dans Azure et dans l'architecture des applications Azure. Les événements sont partout dans l'écosystème logiciel. En général, toute action mise en œuvre entraîne un événement qui peut être capturé et à la suite duquel d'autres actions peuvent être effectuées. Pour poursuivre cette discussion, il est important de commencer par comprendre les bases des événements.

Les événements aident à capturer le nouvel état d'une ressource cible. Un message est une notification légère d'une condition ou d'un changement d'état. Les événements sont différents des messages. Les messages ont trait aux fonctions métier, telles que l'envoi de détails d'une commande à un autre système. Ils contiennent des données brutes et peuvent être volumineux. Mais ils sont différents des événements. Par exemple, une machine virtuelle arrêtée est un événement. La *figure 12.1* illustre cette transition entre l'état actuel et l'état cible :

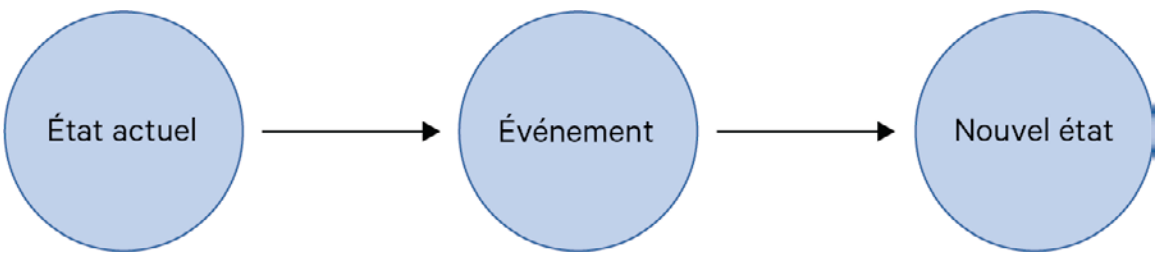


Figure 12.1 : Transition d'un état en raison d'un événement

Les événements peuvent être stockés dans un stockage durable, car les données et les événements historiques peuvent également être utilisés pour trouver des modèles qui apparaissent continuellement. Les événements peuvent être considérés comme des données diffusées en continu. Pour capturer, ingérer et effectuer des analyses sur un flux de données, des composants d'infrastructure spéciaux qui peuvent lire une petite fenêtre de données et fournir des informations sont nécessaires, et c'est là que le service Stream Analytics entre en scène.

Streaming d'événements

Le traitement des événements au fur et à mesure qu'ils sont ingérés et diffusés sur une période de temps fournit des informations en temps réel sur les données. Cette période de temps peut durer 15 minutes ou une heure. Elle est définie par l'utilisateur et dépend des informations à extraire des données. Prenons l'exemple des paiements par carte bancaire. Des millions de cartes bancaires sont utilisées pour payer toutes les minutes et la détection des fraudes peut être exécutée sur des événements en streaming pendant une période de temps d'une ou deux minutes.

Le streaming d'événements fait référence à des services qui peuvent accepter des données au fur et à mesure qu'elles se présentent au lieu de les accepter périodiquement. Par exemple, les flux d'événements doivent être capables d'accepter des informations de température à partir d'appareils au fur et à mesure de leur envoi, plutôt que de placer les données dans une file d'attente ou dans un environnement intermédiaire.

La diffusion d'événements a également la capacité d'interroger des données pendant leur transit. Il s'agit des données temporelles qui sont stockées pendant un certain temps, et les requêtes se produisent sur les données en mouvement, ce qui signifie que les données ne sont pas stationnaires. Cette fonction n'est pas disponible sur d'autres plateformes de données, qui peuvent uniquement interroger des données stockées et non des données temporelles venant d'être ingérées.

Les services de streaming d'événements devraient pouvoir évoluer facilement pour accepter des millions, voire des milliards d'événements. Ils devraient être hautement disponibles de telle sorte que les sources puissent leur envoyer des événements et des données à tout moment. L'ingestion de données en temps réel et la possibilité de travailler sur ces données, plutôt que sur des données stockées dans un emplacement différent, sont la clé du streaming d'événements.

Mais pourquoi avons-nous besoin du streaming d'événements alors qu'il existe déjà tant de plateformes de données avec des capacités avancées d'exécution de requête ? L'un des principaux avantages du streaming d'événements est qu'il fournit des perspectives en temps réel et des informations dont l'utilité dépend du temps. Les mêmes informations identifiées quelques minutes ou heures plus tard seront peut-être moins utiles. Examinons quelques scénarios dans le cadre desquels le travail sur les données entrantes est très important. Ces scénarios ne peuvent pas être résolus efficacement par les plateformes de données existantes :

- **Détection de l'utilisation frauduleuse d'une carte bancaire** : cela doit se produire au moment même où la transaction frauduleuse se produit.
- **Informations de télémétrie des capteurs** : dans le cas des appareils IoT qui envoient des informations vitales sur leurs environnements, l'utilisateur doit être averti au fur et à mesure de la détection d'une anomalie.
- **Tableaux de bord en direct** : le streaming d'événements est nécessaire pour créer des tableaux de bord qui affichent des informations en direct.
- **Télémétrie de l'environnement du datacenter** : cela permettra à l'utilisateur de connaître les intrusions, les failles de sécurité, les défaillances de composants, etc.

Il existe de nombreuses possibilités d'appliquer le streaming d'événements au sein d'une entreprise, et on ne saurait trop souligner son importance.

Event Hubs

Azure Event Hubs est une plateforme de streaming qui fournit des fonctionnalités liées à l'ingestion et au stockage des événements nécessaires au streaming.

Elle peut ingérer des données provenant de différentes sources ; ces sources peuvent être des capteurs IoT ou des applications utilisant le **Kit de développement logiciel** (**SDK**) Event Hubs. Elle prend en charge plusieurs protocoles pour l'ingestion et le stockage de données. Ces protocoles sont des normes du secteur et ils comprennent les éléments suivants :

- **HTTP** : il s'agit d'une option sans état qui ne nécessite pas de session active.
- **Advanced Messaging Queuing Protocol** (**AMQP**) : cela nécessite une session active (c'est-à-dire une connexion établie à l'aide de sockets) et fonctionne avec la sécurité **TLS** (**Transport Layer Security**) et **SSL** (**Secure Socket Layer**).
- **Apache Kafka**: il s'agit d'une plateforme de streaming distribué similaire à Stream Analytics. Toutefois, Stream Analytics est conçu pour exécuter des analyses en temps réel sur plusieurs flux de données provenant de diverses sources, telles que les capteurs de l'Internet des objets et les sites web.

Event Hubs est un service d'ingestion d'événements. Il n'a pas la capacité d'interroger une requête et de générer des résultats à un autre emplacement. Ces tâches sont assurées par Stream Analytics, qui est abordé dans la section suivante.

Pour créer une instance Event Hubs à partir du portail, recherchez les Event Hubs sur Marketplace et cliquez sur **Créer**. Sélectionnez un abonnement et un groupe de ressources existant (ou créez-en un nouveau). Indiquez un nom pour l'espace de noms Event Hubs, la région Azure préférée pour l'héberger, le niveau de tarification (basique ou standard, expliqué plus loin) et le nombre d'unités de débit (expliquées plus loin) :

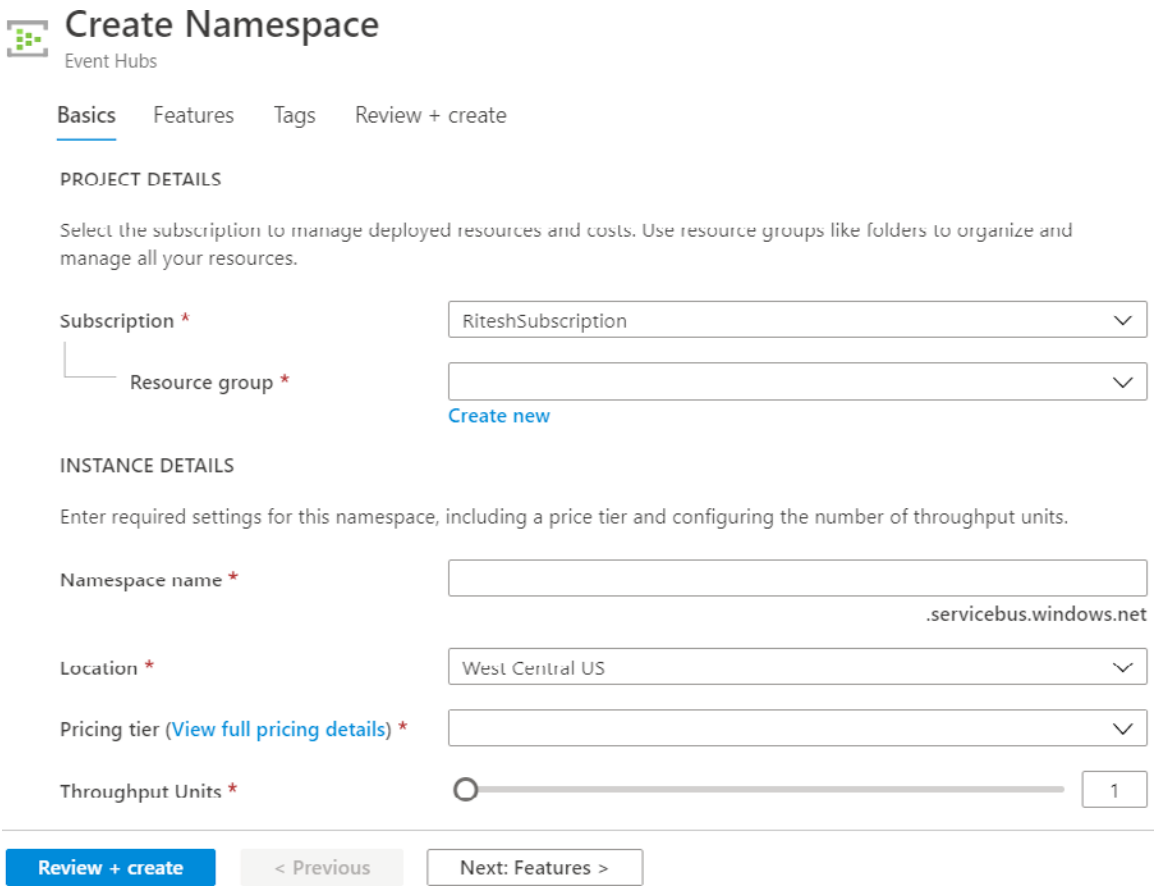


Figure 12.2 : Création d'un espace de noms Event Hubs

Event Hubs, en tant que service PaaS, est hautement distribué, hautement disponible et hautement évolutif.

Event Hubs est fourni avec les deux références SKU ou niveaux de tarification suivants :

- **Basique** : ce niveau est fourni avec un groupe de clients et peut conserver des messages pendant 1 journée. Il peut compter un maximum de 100 connexions négociées.
- **Standard** : ce niveau est fourni avec un maximum de 20 groupes de clients et peut conserver des messages pendant 1 journée, avec un stockage supplémentaire pendant 7 jours. Il peut compter un maximum de 1 000 connexions négociées. Il est également possible de définir des stratégies dans cette référence SKU.

La figure 12 3 montre les différentes références disponibles lors de la création d'un nouvel espace de noms Event Hubs. Celui-ci offre une option pour choisir un niveau de tarification approprié, ainsi que d'autres détails importants :

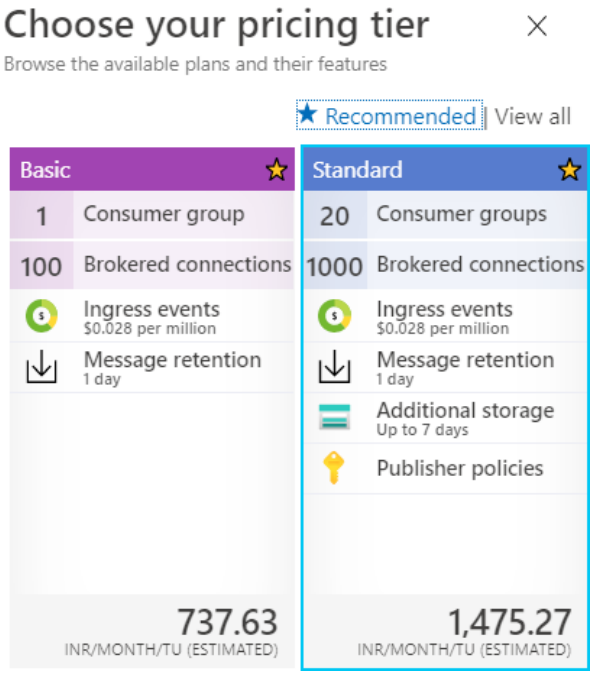


Figure 12.3 : Références Event Hubs

Le débit peut également être configuré au niveau de l'espace de noms. Les espaces de noms sont des conteneurs qui se composent de plusieurs hubs d'événements dans le même abonnement et la même région. Le débit est calculé en tant qu'**unités de débit (TU)**. Chaque TU fournit :

- Jusqu'à 1 Mo par seconde d'entrées ou un maximum de 1 000 événements d'entrées et d'opérations de gestion par seconde.
- Jusqu'à 2 Mo par seconde de sorties ou un maximum de 4 096 événements et d'opérations de gestion par seconde.
- Jusqu'à 84 Go de stockage.

Les TU peuvent varier de 1 à 20 et elles sont facturées sur une base horaire.

Il est important de noter que la référence SKU ne peut pas être modifiée après le provisionnement d'un espace de noms Event Hubs. Toute sélection d'une référence SKU demande une attention et une planification appropriées. Le processus de planification doit inclure la planification du nombre de groupes de clients requis et le nombre d'applications intéressées par la lecture des événements à partir du hub d'événement.

En outre, la SKU standard n'est pas disponible dans chaque région. Il convient de vérifier la disponibilité au moment de la conception et de la mise en œuvre du hub d'événement. L'URL permettant de vérifier la disponibilité de la région est https://azure.microsoft.com/global-infrastructure/services/?products=event-hubs.

Architecture d'Event Hubs

Il existe trois composants principaux de l'architecture Event Hubs : les **producteurs d'événements**, le **hub d'événements** et le **consommateur d'événements**, comme illustré dans le schéma suivant :

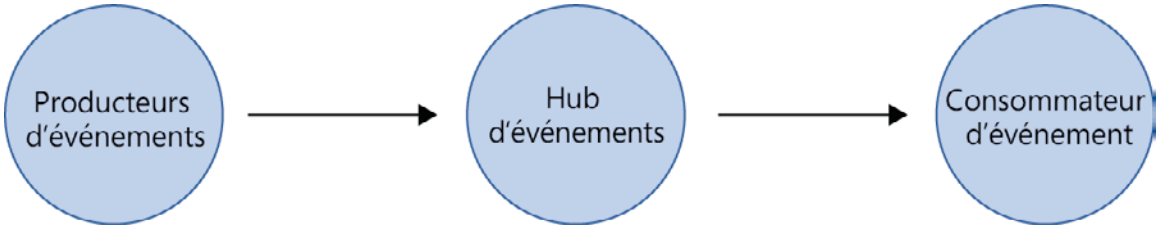


Figure 12.4 : Architecture d'Event Hubs

Les producteurs d'événements génèrent des événements et les envoient au **hub d'événements**. Le **hub d'événements** stocke les événements ingérés et fournit ces données au **consommateur d'événements**. Le **consommateur d'événements** inclut tous les éléments intéressés par ces événements et connectés au **hub d'événements** pour extraire les données.

Les hubs d'événements ne peuvent pas être créés sans espace de noms Event Hubs. L'espace de noms Event Hubs agit comme un conteneur et peut héberger plusieurs Event Hubs. Chaque espace de noms Event Hubs fournit un point de terminaison basé sur REST unique qui est consommé par les clients pour envoyer des données à Event Hubs. Cet espace de noms est le même espace de noms qui est nécessaire pour les éléments Service Bus, tels que les rubriques et les files d'attente.

La chaîne de connexion d'un espace de noms Event Hubs est composée de son URL, de son nom de stratégie et de la clé. Un exemple de chaîne de connexion est indiqué dans le bloc de code suivant :

```
Endpoint=sb://demoeventhubnsbook.servicebus.windows.
net/;SharedAccessKeyName=RootManageSharedAccessKey;SharedAccessKey=M/
E4eeBsr7DAlXcvw6ziFqlSDNbFX6E49Jfti8CRkbA=
```

Cette chaîne de connexion se trouve dans l'**élément de menu** SAS (**Shared Access Signature**) de l'espace de noms. Il peut y avoir plusieurs stratégies définies pour un espace de noms, chacune ayant des niveaux d'accès différents à ce dernier. Les trois niveaux d'accès sont les suivants :

- **Gérer** : permet de gérer le hub d'événements d'un point de vue administratif. Il a également des droits concernant l'envoi et l'écoute des événements.
- **Envoyer** : permet d'écrire des événements dans Event Hubs.
- **Écouter** : permet de lire des événements à partir d'Event Hubs.

Par défaut, la stratégie **`RootManageSharedAccessKey`** est créée lors de la création d'un hub d'événements, comme illustré à la *figure* 12.5. Les stratégies aident à créer un contrôle d'accès granulaire à Event Hubs. La clé associée à chaque stratégie est utilisée par les clients pour déterminer leur identité ; des stratégies supplémentaires peuvent également être créées avec n'importe quelle combinaison des trois niveaux d'accès mentionnés précédemment :

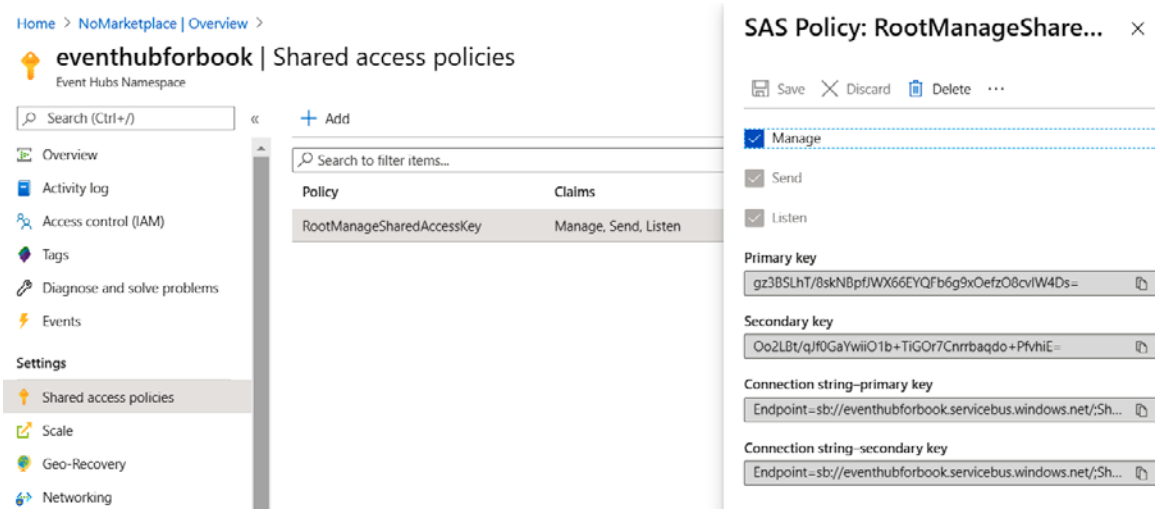


Figure 12.5 : Stratégies d'accès partagé dans Event Hubs

Les hubs d'événements peuvent être créés à partir du service d'espace de noms Event Hubs en effectuant les actions suivantes :

1. Cliquez sur **Event Hubs** dans le menu de gauche et cliquez sur **+ Hub d'événements** dans l'écran qui s'affiche ensuite :

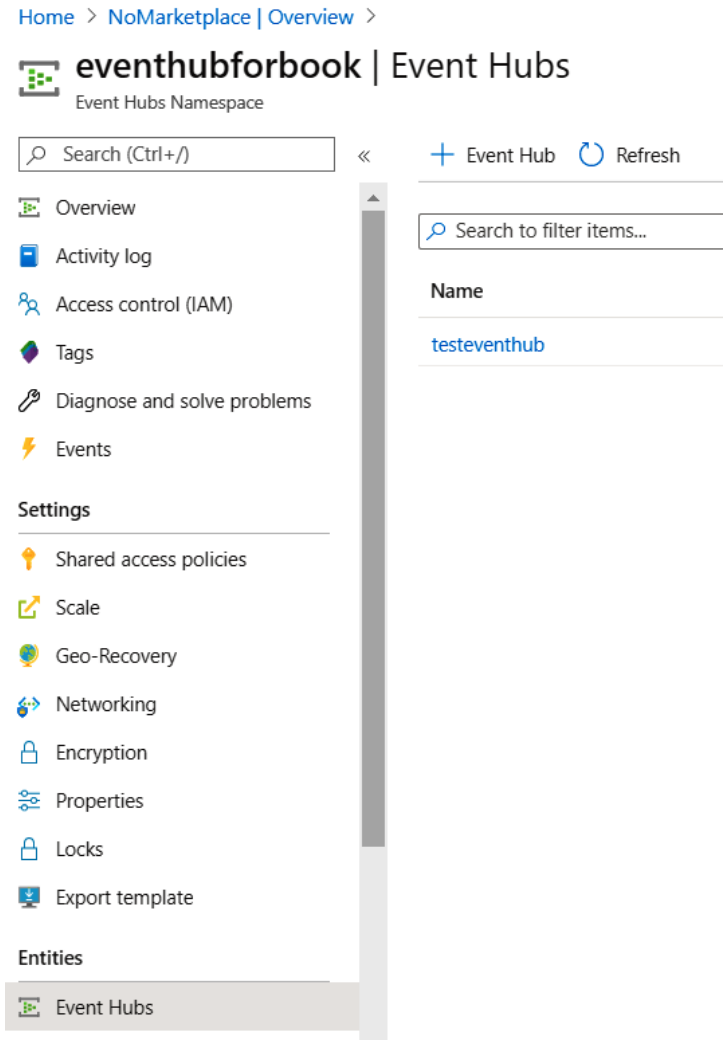


Figure 12.6 : Création d'un hub d'événements dans le portail Azure

2. Ensuite, entrez des valeurs dans les champs **Nombre de partitions** et **Rétention des messages**, avec le nom de votre choix. Ensuite, sélectionnez **Désactivé** pour **Capture**, comme illustré à la *figure 12.7* :

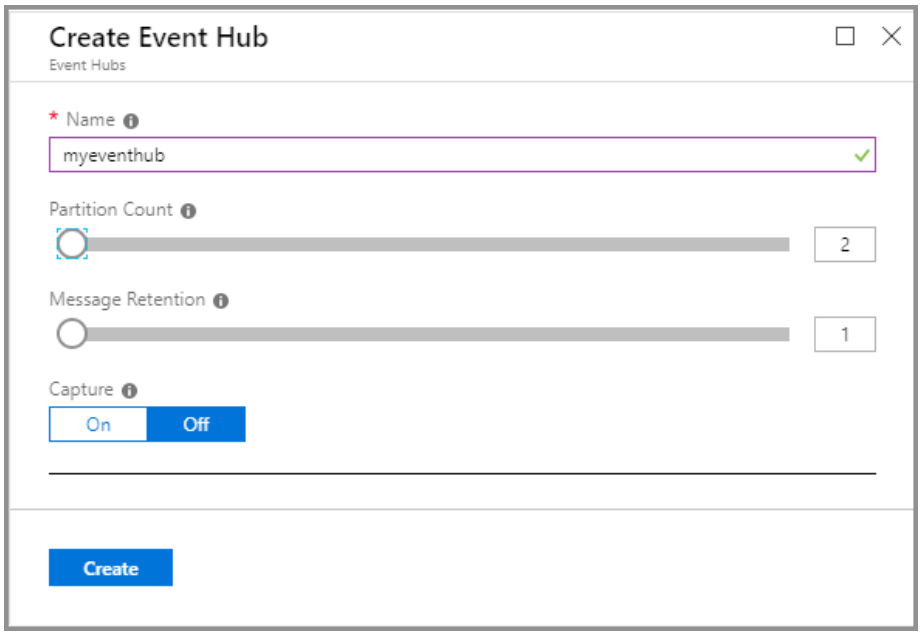


Figure 12.7 : Création d'un hub d'événements

Une fois le hub d'événements créé, vous le verrez dans la liste des hubs d'événements, comme illustré à la *figure 12.8* :

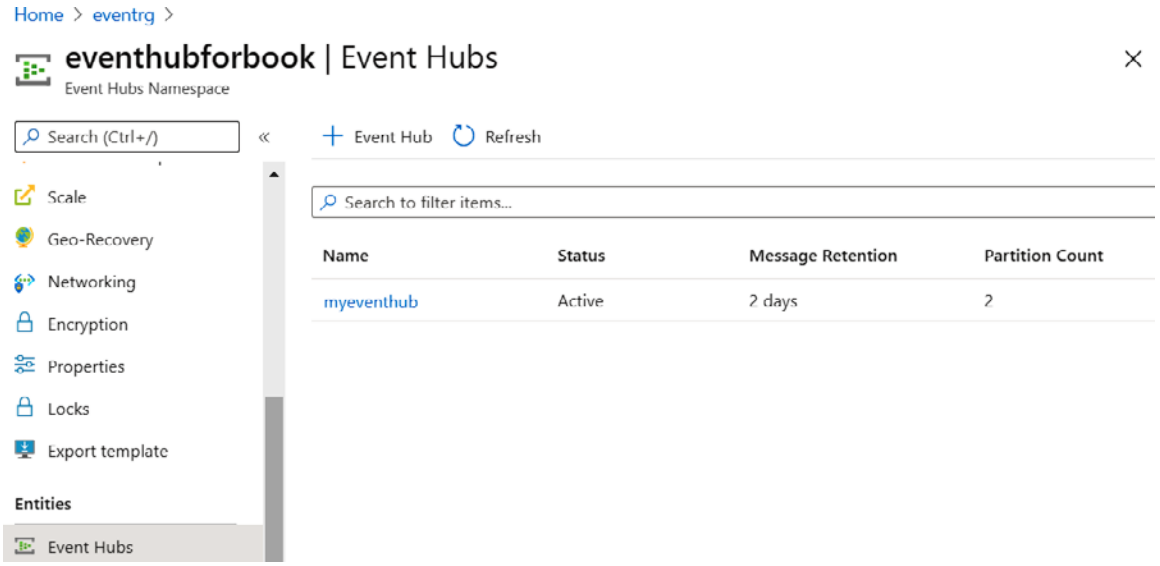


Figure 12.8 : Liste des hubs d'événements créés

Event Hubs permet également le stockage d'événements sur un compte de stockage ou un lac de données directement à l'aide d'une fonctionnalité nommée Capture.

Capture permet d'enregistrer automatiquement des données ingérées dans un compte de stockage ou un lac de données Azure. Cette fonction garantit que l'ingestion et le stockage des événements se produisent en une seule opération, et vous évite de devoir transférer les données dans le stockage séparément.

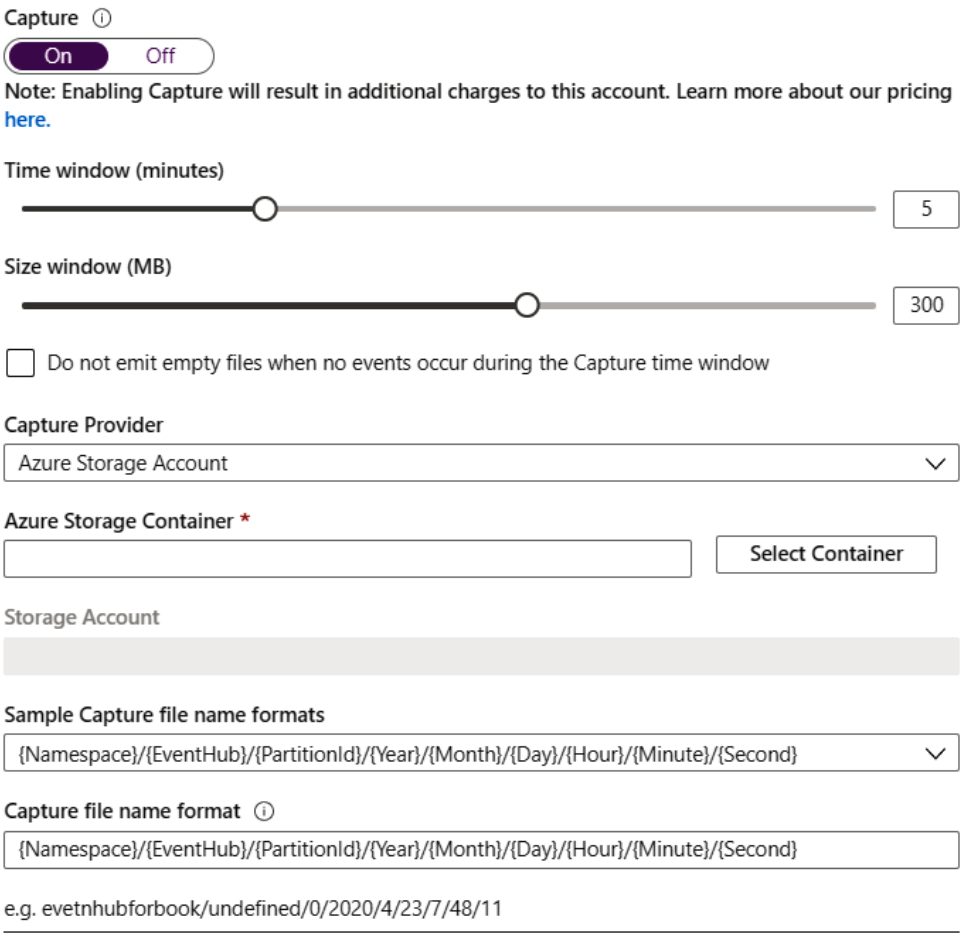


Figure 12.9 : Options de la fonction Capture

Des stratégies distinctes peuvent être attribuées à chaque concentrateur d'événements en ajoutant une nouvelle stratégie au niveau du concentrateur d'événements.

Une fois la stratégie créée, la chaîne de connexion est disponible à partir de l'élément de menu de gauche **Signature d'accès sécurisé** sur le portail Azure.

Étant donné qu'un espace de noms peut compter plusieurs hubs d'événements, la chaîne de connexion d'un hub d'événements individuel sera similaire au bloc de code suivant. La différence ici réside dans la valeur de la clé et dans l'ajout de **`EntityPath`** dans le nom du hub d'événements :

```
Endpoint=sb://azuretwittereventdata.servicebus.windows
=rxEu5K4Y2qsi5wEeOKuOvRnhtgW8xW35UBex4VlIKqg=;EntityPath=myeventhub
```

Nous avons dû laisser l'option **Capture** définie sur **Désactivé** durant la création du hub d'événements, mais celle-ci pourra être activée à nouveau une fois le hub d'événements créé. Elle permet d'enregistrer automatiquement des événements dans Azure Blob Storage ou dans un compte Azure Data Lake Storage. La configuration de la taille et de l'intervalle de temps est indiquée à la *figure 12.10* :

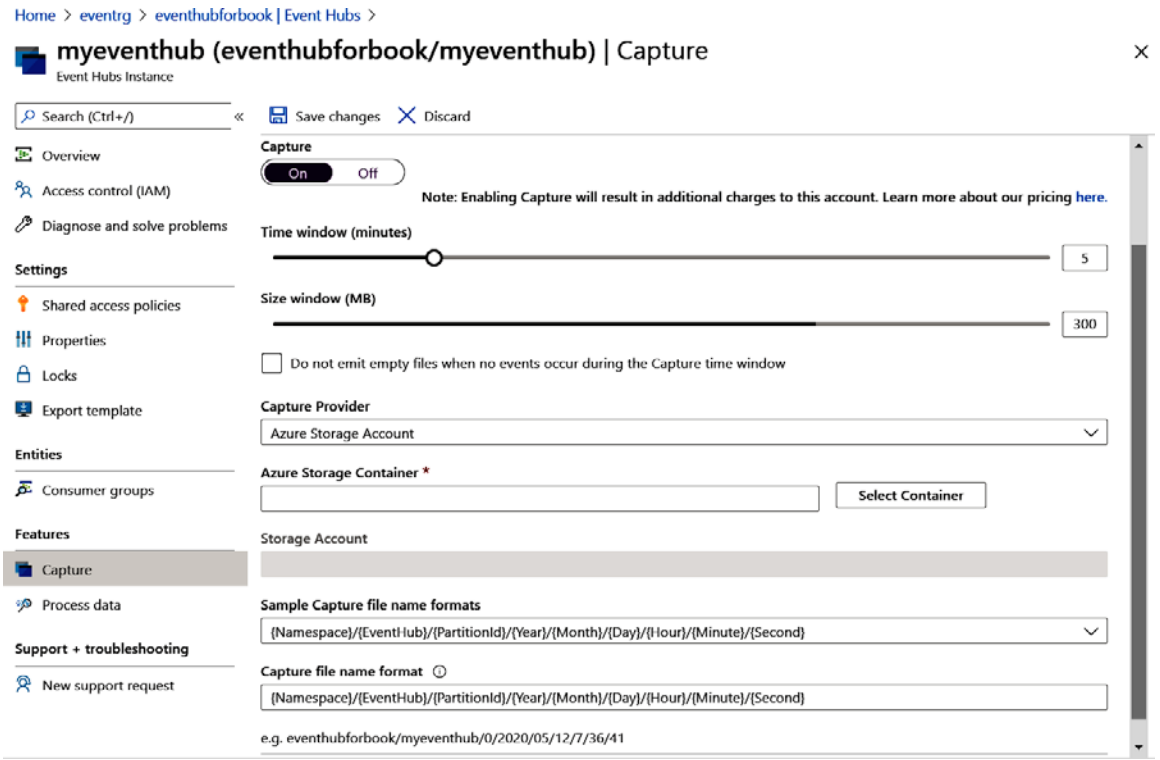


Figure 12.10 : Sélection de la taille et des intervalles de temps pour la capture d'événements

Nous n'avons pas couvert les concepts de partitions et les options de rétention des messages lors de la création des hubs d'événements.

Le partitionnement est un concept important lié à l'évolutivité de n'importe quel magasin de données. Les événements sont conservés dans les hubs d'événements pendant une période spécifique. Si tous les événements sont stockés dans le même magasin de données, il devient extrêmement difficile de dimensionner ce magasin de données. Chaque producteur d'événements se connectera au même magasin de données et y enverra ses événements. Comparez ceci avec un magasin de données qui peut partitionner les mêmes données dans plusieurs magasins de données plus petits, chacun étant identifié de manière unique par une valeur.

Le magasin de données plus petit est nommé **partition**, et la valeur qui définit la partition est connue sous le nom de **clé de partition**. Cette clé de partition fait partie des données d'événement.

Maintenant, les producteurs d'événements peuvent se connecter à l'Event Hub, et en fonction de la valeur de la clé de partition, le Hub d'événements stockera les données dans une partition appropriée. Cela permettra au hub d'événements d'ingérer plusieurs événements en même temps.

Le choix du nombre de partitions est un aspect crucial pour l'évolutivité d'un hub d'événements. La *figure 12.11* montre que les données ingérées sont stockées dans la partition appropriée en interne par Event Hubs à l'aide de la clé de partition :

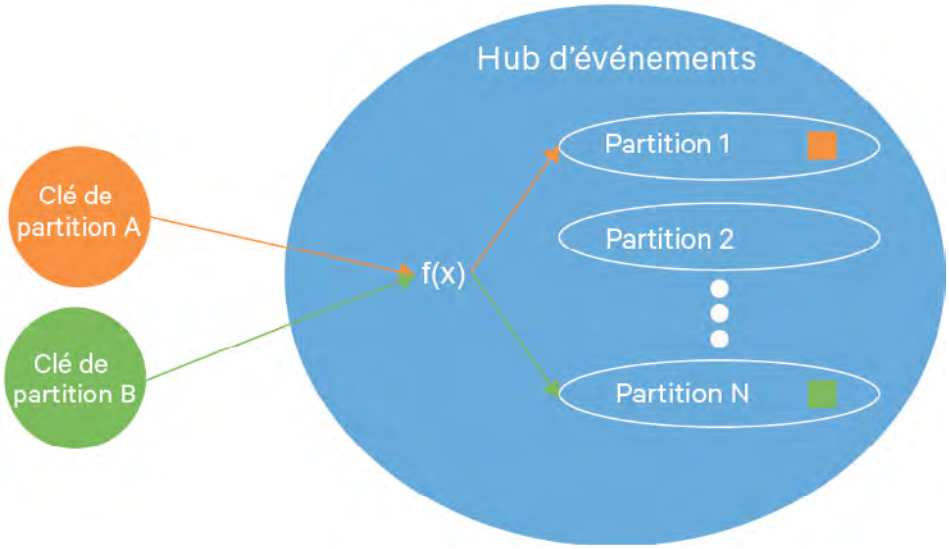


Figure 12.11 : Partitionnement dans un hub d'événements

Il est important de comprendre que la même partition peut avoir plusieurs clés. L'utilisateur décide du nombre de partitions requises et le hub d'événements détermine en interne la meilleure façon d'allouer les clés de partition entre elles. Chaque partition stocke les données de manière ordonnée à l'aide d'un horodatage, et les événements plus récents sont ajoutés vers la fin de la partition.

Il est important de noter qu'il n'est pas possible de modifier le nombre de partitions une fois que le Hub d'événements est créé.

Il est également important de se rappeler que les partitions permettent également d'assurer le parallélisme et la simultanéité pour les applications qui lisent les événements. Par exemple, s'il existe 10 partitions, 10 lecteurs parallèles peuvent lire les événements sans aucune dégradation des performances.

La rétention des messages fait référence à la période pendant laquelle les événements doivent être stockés. Après l'expiration de la période de rétention, les événements sont ignorés.

Groupes de clients

Les clients sont des applications qui lisent des événements à partir d'un hub d'événements. Des groupes de clients sont créés pour que ces derniers se connectent afin de lire les événements. Il peut y avoir plusieurs groupes de clients pour un hub d'événements et chaque groupe de clients a accès à toutes les partitions de celui-ci. Chaque groupe de clients forme une requête sur les événements dans les Hubs d'événements. Les applications peuvent utiliser des groupes de clients et chaque application obtiendra une vue différente des événements de Hub d'événements. Un groupe de clients **`$default`** est créé lors de la création d'un hub d'événements. Une bonne pratique consiste à associer un client à un groupe de clients afin de garantir une performance optimale. Cependant, il est possible d'avoir cinq lecteurs sur chaque partition dans un groupe de clients :

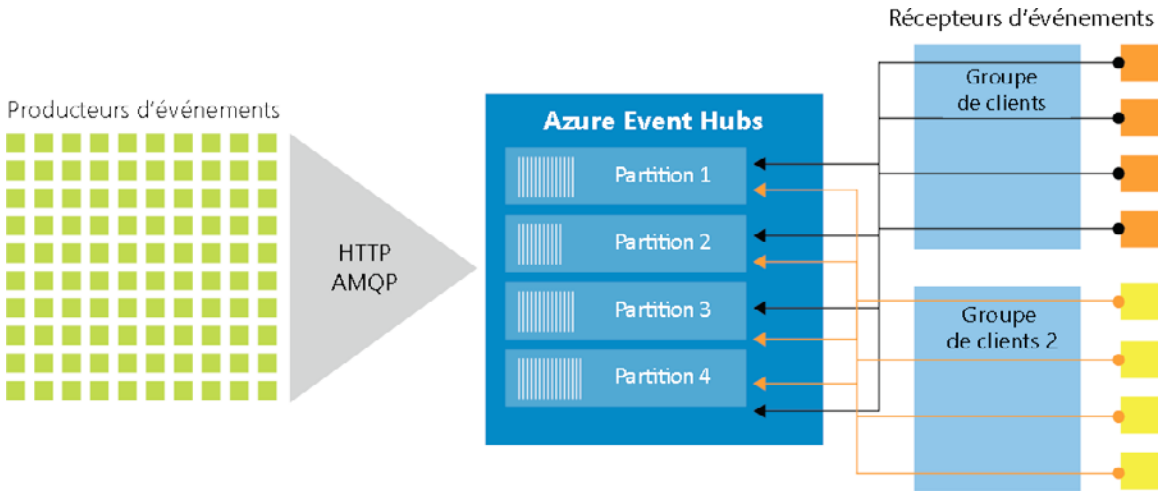


Figure 12.12 : Récepteurs d'événements dans un groupe de clients

Maintenant que vous comprenez ce que sont les groupes de clients, il est temps de vous présenter le concept de débit d'Event Hubs.

Débit

Les partitions aident à garantir l'évolutivité, tandis que le débit permet d'assurer la capacité par seconde. Que signifie la capacité dans le cadre d'Event Hubs ? Il s'agit de la quantité de données qui peuvent être gérées par seconde.

Dans Event Hubs, une unité de débit unique permet les opérations suivantes :

- 1 Mo de données d'ingestion par seconde ou 1 000 événements par seconde (selon la première éventualité)
- 2 Mo de données d'ingestion par seconde ou 4 096 événements par seconde (selon la première éventualité)

L'option d'inflation automatique permet d'augmenter le débit automatiquement si le nombre d'événements entrants/sortants ou la taille totale entrante/sortante franchit un seuil. Au lieu de se bloquer, le débit va se dimensionner. La configuration du débit au moment de la création de l'espace de noms est illustrée à la *figure 12.13*. Encore une fois, toute décision concernant les unités de débit doit faire l'objet d'une réflexion minutieuse :

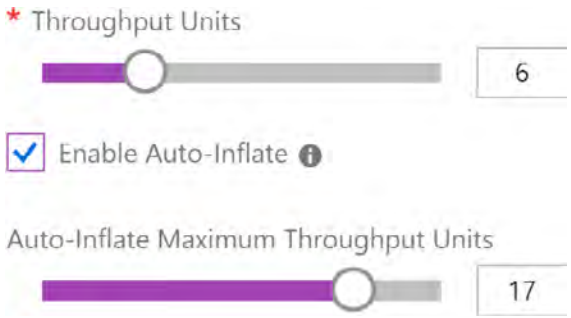


Figure 12.13 : Sélection des unités de débit avec l'inflation automatique

Présentation des bases de Stream Analytics

Event Hubs est une plateforme de streaming de données hautement évolutive, c'est pourquoi nous avons besoin d'un autre service capable de traiter ces événements comme un flux plutôt que comme des données stockées. Stream Analytics aide à traiter et à examiner un flux de Big data, et les tâches Stream Analytics permettent d'exécuter le traitement des événements.

Stream Analytics peut traiter des millions d'événements par seconde et sa mise en route est assez simple. Azure Stream Analytics est un PaaS entièrement géré par Azure. Les clients de Stream Analytics n'ont pas à gérer le matériel et la plateforme sous-jacents.

Chaque travail comprend plusieurs entrées, sorties et une requête, qui transforme les données entrantes en nouvelle sortie. Toute l'architecture de Stream Analytics est illustrée à la *figure 12.14* :

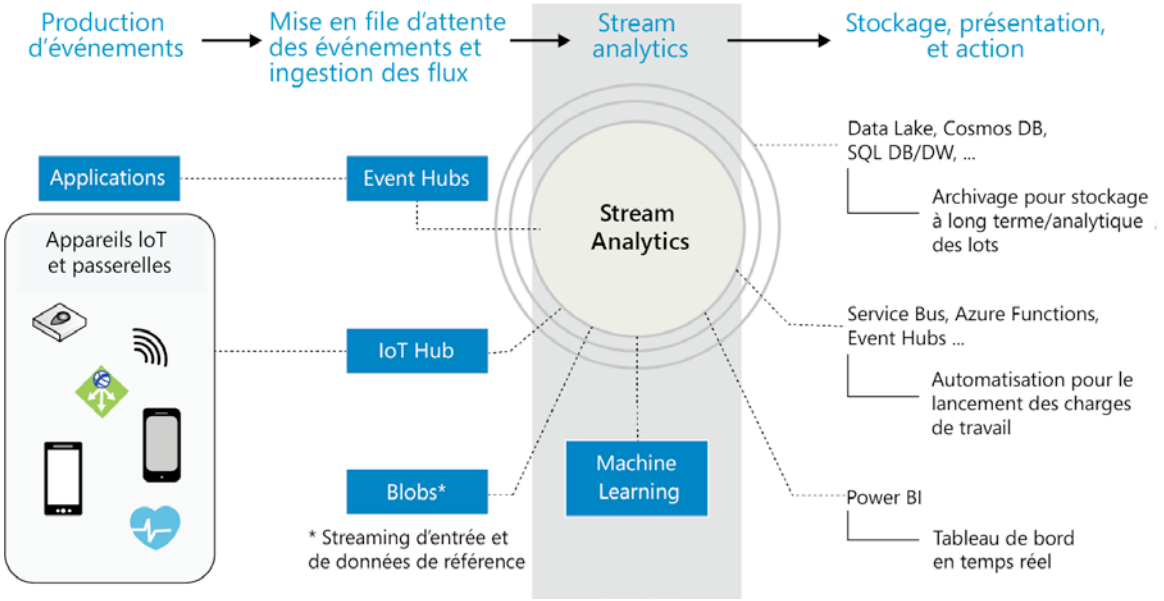


Figure 12.14 : Architecture d'Azure Stream Analytics

À la *figure 12.14*, les sources d'événements sont affichées à l'extrême gauche. Ce sont les sources qui produisent les événements. Il peut s'agir d'appareils IdO, d'applications personnalisées écrites dans n'importe quel langage de programmation, ou d'événements provenant d'autres plateformes Azure, telles que Log Analytics ou Application Insights.

Ces événements doivent d'abord être ingérés dans le système, et il existe de nombreux services Azure qui peuvent aider à ingérer ces données. Nous avons déjà examiné les Event Hubs et comment ces derniers facilitent l'ingestion des données. Il existe d'autres services, tels que IoT Hub, qui aident également à l'ingestion de données spécifiques aux appareils et aux capteurs. IoT Hub et l'ingestion sont abordés en détail dans le *chapitre 11, Conception de solutions IoT*. Ces données ingérées subissent le traitement au fur et à mesure qu'elles parviennent à un flux, et ce traitement est effectué à l'aide de Stream Analytics. La sortie de Stream Analytics pourrait être une plateforme de présentation telle que Power BI, qui montre des données en temps réel aux parties prenantes, ou une plateforme de stockage telle que Cosmos DB, Data Lake Storage ou Azure Storage, à partir de laquelle les données peuvent être lues et traitées ultérieurement par Azure Functions et les files d'attente Service Bus.

Stream Analytics permet de recueillir des informations issues de données ingérées en temps réel pendant une période spécifique et d'identifier des modèles.

Il le fait à travers trois tâches différentes :

- **Entrée** : les données doivent être ingérées dans le processus d'analyse. Les données peuvent provenir d'Event Hubs, d'IoT Hub ou du stockage Azure Blob. Plusieurs entrées de référence séparées à l'aide d'un compte de stockage et d'une base de données SQL peuvent être utilisées pour les données de recherche dans les requêtes.
- **Requête** : c'est là que Stream Analytics effectue la tâche principale qui consiste à analyser les données ingérées et à extraire des informations et des modèles significatifs. Pour cela, il utilise des fonctions JavaScript définies par l'utilisateur, des agrégats JavaScript définis par l'utilisateur, Azure Machine Learning et Azure Machine Learning Studio.
- **Sortie** : le résultat des requêtes peut être envoyé à plusieurs types de destinations différents, les plus importants étant Cosmos DB, Power BI, Synapse Analytics, Data Lake Storage et Functions :

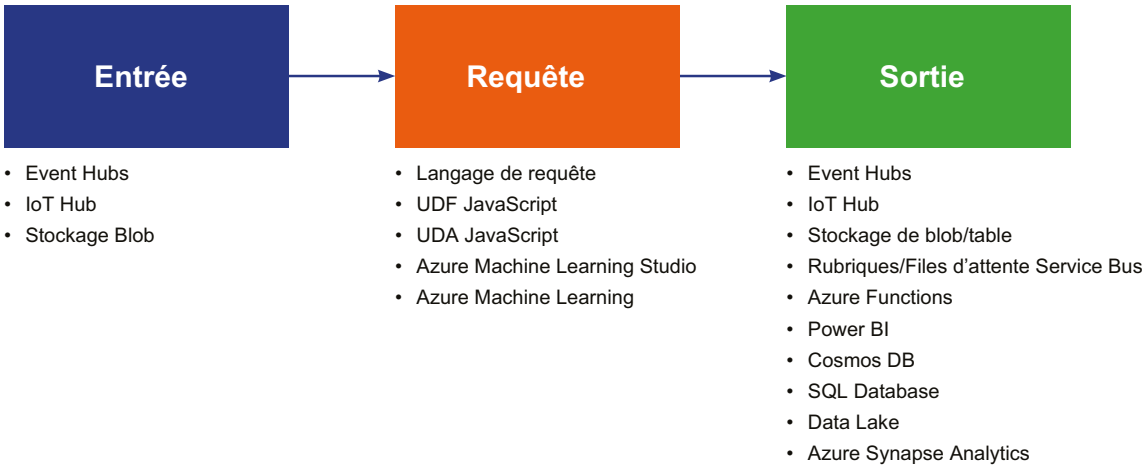


Figure 12.15 : Processus Stream Analytics

Stream Analytics est capable d'ingérer des millions d'événements par seconde et peut exécuter des requêtes sur ces derniers.

Les données d'entrée sont prises en charge dans l'un des trois formats suivants :

- **JavaScript Object Notation (JSON)** : il s'agit d'un format léger, à base de texte clair, lisible par l'homme. Il se compose de paires nom-valeur. Voici un exemple d'événement JSON :

```
{
 "SensorId" : 2,
 "humidity" : 60,
 "temperature" : 26C
}
```

- **Valeurs séparées par des virgules (CSV)** : il s'agit également de valeurs en texte brut, qui sont séparées par des virgules. La *figure 12.16* présente un exemple de CSV. La première ligne est l'en-tête contenant trois champs suivis de deux lignes de données :

```
SensorID, humidity, temperature
2,60,26C
3,65,31C
```

Figure 12.16 : Valeurs en clair

- **Avro** : ce format est similaire à JSON ; cependant, il est stocké dans un format binaire plutôt que dans un format texte :

```
{
      "firstname": "Ritesh",
   "lastname": "Modi",
      "email": "ritesh.modi@outlook.com"
}
```

Toutefois, cela ne signifie pas que Stream Analytics peut ingérer les données uniquement à l'aide de ces trois formats. Il peut également créer des désérialiseurs basés sur .NET personnalisés, à l'aide desquels tout format de données peut être ingéré, selon la mise en œuvre des désérialiseurs. Les étapes que vous pouvez suivre pour écrire un désérialiseur personnalisé sont disponibles à l'adresse https://docs.microsoft.com/azure/stream-analytics/custom-deserializer-examples.

Non seulement Stream Analytics reçoit des événements, mais il fournit également une fonction de requête avancée pour les données qu'il reçoit. Les requêtes peuvent extraire des informations importantes des flux de données temporelles et les générer.

Comme illustré à la *figure 12.17*, il y a un jeu de données d'entrée et un jeu de données de sortie ; la requête déplace les événements de l'entrée vers la sortie. La clause **`INTO`** fait référence à l'emplacement de sortie et la clause **`FROM`** fait référence à l'emplacement d'entrée. Les requêtes sont fortement similaires aux requêtes SQL, ce qui signifie que la courbe d'apprentissage n'est pas trop complexe pour les programmeurs SQL :

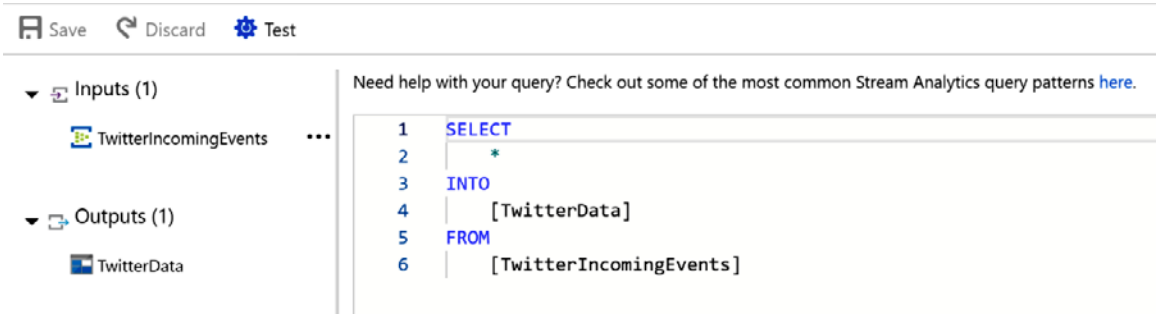


Figure 12.17 : Requête Stream Analytics pour recevoir des données Twitter

Event Hubs fournit des mécanismes pour l'envoi de sorties à partir de requêtes vers des destinations cibles. Au moment de l'écriture de ce manuel, Stream Analytics prend en charge plusieurs destinations pour les événements et les sorties de requête, comme illustré précédemment.

Il est également possible de définir des fonctions personnalisées qui peuvent être réutilisées dans les requêtes. Quatre options sont proposées pour définir des fonctions personnalisées.

- Azure Machine Learning
- Fonctions JavaScript définies par l'utilisateur
- Agrégats JavaScript définis par l'utilisateur
- Azure Machine Learning Studio

L'environnement d'hébergement

Les tâches Stream Analytics peuvent être exécutées sur des hôtes exécutés sur le Cloud ou susceptibles d'être exécutés sur des appareils de périmètre IoT. Les appareils de périmètre IoT sont des appareils proches des capteurs IoT, plutôt que sur le Cloud. La *figure 12.18* illustre le volet **Nouvelle tâche Stream Analytics** :

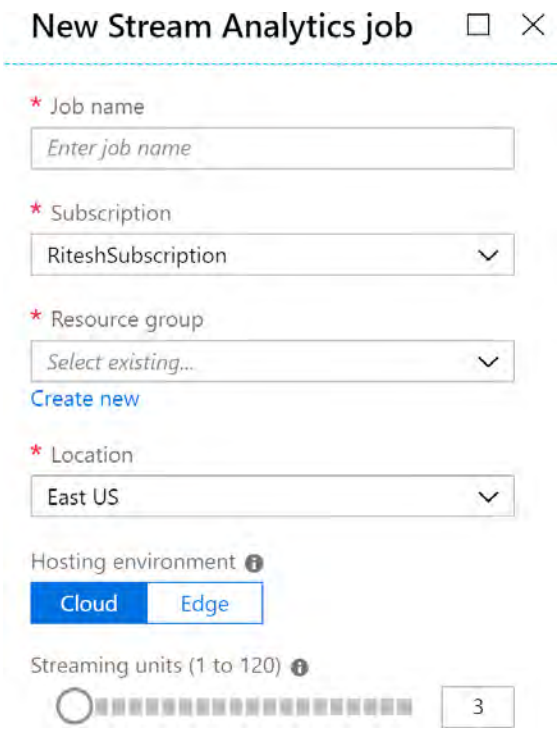


Figure 12.18 : Création d'une tâche Stream Analytics

Examinons en détail les unités de streaming.

Unités de streaming

Sur la *figure 12.18*, vous pouvez voir que les unités de streaming sont la seule configuration propre à Stream Analytics. Les unités de streaming font référence aux ressources (UC et mémoire) affectées à l'exécution d'une tâche Stream Analytics. Les unités de streaming minimales et maximales sont respectivement 1 et 120.

Les unités de streaming doivent être préattribuées en fonction de la quantité de données et du nombre de requêtes exécutées sur ces données, faute de quoi la tâche échouera.

Il est possible de redimensionner les unités de streaming à partir du portail Azure.

Exemple d'application utilisant Event Hubs et Stream Analytics

Dans cette section, nous allons créer un exemple d'application comprenant plusieurs services Azure, notamment Azure Logic Apps, Azure Event Hubs, Azure Storage et Azure Stream Analytics.

Dans cet exemple d'application, nous allons lire tous les tweets contenant le mot « Azure » et les stocker dans un compte de stockage Azure.

Pour créer cette solution, nous devons d'abord configurer toutes les ressources nécessaires.

Provisionnement d'un nouveau groupe de ressources

Accédez au portail Azure, connectez-vous avec vos identifiants valides et cliquez sur **+ Créer une ressource**, puis recherchez le **Groupe de ressources**. Sélectionnez **Resource group** (Groupe de ressources) dans les résultats de la recherche et créez un groupe de ressources. Indiquez ensuite un nom et choisissez un emplacement approprié. Notez que toutes les ressources doivent être hébergées dans le même groupe de ressources et le même emplacement afin de faciliter leur suppression :

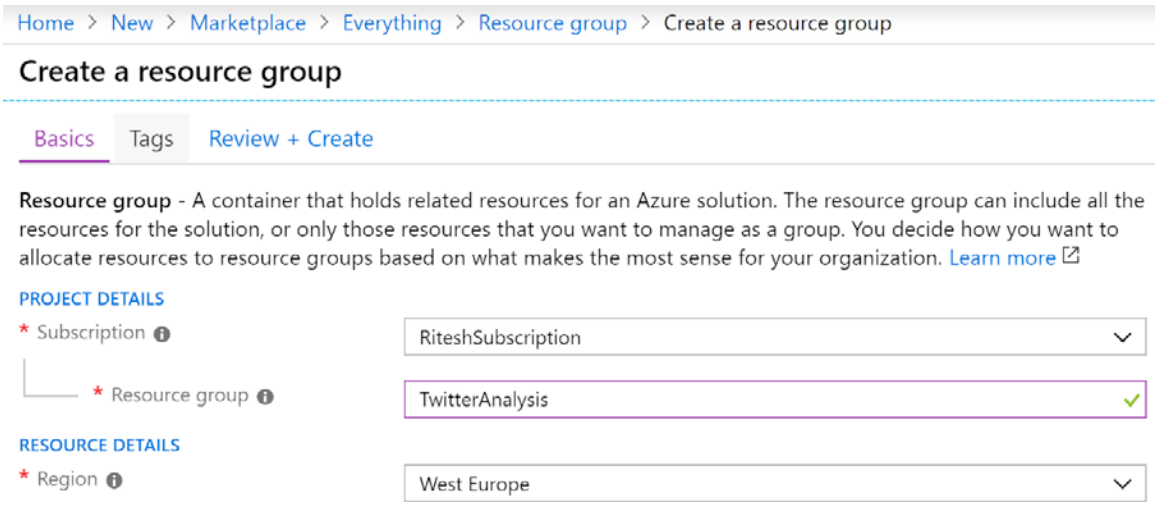


Figure 12.19 : Provisionnement d'un nouveau groupe de ressources dans le portail Azure

Ensuite, nous allons créer un espace de noms Event Hubs.

Création d'un espace de noms Event Hubs

Cliquez sur **+ Créer une ressource** et recherchez **Event Hubs**. Sélectionnez **Event Hubs** dans les résultats de la recherche et créez un nouveau hub d'événements. Indiquez ensuite un nom et un emplacement, puis sélectionnez un abonnement basé sur le groupe de ressources créé précédemment. Sélectionnez **Standard** comme niveau de tarification et sélectionnez **Activer l'inflation automatique**, comme illustré à la *figure 12.20* :

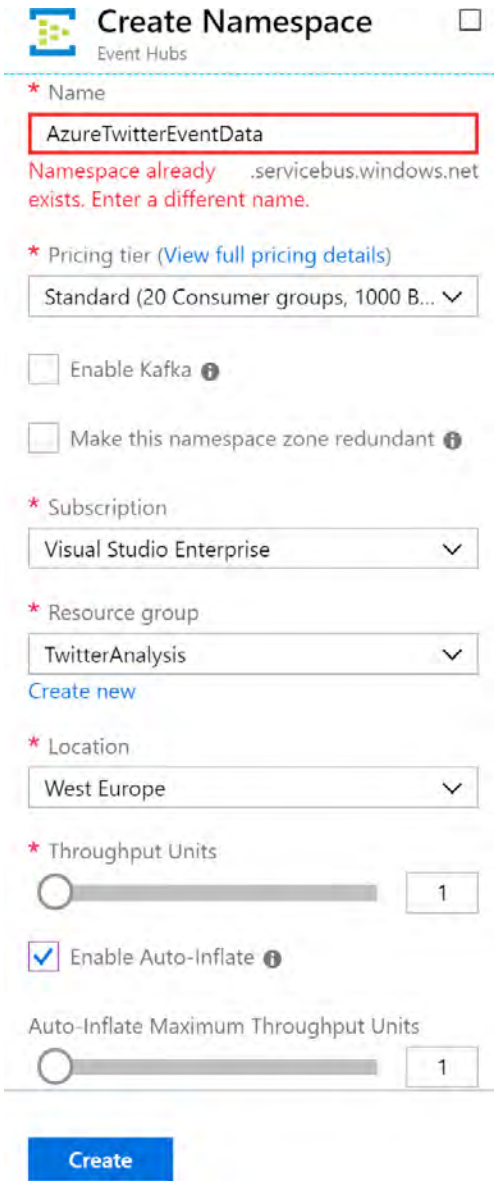


Figure 12.20 : Création d'un espace de noms Event Hubs

À ce stade, un espace de noms Event Hubs a théoriquement été créé. Il s'agit d'une condition préalable à l'existence d'un espace de noms avant qu'un hub d'événements puisse être créé. L'étape suivante consiste à configurer un hub d'événements.

Création d'un hub d'événements

À partir du service d'espace de noms Event Hubs, cliquez sur **Events Hubs** dans le menu de gauche, puis cliquez sur **+ Hubs d'événements** pour créer un nouveau hub d'événements. Nommez-le **azuretwitterdata** et fournissez un nombre optimal de partitions et une valeur pour **Rétention de message** :

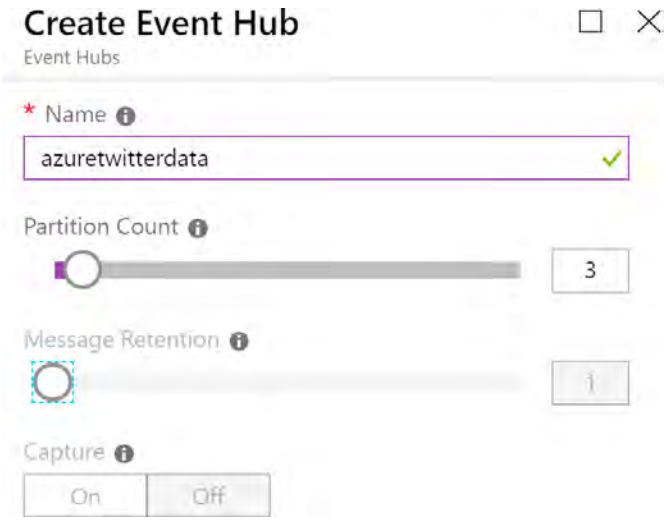


Figure 12.21 : Création du hub d'événements azuretwitterdata

Après cette opération, vous disposerez d'un hub d'événements qui pourra être utilisé pour envoyer des données d'événement, stockées dans un stockage durable, par exemple un lac de données ou un compte de stockage Azure, qui sera utilisé par les services en aval.

Configuration d'une application logique

Une fois le groupe de ressources configuré, cliquez sur **+ Créer une ressource** et recherchez **Logic Apps**. Sélectionnez **Logic Apps** dans les résultats de la recherche et créez une nouvelle application logique. Indiquez ensuite un nom et un emplacement, puis sélectionnez un abonnement basé sur le groupe de ressources créé précédemment. Il est recommandé d'activer **Log Analytics**. Logic Apps est abordé plus en détail dans le *chapitre 11, Solutions Azure utilisant Azure Logic Apps, Event Grid et Functions*. L'application logique est responsable de la connexion à Twitter en utilisant un compte et de la récupération de tous les tweets contenant le terme **Azure** :

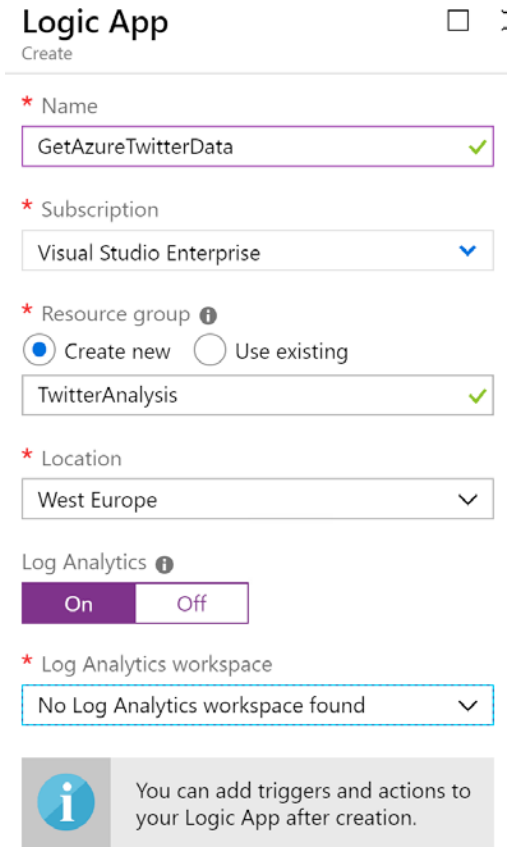


Figure 12.22 : Création d'une application logique

Une fois l'application logique créée, sélectionnez le déclencheur **Quand un nouveau Tweet est publié** sur la surface de conception, connectez-vous et configurez-la comme illustré à la *figure* 12.23. Vous aurez besoin d'un compte Twitter valide pour configurer ce déclencheur :

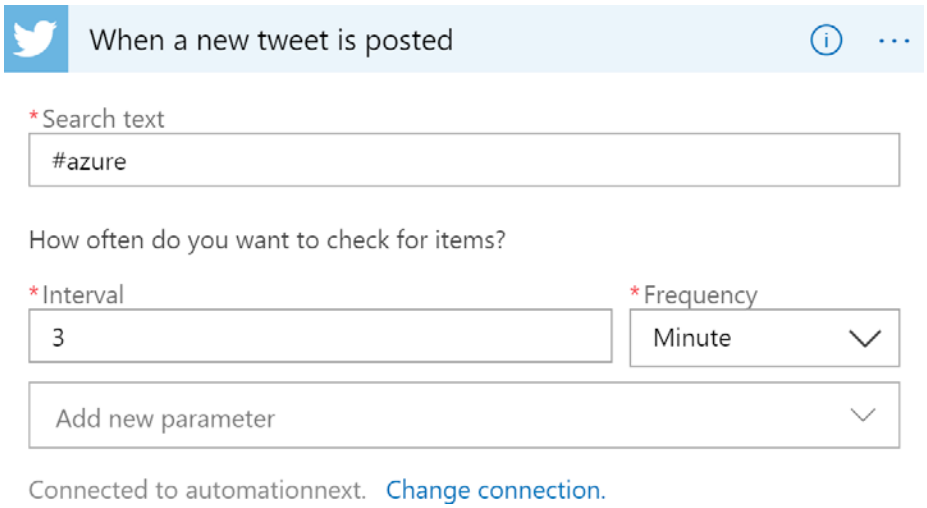


Figure 12.23 : Configuration de la fréquence des tweets entrants

Ensuite, déposez une action **Événement d'envoi** sur la surface de conception. Cette action est responsable de l'envoi de tweets au hub d'événements :

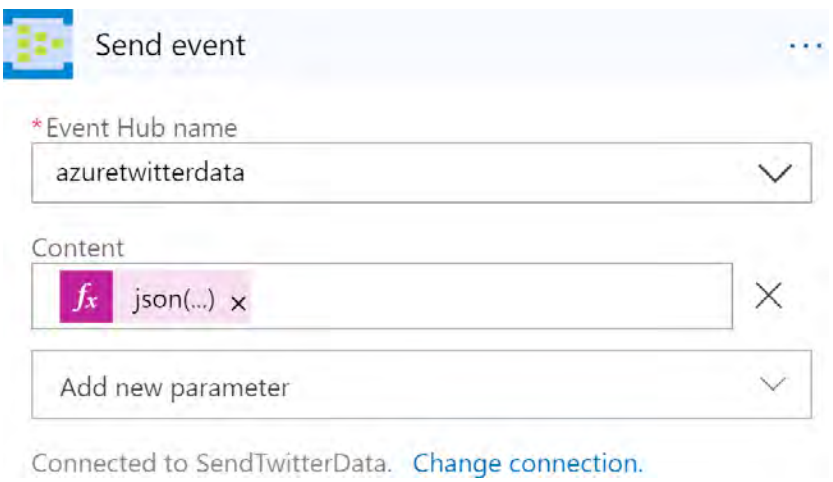


Figure 12.24 : Ajout d'une action pour envoyer des tweets au hub d'événements

Sélectionnez le nom du hub d'événements créé au cours d'une étape antérieure.

La valeur spécifiée dans la zone de texte Contenu est une expression composée dynamiquement à l'aide des fonctions Logic Apps fournies et des données Twitter. Cliquez sur **Ajouter un contenu dynamique** pour afficher une boîte de dialogue afin de composer l'expression :

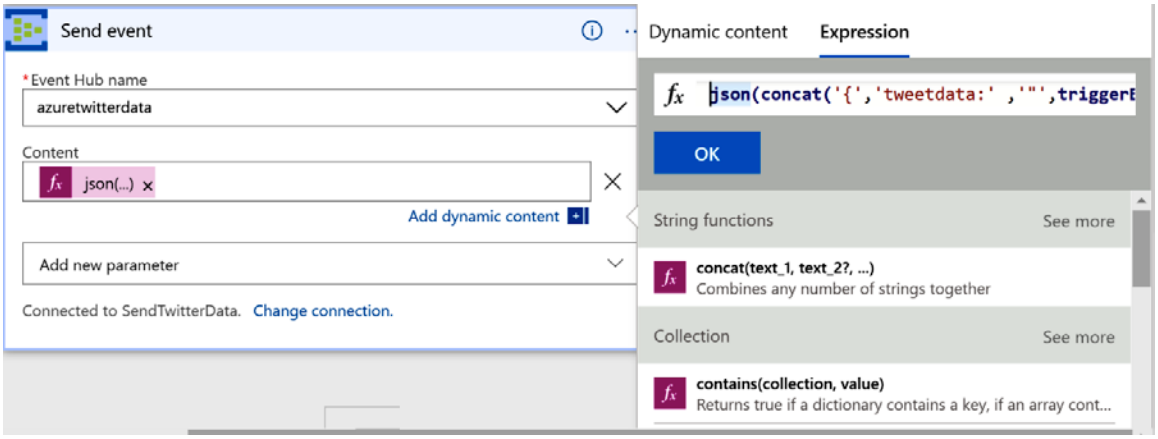


Figure 12.25 : Configuration de l'activité Logic Apps à l'aide d'expressions dynamiques

La valeur de l'expression est la suivante :

```
json(concat('{','tweetdata:' ,'"',triggerBody()?['TweetText'],'"', '}'))
```

À la section suivante, nous allons configurer le compte de stockage.

Configuration d'un compte de stockage

Cliquez sur **+ Créer une ressource** et recherchez **Compte de stockage**. Sélectionnez **Compte de stockage** dans les résultats de la recherche, puis créez un nouveau compte de stockage. Indiquez ensuite un nom et un emplacement, puis sélectionnez un abonnement basé sur le groupe de ressources créé précédemment. En dernier lieu, sélectionnez **StorageV2** pour **Type de compte**, **Standard** pour **Performance** et **Stockage localement redondant** (**LRS**) pour le champ **Réplication**.

Ensuite, nous allons créer un conteneur de stockage Blob pour stocker les données qui sortent de Stream Analytics.

Création d'un conteneur de stockage

Stream Analytics va générer les données en tant que fichiers, qui seront stockés dans un conteneur de stockage d'objets Blob. Un conteneur nommé **twitter** est créé dans le stockage d'objets Blob, comme illustré à la *figure 12.26* :

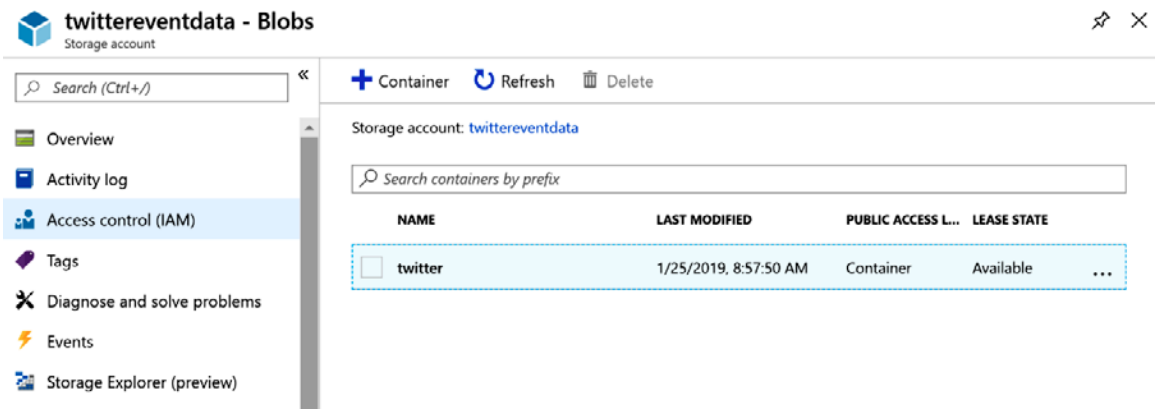


Figure 12.26 : Création d'un conteneur de stockage

Créons une nouvelle tâche Stream Analytics avec un environnement d'hébergement dans le Cloud et configurons les unités de diffusion avec les paramètres par défaut.

Création de tâches Stream Analytics

L'entrée pour cette tâche Stream Analytics provient du hub d'événements, et nous devons donc la configurer dans le menu **Entrées** :

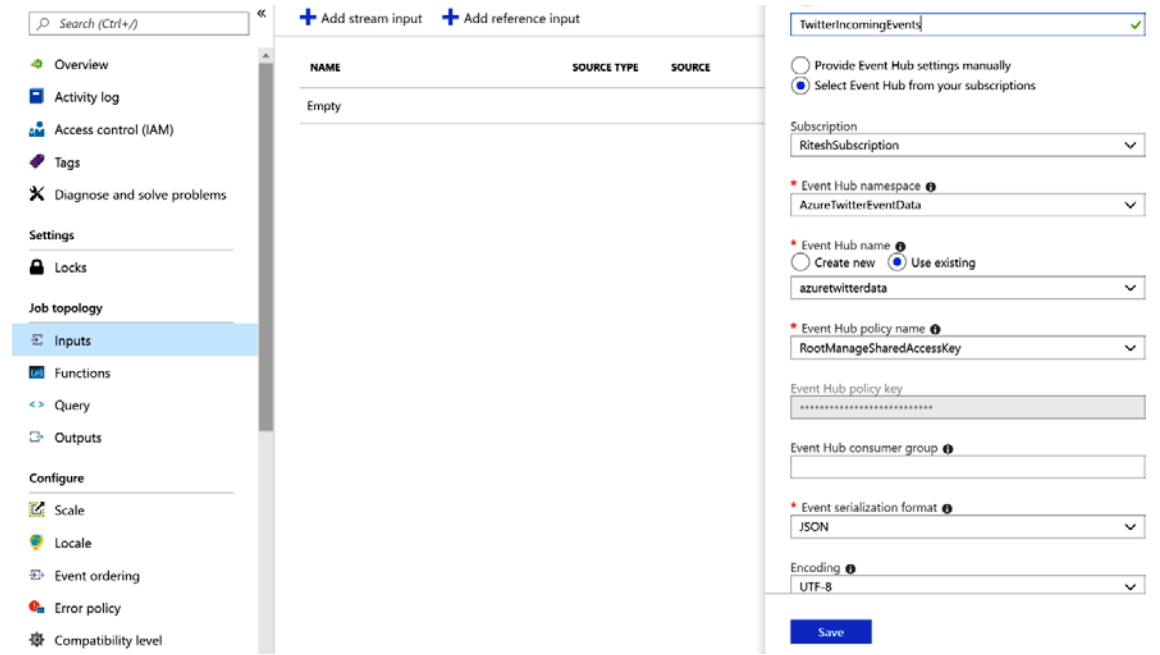


Figure 12.27 : Création d'une tâche Stream Analytics entrante

La sortie pour la tâche Stream Analytics est un compte de stockage de grands objets binaires, nous devons donc configurer la sortie en conséquence. Fournissez un modèle de chemin d'accès qui convient à cet exercice. Nous utilisons par exemple le modèle de chemin d'accès **{datetime:ss}** :

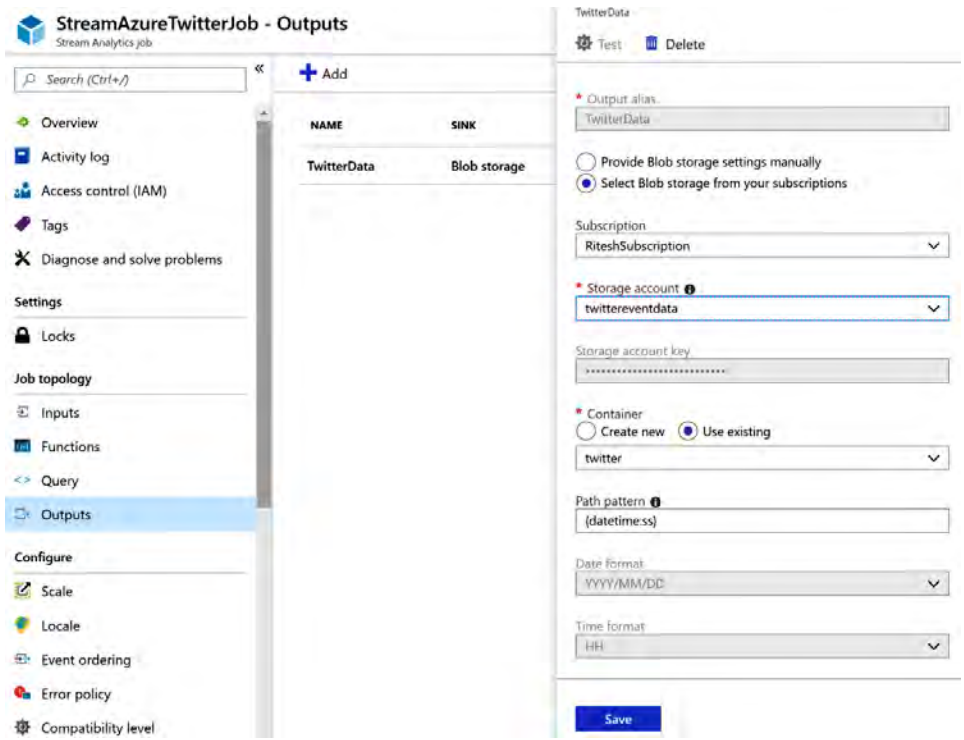


Figure 12.28 :Création d'un compte de stockage Blob en sortie

La requête est assez simple ; vous copiez simplement les données de l'entrée à la sortie :

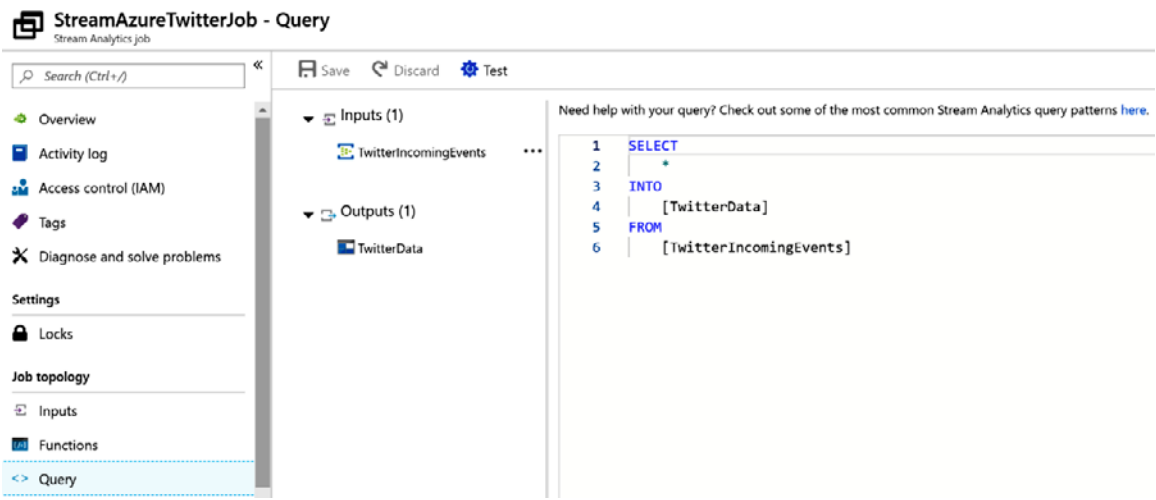


Figure 12.29 : Requête concernant la copie des flux Twitter

Bien que cet exemple implique uniquement la copie de données, il peut y avoir des requêtes plus complexes pour effectuer la transformation avant de charger des données dans une destination.

Cela conclut toutes les étapes de l'application. Vous devriez maintenant être en mesure de l'exécuter.

Exécution de l'application

L'application logique doit être activée et Stream Analytics doit être exécuté. Maintenant, exécutez l'application logique. Cela créera une tâche pour exécuter toutes les activités au sein de l'application, comme illustré à la *figure* 12.30 :

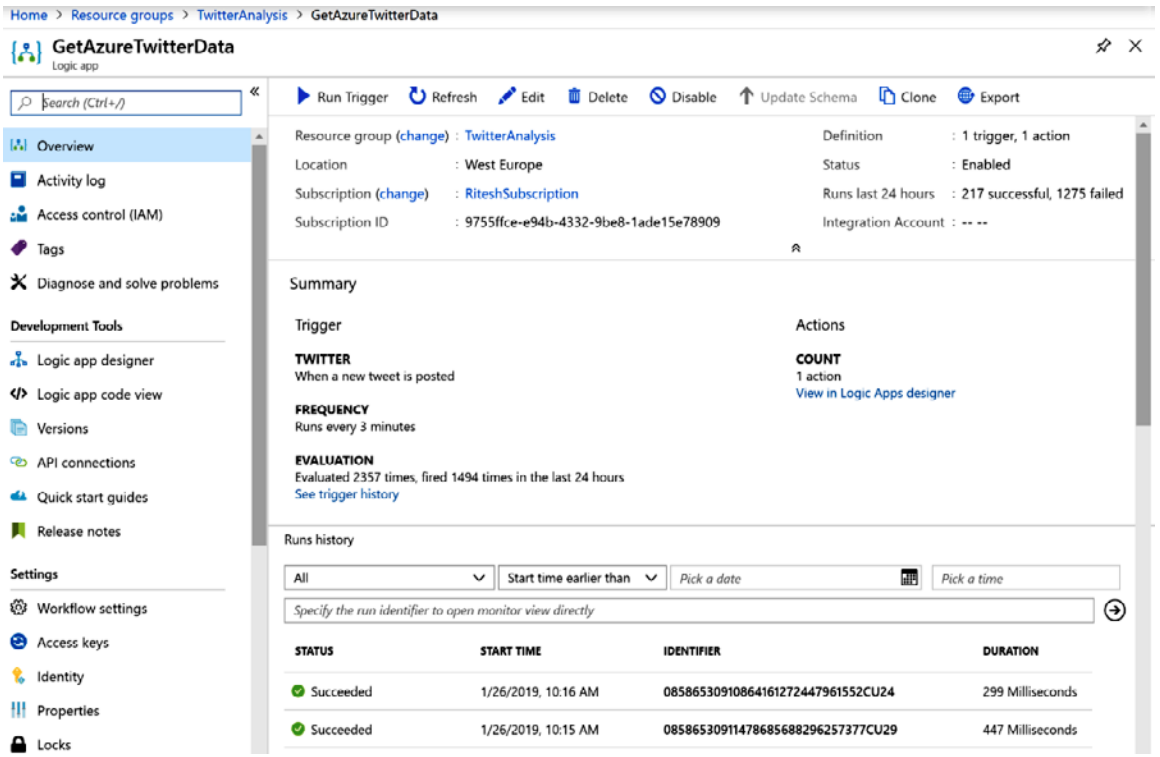


Figure 12.30 : Présentation de l'application GetAzureTwitterData

Le conteneur **Compte de stockage** doit obtenir des données, comme illustré à la *figure 12.31* :

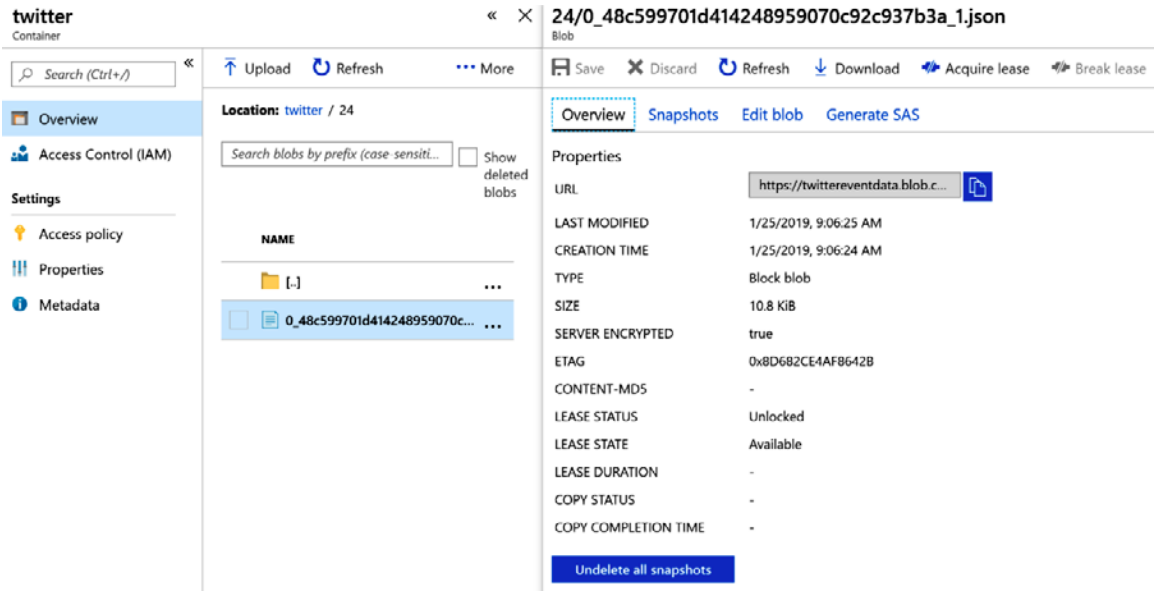


Figure 12.31 : Vérification des données du conteneur Compte de stockage

À titre d'exercice, vous pouvez étendre cet exemple de solution et évaluer le sentiment des tweets toutes les trois minutes. La charge de travail Logic Apps pour un tel exercice serait la suivante :

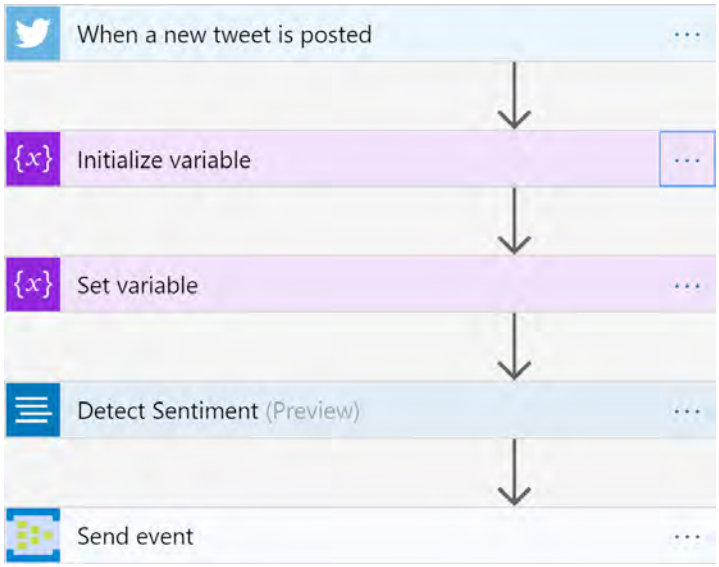


Figure 12.32 : Organigramme dédié à l'analyse des sentiments issus des tweets

Pour détecter les sentiments, vous devez utiliser l'API d'analyse de texte, qui doit être configurée avant d'être utilisée dans Logic Apps.

Résumé

Ce chapitre portait sur des sujets liés aux événements de streaming et de stockage. Les événements constituent aujourd'hui un élément important dans l'architecture globale de la solution. Nous avons abordé les ressources importantes, telles que Event Hubs et Stream Analytics, ainsi que les concepts fondamentaux, tels que les groupes de clients et les débits, ainsi que la création d'une solution end-to-end qui les utilise avec Logic Apps. Vous avez appris que les événements sont déclenchés à partir de plusieurs sources, et qu'afin d'obtenir des informations en temps réel sur les activités et leurs événements connexes, des services tels qu'Event Hubs et Stream Analytics jouent un rôle important. Dans le chapitre suivant, nous apprendrons à intégrer Azure DevOps et Jenkins, ainsi qu'à mettre en œuvre certaines des meilleures pratiques du secteur tout en développant des solutions.

13

Intégration de DevOps Azure

Dans le chapitre précédent, vous avez découvert les événements de Big Data et leur relation avec les services Event Hubs et Stream Analytics d'Azure. Le développement de logiciels est une tâche complexe qui nécessite divers processus et outils, et implique des collaborateurs travaillant dans différents services. Ces derniers doivent tous se regrouper et travailler de concours. Compte tenu de toutes ces variables, les risques sont élevés lors de la livraison aux clients finaux. Une simple omission ou une configuration inappropriée peut entraîner la panne de l'application. Ce chapitre aborde l'adoption et la mise en œuvre de bonnes pratiques destinées à réduire ce risque de manière considérable et à s'assurer qu'un logiciel de haute qualité sera systématiquement livré au client.

Avant d'entrer dans les détails relatifs à DevOps, voici une liste des problèmes rencontrés par les éditeurs de logiciels et qui peuvent être résolus grâce à DevOps :

- Les organisations rigides sont réfractaires au changement
- Les processus sont chronophages
- Des équipes isolées travaillent de façon cloisonnée
- La conception est monolithique et les déploiements sont de type Big Bang
- L'exécution manuelle
- Le manque d'innovation

Dans ce chapitre, nous allons aborder les thèmes suivants :

- DevOps
- Pratiques DevOps
- Azure DevOps
- Préparation DevOps
- DevOps pour solutions PaaS
- Solutions basées sur DevOps pour machine virtuelle (IaaS)
- Solutions basées sur DevOps pour conteneur (IaaS)
- DevOps Azure et Jenkins
- Azure Automation
- Outils Azure pour DevOps

DevOps

À l'heure actuelle, le secteur ne propose aucune définition officielle du terme DevOps. Chaque organisation a formulé sa propre définition de DevOps et a tenté d'implémenter ce système. Chacune a son propre point de vue et pense avoir implémenté DevOps une fois qu'elle a mis en œuvre la gestion de l'automatisation et de la configuration, et qu'elle a utilisé des processus Agile.

En m'appuyant sur mon expérience professionnelle des projets DevOps dans le secteur, je suis parvenu à déduire la définition suivante : DevOps désigne le mécanisme de livraison des systèmes logiciels. Il consiste à rassembler les personnes, à les faire collaborer et à leur permettre de communiquer, afin qu'elles travaillent ensemble pour concrétiser une vision et un objectif communs. Il s'agit de prendre des responsabilités et des engagements communs. Il convient de mettre en œuvre les processus qui favorisent la collaboration et un état d'esprit orienté vers le service. Cela permet de mettre en place des mécanismes de livraison qui apportent agilité et flexibilité au sein de l'organisation. Contrairement à ce que l'on croit souvent, DevOps ne repose pas sur des outils, des technologies et l'automatisation. Ces derniers constituent des catalyseurs qui favorisent la collaboration, la mise en œuvre de processus Agile et une livraison plus rapide et plus efficace au client.

Plusieurs définitions différentes de DevOps sont disponibles sur Internet, et aucune d'elles n'est incorrecte. DevOps ne fournit pas un cadre ou une méthode. Il s'agit d'un ensemble de principes et de pratiques qui, lorsqu'ils sont implémentés au sein d'une organisation, ou dans le cadre d'un engagement ou d'un projet, permettent d'atteindre un objectif ou de concrétiser une vision, à la fois concernant DevOps et l'organisation. Ces principes et ces pratiques n'exigent aucun processus, outil, technologie ou environnement particulier. DevOps fournit une direction, qui peut être suivie à l'aide de n'importe quel outil, technologie et processus, bien que certains d'entre eux soient plus appropriés pour concrétiser la vision des principes et des pratiques DevOps.

Bien que les pratiques DevOps puissent être implémentées dans n'importe quelle organisation fournissant des produits et des services aux clients, ce manuel envisage DevOps du point du vue du développement logiciel et du département des opérations de toute organisation.

Donc, qu'est-ce que le DevOps ? Il s'agit d'un ensemble de principes et de pratiques réunissant dès le départ toutes les équipes, y compris celles des développeurs et des opérations, afin de permettre une livraison end-to-end plus rapide et plus efficace d'un projet de qualité au client final, et ce systématiquement et de manière cohérente et prévisible, ce qui a pour effet de raccourcir le délai de commercialisation et donc d'obtenir un avantage concurrentiel.

La définition précédente de DevOps ne fait référence à aucun processus, outil ou technologie spécifique. Elle ne prescrit aucune méthodologie ou environnement.

La mise en œuvre des principes et des pratiques DevOps dans toute organisation vise à s'assurer que les exigences des parties prenantes (y compris des clients) et leurs attentes sont satisfaites, et ce de manière efficace.

Les exigences et les attentes des clients sont satisfaites dans les cas suivants :

- Le client obtient les fonctions souhaitées
- Le client obtient les fonctions voulues quand il le souhaite
- Le client obtient des mises à jour plus rapides des fonctions
- La qualité de la prestation est élevée

Lorsqu'une organisation est capable de répondre aux attentes susmentionnées, les clients sont satisfaits et lui restent fidèles. Cela permet ainsi de renforcer la compétitivité de l'organisation sur le marché, ce qui se traduit par une valorisation de la marque et une meilleure cote. Cela a un impact direct sur le chiffre d'affaires et sur les bénéfices nets de l'organisation. L'organisation peut investir davantage dans l'innovation et les commentaires des clients, ce qui permet d'apporter des modifications aux systèmes et aux services afin qu'ils restent pertinents.

La mise en œuvre des principes et des pratiques DevOps dans toute organisation s'inspire de son écosystème environnant. Cet écosystème se compose de l'industrie et des domaines auxquels appartient l'organisation.

DevOps s'appuie sur un ensemble de principes et de pratiques. Nous étudierons en détail ces principes et pratiques un peu plus loin dans ce chapitre. Les principes de base de DevOps sont :

- **Agilité** : le fait d'être agile accroît la flexibilité globale des changements et garantit que l'adaptabilité augmente dans tous les environnements en constante évolution et est productive. Les processus agiles ont une durée de travail plus courte et permettent de détecter les problèmes éventuels plus tôt dans le cycle de vie de développement, ce qui réduit la dette technique.
- **Automatisation** : l'adoption d'outils et de l'automatisation accroît l'efficacité et la prévisibilité globales du processus et du produit final. Cela permet d'accélérer, de faciliter et de réduire les coûts des tâches.
- **Collaboration** : la collaboration désigne un référentiel commun, une rotation des responsabilités professionnelles, un partage des données et des informations, ainsi que d'autres aspects qui améliorent la productivité de chaque membre de l'équipe, favorisant l'efficacité globale de la livraison du produit.
- **Commentaires** : cela fait référence à des boucles de commentaires rapides et précoces entre plusieurs équipes afin de déterminer ce qui fonctionne et ce qui ne fonctionne pas. Les commentaires aident les équipes à hiérarchiser les problèmes et à les corriger dans les versions ultérieures.

Les pratiques de base de DevOps sont les suivantes :

- **Intégration continue** : il s’agit du processus de validation et de vérification de la qualité et de l’exactitude du code entré dans le référentiel par les développeurs. L’intégration peut être programmée, manuelle ou continue. « Continue » signifie que le processus vérifie les différents attributs de qualité chaque fois qu’un développeur entre le code, tandis que « programmée » signifie que les vérifications ont lieu selon un programme planifié. « Manuelle » fait référence à l’exécution manuelle par un administrateur ou un développeur.
- **Gestion de la configuration** : il s’agit d’une facette importante de DevOps qui fournit des conseils pour la configuration de l’infrastructure et des applications, soit en allant chercher des configurations à partir de serveurs de gestion de configuration, soit en les planifiant. La gestion de la configuration doit ramener l’environnement à l’état attendu chaque fois qu’il est exécuté.
- **Livraison en continu** : la livraison continue fait référence à l’état de préparation d’une application pour son déploiement dans tout environnement, qu’il soit existant ou nouveau. La livraison est généralement exécutée au moyen d’une définition de version dans des environnements inférieurs comme le développement et les tests.
- **Déploiement continu** : le déploiement continu fait référence à la possibilité de déployer automatiquement l’environnement et l’application en production. Il est généralement exécuté au moyen d’une définition de version dans l’environnement de production.
- **Apprentissage continu** : il s’agit du processus de compréhension des problèmes rencontrés par les opérations et les clients, et de leur transmission aux équipes de développement et de test afin qu’elles puissent les résoudre dans les versions suivantes dans le but d’améliorer l’intégrité et la facilité d’utilisation globales de l’application.

L’essence de DevOps

DevOps n’est pas un nouveau paradigme. Toutefois, il gagne en popularité et suscite de plus en plus d’intérêt. Cette solution n’avait jamais autant été adoptée et de nombreuses entreprises empruntent désormais ce parcours. J'ai volontairement fait référence à DevOps en tant que parcours car il existe différents niveaux de maturité au sein de DevOps. Bien qu'un déploiement continu et une livraison réussis soient considérés comme étant les niveaux de maturité les plus élevés de ce parcours, l'adoption d'un contrôle du code source et le développement d'un logiciel agile constituent de solides points de départ dans le parcours DevOps.

L'un des concepts principaux de DevOps consiste à lever les barrières entre les développeurs et les équipes des opérations. Il s'agit de favoriser une collaboration étroite entre plusieurs équipes. Il s'agit de rompre avec l'état d'esprit selon lequel il incombe au développeur d'écrire du code uniquement et de le transmettre à l'équipe des opérations qui le testera, puis le déploiera. Il s'agit également de cesser de penser que les opérations n'ont aucun rôle à jouer au sein des activités de déploiement. Les Opérations doivent s'impliquer dans la planification du produit et doivent être mises au courant des fonctions à venir dans la nouvelle solution. Elles doivent également transmettre en permanence des commentaires aux développeurs en ce qui concerne les enjeux opérationnels, afin de remédier à ces problèmes dans les versions suivantes. Elles doivent influer sur la conception du système, afin de garantir son fonctionnement optimal. De même, les développeurs doivent aider l'équipe des opérations à déployer le système et à résoudre tout incident éventuel.

La définition de DevOps évoque une livraison des systèmes end-to-end plus rapide et plus efficace aux parties prenantes. Elle ne précise pas dans quelle mesure cette livraison doit être effectivement rapide et efficace. Celle-ci doit être suffisamment rapide selon le domaine de l'organisation, le secteur, la segmentation des clients et les besoins. Pour certaines organisations, un rythme trimestriel est suffisamment rapide, tandis que pour d'autres, la fréquence doit être hebdomadaire. Les deux sont valides d'un point de vue DevOps, et ces organisations peuvent déployer des processus et des technologies pertinents pour respecter les dates de lancement qu'elles se sont fixées. DevOps ne prescrit aucun délai précis pour **l'intégration et le déploiement continus** (**CI/CD**). Les organisations doivent identifier quelle serait la mise en œuvre optimale des principes et des pratiques DevOps sur la base de leur projet, de leur engagement et de leur vision organisationnelle dans leur globalité.

La définition évoque aussi une livraison end-to-end. Cela signifie que tout, de la planification et de la livraison du système jusqu'aux services et aux opérations, doit faire partie intégrante de l'adoption de DevOps. Les processus doivent permettre une plus grande flexibilité, modularité et agilité dans le cycle de vie du développement de l'application. Bien que les organisations soient libres d'opter pour le processus qu'elles jugent le plus approprié (en cascade, agile, Scrum ou autre), elles tendent généralement à favoriser les processus agiles avec une prestation axée sur les itérations. Cela permet une livraison plus rapide en unités plus petites, qui sont bien plus faciles à tester et à gérer qu'une livraison conséquente.

DevOps évoque régulièrement les clients finaux, et ce d'une manière constante et prévisible. Cela signifie que les organisations doivent continuellement fournir aux clients des fonctions nouvelles et améliorées, grâce à l'automatisation. Nous ne pouvons pas atteindre la cohérence et la prévisibilité sans recourir à l'automatisation. Les tâches manuelles doivent être éliminées, afin de garantir un niveau élevé de cohérence et de prévisibilité. L'automatisation doit également être gérée end-to-end, pour éviter les pannes. Cela indique également que le système doit être modulaire pour permettre une livraison plus rapide, grâce à sa fiabilité, sa disponibilité et son évolutivité. Les tests jouent un rôle important pour garantir une livraison cohérente et prévisible.

Le résultat final de la mise en œuvre de ces pratiques et principes est de permettre à l'organisation de répondre aux attentes et exigences des clients. L'organisation est capable de se développer plus rapidement que la concurrence et améliore la qualité et la capacité de ses produits et services, au moyen d'innovations et de progrès continus.

Maintenant que vous comprenez l'objectif de DevOps, passons à ses pratiques de base.

Pratiques DevOps

DevOps se compose de plusieurs pratiques, chacune offrant une fonction distincte dans l'ensemble du processus. Le diagramme suivant montre la relation entre elles. La gestion de la configuration, l'intégration continue et le déploiement continu constituent des pratiques fondamentales permettant la mise en œuvre de DevOps. Lorsque nous offrons des services logiciels combinant ces trois services, nous parvenons à assurer une livraison continue. La livraison continue est la capacité et le niveau de maturité d'une organisation qui dépend de la maturité de la gestion de la configuration, de l'intégration continue et du déploiement continu. Des commentaires continus à tous les stades forment une boucle qui permet d'offrir une qualité de service supérieure aux clients. Ces commentaires doivent concerner toutes les pratiques DevOps. Examinons en détail chacune de ces capacités et pratiques DevOps :

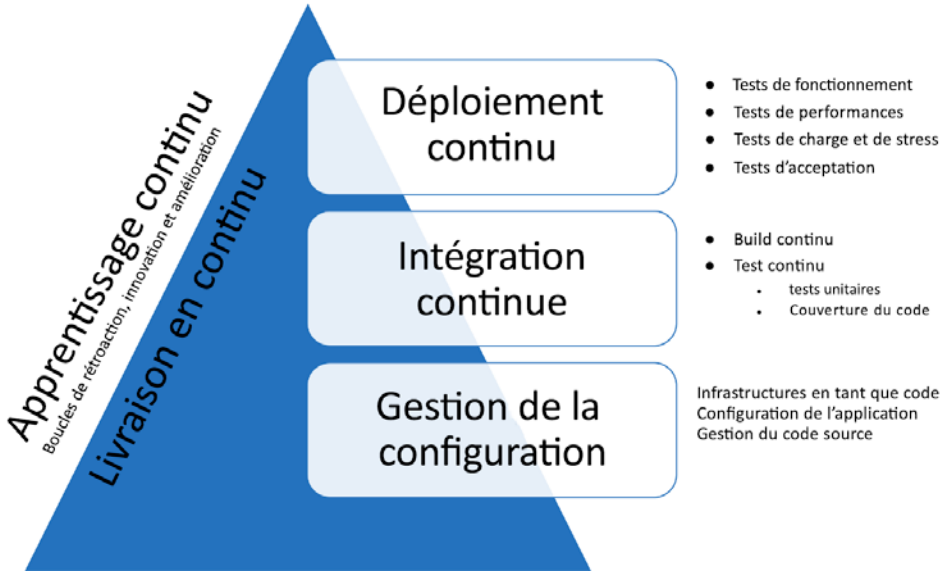


Figure 13.1 : Pratiques DevOps

Gestion de la configuration

Les services et les applications métier nécessitent un environnement dans lequel ils peuvent être déployés. En général, l'environnement est une infrastructure composée de plusieurs serveurs, ordinateurs, réseaux, espaces de stockage, conteneurs et de bien d'autres services fonctionnant ensemble de telle sorte que les applications métier peuvent y être déployées. Les applications métier se composent de plusieurs services exécutés sur plusieurs serveurs, sur site ou dans le Cloud, et chaque service a sa propre configuration et ses propres exigences du point de vue de la configuration de l'infrastructure. En bref, les infrastructures et les applications sont toutes deux nécessaires pour fournir des systèmes aux clients et chacune d'elles dispose de sa propre configuration. En cas de défaut de la configuration, il est possible que l'application ne fonctionne pas comme prévu, ce qui risque d'entraîner des temps d'arrêt et des pannes. En outre, le processus de **gestion du cycle de vie des applications** (**ALM**) préconisant de recourir à plusieurs étapes et environnements, une application doit être déployée dans plusieurs environnements avec diverses configurations. L'application est déployée dans l'environnement de développement afin que les développeurs puissent voir le résultat de leur travail. L'application est ensuite déployée sur plusieurs environnements de test avec des configurations différentes pour les tests fonctionnels, les tests de charge et de stress, les tests de performances, les tests d'intégration et bien plus encore. Elle est également déployée dans l'environnement de préproduction pour réaliser des tests d'acceptation de l'utilisateur et enfin dans l'environnement de production. Il est essentiel qu'une application puisse être déployée dans plusieurs environnements sans qu'aucune modification manuelle de sa configuration ne soit nécessaire.

La gestion de la configuration fournit un ensemble de processus et d'outils et permet de veiller à ce que chaque environnement et application obtienne sa propre configuration. La gestion de la configuration permet d'établir le suivi des éléments de configuration et tout changement dans l'environnement doit être considéré comme étant un élément de configuration en soi. La gestion de la configuration définit également les relations entre les éléments de configuration et la manière dont les modifications apportées à ces derniers auront un impact sur les autres.

Utilisation de la gestion de la configuration

La gestion de la configuration facilite la mise en œuvre des éléments suivants :

- **Infrastructure en tant que code** : lorsque le processus de mise en service de l'infrastructure et sa configuration sont représentés par l'intermédiaire d'un code et que ce même code suit le processus de cycle de vie de l'application, il est qualifié d'**Infrastructure en tant que Code** (**IaC**). IaC aide à automatiser le provisionnement et la configuration de l'infrastructure. Elle représente également l'ensemble de l'infrastructure dans le code qui peut être stockée dans un référentiel et soumise à un contrôle de version. Cela permet aux utilisateurs d'utiliser les configurations d'environnement précédentes lorsque nécessaire. Elle permet également la mise en service d'un environnement à plusieurs reprises, de façon cohérente et prévisible. Tous les environnements mis en service de la sorte sont cohérents et égaux à tous les stades de l'ALM. Plusieurs outils permettent de réaliser l'IaC, y compris les modèles ARM, Ansible et Terraform.

- **Déploiement et configuration de l'application** : le déploiement d'une application et sa configuration sont les étapes qui suivent la mise en service de l'infrastructure. Par exemple, le déploiement et la configuration d'une application peuvent consister à déployer un paquet **`webdeploy`** sur un serveur, à déployer un schéma SQL Server et des données (**`bacpac`**) sur un autre serveur, ou encore à modifier la chaîne de connexion SQL sur le serveur web pour représenter le SQL Server approprié. La gestion de la configuration stocke les valeurs de la configuration de l'application pour chaque environnement dans lequel celle-ciest déployée.

La configuration appliquée doit également être surveillée. La configuration attendue et souhaitée doit être maintenue en permanence. Toute dérive de cette configuration attendue et souhaitée rendrait l'application non disponible. La gestion de la configuration est également capable de détecter cette faille et de reconfigurer l'application et l'environnement pour restaurer l'état souhaité.

Une fois la gestion de la configuration automatisée mise en place, personne dans l'équipe n'aura à déployer ni à configurer les environnements et les applications dans la production. L'équipe des opérations n'est pas tributaire de l'équipe de développement, et elle ne doit pas non plus s'appuyer sur une documentation de déploiement laborieuse.

Le contrôle de code source constitue un autre aspect de la gestion de la configuration. Les services et applications métier comprennent le code et d'autres éléments. Plusieurs membres de l'équipe travaillent sur les mêmes fichiers. Le code source doit être à jour en tout temps et doit être accessible aux membres de l'équipe authentifiés uniquement. Le code, ainsi que d'autres éléments, sont des éléments de configuration en soi. Le contrôle de la source favorise la collaboration et la communication au sein de l'équipe, car tout le monde a connaissance du travail d'autrui et les conflits sont résolus à un stade précoce.

La gestion de la configuration se divise en deux grandes catégories :

- À l'intérieur de la machine virtuelle
- À l'extérieur de la machine virtuelle

Outils de gestion de la configuration

Les outils disponibles pour la gestion de la configuration à l'intérieur de la machine virtuelle seront abordés ultérieurement.

Configuration de l'état souhaité

La configuration de l'état souhaité (**DSC**) est une plateforme de gestion de la configuration de Microsoft, conçue comme une extension de PowerShell. DSC a été initialement lancée dans le cadre de **Windows Management Framework** (**WMF**) 4.0. Elle est disponible dans le cadre des versions Windows Management Framework (WMF) 4.0 et 5.0 pour tous les systèmes d'exploitation Windows antérieurs à Windows 2008 R2. WMF 5.1 est disponible de manière native sur Windows Server 2016/2019 et Windows 10.

Chef, Puppet et Ansible

Outre DSC, il existe une multitude d'outils de gestion de la configuration, comme Chef, Puppet et Ansible, qui sont pris en charge par Azure. Ces outils ne sont pas abordés en détail dans ce manuel. Pour en savoir plus sur ces outils, consultez l'adresse https://docs.microsoft.com/azure/virtual-machines/windows/infrastructure-automation.

Les outils disponibles pour la gestion de la configuration à l'extérieur de la machine virtuelle sont indiqués plus bas.

Modèles ARM

Les modèles ARM sont principalement des moyens de mettre en service des ressources dans ARM. Les modèles ARM fournissent un modèle déclaratif grâce auquel les ressources et leurs configurations, scripts et extensions sont définis. Les modèles ARM s'appuient sur le format **JavaScript Object Notation** (**JSON**). Ils utilisent la syntaxe et les conventions JSON pour déclarer et configurer les ressources. Les fichiers JSON sont basés sur des fichiers texte lisibles et conviviaux pour les utilisateurs. Ils peuvent être stockés dans un référentiel de code source et disposent d'un contrôle de version. Ils permettent également de représenter l'infrastructure en tant que code qui peut être utilisée pour mettre en service les ressources dans des groupes de ressource Azure plusieurs fois, et ce de manière prévisible, cohérente et uniforme.

Les modèles fournissent la flexibilité nécessaire afin qu'une solution soit générique et modulaire de par sa conception et sa mise en œuvre. Les modèles permettent d'accepter les paramètres des utilisateurs, de déclarer des variables internes, de définir des dépendances entre les ressources, de relier les ressources au sein de groupes de ressources identiques ou différents et d'exécuter d'autres modèles. Ils fournissent également des expressions de type langage script et des fonctions qui les rendent personnalisables et dynamiques lors de l'exécution. Deux chapitres sont dédiés aux modèles ARM dans ce livre : le *Chapitre 15*, *Déploiements multi-abonnements à l'aide des modèles ARM* et le *Chapitre 16*, *Conception et application de modèles ARM modulaires*.

Concentrons-nous maintenant sur le prochain principe DevOps important : l'intégration continue.

Intégration continue

Plusieurs développeurs écrivent du code qui finit par être stocké dans un référentiel commun. Ce code est normalement archivé ou publié dans le référentiel lorsque les développeurs finalisent leurs fonctions. Cela peut prendre un seul ou plusieurs jours, voire plusieurs semaines. Certains développeurs peuvent travailler sur la même fonction et suivre les mêmes pratiques de publication et d'enregistrement du code en quelques jours ou quelques semaines. Cela peut nuire à la qualité du code. L'échec rapide constitue l'un des principes de DevOps. Les développeurs doivent enregistrer et publier le code dans le référentiel fréquemment, puis le compiler pour vérifier la présence de tout bogue éventuel et si le code est compatible avec celui écrit par leurs collègues. Si un développeur ne suit pas cette pratique, le code de sa machine deviendra très volumineux et sera difficile à intégrer à un autre code. En outre, si la compilation échoue, il est difficile et long de remédier aux problèmes qui en découlent.

Intégration du code

L'intégration continue a résolu ces types de problèmes. L'intégration continue permet de compiler et de valider le code publié et enregistré par un développeur en lui faisant passer une série d'étapes de validation. L'intégration continue crée un flux de processus comportant plusieurs étapes. L'intégration continue est composée de builds et de tests automatisés et continus. Normalement, la première étape consiste à compiler le code. Une fois la compilation réussie, chaque étape doit valider le code d'un point de vue spécifique. Par exemple, les tests unitaires peuvent être exécutés sur le code compilé, et la couverture de code peut être exécutée pour vérifier quels parcours de code sont exécutés par les tests unitaires. Ces opérations peuvent révéler si des tests unitaires complets sont rédigés ou s'il est possible d'ajouter d'autres tests unitaires. Le résultat final de l'intégration continue est la création de paquets de déploiement qui peuvent être utilisés par le déploiement continu afin de les déployer sur plusieurs environnements.

Enregistrement fréquent du code

Les développeurs sont encouragés à enregistrer leur code plusieurs fois par jour et non au bout de plusieurs jours ou semaines. L'intégration continue lance l'exécution de l'intégralité du pipeline immédiatement après l'enregistrement ou la publication du code. Si la compilation réussit et que les tests de code et autres activités qui font partie du pipeline sont exécutés sans erreurs, le code est déployé dans un environnement de tests et les tests d'intégration y sont exécutés.

Productivité accrue

L'intégration continue augmente la productivité des développeurs. Ces derniers n'ont pas besoin de compiler manuellement leur code, d'exécuter plusieurs types de tests les uns après les autres, ni de créer des packages externes. Elle réduit également les risques de bogues dans le code et de tout autre vice. Elle fournit une rétroaction rapide aux développeurs sur la qualité de leur code. Dans l'ensemble, la qualité des livrables est élevée et ces derniers sont remis plus rapidement grâce à l'adoption d'une pratique d'intégration continue. Voici un exemple de pipeline d'intégration continue :

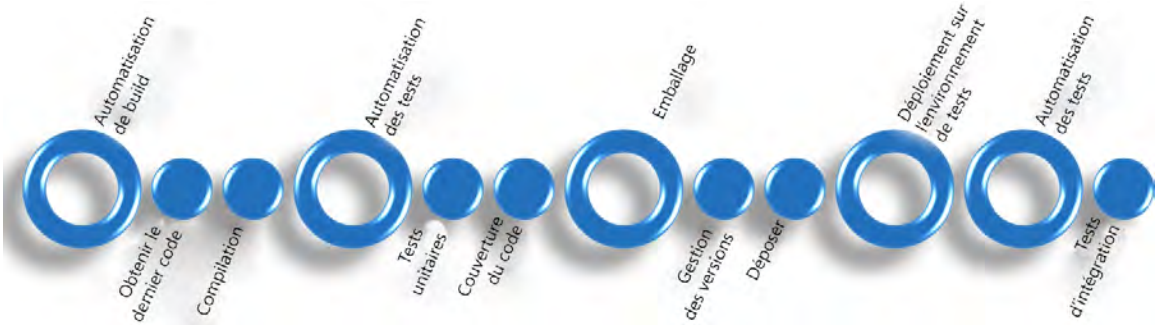


Figure 13.2 : Pipeline d'intégration continue

Automatisation de build

La génération automatisée se compose de plusieurs tâches exécutées dans l'ordre. En règle générale, la première tâche consiste à récupérer le dernier code source dans le référentiel. Le code source peut comporter plusieurs projets et fichiers. Il est compilé afin de générer des éléments tels que des fichiers exécutables, des bibliothèques de liens dynamiques, des assemblages, etc. Une génération automatisée réussie implique qu'il n'existe plus aucune erreur de compilation dans le code.

La génération automatisée peut comporter un plus grand nombre d'étapes, selon la nature et le type de projet.

Automatisation des tests

L'automatisation des tests se compose de tâches de validation des différents aspects du code. Ces tâches sont liées aux tests de code d'un autre point de vue et sont exécutées de manière séquentielle. En règle générale, la première étape consiste à exécuter une série de tests unitaires sur le code. Les tests unitaires désignent le processus de test de la plus petite dénomination d'une fonction, afin de valider son comportement isolé des autres fonctions. Ils peuvent être automatisés ou manuels, en sachant qu'il est préférable d'avoir recours à des tests unitaires automatisés.

La couverture du code est un autre type de test automatisé qui peut être exécuté sur le code afin de connaître la quantité de code exécutée lors de l'exécution des tests unitaires. Elle est généralement représentée sous forme de pourcentage et se réfère à la quantité de code qui peut être testée au moyen de tests unitaires. Si la couverture du code n'est pas proche de 100 %, cela signifie que le développeur n'a pas écrit de test unitaire pour ce comportement ou bien que le code découvert n'est pas du tout nécessaire.

L'exécution réussie de l'automatisation de test, c'est-à-dire s'il n'y a aucune défaillance importante du code, doit entraîner le début de l'exécution des tâches d'empaquetage. L'automatisation de test peut comporter un plus grand nombre d'étapes, en fonction de la nature et du type de projet.

Emballage

L'empaquetage désigne le processus de génération des éléments déployables comme les paquets `MSI`, `NuGet` et `webdeploy`, les paquets de base de données, la gestion de leurs versions et leur stockage dans un emplacement afin qu'ils puissent être consommés par d'autres canaux et processus.

Une fois le processus d'intégration continue terminé, le processus de déploiement continu commence, comme nous allons voir dans la section suivante.

Déploiement continu

Lorsque le processus atteint le déploiement continu, l'intégration continue a permis de s'assurer que tous les bits d'une application étaient entièrement fonctionnels et pouvaient passer à d'autres activités de déploiement continu. Le déploiement continu se réfère à la capacité de déployer des applications métier et des services aux environnements de préproduction et de production grâce à l'automatisation. Par exemple, le déploiement continu peut mettre en service et configurer l'environnement de pré-production, y déployer l'application et la configurer. Après avoir effectué plusieurs validations, comme des tests fonctionnels et des tests de performance sur l'environnement de préproduction, l'environnement de production est mis en service, configuré, et l'application est déployée grâce à l'automatisation. Le processus de déploiement ne comporte aucune étape manuelle. Chaque tâche de déploiement est automatisée. Le déploiement continu peut mettre en service l'environnement et déployer l'application à partir de zéro, tandis qu'il peut simplement déployer les modifications delta dans l'environnement existant si l'environnement existe déjà.

Tous les environnements sont mis en service grâce à l'automatisation à l'aide de l'Infrastructure en tant que code (IaC). Cela garantit que tous les environnements, que ce soit le développement, le test, la pré-production ou la production sont identiques. De même, l'application est déployée via l'automatisation, ce qui permet de s'assurer qu'elle est également déployée de manière uniforme sur tous les environnements. La configuration de ces environnements peut être différente pour l'application.

Le déploiement continu est généralement intégré grâce à l'intégration continue. Une fois que l'intégration continue a généré les packages déployables finaux, le déploiement continu est lancé et démarre son propre pipeline. Ce pipeline est appelé le pipeline de lancement. Le pipeline de mise en production se compose de plusieurs environnements, chacun étant constitué de tâches destinées à mettre en service l'environnement, à le configurer, à déployer des applications, à les configurer, à exécuter la validation opérationnelle sur les environnements et à tester l'application sur plusieurs environnements.

Le recours au déploiement continu apporte des avantages conséquents. Le processus de déploiement global est hautement fiable, ce qui permet une diffusion plus rapide et sans risque dans l'environnement de production. Les risques sont considérablement réduits. L'équipe est soumise à des niveaux de stress moindres et il est possible de revenir à l'environnement de travail précédent si des problèmes sont détectés dans la version actuelle :

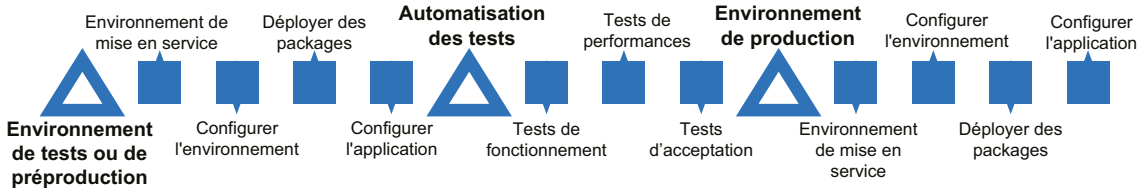


Figure 13.3 : Pipeline de déploiement continu

Bien que chaque système exige sa propre configuration de pipeline de mise en production, un court exemple de déploiement continu est illustré dans le diagramme précédent. Il est important de noter qu'en général, la mise en service et la configuration de plusieurs environnements font partie du pipeline de lancement et il est nécessaire d'obtenir des approbations avant de passer à l'environnement suivant. Le processus d'approbation peut être manuel ou automatisé, selon la maturité de l'organisation.

Nous nous pencherons ensuite sur les aspects liés à l'environnement de tests.

Déploiement de l'environnement de tests

Le pipeline de mise en production démarre dès que le dépôt est disponible à la suite de l'intégration continue et la première étape doit consister à réunir tous les éléments du dépôt. Ensuite, un nouvel environnement de tests sans système d'exploitation pourra être utilisé ou il sera possible de réutiliser un environnement existant. Cela dépendra encore une fois du type de projet et de la nature des tests qui doivent être exécutés sur cet environnement. L'environnement est mis en service et configuré. Les éléments de l'application sont déployés et configurés.

Automatisation des tests

Après le déploiement d'une application, une série de tests peut être exécutée sur l'environnement. Parmi les tests exécutés figure un test fonctionnel. Les tests fonctionnels visent d'abord à valider l'exhaustivité de la fonction et le bon fonctionnement de l'application. Ces tests sont écrits en tenant compte des exigences du client. Une autre série de tests qui peut être exécutée est liée à l'évolutivité et à la disponibilité de l'application. Cela comprend généralement les tests de charge, les tests de stress et les tests de performance. Cela doit également inclure la validation opérationnelle de l'environnement d'infrastructure.

Déploiement de l'environnement de préproduction

Cette opération est fortement similaire au déploiement de l'environnement de tests, sauf que les valeurs de configuration de l'environnement et de l'application diffèrent.

Tests d'acceptation

Les tests d'acceptation sont généralement réalisés par des parties prenantes de l'application, et il peut s'agir d'un processus manuel ou automatisé. Cette étape permet de vérifier, du point de vue du client, si l'application fonctionne correctement et dans son intégralité.

Déploiement sur l'environnement de production

Une fois que le client donne son approbation, les mêmes étapes que celles inhérentes au déploiement de l'environnement de tests et de test intermédiaire sont exécutées, à la différence que les valeurs de configuration de l'environnement et de l'application sont spécifiques à l'environnement de production. Une validation est effectuée après le déploiement afin de s'assurer que l'application s'exécute selon les attentes.

La livraison continue est un principe DevOps important qui ressemble beaucoup au déploiement continu, avec quelques différences. La section suivante traite de la livraison continue.

Livraison en continu

La livraison continue et le déploiement continu peuvent vous sembler similaires. Néanmoins, il s'agit de deux processus différents. Le déploiement continu fait référence au déploiement dans plusieurs environnements et en dernier lieu dans l'environnement de production via l'automatisation, tandis que la livraison continue désigne la capacité à générer des paquets d'application facilement déployables dans n'importe quel environnement. Afin de générer des éléments facilement déployables, l'intégration continue doit être mise en œuvre pour générer des éléments d'application. Un environnement, nouveau ou existant, doit être utilisé pour déployer ces éléments, des tests fonctionnels doivent être réalisés en plus de tests de performance et d'acceptation utilisateur via l'automatisation. Une fois ces activités exécutées avec succès et sans erreur, le package d'application est considéré comme étant prêt au déploiement. La livraison continue comprend l'intégration continue ainsi que le déploiement dans un environnement en vue des validations finales. Elle permet d'obtenir des commentaires plus rapidement à la fois des opérations et de l'utilisateur final. Ce feedback peut ensuite être utilisé pour mettre en place les itérations ultérieures.

La section suivante traite de l'apprentissage continu.

Apprentissage continu

Compte tenu de toutes les pratiques DevOps susmentionnées, il est possible de créer d'excellentes applications métier et de les déployer automatiquement dans l'environnement de production. Cependant, les avantages de DevOps seront de courte durée si aucun principe d'amélioration continue et de feedback n'est appliqué. Il est extrêmement important que des feedbacks sur le comportement de l'application soient communiqués en temps réel à l'équipe de développement, que ces commentaires proviennent des utilisateurs finaux ou de l'équipe des opérations.

Les commentaires doivent être communiqués aux équipes, en indiquant de manière pertinente ce qui fonctionne et surtout, ce qui ne fonctionne pas.

L'architecture et la conception d'une application doivent être bâties en tenant compte de la surveillance, des audits et de la télémétrie. L'équipe des opérations doit recueillir des données de télémétrie issues de l'environnement de production, saisir tout bogue ou problème et en informer l'équipe de développement afin que celle-ci applique des correctifs dans les versions ultérieures.

L'apprentissage continu permet de rendre une application robuste et résistante aux failles. Elle permet de s'assurer que l'application répond aux exigences des consommateurs. La *figure 13.4* montre la boucle de feedback qui doit être implémentée entre les différentes équipes :

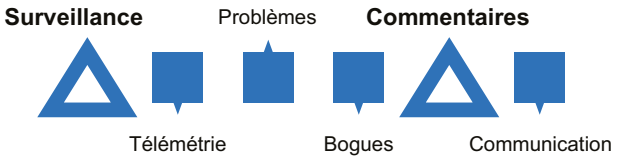


Figure 13.4 : Boucle de feedback

Après avoir passé en revue les pratiques importantes liées à DevOps, passons aux outils et services qui les rendent possibles.

Azure DevOps

Concentrons-nous sur un autre service en ligne révolutionnaire qui permet l'intégration continue, le déploiement continu et la livraison continue, en toute fluidité : Azure DevOps. En réalité, il serait plus approprié de définir cette solution comme une suite de services disponibles sous un seul et même nom. Azure DevOps est un PaaS fourni par Microsoft et hébergé dans le Cloud. Ce même service est disponible en tant que **Team Foundation Services** (**TFS**) sur site. Tous les exemples présentés dans ce livre utilisent Azure DevOps.

Selon Microsoft, Azure DevOps est une plateforme de collaboration basée dans le Cloud, qui aide les équipes à partager du code, à établir le suivi de leur travail et à lancer un logiciel. Azure DevOps est le nouveau nom de **Visual Studio Team Services** (**VSTS**). Azure DevOps est un outil de développement logiciel d'entreprise et un service qui permet aux organisations de fournir des espaces d'automatisation pour leur processus de gestion du cycle de vie de l'application end-to-end, de la planification au déploiement de l'application, tout en obtenant un feedback en temps réel de la part des systèmes logiciels. Cela augmente la maturité et la capacité d'une organisation à fournir des systèmes logiciels de haute qualité à leurs clients.

Une livraison de logiciels réussie implique de regrouper efficacement de nombreux processus et activités. Cela inclut la mise en œuvre de processus agiles, une plus grande collaboration entre les équipes, une transition automatique et transparente des éléments d'une phase d'ALM à une autre et des déploiements sur plusieurs environnements. Il est important de suivre et d'établir des rapports sur ces activités afin de mesurer et d'améliorer les processus de livraison. Avec Azure DevOps, cette étape est simple et facile. Il fournit toute une suite de services qui permet :

- La collaboration entre chaque membre de l'équipe en fournissant une interface unique pour la gestion de l'ensemble du cycle de vie de l'application
- La collaboration entre les équipes de développement à l'aide des services de gestion de code source
- La collaboration entre les équipes de tests à l'aide des services de gestion de test
- La validation automatique du code et l'empaquetage grâce à l'intégration continue, à l'aide des services de gestion de version
- L'automatisation de la validation de la fonction de l'application, du déploiement et de la configuration de plusieurs environnements au moyen du déploiement et de la livraison continus, à l'aide des services de gestion de la mise en production
- Le suivi et la gestion des éléments de travail à l'aide de services de gestion des tâches

Le tableau suivant montre tous les services disponibles pour un projet à partir de la barre de navigation gauche d'**Azure DevOps** :

Service	Description
Tableaux	Les tableaux de bord permettent de planifier le projet en affichant l'avancement actuel des tâches, des backlogs et des témoignages d'utilisateurs, ainsi que les informations Sprint. Il fournit également un processus Kanban et décrit les tâches en cours et terminées.
Repos	Repos permet de gérer les référentiels. Il prend en charge la création de branches supplémentaires, leur fusion, la résolution des conflits de code et la gestion des autorisations. Un projet peut présenter plusieurs référentiels.
Pipelines	Les pipelines de builds et de versions sont créés et gérés à partir de pipelines. Ils permettent d'automatiser le processus de builds et de versions. Un projet peut présenter plusieurs pipelines de builds et de versions.
Plans de test	Tous les artefacts liés aux tests, ainsi que leur gestion, sont disponibles dans les plans de test.
Artefacts	Les packages NuGet et d'autres artefacts sont stockés et gérés ici.

Tableau 13.1 : Liste des services DevOps Azure

Une organisation dans Azure DevOps est une limite de sécurité et un conteneur logique qui fournit tous les services nécessaires à l'implémentation d'une stratégie DevOps. Azure DevOps permet la création de plusieurs projets au sein d'une seule organisation. Par défaut, un référentiel est créé avec la création d'un projet. Cependant, Azure DevOps permet la création de référentiels supplémentaires dans un projet unique. La relation entre l'**organisation DevOps Azure**, les **projets** et le **référentiel** est illustrée à la *figure* 13.5 :

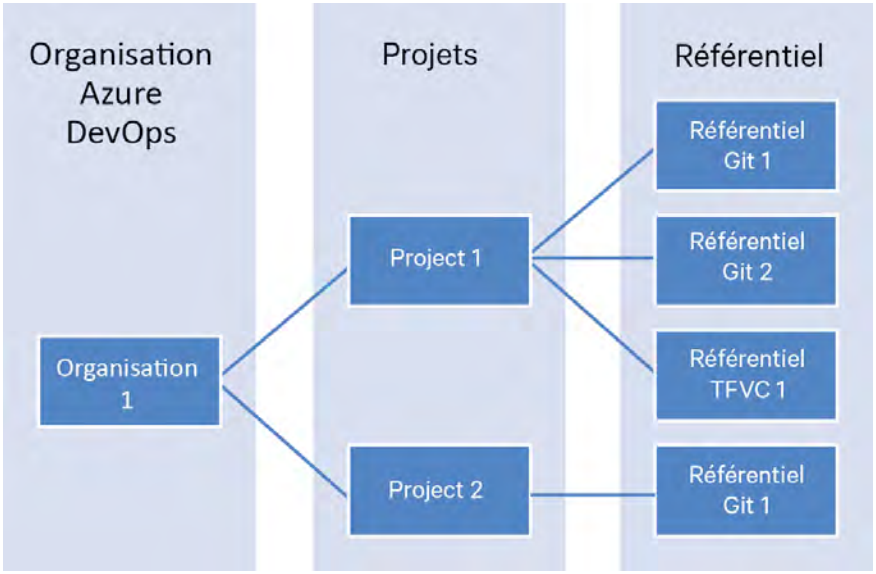


Figure 13.5 : Relation entre l'organisation Azure DevOps, les projets et le référentiel

Azure DevOps fournit deux types de référentiels :

- Git
- Team Foundation Version Control (TFCV)

Il offre également la possibilité de choisir entre le référentiel de contrôle de source Git ou TFVC. Il est possible de combiner les référentiels TFS et TFVC disponibles dans un projet unique.

TFVC

TFVC est la méthode d'implémentation du contrôle de version classique et centralisée, qui comporte un référentiel central sur lequel les développeurs travaillent directement en mode connecté afin d'enregistrer leurs modifications. Si le référentiel central est hors ligne ou non disponible, les développeurs ne peuvent pas enregistrer leur code et doivent attendre que celui-ci soit à nouveau en ligne et disponible. D'autres développeurs peuvent voir uniquement le code enregistré. Les développeurs peuvent regrouper plusieurs modifications dans un seul ensemble de modifications, afin d'enregistrer les modifications de code regroupées de manière logique sous une seule et même modification. TFVC verrouille les fichiers de code qui font l'objet de modifications. D'autres développeurs peuvent lire un fichier verrouillé, mais ils ne peuvent pas le modifier. Ils doivent attendre que la modification précédente soit terminée et que le fichier soit ainsi déverrouillé pour pouvoir le modifier à leur tour. L'historique des enregistrements et des modifications est disponible dans le référentiel central, tandis que les développeurs disposent d'une copie de travail des fichiers, mais pas de l'historique.

TFVC est particulièrement adapté aux grandes équipes qui travaillent sur les mêmes projets. Cela permet de contrôler le code source dans un emplacement central. Cela fonctionne également très bien pour les projets de longue durée, car l'historique est géré dans un emplacement central. TFVC fonctionne sans problème avec les fichiers binaires et volumineux.

Git

Git, quant à lui, est une méthode de contrôle de version moderne et distribuée, dans laquelle les développeurs peuvent travailler sur leurs propres copies locales du code ainsi que sur l'historique, en mode hors connexion. Les développeurs peuvent travailler hors connexion sur leur clone local du code. Chaque développeur dispose d'une copie locale du code et de l'ensemble de son historique et ils travaillent sur leurs modifications avec ce référentiel local. Il peut valider son code dans le référentiel local. Il peut se connecter au référentiel central pour synchroniser leur référentiel local au besoin. Cela permet à chaque développeur de travailler sur n'importe quel fichier, avec sa copie locale. La ramification Git ne crée pas d'autre copie du code original et est extrêmement rapide à créer.

GIT fonctionne aussi bien avec les petites que les grandes équipes. La ramification et la fusion sont un jeu d'enfant, grâce à des options avancées.

GIT est une méthode d'utilisation du contrôle de source recommandée, du fait de la fonction enrichie qu'elle offre. Nous allons utiliser Git comme référentiel pour notre exemple d'application dans ce manuel. Dans la section suivante, nous vous proposerons une présentation détaillée de la mise en œuvre de l'automatisation par l'intermédiaire de DevOps.

Préparation à DevOps

Par la suite, nous allons mettre l'accent sur l'automatisation des processus et des déploiements à l'aide de différents modèles dans Azure. En voici quelques exemples :

- DevOps pour solutions IaaS
- DevOps pour solutions PaaS
- Solutions basées sur DevOps pour conteneur

Généralement, il existe des services partagés, qui ne sont pas propres à une application. Ces services sont consommés par plusieurs applications de différents environnements, comme le développement, le test et la production. Le cycle de vie de ces services partagés est différent pour chaque application. Par conséquent, ils disposent de référentiels de contrôle de version, d'une base de code, d'un build et d'une gestion des publications distincts. Ils suivent leur propre cycle « planifier/concevoir/développer/tester/publier ».

Les ressources qui font partie de ce groupe sont mises en service au moyen de modèles ARM, de PowerShell et de configurations DSC.

Le processus global de conception de ces composants communs est indiqué ici :

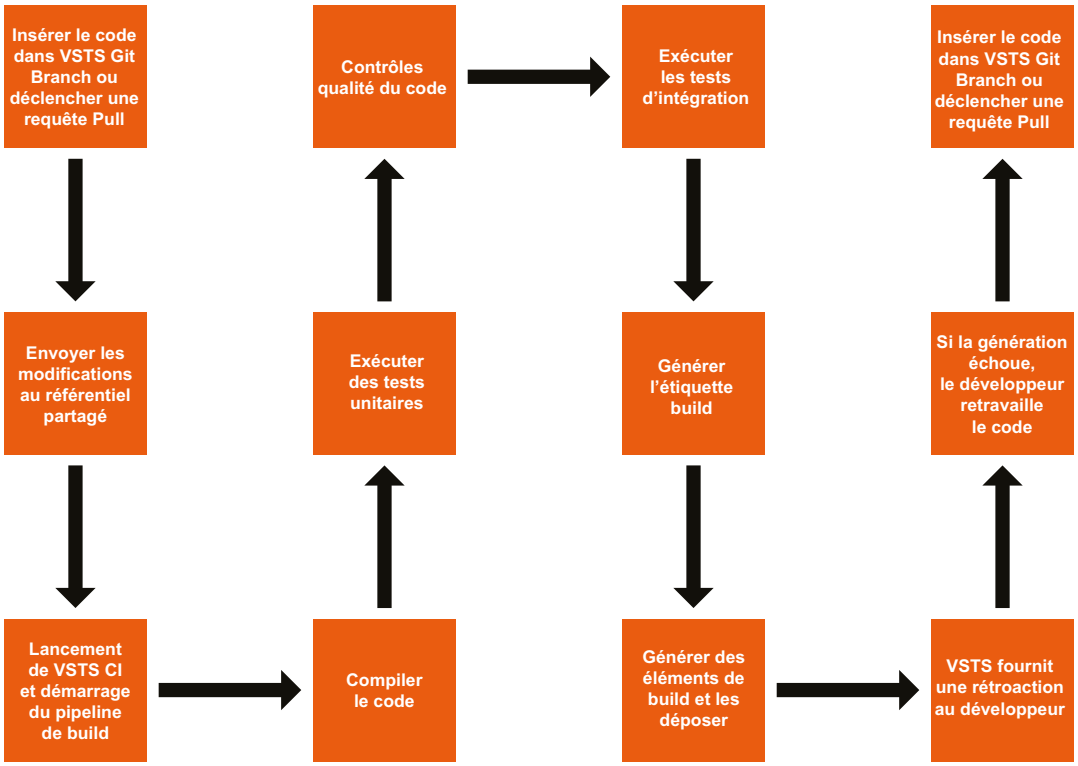


Figure 13.6 : Processus global de conception de ces composants communs

Le processus de publication est illustré à la *figure* 13.7 :

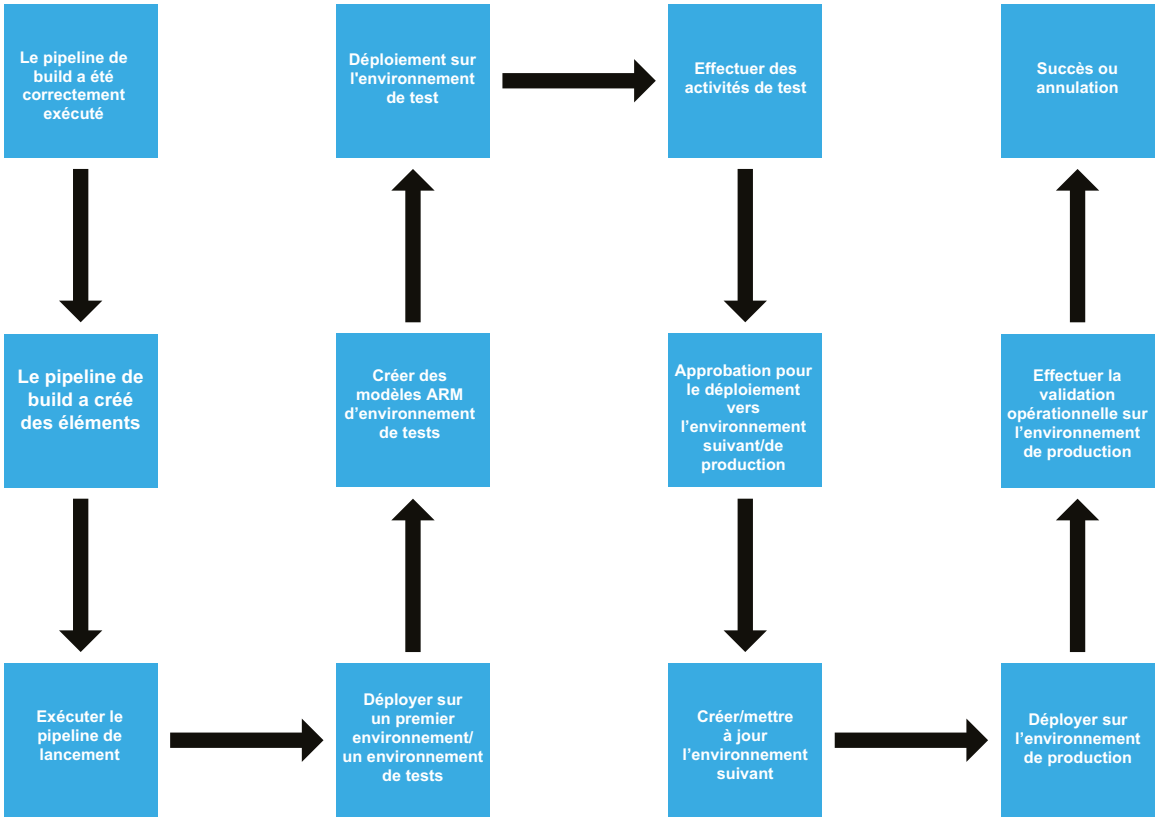


Figure 13.7 : Processus de publication

Dans le parcours DevOps, il est important de comprendre et de mettre en service les composants et les services communs avant de se lancer dans un engagement, produit ou service logiciel.

La première étape de la mise en route d'Azure DevOps consiste à configurer une organisation.

Organisations Azure DevOps

Un système de contrôle de version est nécessaire pour collaborer au niveau du code. Azure DevOps fournit des versions centralisées et décentralisées des systèmes de contrôle. Azure DevOps fournit également des services d'orchestration pour bâtir et exécuter des pipelines de version et de mise en production. Il s'agit d'une plateforme mature qui permet d'organiser le contrôle de toutes les versions liées à DevOps, aux publications et aux éléments de travail connexes. Une fois qu'une organisation est configurée dans Azure DevOps, un projet Azure DevOps doit être créé pour contenir tous les éléments liés au projet.

Une organisation DevOps Azure peut être mise en service en visitant https://dev.azure.com.

Une organisation Azure DevOps est la limite administrative et de gestion de haut niveau qui assure la sécurité, l'accès et la collaboration entre les membres d'une équipe appartenant à une organisation. Il peut y avoir plusieurs projets au sein d'une organisation et chaque projet comprend plusieurs équipes.

Mise en service d'Azure Key Vault

Il est déconseillé de stocker des secrets, des certificats, des données d'identification et toute autre information sensible dans les fichiers de configuration du code, les bases de données ou dans tout autre système de stockage général. Il est conseillé de stocker ces données importantes dans une chambre forte spécifiquement conçue pour stocker des secrets et des informations d'identification. Azure Key Vault fournit un tel service. Azure Key Vault est disponible en tant que ressource et en tant que service auprès d'Azure. Passons maintenant à l'exploration des options de stockage pour les configurations.

Mise en service d'un serveur/service de gestion de configuration

Un serveur/service de gestion de configuration fournissant un stockage des configurations ainsi que l'application de ces configurations aux différents environnements est toujours une bonne stratégie pour automatiser les déploiements. DSC sur des machines virtuelles personnalisées, DSC sur Azure Automation, Chef, Puppet et Ansible sont autant d'options qui peuvent être utilisées en toute transparence sur Azure, tant pour des environnements Windows que Linux. Ce manuel utilise DSC comme outil de gestion de configuration à toutes fins, et il fournit un serveur pull qui contient tous les documents de configuration (fichiers MOF) pour l'exemple d'application. Il gère également la base de données de toutes les machines virtuelles et les conteneurs qui sont configurés et enregistrés avec le serveur pull, afin d'y extraire les documents de configuration. Le gestionnaire de configuration local sur ces machines virtuelles et conteneurs cibles vérifie périodiquement la disponibilité de nouvelles configurations et détecte toute dérive éventuelle dans la configuration actuelle, qui sera alors signalée au serveur pull. Il dispose également de capacités de reporting intégrées qui fournissent des informations sur les nœuds conformes et non conformes au sein d'une machine virtuelle. Un serveur pull est une application web générale qui héberge un point de terminaison de serveur pull DSC. Nous discuterons ensuite d'une technique de surveillance des processus en temps réel avec Log Analytics.

Log Analytics

Log Analytics est un service d'audit et de surveillance fourni par Azure qui permet d'obtenir des informations en temps réel sur toutes les modifications, dérives et événements survenus au sein des machines virtuelles et des conteneurs. Il fournit un espace de travail et un tableau de bord centralisés pour les administrateurs IT, qui peuvent ainsi consulter, rechercher et affiner les résultats d'une recherche sur toute modification, faille et événement survenu(e) sur ces machines virtuelles. Il fournit également des agents qui sont déployés sur les machines virtuelles et les conteneurs cibles. Une fois déployés, ces agents commencent à envoyer la totalité des modifications, événements et failles à l'espace de travail centralisé. Examinons les options de stockage pour le déploiement de plusieurs applications.

Comptes Azure Storage

Azure Storage est un service fourni par Azure qui permet de stocker des fichiers comme objets blob. Tous les scripts et le code destinés à l'automatisation de la mise en service, du déploiement et de la configuration de l'infrastructure, ainsi que l'exemple d'application, sont stockés dans le référentiel Azure DevOps Git, et empaquetés et déployés dans un compte Azure Storage. Azure fournit des ressources d'extension de script PowerShell qui peuvent télécharger automatiquement les scripts PowerShell et DSC et les exécuter sur des machines virtuelles lors de l'exécution de modèles ARM. Ce stockage agit comme un espace de stockage commun à tous les déploiements pour plusieurs applications. Le stockage de scripts et de modèles dans un compte de stockage garantit qu'ils peuvent être utilisés dans tous les projets quels qu'ils soient dans Azure DevOps. Explorons l'importance des images dans la section suivante.

Images Docker du système d'exploitation

Les images de la machine virtuelle et du conteneur doivent être générées dans le cadre du pipeline de version et de mise en production des services communs. Des outils tels que Packer et Docker Build peuvent être utilisés pour générer ces images.

Outils de gestion

Tous les outils de gestion, comme Kubernetes, DC/OS, Docker Swarm et ITIL doivent être mis en service avant la création et le déploiement de la solution.

Pour conclure cette section sur la préparation DevOps, explorons les outils de gestion. Plusieurs choix sont possibles pour chaque activité au sein d'un écosystème DevOps et ils doivent être activés dans le cadre du parcours DevOps. Cette étape ne doit pas être reléguée au second plan, mais doit plutôt faire partie de la planification DevOps

DevOps pour solutions PaaS

L'architecture type pour les services d'application Azure PaaS est basée sur la *figure* 13.8 :

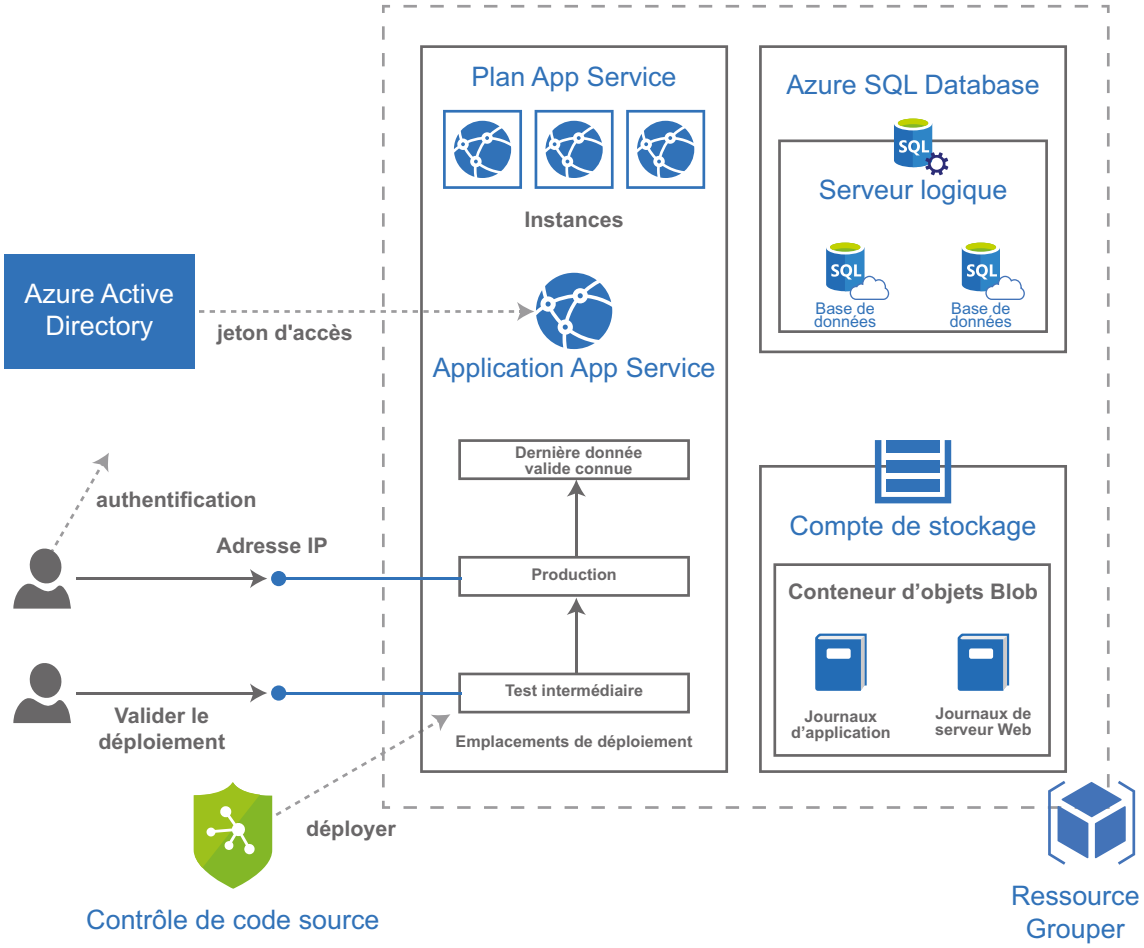


Figure 13.8 : Architecture type d'un service d'application Azure PaaS

L'architecture illustre certains composants essentiels, tels qu'Azure SQL, les comptes de stockage et le système de contrôle des versions, lesquels font partie de l'architecture de la solution dans le Cloud basée sur Azure App Service. Ces éléments doivent être créés à l'aide de modèles ARM. Ces modèles ARM doivent être intégrés à la stratégie globale de gestion de la configuration. Ils peuvent disposer de leurs propres pipelines de gestion des versions et de la mise en production, qui ressemblent aux pipelines présentés à la *figure* 13.9 :

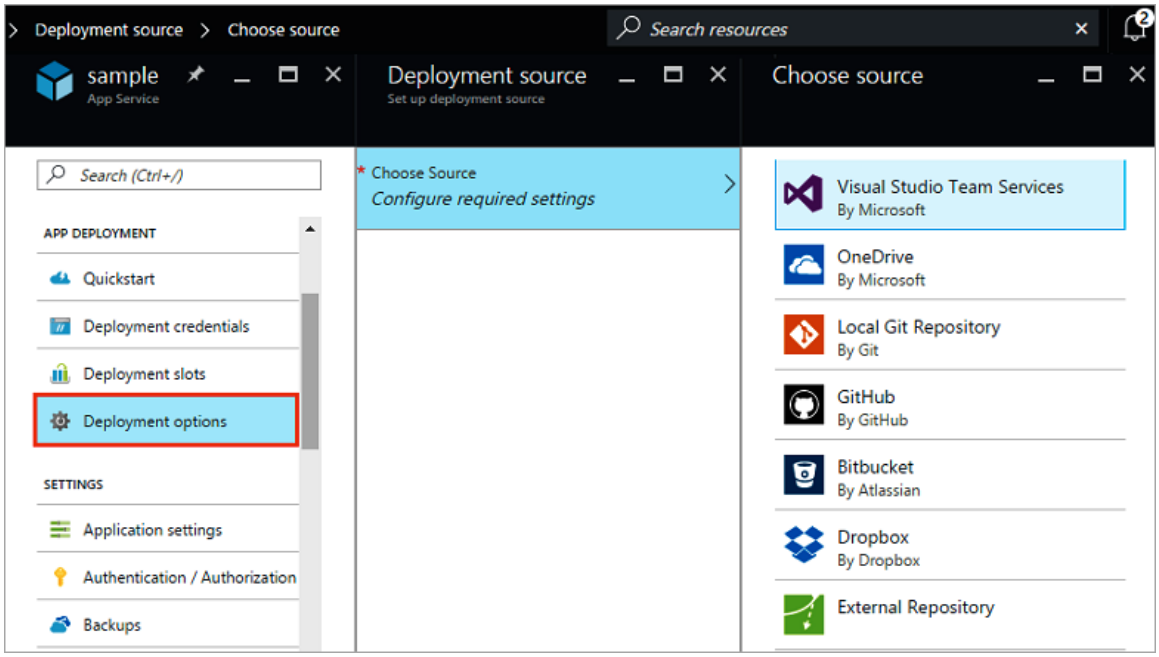


Figure 13.9 : Choix des options de déploiement pour le service d'application

Maintenant que nous avons exploré les différentes options de source de déploiement, découvrons comment déployer des solutions dans le Cloud sur Azure.

Azure App Service

Azure App Service fournit des services d'hébergement gérés pour des solutions dans le Cloud. Il s'agit d'une plateforme entièrement gérée en vue de la mise en service et du déploiement de solutions Cloud. Azure App Service élimine les tâches fastidieuses liées à la création et à la gestion des infrastructures et fournit des **contrats de niveau de service** (**SLA**) minimaux pour l'hébergement de vos solutions dans le Cloud.

Emplacements de déploiement

Azure App Service fournit des emplacements qui facilitent et simplifient les déploiements. Il existe plusieurs emplacements, et l'échange entre les emplacements se fait au niveau de DNS. Cela signifie que tout ce qui se trouve dans l'emplacement de production peut être échangé avec un emplacement de préproduction en permutant simplement les entrées DNS. Cela permet de déployer la solution Cloud personnalisée dans l'environnement de préproduction, puis, après avoir effectué toutes les vérifications et tous les tests, ceux-ci peuvent aisément être transmis à l'environnement de production en cas de réussite. Toutefois, si un problème survient dans l'environnement de production après le basculement, toutes les valeurs positives précédentes issues de l'environnement de production peuvent être restaurées, en basculant de nouveau. Examinons maintenant l'offre de base de données d'Azure et certaines de ses fonctions clés.

Azure SQL

Azure SQL est un service SQL PaaS fourni par Azure pour héberger les bases de données. Azure fournit une plateforme sécurisée afin d'héberger les bases de données et prend entièrement en charge la disponibilité, la fiabilité et l'évolutivité du service. Avec Azure SQL, il est inutile de mettre en service des machines virtuelles personnalisées, de déployer un serveur SQL Server et de le configurer. Plutôt, l'équipe Azure s'en charge en arrière-plan et les gère à votre place. Elle fournit également un service de pare-feu pour assurer la sécurité, et seule une adresse IP autorisée par le pare-feu peut se connecter au serveur et accéder à la base de données. Les machines virtuelles mises en service afin d'héberger des applications Web disposent d'adresses IP publiques qui leur sont attribuées et ajoutées aux règles de pare-feu Azure SQL de manière dynamique. Azure SQL Server et sa base de données sont créés lors de l'exécution du modèle ARM. Ensuite, nous aborderons les pipelines de build et de lancement.

Pipelines de build et de lancement

Dans cette section, un nouveau pipeline de build est créé. Il compile et valide une application ASP.NET MVC, puis génère des paquets pour le déploiement. Après la génération du paquet, une définition de version garantit que le déploiement vers le premier environnement se produit dans App Service et Azure SQL dans le cadre du déploiement continu.

Il existe deux façons de créer des pipelines de build et de lancement :

1. Utiliser l'éditeur classique
2. Utiliser des fichiers YAML

Les fichiers YAML offrent plus de flexibilité pour la création de pipelines de build et de lancement.

La structure de projet de l'exemple d'application est illustrée à la *figure 13.10* :

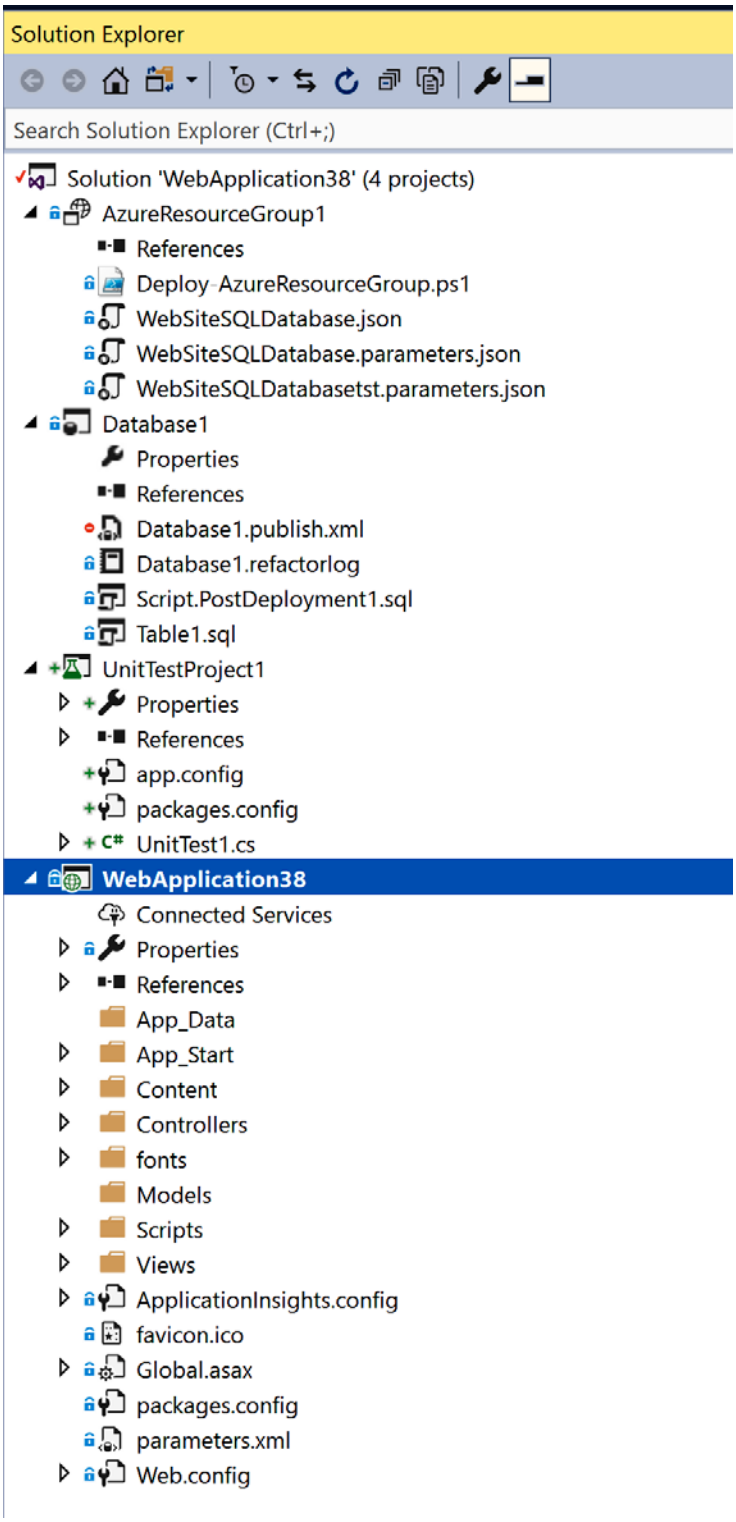


Figure 13.10 : Structure de projet d'un exemple d'application

Dans le cadre de ce projet, il y a une application ASP.NET MVC. Il s'agit de l'application principale, qui se compose de pages d'application. Les paquets Web Deploy sont générés depuis ce projet à partir des pipelines de build et ils finissent sur Web Apps. D'autres projets font également partie de la solution, comme indiqué ci-après :

- **Projet de test unitaire** : code pour le test unitaire de l'application ASP.NET MVC. Les assemblages de ce projet seront générés et exécutés dans l'exécution de la build.
- **Projet de base de données SQL** : code lié au schéma, à la structure et aux données principales de la base de données SQL. Les fichiers **DacPac** seront générés à partir de ce projet à l'aide de la définition de version.
- **Projet Azure Resource Group** : modèles ARM et code de paramètre pour mettre en service l'intégralité de l'environnement Azure sur lequel l'application ASP.NET MVC et les tables SQL sont créées.

Le pipeline de build est illustré à la *figure 13.11* :

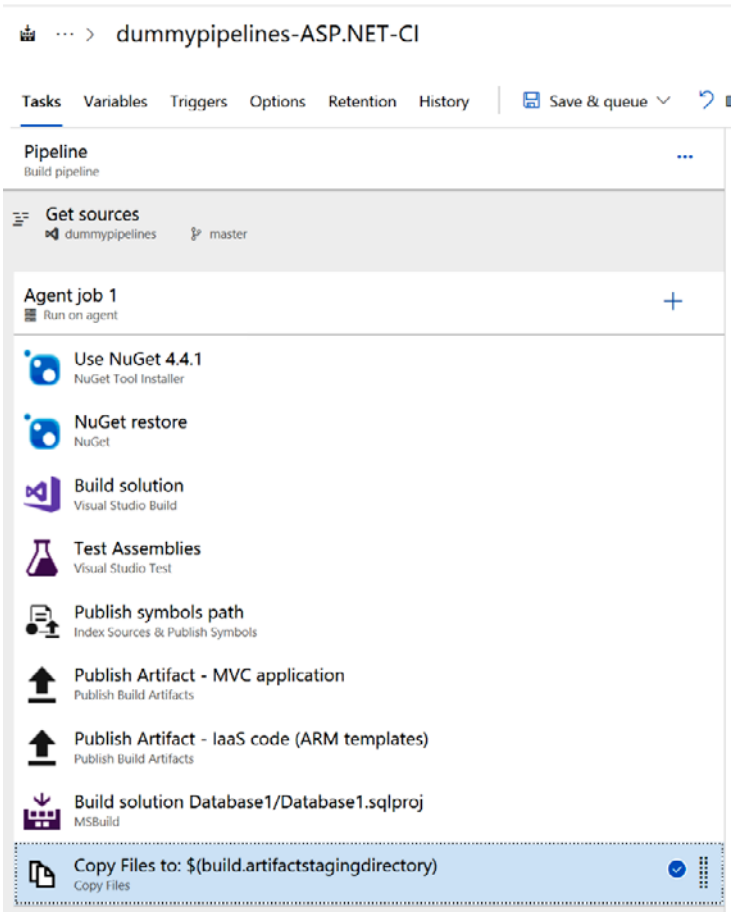
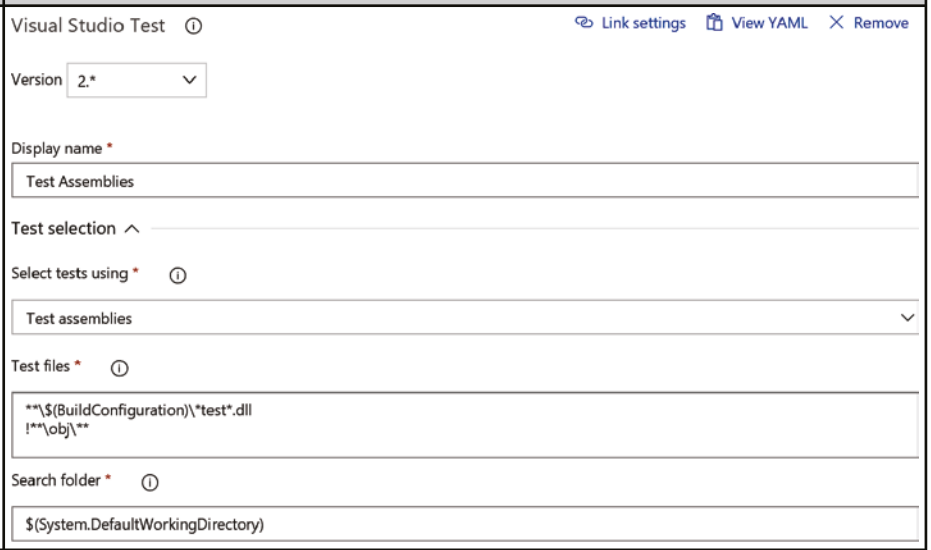
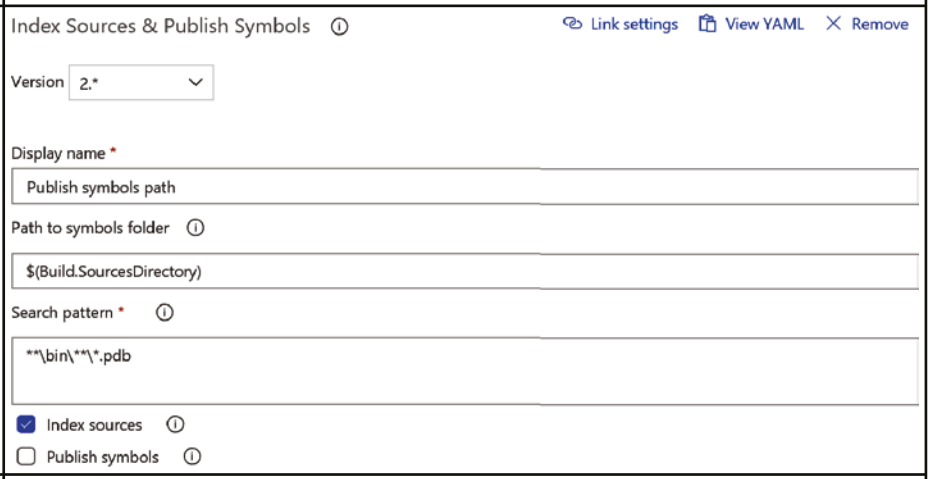
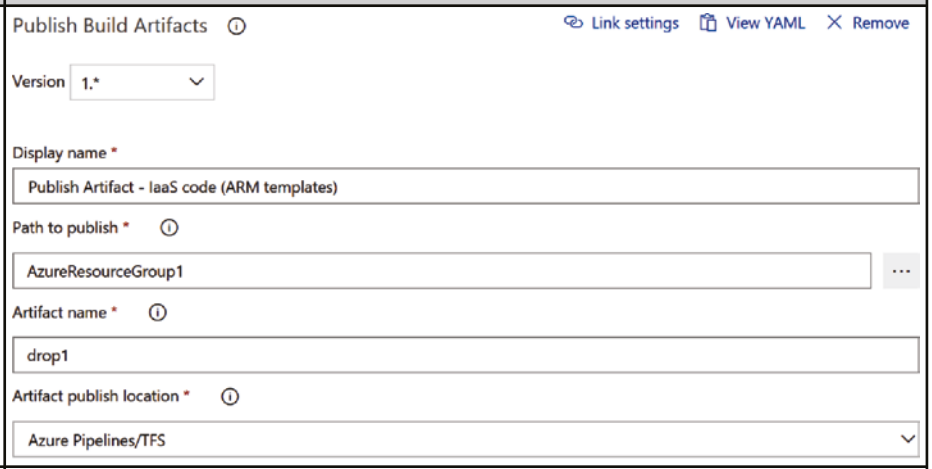
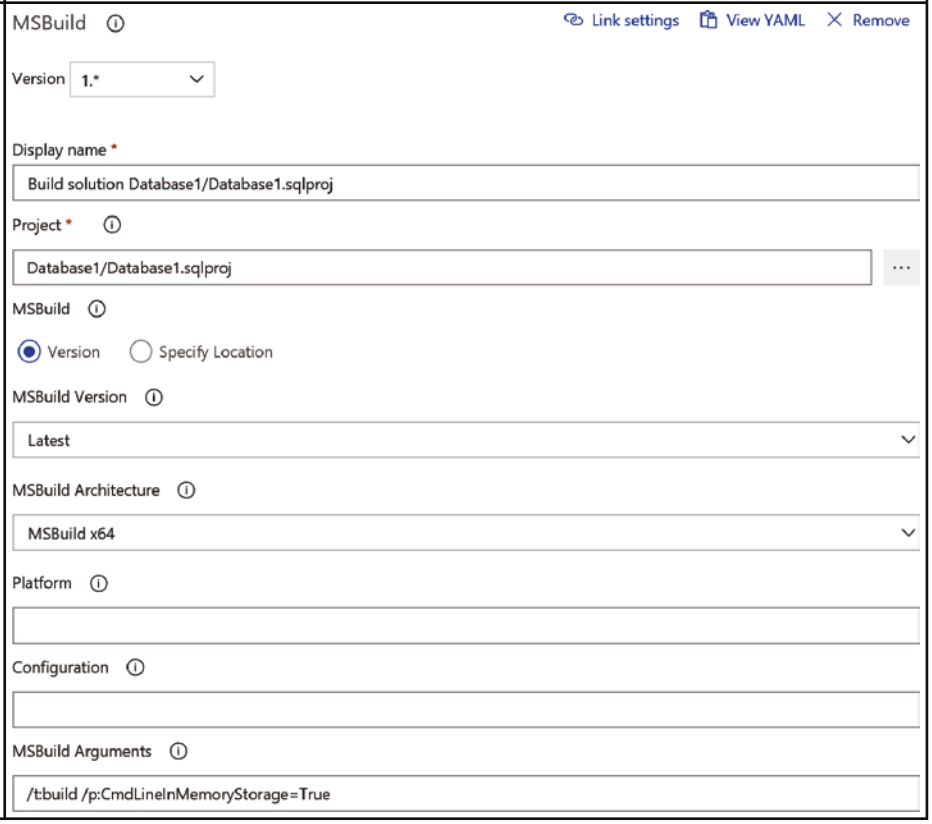


Figure 13.11 : Pipeline de build de l'application ASP.NET MVC

La configuration de chaque tâche est illustrée dans le *tableau* 13.2 :

Nom de la tâche	Configuration de la tâche
`Utiliser NuGet 4.4.1`	NuGet Tool Installer Link settings View YAML Remove Version 0.* Display name * Use NuGet 4.4.1 Version of NuGet.exe to install * 4.4.1 Always download the latest matching version Control Options Output Variables
`Restaurer NuGet`	NuGet Link settings View YAML Remove Version 2.* Display name * NuGet restore Command * restore Path to solution, packages.config, or project.json * ***.sln
`Solution de build`	Visual Studio Build Link settings View YAML Remove Version 1.* Display name * Build solution Solution * WebApplication38/WebApplication38.csproj Visual Studio Version Latest MSBuild Arguments /p:DeployOnBuild=true /p:WebPublishMethod=Package /p:PackageAsSingleFile=true /p:SkipInvalidConfigurations=true /p:PackageLocation="$(build.artifactstagingdirectory)\\" Platform $(BuildPlatform) Configuration $(BuildConfiguration)

Nom de la tâche	Configuration de la tâche
Assemblys de test	
Publier le chemin des symboles	
Publier un élément - Application MVC	

Nom de la tâche	Configuration de la tâche
`Publier un élément - Code IaaS (modèles ARM)`	
`Build solution Database1/ Database1 .sqlproj`	

Nom de la tâche	Configuration de la tâche
`Copier des fichiers dans : $(build.artifactstaging directory)`	Copy Files Link settings View YAML Remove Version 2.* Display name * Copy Files to: $(build.artifactstagingdirectory) Source Folder Contents * ***.dacpac Target Folder * $(build.artifactstagingdirectory)

Tableau 13.2 : Configuration des tâches de pipeline de build

Le pipeline de build est configuré pour s'exécuter automatiquement dans le cadre de l'intégration continue, comme illustré à la *figure* 13.12 :

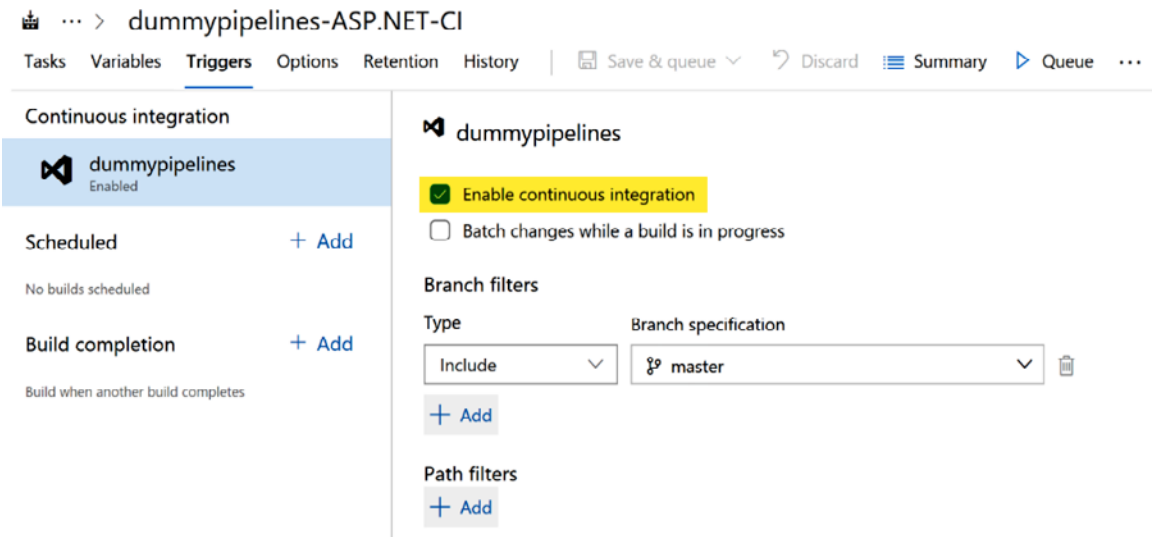


Figure 13.12 : Activation de l'intégration continue dans le pipeline de build

La définition de version se compose de plusieurs environnements, tels que le développement, les tests, **les tests d'intégration de système** (**SIT**), **les tests d'acceptation par les utilisateurs** (**UAT**), la préproduction et la production. Les tâches sont assez similaires dans chaque environnement, avec l'ajout de tâches spécifiques à cet environnement. Par exemple, un environnement de tests contient des tâches supplémentaires liées à l'interface utilisateur, ainsi que des tests fonctionnels et d'intégration, par rapport à un environnement de développement.

La définition de version pour cette application est illustrée à la *figure* 13.13 :

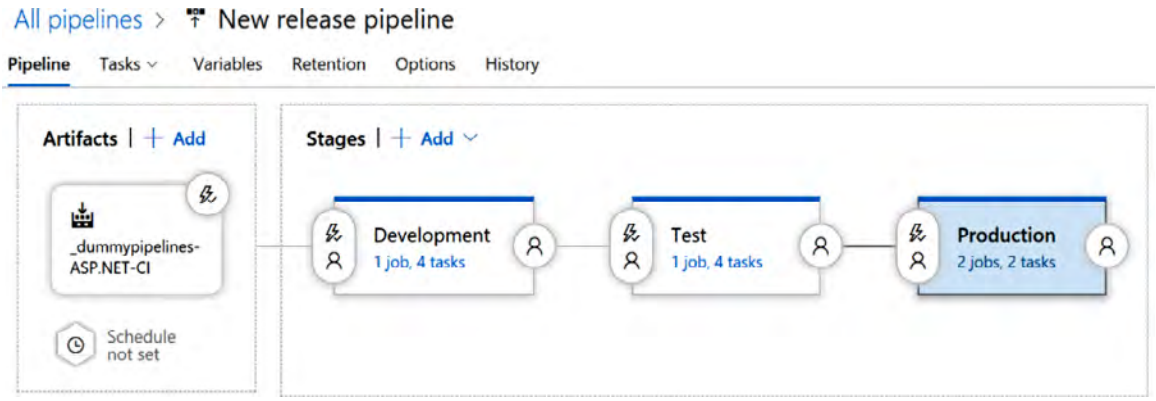


Figure 13.13 : Définition de version

Les tâches de version pour un environnement unique sont illustrées à la *figure* 13.14 :

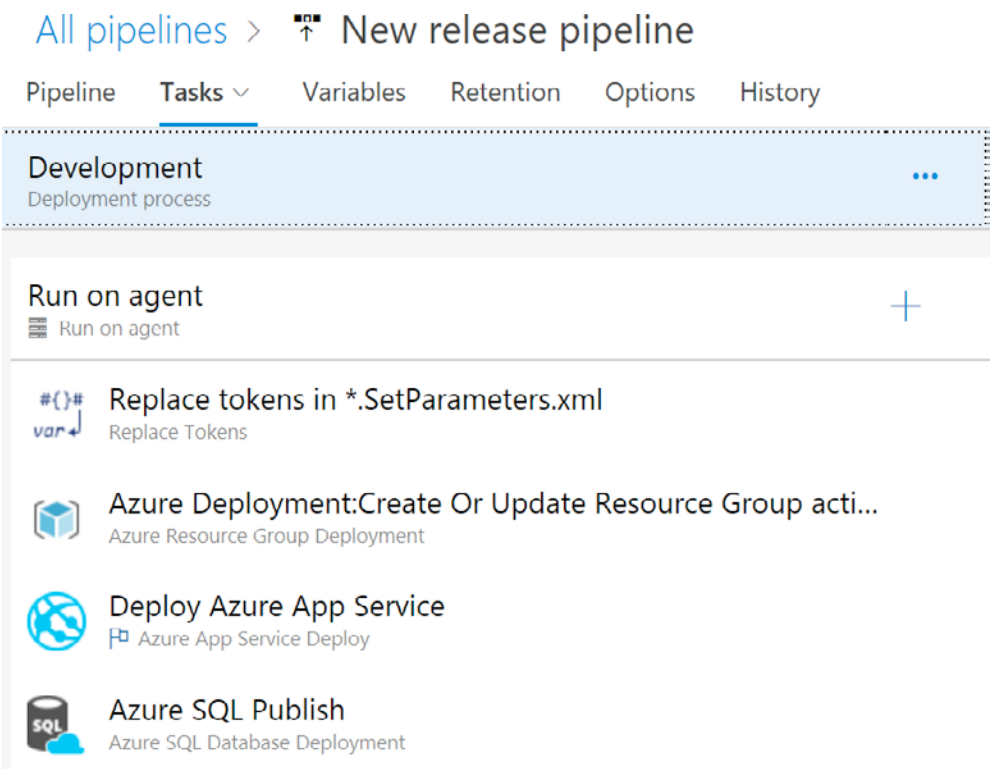
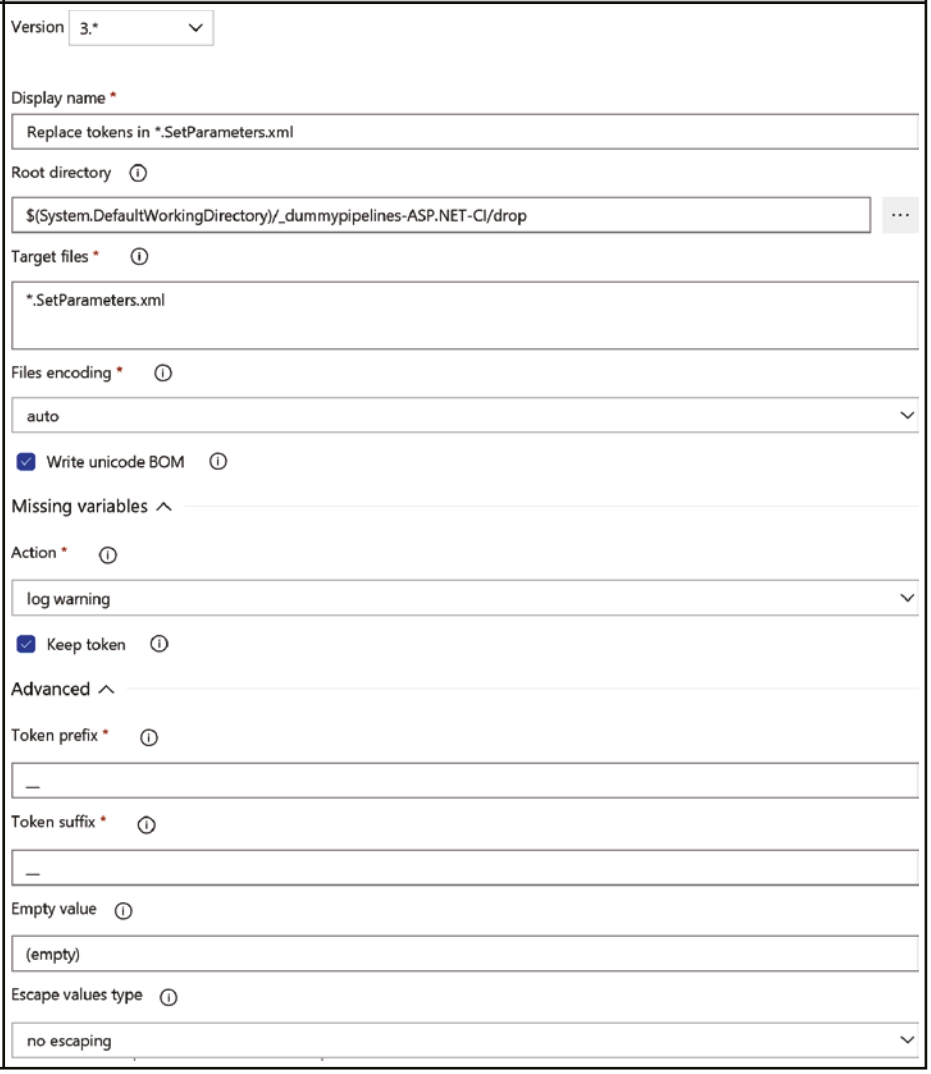
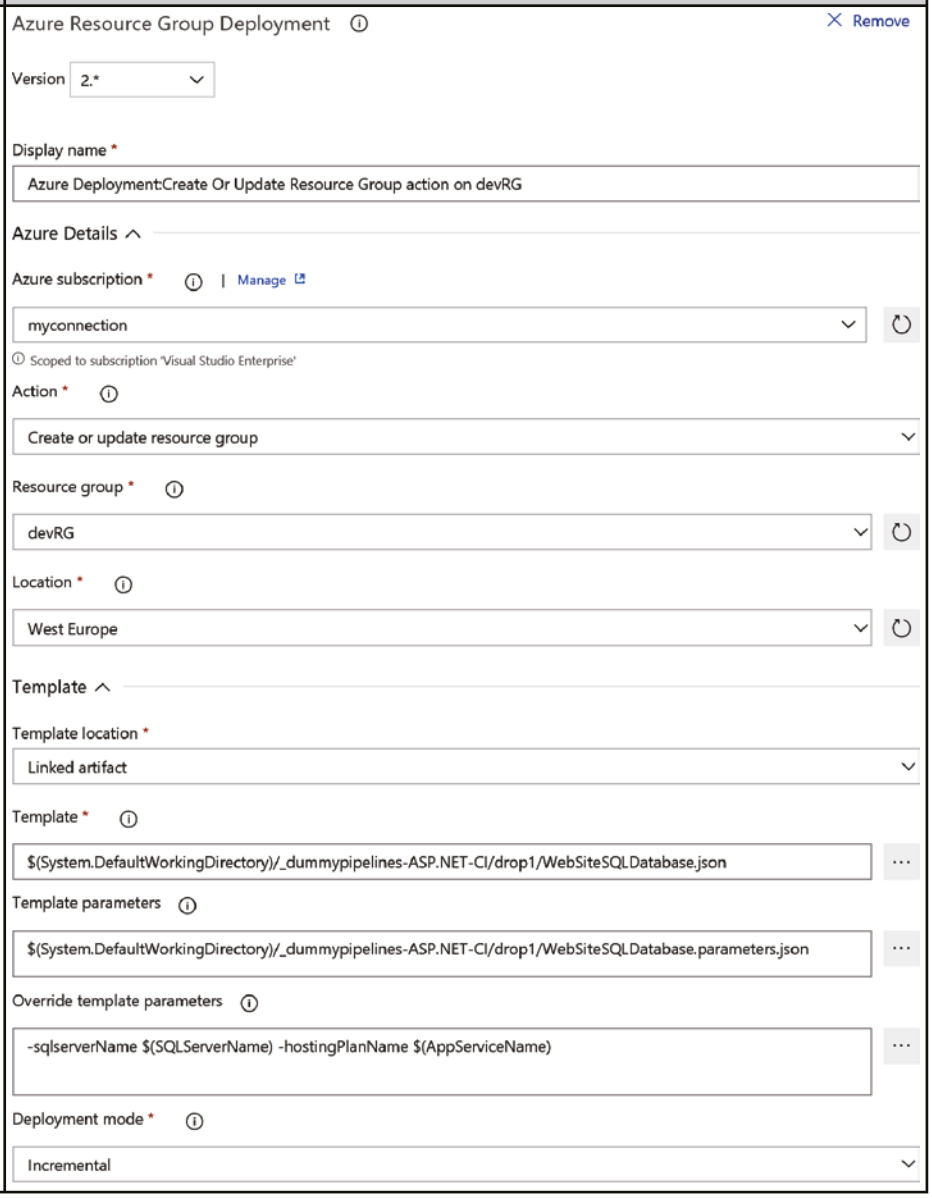
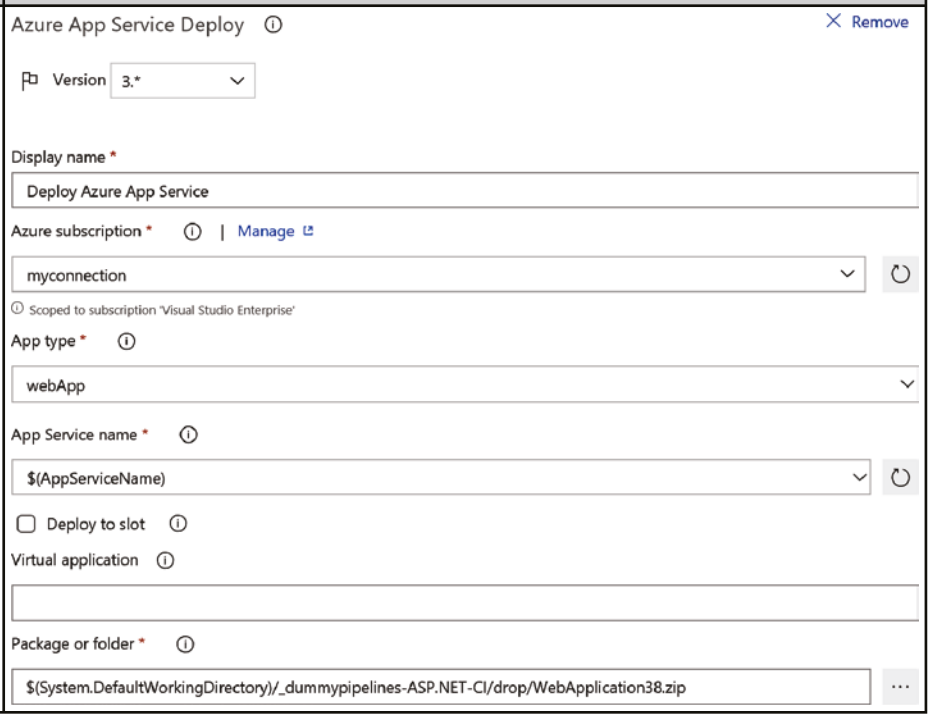


Figure 13.14 : Tâches de version d'un environnement unique

La configuration pour chacune des tâches est répertoriée ici :

Nom de la tâche	Configuration de la tâche
`Remplacez les jetons dans *.SetParameters.xml` (Il s'agit d'une tâche installée à partir de MarketPlace.)	

Nom de la tâche	Configuration de la tâche
Déploiement Azure : créer ou mettre à jour l'action du groupe de ressources sur devRG	

Nom de la tâche	Configuration de la tâche
`Déployer Azure App Service`	

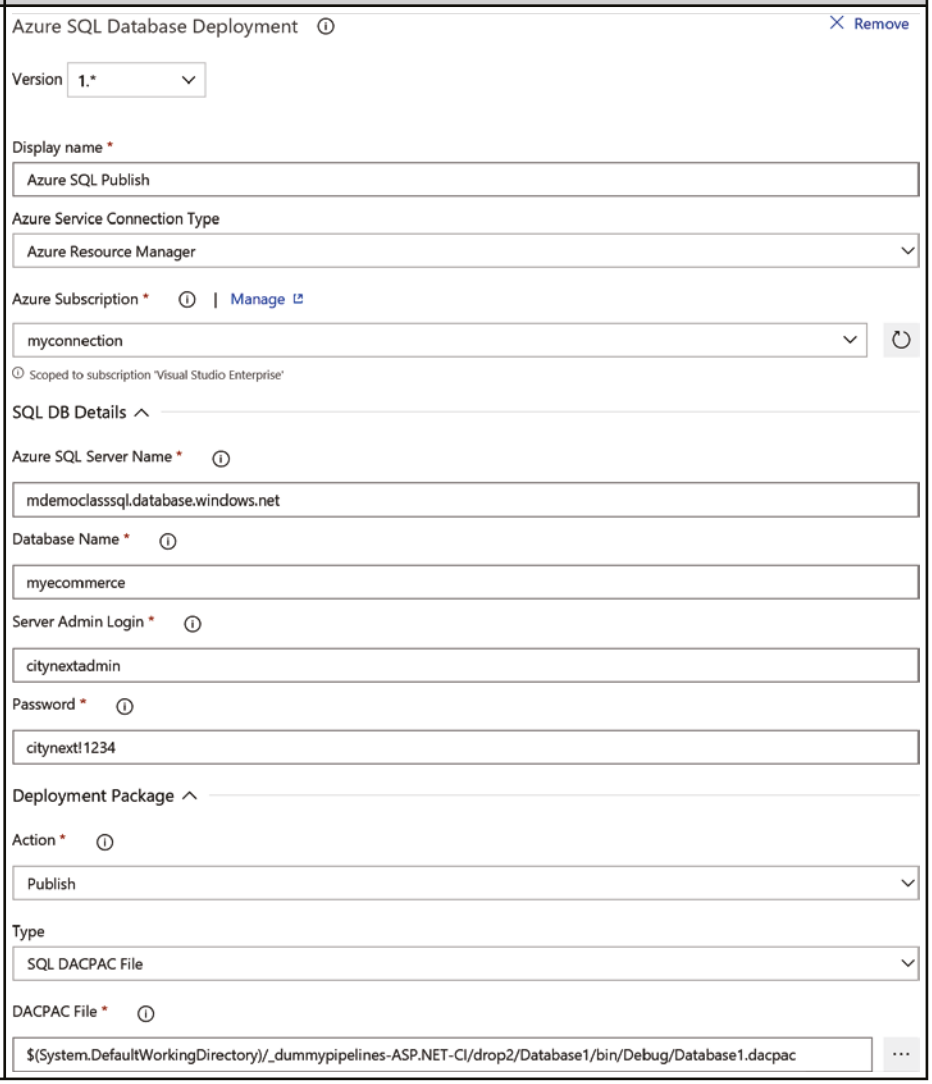
Nom de la tâche	Configuration de la tâche
`Azure SQL Publish`	

Tableau 13.3 : Configuration des tâches du pipeline de lancement

Dans cette section, vous avez vu comment configurer des pipelines de build et de publication dans Azure DevOps. Dans la section suivante, l'accent sera mis sur différentes architectures, telles que IaaS, les conteneurs et différents scénarios.

DevOps pour IaaS

IaaS implique la gestion et l'administration conjointes de l'infrastructure de base et des applications. Plusieurs ressources et composants doivent être mis en service, configurés et déployés sur plusieurs environnements. Il est important de comprendre l'architecture avant de poursuivre.

L'architecture type d'une solution de machine virtuelle IaaS est illustrée ici :

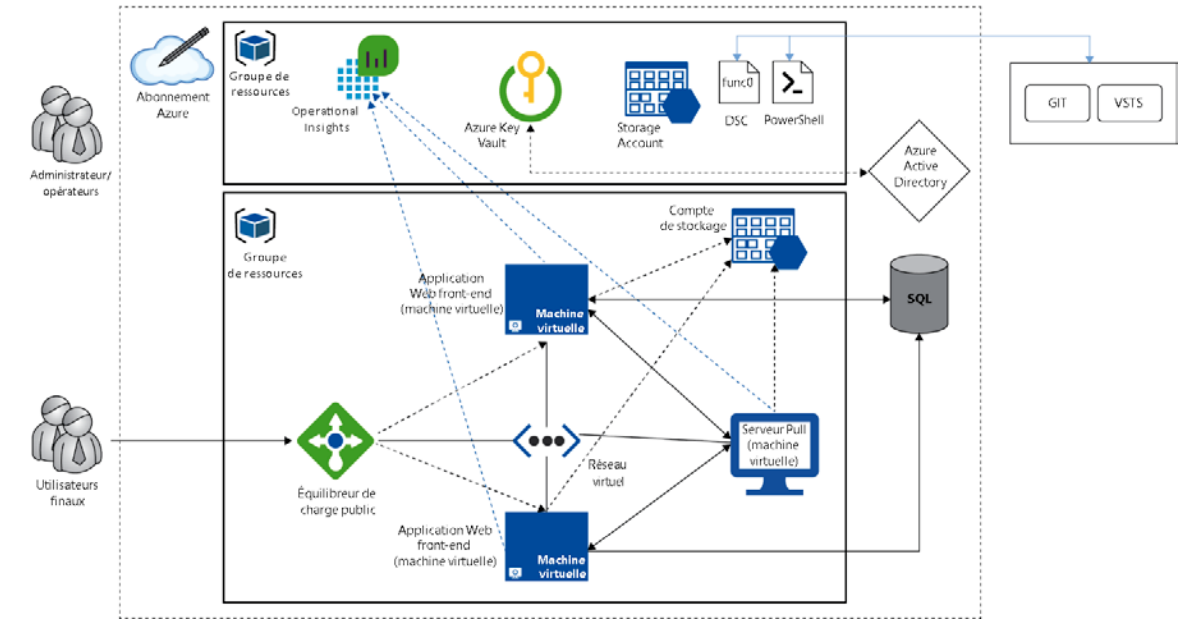


Figure 13.15 : Architecture d'une solution basée sur des machines virtuelles IaaS

Chacun des composants répertoriés dans l'architecture est abordé dans la section suivante.

Machines virtuelles Azure

Des machines virtuelles Azure qui hébergent des applications web, des serveurs d'application, des bases de données et d'autres services sont mises en service à l'aide des modèles ARM. Elles sont connectées à un réseau virtuel, qui leur attribue une adresse IP privée. L'adresse IP publique des machines virtuelles est facultative, car elles sont reliées à un équilibreur de charge public. Des agents Operational Insights sont installés sur des machines virtuelles afin de les surveiller. Des scripts PowerShell sont également exécutés sur les machines virtuelles téléchargées à partir d'un compte de stockage disponible dans un autre groupe de ressources pour ouvrir les ports de pare-feu pertinents, télécharger les packages appropriés et installer des certificats locaux afin de sécuriser l'accès au moyen de PowerShell. L'application Web est configurée de sorte à s'exécuter sur le port fourni de ces machines virtuelles. Le numéro de port de l'application Web et toute sa configuration sont extraits d'un serveur pull DSC et attribués de manière dynamique.

Équilibreurs de charge publics Azure

Un équilibreur de charge public est relié à certaines machines virtuelles afin de leur envoyer des requêtes de manière cyclique. Cela est généralement nécessaire pour les applications web frontales et les API. Une adresse IP publique et un nom DNS peuvent être attribués à l'équilibreur de charge pour lui permettre de diffuser des requêtes Internet. Il accepte les requêtes web HTTP sur un port différent et les achemine aux machines virtuelles. Il sonde également les protocoles HTTP de certains ports en s'appuyant sur des chemins d'application fournis. Les règles **Network Address Translation** (**NAT**) peuvent également être appliquées afin de permettre la connexion aux machines virtuelles via des bureaux distants.

Azure Application Gateway est une ressource alternative à l'équilibreur de charge public Azure. Les passerelles d'application sont des équilibreurs de charge de couche 7 et fournissent des fonctions telles que la terminaison SSL, l'affinité de session et le routage basé sur l'URL. Nous allons examiner en détail le pipeline de build dans la section suivante.

Pipeline de build

Un pipeline de build type d'une solution de machine virtuelle IaaS est illustré ci-après. Un pipeline de lancement est créé dès lors qu'un développeur publie son code dans le référentiel. Le pipeline de build démarre automatiquement dans le cadre de l'intégration continue. Il compile et génère le code, exécute des tests unitaires sur celui-ci, contrôle sa qualité et crée des documents à partir des commentaires relatifs au code. Il déploie les nouveaux binaires dans l'environnement de développement (notez que cet environnement de développement n'est pas nouvellement créé), modifie la configuration, exécute des tests d'intégration et génère des étiquettes de conception, pour faciliter l'identification. Il dépose ensuite les éléments générés dans un emplacement accessible au pipeline de lancement. En cas de problème durant l'exécution de l'une des étapes de ce pipeline, le développeur sera informé via le feedback du pipeline de build, ce qui lui permettra de procéder à des modifications en conséquence et de les publier à nouveau. Ce pipeline de build doit présenter un résultat d'échec ou de réussite selon la gravité des erreurs détectées, ce qui dépend de chaque organisation : Un pipeline de build classique est illustré à la *figure 13.16* :

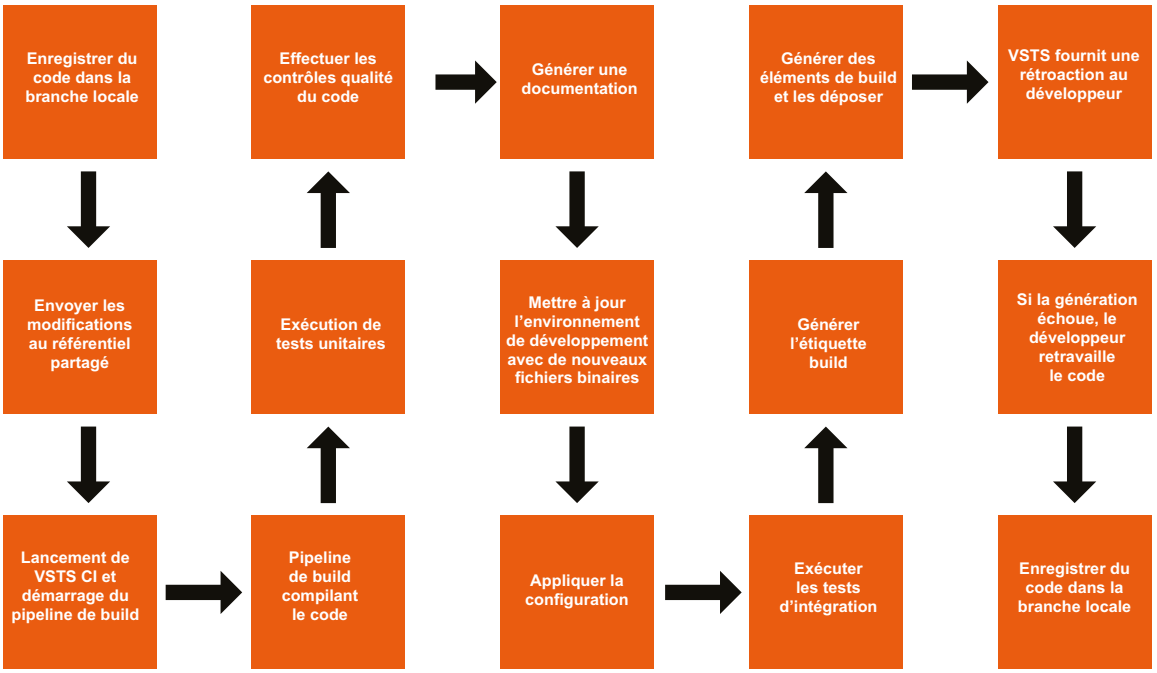


Figure 13.16 : Pipeline de build IaaS type

Après avoir abordé le pipeline de build, nous allons découvrir la mise en œuvre d'un pipeline de lancement.

Pipeline de lancement

Un pipeline de lancement type d'un déploiement de machine virtuelle IaaS est illustré ci-après. Un pipeline de lancement démarre après l'achèvement du pipeline de build. La première étape du pipeline de lancement consiste à recueillir les éléments générés par le pipeline de build. Il s'agit généralement de documents d'assemblage, et de fichiers binaires et de configuration déployables. Le pipeline de lancement s'exécute et crée ou met à jour le premier environnement, qui est généralement un environnement de test. Il utilise des modèles Azure Resource Manager pour configurer tous les services et toutes les ressources IaaS et PaaS sur Azure, et les configure également. Il permet également l'exécution de scripts et de la configuration DSC une fois que les machines virtuelles ont été créées en tant qu'étapes post-création. Cela permet de configurer l'environnement au sein de la machine virtuelle et du système d'exploitation. À ce stade, les binaires d'application provenant du pipeline de build sont déployés et configurés. Différents tests automatisés sont réalisés pour vérifier que la solution fonctionne et si le résultat est jugé satisfaisant, le pipeline passe au déploiement vers l'environnement suivant, après avoir obtenu les autorisations nécessaires. Les mêmes étapes sont exécutées dans l'environnement suivant, y compris dans l'environnement de production. Enfin, des tests de validation opérationnelle sont exécutés en production afin de garantir que l'application fonctionne comme prévu, sans écart.

À ce stade, tout problème ou bogue doit être rectifié et l'ensemble du cycle doit être répété ; toutefois, si ces opérations ne sont pas effectuées dans les délais impartis, le dernier instantané connu doit être restauré dans l'environnement de production afin de minimiser les temps d'arrêt. Un pipeline de lancement classique est illustré à la *figure* 13.17 :

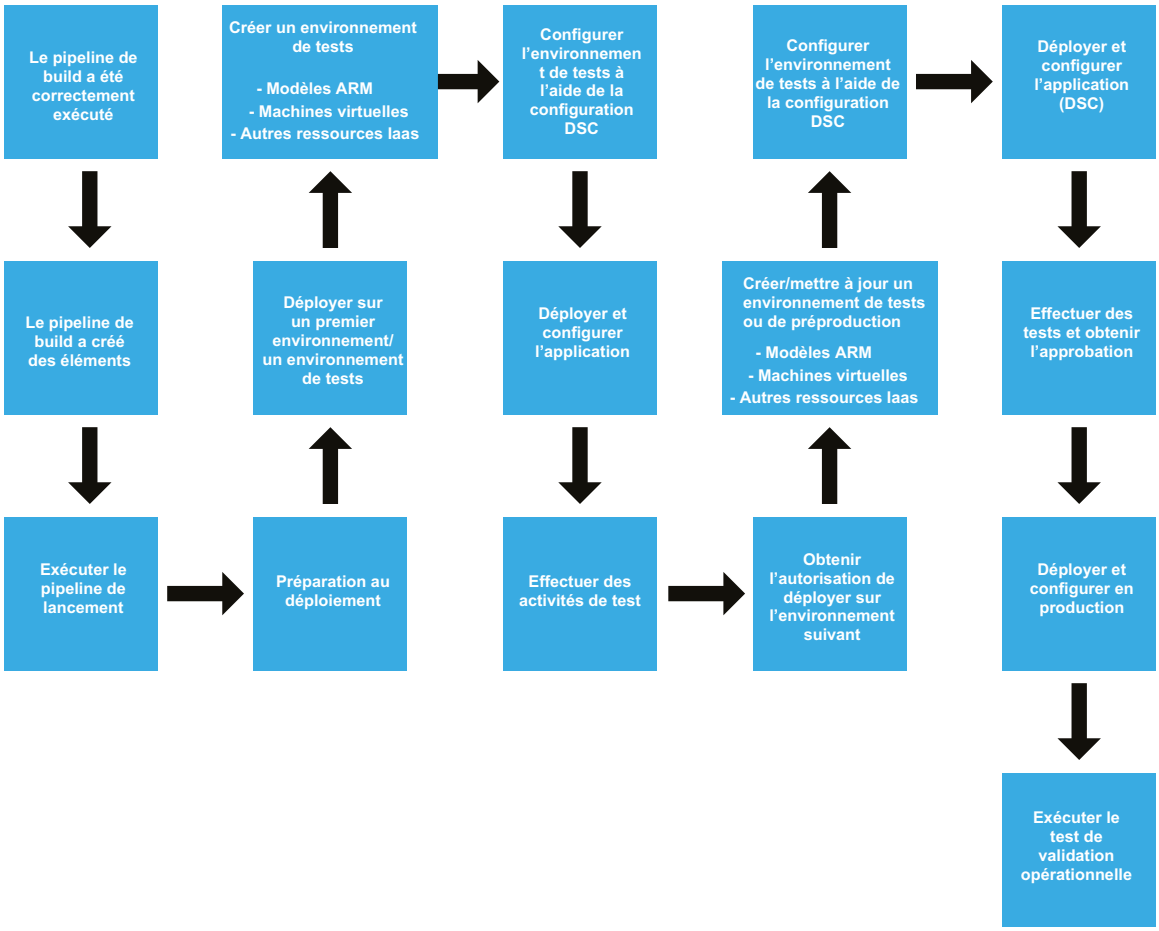


Figure 13.17 : Pipeline de lancement IaaS type

Cette section conclut le processus DevOps pour les solutions IaaS et le chapitre suivant sera consacré aux conteneurs sur les machines virtuelles. Veuillez noter que les conteneurs peuvent également s'exécuter sur une PaaS, par exemple App Service et Azure Functions.

DevOps avec des conteneurs

Dans une architecture type, les runtimes de conteneur sont déployés sur des machines virtuelles et les conteneurs sont exécutés en leur sein. L'architecture type d'une solution de conteneur IaaS est illustrée ci-après :

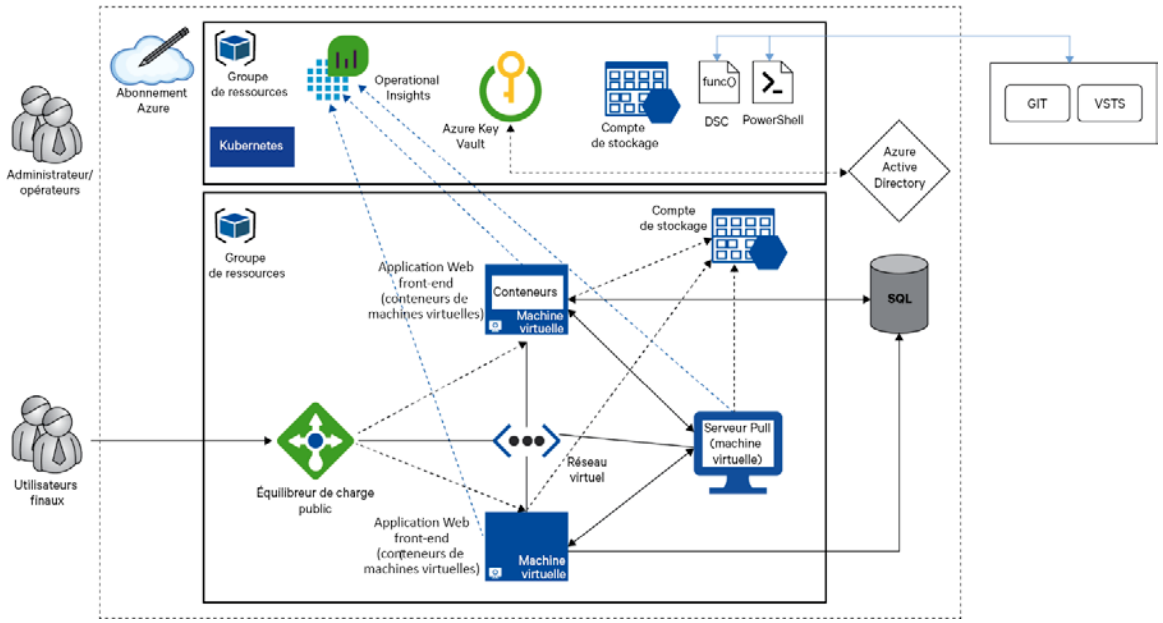


Figure 13.18 : Solutions IaaS basées sur les conteneurs

Ces conteneurs sont gérés par des orchestrateurs de conteneurs tels que Kubernetes. Les services de surveillance sont assurés par Log Analytics et tous les secrets ainsi que toutes les clés sont stockés dans Azure Key Vault. Il existe également un serveur d'extraction, qui peut se trouver sur une machine virtuelle ou sur Azure Automation, lequel fournit des informations de configuration aux machines virtuelles.

Conteneurs

Les conteneurs sont une technologie de virtualisation ; toutefois, ils ne permettent pas de virtualiser un serveur physique. En fait, les conteneurs sont une virtualisation au niveau du système d'exploitation. Par conséquent, les conteneurs partagent le noyau du système d'exploitation fourni par l'hôte, entre eux et avec l'hôte. Plusieurs conteneurs s'exécutant sur un hôte (physique ou virtuel) partagent le noyau du système d'exploitation hôte. Un seul noyau de système d'exploitation est fourni par l'hôte et utilisé par tous les conteneurs qui y sont exécutés.

Les conteneurs sont également complètement isolés de l'hôte et des autres conteneurs, plus ou moins comme une machine virtuelle. Les conteneurs utilisent des espaces de noms du système d'exploitation, et des groupes de contrôle sous Linux, pour donner l'impression d'un nouvel environnement de système d'exploitation. Ils utilisent également des techniques de virtualisation de système d'exploitation spécifiques sur Windows. Chaque conteneur obtient sa propre copie des ressources du système d'exploitation.

Docker

Docker fournit des fonctions de gestion pour les conteneurs. Il est composé de deux programmes exécutables :

- Le démon Docker
- Le client Docker

Le démon Docker constitue la force motrice de la gestion des conteneurs. Il s'agit d'un service de gestion capable de gérer toutes les activités relatives aux conteneurs sur l'hôte. Le client Docker interagit avec le démon Docker et est responsable de la capture et de la transmission d'entrées dans le démon Docker. Le démon Docker fournit le runtime, les bibliothèques, les pilotes graphiques et les moteurs capables de créer, gérer et surveiller les conteneurs et les images sur le serveur hôte. Il peut également créer des images personnalisées qui sont utilisés pour développer et livrer des applications sur plusieurs environnements.

Le Dockerfile

Le **`Dockerfile`** est un bloc constitutif fondamental qui permet de créer des images de conteneur. Il s'agit d'un fichier texte lisible sans aucune extension, qui se nomme **`Dockerfile`**. Bien qu'il existe un mécanisme pour le nommer différemment, en général, il est nommé **`Dockerfile`**. DockerFile contient les instructions de création d'une image de conteneur personnalisé à partir de l'image de base. Ces instructions sont exécutées séquentiellement de haut en bas par le démon Docker. Les instructions font référence à la commande et à ses paramètres, par exemple **`COPY`**, **`ADD`**, **`RUN`**, et **`ENTRYPOINT`**. DockerFile active des pratiques d'infrastructure en tant que code (IaC) en convertissant le déploiement d'applications et la configuration en instructions pouvant être versionnées et stockées dans un référentiel de code source. Nous allons maintenant aborder les étapes de build dans la section suivante.

Pipeline de build

Il n'existe aucune différence du point de vue de la conception entre le conteneur et une solution basée sur une machine virtuelle. L'étape de build reste la même. Un pipeline de lancement type d'un déploiement basé sur un conteneur IaaS est illustré ci-après.

Pipeline de lancement

La seule différence entre un pipeline de lancement type pour un déploiement IaaS basé sur les conteneurs et le pipeline de lancement est la gestion de l'image de conteneur et la création de conteneurs utilisant DockerFile et Docker Compose. Des utilitaires de gestion des conteneurs avancés, tels que Docker Swarm, DC/OS et Kubernetes, peuvent également être déployés et configurés dans le cadre de la gestion des publications. Toutefois, il convient de noter que ces outils de gestion des conteneurs doivent faire partie du pipeline de lancement des services partagés, comme nous l'avons déjà vu. La *figure 13.19* illustre un pipeline de lancement type pour une solution basée sur un conteneur :

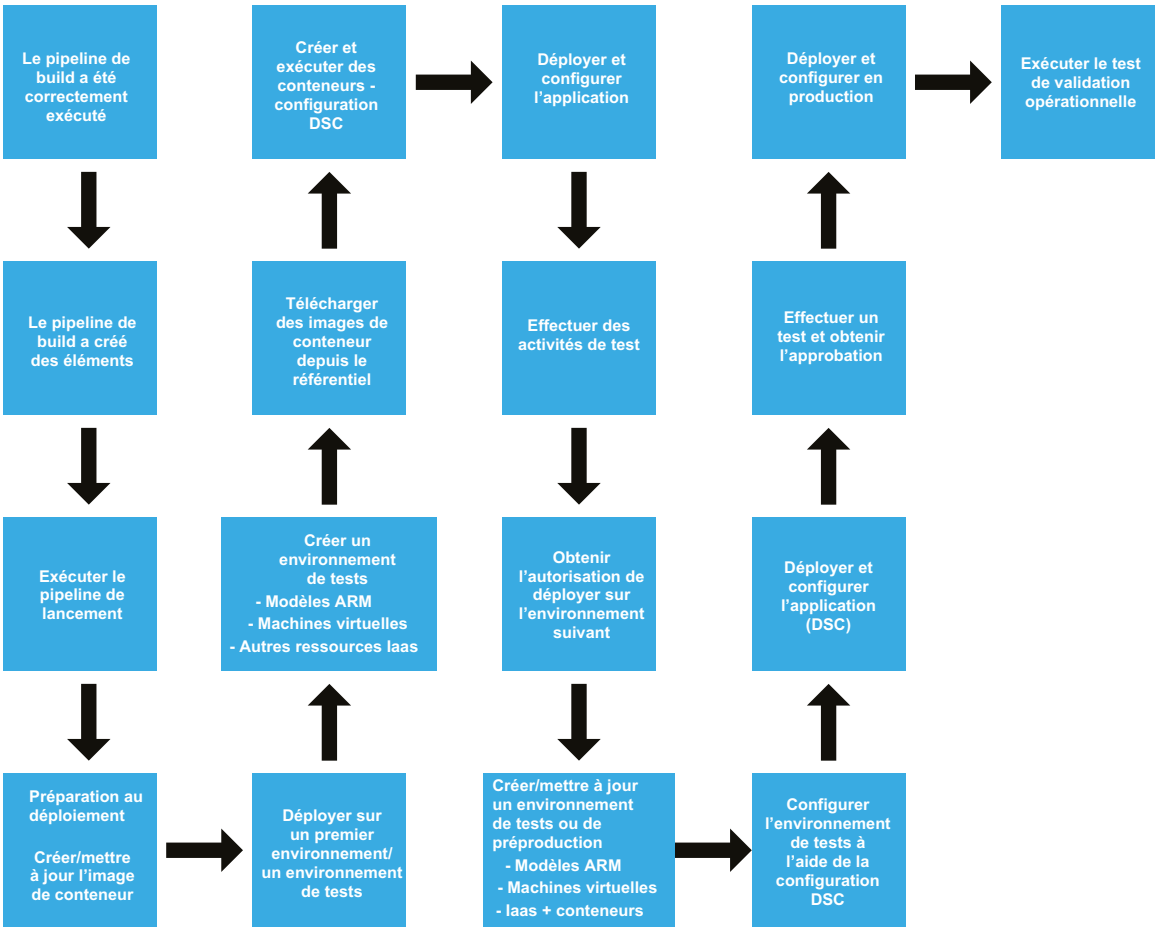


Figure 13.19 : Pipeline de lancement type basé sur des conteneurs

La section suivante aborde l'intégration avec d'autres ensembles d'outils, tels que Jenkins.

DevOps Azure et Jenkins

DevOps Azure est un orchestrateur de plateforme ouverte qui s'intègre parfaitement à d'autres outils d'orchestrateur. Cette solution fournit toutes les infrastructures et les fonctions nécessaires qui s'intègrent également avec Jenkins. Les organisations avec des pipelines CI/CD bien établis et construits sur Jenkins peuvent les réutiliser avec les fonctions avancées mais simples d'utilisation d'Azure DevOps pour les orchestrer.

Jenkins peut être utilisé comme référentiel et peut exécuter des pipelines CI/CD dans Azure DevOps. De plus, il est possible d'avoir un référentiel dans Azure DevOps et d'exécuter des pipelines CI/CD dans Jenkins.

La configuration Jenkins peut être ajoutée dans Azure DevOps en tant que crochets de service et chaque fois qu'une modification de code est validée dans le référentiel Azure DevOps, elle peut déclencher des pipelines dans Jenkins. La *figure* 13.20 illustre la configuration de Jenkins à partir de la section de configuration du crochet de service Azure DevOps :

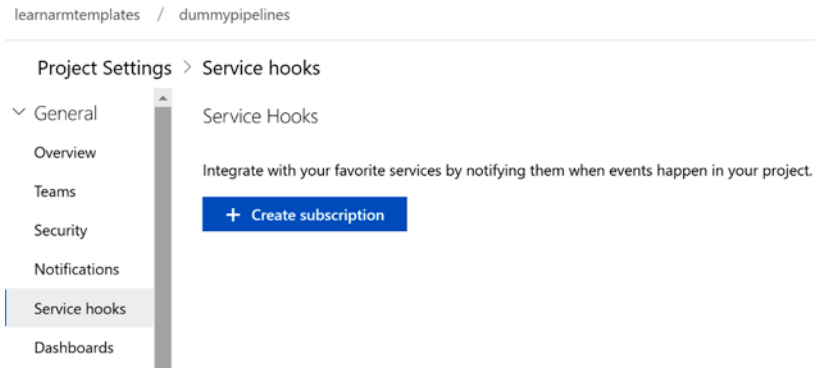


Figure 13.20 : Configuration de Jenkins

Il existe plusieurs déclencheurs qui exécutent les pipelines dans Jenkins ; l'un d'eux est **Transmis par le code**, comme illustré à la *figure* 13.21 :

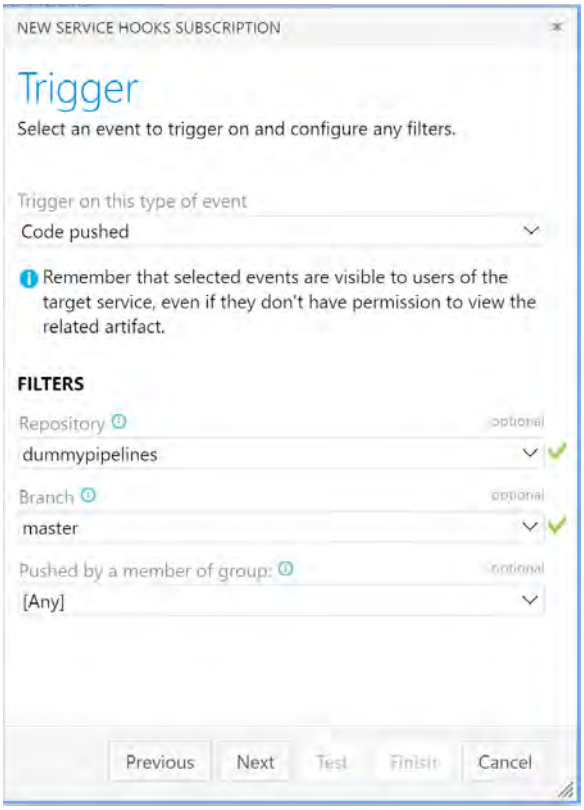


Figure 13.21 : Déclencheur transmis par le code exécuté

Il est également possible de déployer sur la machine virtuelle Azure et d'exécuter des pipelines de version DevOps Azure, tel qu'expliqué ici : https://docs.microsoft.com/azure/virtual-machines/linux/tutorial-build-deploy-jenkins

Jenkins doit déjà être déployé avant d'être utilisé dans n'importe quel scénario. Le processus de déploiement sur Linux est décrit à l'adresse https://docs.microsoft.com/azure/virtual-machines/linux/tutorial-jenkins-github-docker-cicd.

La section suivante se concentrera davantage sur les outils et les services relatifs à la gestion de la configuration. Azure Automation fournit des services DSC tels que le serveur pull.

Azure Automation

Azure Automation est la plateforme de Microsoft pour toute implémentation de l'automatisation au niveau de déploiements dans le Cloud, sur site et hybrides. Azure Automation est une plateforme d'automatisation mature qui fournit des fonctions enrichies pour les éléments suivants :

- Définition des actifs, tels que les variables, les connexions, les informations d'identification, les certificats et les modules
- Mise en œuvre de procédures opérationnelles en utilisant Python, des scripts PowerShell et des charges de travail de PowerShell
- Fourniture d'interfaces utilisateur pour créer des procédures opérationnelles
- Gestion du cycle de vie complet des procédures opérationnelles, notamment la conception, le test et la publication
- Planifier des procédures opérationnelles
- Possibilité d'exécuter des procédures opérationnelles n'importe où, sur le Cloud ou sur site
- DSC comme plateforme de gestion de la configuration
- Gestion et configuration des environnements : Windows et Linux, applications et déploiement
- Possibilité d'étendre l'automatisation Azure par importation des modules personnalisés

Azure Automation fournit un serveur pull DSC qui permet de créer un serveur de gestion de la configuration centralisée comprenant les configurations des nœuds et des machines virtuelles, ainsi que leurs composants.

Cette solution implémente le modèle hub-and-spoke dans lequel les nœuds peuvent se connecter au serveur pull DSC et télécharger des configurations qui leur ont été attribuées, puis les reconfigurer afin de refléter l'état souhaité. Toute modification ou tout écart au sein de ces nœuds est corrigé(e) automatiquement par les agents DSC lors de leur exécution suivante. De cette façon, les administrateurs n'ont pas besoin de surveiller activement l'environnement à la recherche de tout écart éventuel.

DSC fournit un langage déclaratif dans lequel vous définissez l'intention et la configuration, mais pas la manière d'exécuter et d'appliquer ces configurations. Ces configurations sont basées sur le langage PowerShell et facilitent le processus de gestion de la configuration.

Dans cette section, nous allons étudier une implémentation simple d'Azure Automation DSC afin de configurer une machine virtuelle pour installer et configurer le serveur web (IIS) et créer un fichier **`index.htm`** informant les utilisateurs que le site est en maintenance.

Vous découvrirez ensuite comment configurer un compte Azure Automation.

Mise en service d'un compte Azure Automation

Créez un compte Azure Automation à partir du portail Azure ou de PowerShell au sein d'un groupe de ressources nouveau ou existant. Vous remarquerez à la *figure* 13.22 qu'Azure Automation fournit des éléments de menu pour DSC :

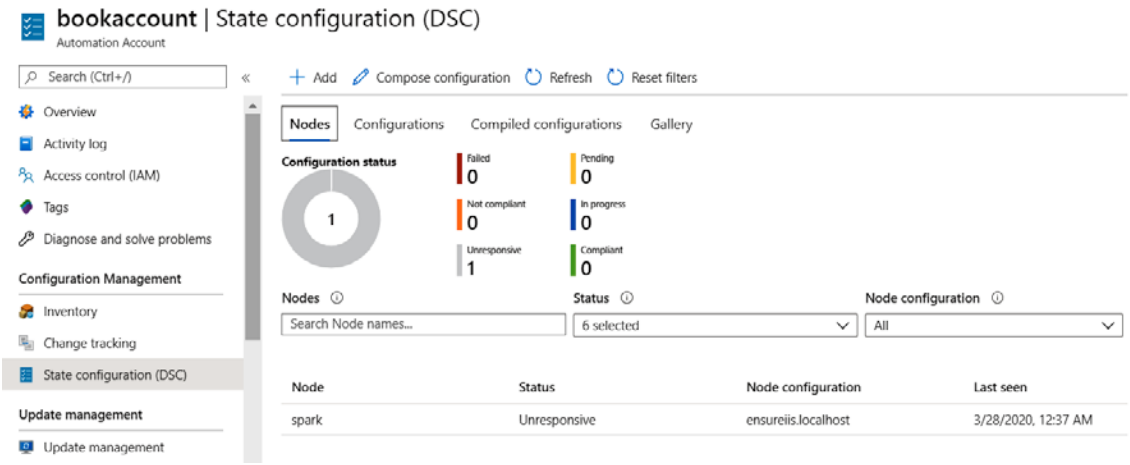


Figure 13.22 : DSC dans un compte Azure Automation

Il comprend les éléments suivants :

- **Nœuds DSC** : ceux-ci répertorient toutes les machines virtuelles et les conteneurs qui sont inscrits dans le serveur pull Azure Automation DSC. Ces machines virtuelles et conteneurs sont gérés à l'aide des configurations du serveur pull DSC actuel.
- **Configurations DSC** : celles-ci répertorient toutes les configurations PowerShell brutes importées et chargées dans le serveur pull DSC. Elles sont en format lisible et ne sont pas dans un état compilé.
- **Configurations de nœud DSC** : celles-ci répertorient toutes les compilations de configurations DSC disponibles sur le serveur pull à affecter à des nœuds, qu'il s'agisse de machines virtuelles ou de conteneurs. Une configuration DSC produit des fichiers MOF une fois les compilations effectuées, ceux-ci étant ensuite utilisés pour configurer les nœuds.

Après avoir configuré un compte Azure Automation, nous pouvons créer un exemple de configuration DSC, comme illustré à la section suivante.

Création d'une configuration DSC

L'étape suivante consiste à écrire une configuration DSC à l'aide de n'importe quel éditeur PowerShell pour tenir compte de l'intention de la configuration. Dans cet exemple, une configuration unique **`ConfigureSiteOnIIS`** est créée. Elle importe le module DSC de base **`PSDesiredStateConfiguration`**, composé des ressources utilisées dans la configuration. Elle déclare également un nœud de serveur web. Une fois cette configuration chargée et compilée, elle génère une configuration DSC dénommée **`ConfigureSiteOnIISwebserver`**. Cette configuration peut ensuite être appliquée aux nœuds.

La configuration se compose de plusieurs ressources. Ces ressources configurent le nœud cible. Les ressources installent un serveur web, ASP.NET et le cadre, et créent un fichier **`index.htm`** dans le répertoire **`inetpub\wwwroot`** avec le contenu indiquant que le site est en cours de maintenance. Pour plus d'informations sur l'écriture de la configuration DSC, accédez à https://docs.microsoft.com/powershell/scripting/dsc/getting-started/wingettingstarted?view=powershell-7.

La liste de code suivante indique la configuration complète décrite dans le paragraphe précédent. Cette configuration sera chargée sur le compte Azure Automation :

```
Configuration ConfigureSiteOnIIS {
    Import-DscResource -ModuleName 'PSDesiredStateConfiguration'
    Node WebServer {
      WindowsFeature IIS
        {
            Name = "Web-Server"
            Ensure = "Present"
        }
        WindowsFeature AspDotNet
        {
            Name = "net-framework-45-Core"
            Ensure = "Present"
            DependsOn = "[WindowsFeature]IIS"
        }
```

```
        WindowsFeature AspNet45
        {
            Ensure          = "Present"
            Name            = "Web-Asp-Net45"
            DependsOn = "[WindowsFeature]AspDotNet"
        }
        File IndexFile
        {
            DestinationPath = "C:\inetpub\wwwroot\index.htm"
            Ensure = "Present"
            Type = "File"
            Force = $true
            Contents = "<HTML><HEAD><Title> Website under construction.</
Title></HEAD><BODY> '
             <h1>If you are seeing this page, it means the website is under
maintenance and DSC Rocks !!!!!</h1></BODY></HTML>"
        }
   }
}
```

Une fois l'exemple de configuration DSC créé, il doit être importé dans Azure Automation, comme illustré à la section suivante.

Importation d'une configuration DSC

La configuration DSC n'est toujours pas prise en charge dans Azure Automation. Elle est disponible sur certaines machines locales. Elle doit être chargée dans les configurations DSC Azure Automation. Azure Automation fournit une cmdlet de commande **`Import-AzureRMAutomationDscConfiguration`** pour importer la configuration dans Azure Automation :

```
Import-AzureRmAutomationDscConfiguration -SourcePath "C:\DSC\AA\DSCfiles\
ConfigureSiteOnIIS.ps1" -ResourceGroupName "omsauto" -AutomationAccountName
"datacenterautomation" -Published -Verbose
```

Les commandes vont importer la configuration dans Azure Automation. Une fois l'importation effectuée, la configuration DSC doit être compilée afin de pouvoir être affectée aux serveurs pour les contrôles de conformité et la réparation automatique.

Compilation de la configuration DSC

Une fois la configuration DSC disponible dans Azure Automation, vous pouvez être invité à la compiler. Azure Automation prévoit une autre cmdlet de commande pour ce faire. Utilisez la cmdlet de commande **`Start-AzureRmAutomationDscCompilationJob`** pour compiler la configuration importée. Le nom de la configuration doit être identique au nom de la configuration chargée. La compilation crée un fichier MOF nommé d'après la configuration et le nom du nœud, c'est-à-dire, dans ce cas le serveur web **`ConfigureSiteOnIIS`**. L'exécution de la commande est affichée ici :

```
Start-AzureRmAutomationDscCompilationJob -ConfigurationName ConfigureSiteOnIIS
-ResourceGroupName "omsauto" -AutomationAccountName "datacenterautomation"
-Verbose
```

Vous avez effectué la configuration de nœud DSC. Dans la section suivante, vous apprendrez à affecter des configurations aux nœuds.

Affectation de configurations aux nœuds

Désormais, les configurations DSC compilées peuvent être appliquées à des nœuds. Utilisez **`Registre-AzureRmAutomationDscNode`** pour affecter la configuration à un nœud. Le paramètre **`NodeConfigurationName`** identifie le nom de la configuration qui doit être appliqué au nœud. Il s'agit d'une cmdlet de commande efficace, qui peut également configurer l'agent DSC **`localconfigurationmanager`** sur les nœuds avant de pouvoir télécharger des configurations et les appliquer. Il existe plusieurs paramètres **`localconfigurationmanager`** qui peuvent être configurés et des informations sont disponibles sur ce sujet à l'adresse https://devblogs.microsoft.com/powershell/understanding-meta-configuration-in-windows-powershell-desired-state-configuration.

Examinons la configuration ci-dessous :

```
Register-AzureRmAutomationDscNode -ResourceGroupName "omsauto"
-AutomationAccountName "datacenterautomation" -AzureVMName testtwo
-ConfigurationMode ApplyAndAutocorrect -ActionAfterReboot ContinueConfiguration
-AllowModuleOverwrite $true -AzureVMResourceGroup testone -AzureVMLocation
"West Central US" -NodeConfigurationName "ConfigureSiteOnIIS.WebServer"
-Verbose
```

Maintenant, nous pouvons vérifier si la configuration a été appliquée aux serveurs en parcourant le site web nouvellement déployé à l'aide d'un navigateur. Une fois le test terminé, passons à la validation des connexions.

Validation

Le cas échéant, les groupes de sécurité réseau et les pare-feu sont ouverts et activés pour le port 80 et une adresse IP publique est attribuée à la machine virtuelle. Le site web par défaut peut être parcouru à l'aide de l'adresse IP. Si tel n'est pas le cas, connectez-vous à la machine virtuelle utilisée pour appliquer la configuration DSC et accédez à **`http://localhost`**.

La page suivante devrait alors s'afficher :

Figure 13.23 : Localhost

Il s'agit d'un système de gestion de la configuration efficace, qui permet, sans l'écriture d'un code significatif, de créer une configuration une fois, qui pourra être appliquée à ce serveur ainsi qu'à d'autres, lesquels s'exécuteront dans l'état souhaité sans qu'aucune intervention manuelle ne soit nécessaire. Dans la section suivante, nous allons aborder les différents outils disponibles pour Azure DevOps.

Outils pour DevOps

Comme mentionné auparavant, Azure est une plateforme mature et enrichie qui prend en charge les éléments suivants :

- Plusieurs choix de langages
- Plusieurs choix de systèmes d'exploitation
- Plusieurs choix d'outils et d'utilitaires
- Plusieurs modèles pour déployer des solutions (par exemple des services app, conteneurs, machines virtuelles et des micro-services)

Compte tenu de toutes ces options et choix, Azure est à la fois :

- **Un Cloud ouvert** : ouvert aux deux open source, et aux produits, outils et services Microsoft et tiers.
- **Un Cloud flexible** : il est assez facile pour les utilisateurs professionnels et les développeurs de l'utiliser avec leurs compétences et connaissances existantes.
- **Une capacité de gestion unifiée** : elle fournit des fonctions de surveillance et de gestion sans effort.

Tous les services et capacités mentionnés ici sont essentiels pour réussir la mise en œuvre de DevOps. La *figure 13.24* montre les outils et utilitaires open source qui peuvent être utilisés pour les différentes phases de gestion du cycle de vie des applications et de DevOps dans son ensemble :

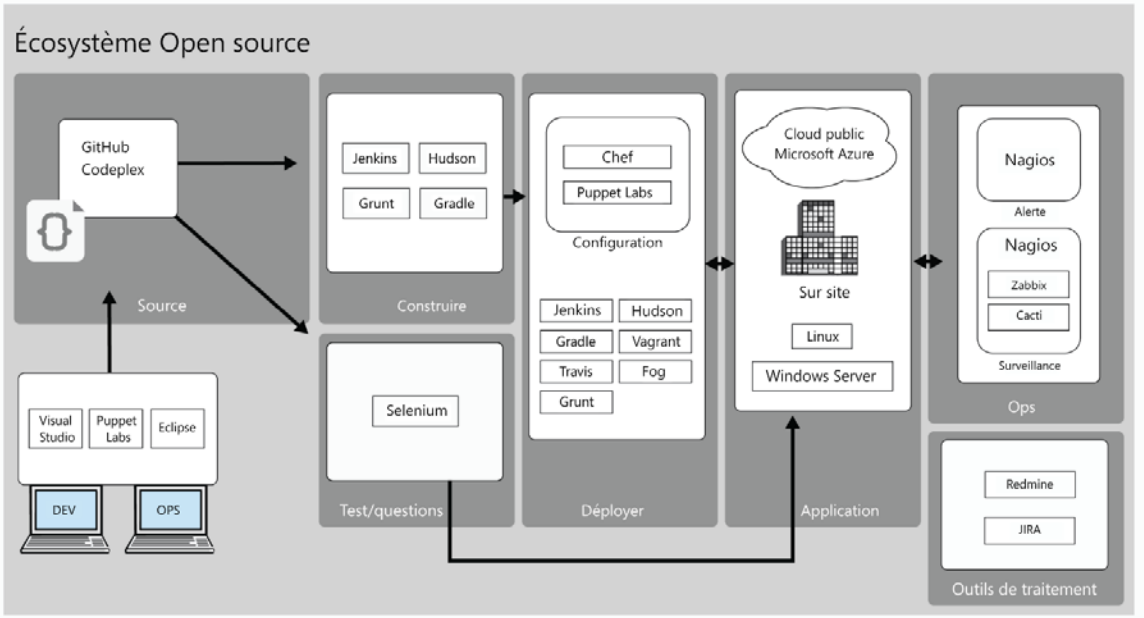


Figure 13.24 : Outils et utilitaires open source

La figure 13.24 montre les outils et utilitaires Microsoft qui peuvent être utilisés pour les différentes phases de gestion du cycle de vie des applications et de DevOps dans son ensemble. Encore une fois, il s'agit simplement d'une modeste représentation de tous ces outils et utilitaires, et de nombreuses autres options sont disponibles, comme suit :

- Orchestration du build Azure DevOps pour concevoir un pipeline de build
- Gestionnaire de tests Microsoft, Pester pour test
- Modèles DSC, PowerShell et ARM pour la gestion de déploiement ou de configuration
- Log Analytics, Application Insights et **System Center Operations Manager** (**SCOM**) pour la surveillance et la génération d'alertes
- Azure DevOps et System Center Service Manager pour la gestion des processus :

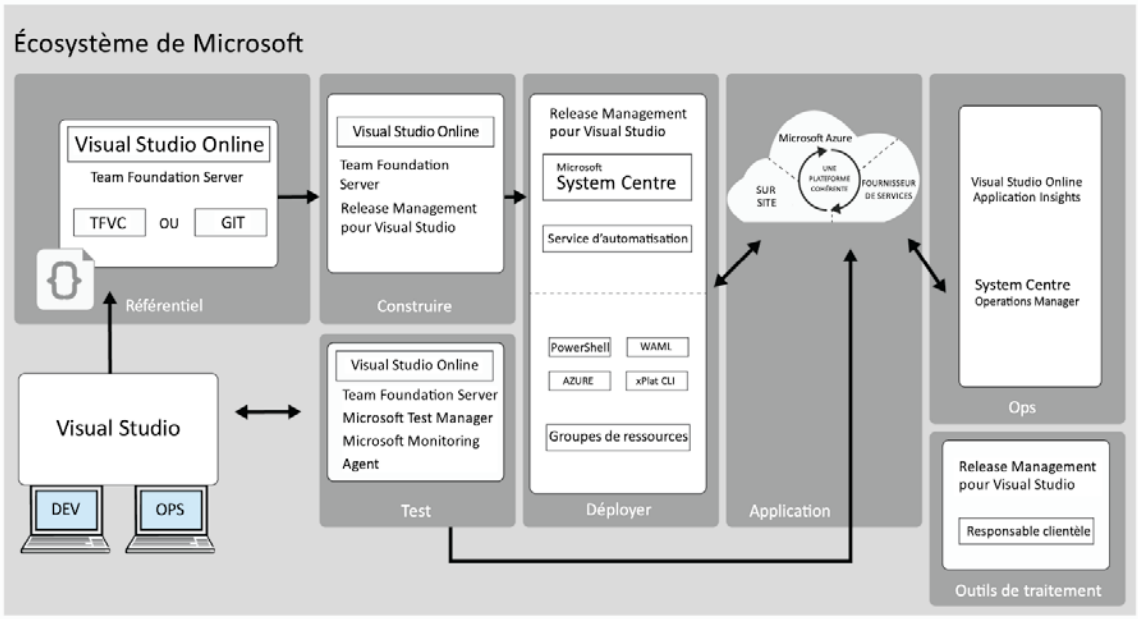


Figure 13.25 : Outils et utilitaires Microsoft

Il existe de nombreux outils disponibles pour chacune des pratiques DevOps. Cette section vous a présenté certains de ces outils et leur configuration.

Résumé

DevOps a de plus en plus de poids et gagne en importance dans le secteur. La plupart des organisations se sont rendu compte de ses avantages et sont impatientes de mettre en œuvre les principes DevOps. Parallèlement, la plupart d'entre elles adoptent le Cloud. Azure est une plateforme Cloud riche et mature qui fournit des services DevOps, afin de simplifier sa mise en place au sein des organisations.

Dans ce chapitre, DevOps a été abordé, ainsi que ses pratiques de base, telles que la gestion de la configuration, l'intégration continue, la livraison en continu et le déploiement. Nous avons également discuté des différentes solutions Cloud basées sur PaaS, une machine virtuelle IaaS et un conteneur, ainsi que de leurs ressources Azure et pipelines de build et de lancement.

Dans ce chapitre, nous avons expliqué la gestion de la configuration et nous avons abordé les services DSC d'Azure Automation, ainsi que le recours aux serveurs pull pour configurer des machines virtuelles automatiquement. Enfin, nous avons abordé l'ouverture et la flexibilité d'Azure qui propose un choix en termes de langages, d'outils et de systèmes d'exploitation.

Au cours du chapitre suivant, nous passerons en revue les détails de Kubernetes et ses composants et interactions, en plus des considérations de conception et de déploiement de l'application sur Kubernetes.

14

Architecture des solutions Azure Kubernetes

Les conteneurs sont l'un des composants d'infrastructure qui ont le plus fait parler d'eux au cours des dix dernières années. Il ne s'agit pas d'une nouvelle technologie puisqu'ils existent depuis un certain temps. Ils sont répandus dans le monde de Linux depuis plus de deux décennies. Les conteneurs n'étaient pas bien connus dans la communauté des développeurs en raison de leur complexité et du fait qu'il n'y avait pas beaucoup de documentation à leur sujet. Cependant, vers le début de cette décennie, en 2013, une entreprise nommée Docker a été lancée et a changé la perception et l'adoption des conteneurs dans le monde du développement.

Docker a écrit un « enveloppeur » d'API robuste au-dessus de conteneurs LXC Linux existants, ce qui permet aux développeurs de créer, de gérer et de détruire des conteneurs facilement à partir de l'interface de ligne de commande. Lorsque l'on conteneurise des applications, le nombre de conteneurs dont nous disposons peut augmenter considérablement avec le temps, et nous pouvons arriver à un point où il est nécessaire de gérer des centaines, voire des milliers de conteneurs. C'est là que les orchestrateurs de conteneurs entrent en jeu, et Kubernetes en fait partie. Grâce à Kubernetes, nous pouvons automatiser le déploiement, la mise à l'échelle, la mise en réseau et la gestion des conteneurs.

Dans ce chapitre, nous allons aborder :

- Les concepts d'introduction des conteneurs
- Les concepts de Kubernetes
- Les éléments importants qui font fonctionner Kubernetes
- L'architecture des solutions à l'aide d'Azure Kubernetes Service

Maintenant que vous savez à quoi sert Kubernetes, revenons aux bases et discutons des conteneurs, de la façon dont ils sont orchestrés à l'aide de Kubernetes, et plus encore.

Présentation des conteneurs

Les conteneurs sont une virtualisation au niveau du système d'exploitation. Ils sont hébergés sur un système d'exploitation fonctionnant soit sur un serveur physique, soit sur un serveur virtuel. La nature de la mise en œuvre dépend du système d'exploitation hôte. Par exemple, les conteneurs Linux sont inspirés des cgroups, tandis que les conteneurs Windows sont des machines virtuelles presque légères et peu encombrantes.

Les conteneurs sont réellement multiplateformes. Les applications en conteneurs peuvent s'exécuter sur n'importe quelle plateforme, comme Linux, Windows ou Mac, de manière uniforme sans que des modifications soient nécessaires, ce qui les rend hautement portables. Cette technologie est donc parfaite pour les organisations, car elle ne dépend pas des plateformes.

En outre, les conteneurs peuvent fonctionner dans n'importe quel environnement Cloud ou sur site sans qu'il soit nécessaire de les modifier. Cela signifie que les organisations ne sont pas non plus liées à un seul fournisseur de services Cloud si elles mettent en place des conteneurs comme plateforme d'hébergement dans le Cloud. Elles peuvent déplacer leur environnement du site local vers le Cloud.

Les conteneurs offrent tous les avantages généralement disponibles avec les machines virtuelles. Ils possèdent leurs propres adresses IP, noms DNS, identités, piles de réseau, systèmes de fichiers et autres composants qui donnent aux utilisateurs l'impression d'utiliser un tout nouvel environnement de système d'exploitation. Sous le capot, le moteur d'exécution de Docker virtualise plusieurs composants du système d'exploitation au niveau du noyau pour donner cette impression.

Les organisations qui adoptent la technologie des conteneurs bénéficient ainsi d'avantages importants, et Docker est l'un des précurseurs à cet égard. Il existe d'autres options pour le runtime de conteneur, comme CoreOS Rkt (prononcé Rocket, plus en production), Mesos Containerizer et les conteneurs LXC. Les organisations peuvent adopter la technologie avec laquelle elles se sentent à l'aise.

Auparavant, les conteneurs n'étaient pas disponibles dans l'écosystème Windows, mais ils sont devenus compatibles seulement pour Windows 10 et Windows Server 2016. Cependant, les conteneurs jouent désormais un rôle important dans le monde de Windows.

Comme mentionné dans l'introduction, les conteneurs doivent être bien surveillés, régis et gérés, tout comme n'importe quel autre élément d'infrastructure au sein d'un écosystème. Il est nécessaire de déployer un orchestrateur, comme Kubernetes, qui peut vous aider à le faire facilement. Dans la section suivante, vous découvrirez les notions de base de Kubernetes, y compris ses avantages.

Notions de base de Kubernetes

De nombreuses organisations se demandent encore *si elles ont besoin de Kubernetes, ou même d'un orchestrateur de conteneurs.* Lorsque nous envisageons la gestion des conteneurs à grande échelle, nous devons penser à plusieurs points, tels que l'évolutivité, l'équilibrage des charges, la gestion du cycle de vie, la livraison continue, l'enregistrement et la surveillance, etc.

Vous vous demandez peut-être *si les conteneurs ne sont pas censés faire tout cela.* La réponse est que les conteneurs ne sont qu'une partie de bas niveau du casse-tête. Les vrais avantages sont obtenus grâce aux outils qui utilisent les conteneurs. En fin de compte, nous avons besoin de quelque chose pour nous aider à orchestrer le tout.

Kubernetes est un mot grec, Κυβερνήτης , qui signifie « timonier » ou « capitaine du navire ». Dans l'esprit du thème maritime des conteneurs Docker, Kubernetes est le capitaine du navire. Kubernetes est souvent écrit K8s, 8 représentant les huit lettres entre « K » et « s » dans le mot « Kubernetes ».

Comme nous l'avons mentionné précédemment, les conteneurs sont plus agiles que les machines virtuelles. Ils peuvent être créés en quelques secondes et détruits tout aussi rapidement. Ils ont un cycle de vie similaire à celui des machines virtuelles. Cependant, ils doivent être surveillés, régis et gérés activement dans un environnement.

Il est possible de les gérer à l'aide de vos outils existants. Malgré cela, des outils spécialisés, tels que Kubernetes, peuvent offrir de précieux avantages :

- Kubernetes est de nature autorégénératrice. Lorsqu'un pod (lire « conteneur » pour l'instant) arrive dans un environnement Kubernetes, Kubernetes veille à ce qu'un nouveau pod soit créé ailleurs, soit sur le même nœud, soit sur un autre nœud, pour répondre aux demandes au nom de l'application.
- Kubernetes facilite également le processus de mise à niveau d'une application. Il propose des fonctionnalités prêtes à l'emploi pour vous aider à effectuer plusieurs types de mises à niveau avec la configuration d'origine.
- Il permet d'effectuer des déploiements bleu-vert. Dans ce type de déploiement, Kubernetes déploiera la nouvelle version de l'application aux côtés de l'ancienne, et une fois qu'il aura la certitude que la nouvelle application fonctionne comme prévu, un changement de DNS sera effectué pour passer à la nouvelle version de l'application. L'ancien déploiement de l'application peut continuer à exister à des fins de restauration.
- Kubernetes permet également de mettre en œuvre une stratégie de déploiement de mise à niveau propagée. Dans le cas présent, Kubernetes déploiera la nouvelle version de l'application un serveur à la fois et supprimera l'ancien déploiement un serveur à la fois. Il continuera cette activité jusqu'à ce qu'il n'y ait plus de serveurs provenant de l'ancien déploiement.
- Kubernetes peut être déployé dans un datacenter sur site ou dans le Cloud à l'aide d'une **infrastructure en tant que service** (**IaaS**). Cela signifie que les développeurs créent d'abord un groupe de machines virtuelles, puis déploient Kubernetes sur ces machines. Il existe également une autre approche consistant à utiliser Kubernetes comme **plateforme en tant que service** (**PaaS**). Azure fournit un service PaaS connu nommé **Azure Kubernetes Service** (**AKS**), qui fournit un environnement Kubernetes prêt à l'emploi pour les développeurs.

En ce qui concerne le déploiement, Kubernetes peut être déployé de deux manières :

- **Clusters non gérés** : des clusters non gérés peuvent être créés en installant Kubernetes et tout autre paquet pertinent sur une machine sans système d'exploitation ou sur une machine virtuelle. Dans un cluster non géré, il y aura des nœuds maîtres et des nœuds de travail. Ces nœuds collaborent pour orchestrer les conteneurs. Si vous vous demandez comment cela est mis en œuvre, nous explorerons plus loin dans ce chapitre l'architecture complète de Kubernetes. Pour l'instant, il suffit de savoir qu'il existe des nœuds maîtres et des nœuds de travail.

- **Clusters gérés** : les clusters gérés sont normalement fournis par le fournisseur de services Cloud, qui gère l'infrastructure pour vous. Dans Azure, ce service est appelé AKS. Azure apportera un soutien actif en ce qui concerne la mise à jour et la gestion de l'infrastructure. Avec l'IaaS, les organisations doivent garantir la disponibilité et l'évolutivité des nœuds et de l'infrastructure. Dans le cas d'AKS, le composant maître n'est pas visible, car il est géré par Azure. Toutefois, les nœuds de travail (esclaves) sont visibles et déployés dans un groupe de ressources distinct, ce qui vous permet d'y accéder si nécessaire.

Voici certains des principaux avantages de l'utilisation d'AKS par rapport aux clusters non gérés :

- Si vous utilisez des clusters non gérés, vous devez vous efforcer de rendre la solution hautement disponible et évolutive. En outre, vous devez disposer d'une gestion adéquate des mises à jour pour l'installation des mises à jour et des correctifs. En revanche, dans AKS, Azure gère tout cet aspect, ce qui permet aux développeurs de gagner du temps et d'être plus productifs.
- Intégration native avec d'autres services, comme Azure Container Registry pour stocker vos images de conteneurs en toute sécurité, Azure DevOps pour intégrer les pipelines CI/CD, Azure Monitor pour la journalisation, et Azure Active Directory pour la sécurité.
- Évolutivité et démarrage plus rapide.
- Prise en charge des jeux de mise à l'échelle de machine virtuelle.

Bien qu'il n'y ait aucune différence en termes de fonctionnalité de base entre ces deux déploiements, le déploiement IaaS permet d'ajouter immédiatement de nouveaux plug-ins et de nouvelles configurations, alors que l'équipe Azure peut mettre un peu de temps pour effectuer la même opération avec AKS. De plus, les nouvelles versions plus récentes de Kubernetes sont disponibles dans AKS assez rapidement, sans délai important.

Nous avons couvert les bases de Kubernetes. À ce stade, vous vous demandez peut-être comment Kubernetes parvient à faire tout cela. Dans la prochaine section, nous examinerons les composants de Kubernetes et la façon dont ils fonctionnent ensemble.

Architecture Kubernetes

La première étape pour comprendre Kubernetes consiste à se familiariser avec son architecture. Nous examinerons en détail chaque composant dans la prochaine section, mais une présentation globale de l'architecture vous aidera à comprendre l'interaction entre les composants.

Clusters Kubernetes

Kubernetes a besoin de nœuds physiques ou virtuels pour l'installation de deux types de composants :

- Composants du plan de contrôle Kubernetes, ou composants maîtres
- Nœuds de travail Kubernetes, ou composants esclaves

La figure 14.1 est un diagramme qui offre une vue d'ensemble de l'architecture de Kubernetes. Nous reviendrons plus en détail sur les composants plus tard :

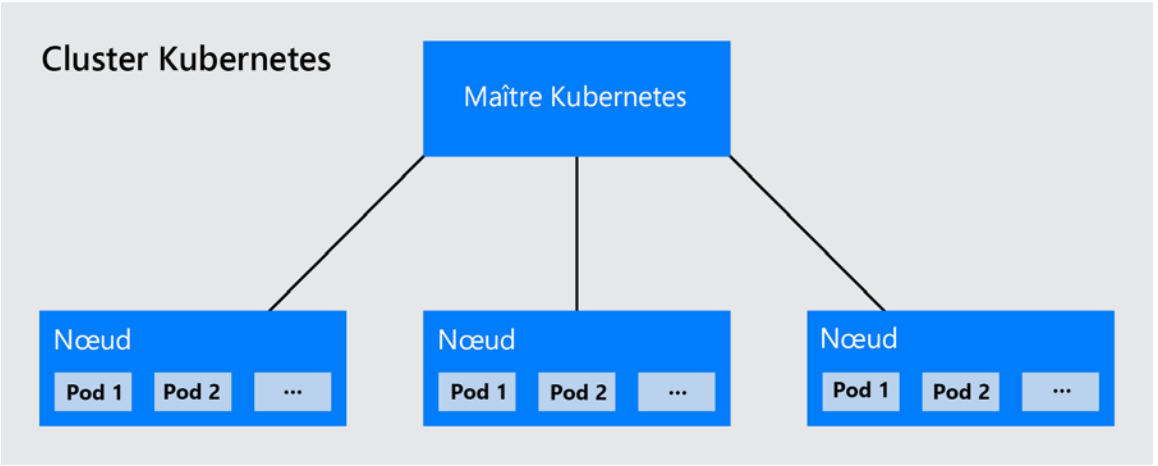


Figure 14.1 : Vue d'ensemble du cluster Kubernetes

Les composants du plan de contrôle sont responsables de la gestion et de la gouvernance de l'environnement Kubernetes et des composants Kubernetes esclaves.

L'ensemble des nœuds, soit le nœud maître et les nœuds esclaves, forme le cluster. En d'autres termes, un cluster est un ensemble de nœuds. Les clusters sont virtuels ou physiques, connectés les uns aux autres et accessibles à l'aide de la pile de mise en réseau TCP. Le monde extérieur n'aura aucune idée de la taille ou de la capacité de votre cluster, ni même du nom des nœuds de travail. La seule chose que les nœuds connaissent, c'est l'adresse du serveur d'API avec laquelle ils interagissent avec le cluster. Pour eux, le cluster est un grand ordinateur qui exécute leurs applications.

C'est Kubernetes qui décide en interne d'une stratégie appropriée en utilisant des contrôleurs pour choisir un nœud valide et sain capable de faire fonctionner l'application sans problème.

Les composants du plan de contrôle peuvent être installés dans une configuration à haute disponibilité. Jusqu'à présent, nous avons vu les clusters et leur fonctionnement. Dans la section suivante, nous allons examiner les composants d'un cluster.

Composants de Kubernetes

Les composants de Kubernetes sont divisés en deux catégories : les composants maîtres et les composants de nœuds. Les composants maîtres constituent le plan de contrôle du cluster. Le plan de contrôle est responsable de la gestion des nœuds de travail et des pods dans le cluster. L'autorité décisionnelle d'un cluster est le plan de contrôle, et elle s'occupe également de la détection et des réponses en lien avec les événements de cluster. La *figure* 14.2 décrit l'architecture complète d'un cluster Kubernetes :

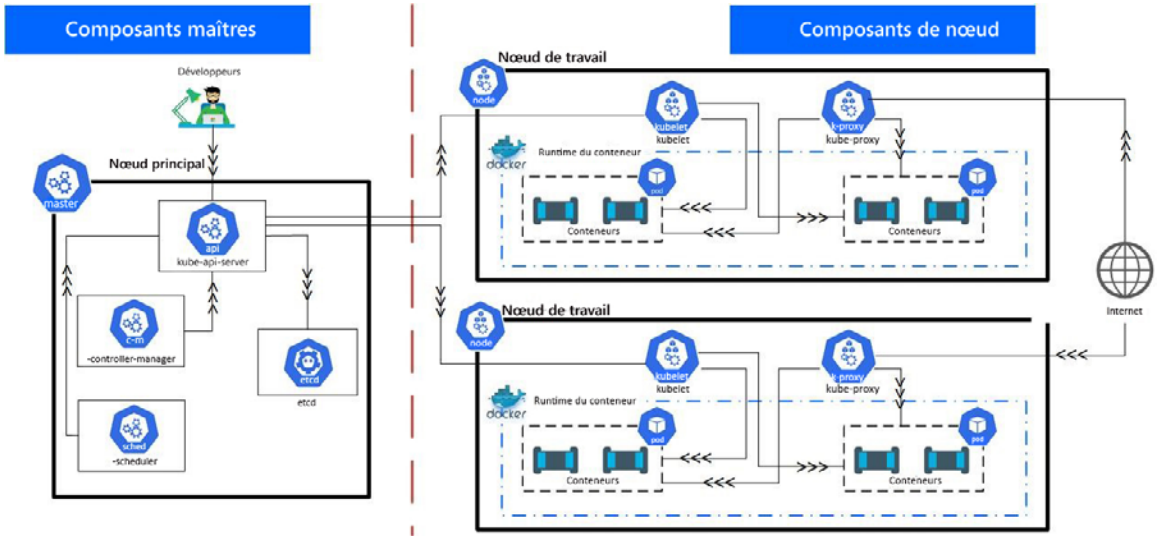


Figure 14.2 : Architecture Kubernetes

Vous devez comprendre chacun de ces composants pour gérer correctement un cluster. Examinons maintenant les composants maîtres :

- **Serveur d'API** : le serveur d'API est sans aucun doute le cerveau de Kubernetes. Il s'agit du composant central qui permet toutes les activités au sein de Kubernetes. Chaque demande de client, à quelques exceptions près, aboutit au serveur d'API, qui décide du flux de la demande. Il est seul responsable de l'interaction avec le serveur etcd.
- **etcd** : etcd est le magasin de données de Kubernetes. Seul le serveur d'API est autorisé à communiquer avec etcd, et le serveur d'API peut effectuer des activités **Créer**, **Lire**, **Mettre à jour** et **Supprimer** (**CLMS**) sur etcd. Lorsqu'une requête aboutit sur le serveur d'API, après validation, ce serveur peut effectuer toutes les opérations CLMS, en fonction de la requête etcd. etcd est un magasin de données distribué et hautement disponible. Il peut y avoir plusieurs installations d'etcd comprenant chacune une copie des données, et chacune d'elles peut répondre aux demandes du serveur d'API. Dans la *figure* 14.3, vous pouvez voir qu'il y a plusieurs instances en cours d'exécution dans le plan de contrôle pour offrir une haute disponibilité :

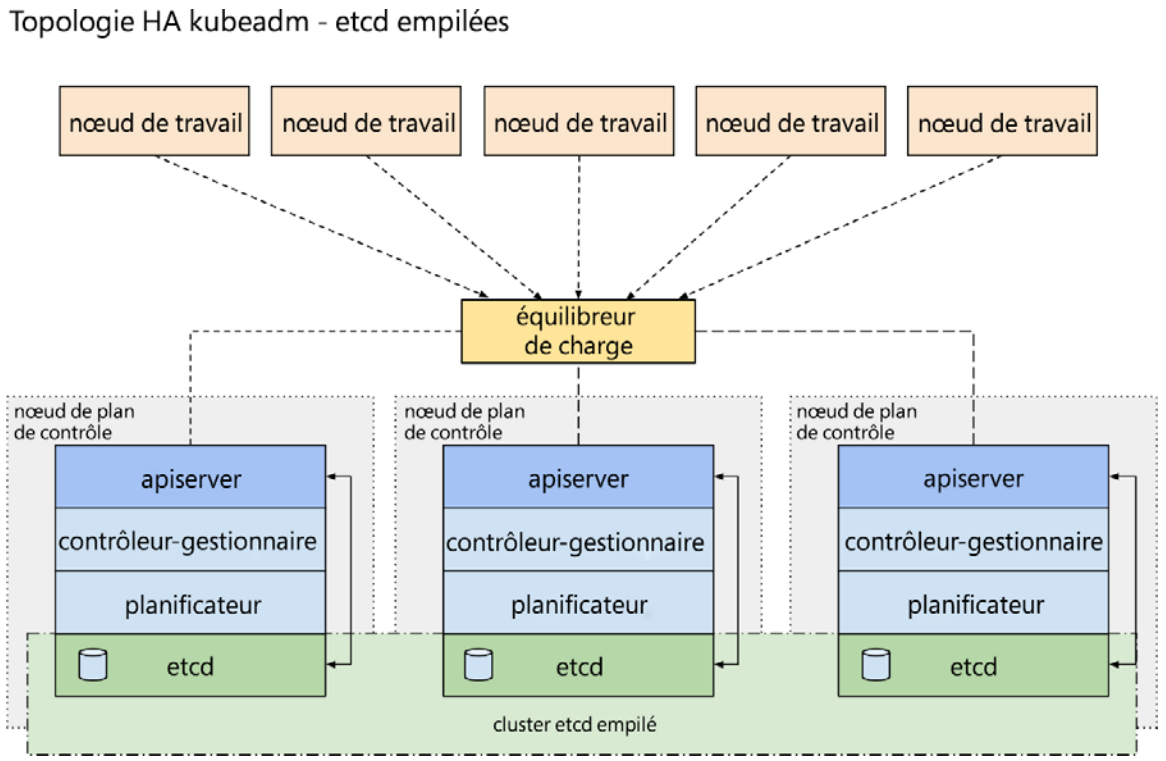


Figure 14.3 : Rendre le plan de contrôle hautement disponible

- **Gestionnaire de contrôleur** : le gestionnaire de contrôleur est la force motrice de Kubernetes. Tandis que le serveur d'API reçoit les requêtes, le véritable travail dans Kubernetes est effectué par le gestionnaire de contrôleur. Comme son nom l'indique, le gestionnaire de contrôleur est le gestionnaire des contrôleurs. Il y a plusieurs contrôleurs dans un nœud maître de Kubernetes, et chacun est responsable de la gestion d'un seul contrôleur.

  La principale responsabilité d'un contrôleur est la gestion d'une ressource unique dans un environnement Kubernetes. Par exemple, il y a un gestionnaire de contrôleur de réplication pour gérer les ressources du contrôleur de réplication, et un contrôleur de ReplicaSet pour gérer les ReplicaSets dans un environnement Kubernetes. Le contrôleur surveille le serveur d'API et, lorsqu'il reçoit une demande pour une ressource qu'il gère, le contrôleur effectue sa tâche.

L'une des principales responsabilités des contrôleurs est de continuer à fonctionner en boucle et de s'assurer que Kubernetes est dans l'état souhaité. S'il y a un écart par rapport à l'état souhaité, les contrôleurs doivent ramener Kubernetes à l'état souhaité. Un contrôleur de déploiement surveille toutes les nouvelles ressources de déploiement créées par le serveur d'API. Si une nouvelle ressource de déploiement est trouvée, le contrôleur de déploiement crée une nouvelle ressource ReplicaSet et s'assure que le ReplicaSet est toujours dans l'état souhaité. Un contrôleur de réplication fonctionne en boucle et vérifie si le nombre réel de pods dans l'environnement correspond au nombre de pods souhaité. Si un pod cesse de fonctionner pour une raison quelconque, le contrôleur de la réplication constatera que le nombre réel de pods a diminué d'une unité et programmera un nouveau Pod dans le même nœud ou dans un autre nœud.

- **Planificateur** : la tâche d'un planificateur consiste à planifier les pods sur les nœuds esclaves de Kubernetes. Il n'est pas responsable de la création de pods. Il est purement responsable de l'assignation des pods aux nœuds esclaves de Kubernetes. Il le fait en tenant compte de l'état actuel des nœuds, de leur pourcentage d'utilisation, des ressources dont ils disposent ainsi que de la définition du pod. Un pod peut avoir une préférence concernant un nœud spécifique, et le planificateur tiendra compte de ces demandes lors de l'affectation de pods aux nœuds.

Nous allons maintenant explorer les composants des nœuds qui sont déployés dans chacun des nœuds de travail du cluster :

- **Kubelet** : alors que le serveur d'API, le planificateur, les contrôleurs, etc. sont déployés sur les nœuds maîtres, les kubelets sont déployés sur les nœuds esclaves. Ils agissent en tant qu'agents pour les composants maîtres de Kubernetes et sont responsables de la gestion locale des pods sur les nœuds. Il y a un kubelet sur chaque nœud. Un kubelet prend les commandes des composants maîtres, comme le serveur d'API et le gestionnaire de contrôleurs, et leur fournit des informations sur l'intégrité, la surveillance et la mise à jour des nœuds et des pods. Il s'agit du canal de communication administrative entre les nœuds maîtres et esclaves.
- **kube-proxy** : kube-proxy, tout comme les kubelets, est déployé sur des nœuds esclaves. Il est responsable de la surveillance des pods et des services, ainsi que de la mise à jour des tableaux d'ip et des règles netfilter du pare-feu local en fonction de tout changement dans la disponibilité des pods et des services. Cela garantit que les informations d'acheminement sur les nœuds sont mises à jour au fur et à mesure de la création ou de la suppression de nœuds et des services.

- **Exécution du conteneur** : l'écosystème actuel compte de nombreux vendeurs et fournisseurs de conteneurs. Docker est le plus célèbre d'entre eux, bien que d'autres gagnent également en popularité. C'est pourquoi, dans notre architecture, nous avons marqué l'exécution du conteneur avec le logo Docker. Kubernetes est un orchestrateur de conteneurs générique. Il ne peut pas être étroitement associé à un seul fournisseur de conteneurs, comme Docker. Il doit être possible d'utiliser n'importe quel runtime de conteneur sur les nœuds esclaves pour gérer le cycle de vie des conteneurs.

Pour exécuter des conteneurs dans des pods, une norme du secteur nommée **interface de runtime de conteneur (CRI)** a été développée et est utilisée par toutes les grandes entreprises. La norme prévoit des règles à suivre pour avoir une interopérabilité avec des orchestrateurs comme Kubernetes. Les kubelets ne savent pas quels binaires de conteneurs sont installés sur les nœuds. Il peut s'agir de binaires Docker ou de tout autre binaire.

Étant donné que ces exécutions de conteneurs ont été développées selon une norme industrielle commune, quelle que soit l'exécution utilisée, kubelets pourra communiquer avec l'exécution du conteneur. Cela dissocie la gestion des conteneurs de la gestion des clusters Kubernetes. Les responsabilités de l'exécution du conteneur comprennent la création de conteneurs, la gestion de la pile de réseaux des conteneurs et la gestion du réseau de pont. Dans la mesure où la gestion des conteneurs est distincte de la gestion des clusters, Kubernetes ne perturbe pas les responsabilités de l'exécution des conteneurs.

Les composants que nous avons abordés s'appliquent à la fois aux clusters AKS gérés et non gérés. Toutefois, les composants maîtres ne sont pas exposés à l'utilisateur final, car Azure gère tout cela dans le cas d'AKS. Plus loin dans ce chapitre, nous aborderons l'architecture d'AKS. Vous en apprendrez davantage sur les clusters non gérés et vous comprendrez plus clairement les différences entre ces systèmes.

Ensuite, vous découvrirez quelques-unes des ressources les plus importantes de Kubernetes, qu'on appelle souvent primitives, des connaissances qui s'appliquent à la fois aux clusters non gérés et AKS.

Primitives Kubernetes

Vous avez appris que Kubernetes est un système d'orchestration utilisé pour déployer et gérer les conteneurs. Kubernetes définit un ensemble d'éléments constitutifs, qui sont également connus sous le nom de primitives. Conjointement, ces primitives peuvent nous aider à déployer, à gérer et à mettre à l'échelle des applications en conteneurs. Nous allons maintenant examiner chacune de ces primitives et détailler leurs rôles.

Pod

Les pods sont l'unité la plus basique du déploiement dans Kubernetes. La question immédiate qui se pose à un esprit curieux est de savoir en quoi un pod est différent d'un conteneur. Les pods sont des wrappers au-dessus des conteneurs. En d'autres termes, les conteneurs sont contenus dans les pods. Il peut y avoir plusieurs conteneurs dans un pod. Toutefois, la bonne pratique consiste à utiliser un pod par conteneur. Cela ne signifie pas que nous ne pouvons pas avoir plus d'un conteneur dans un pod. Vous pouvez également mettre plusieurs conteneurs dans un pod en définissant un conteneur principal et des conteneurs auxiliaires. De plus, il existe des modèles, comme les modèles de « sidecar », qui peuvent être mis en œuvre avec des pods multiconteneurs.

Chaque pod possède sa propre adresse IP et sa propre pile de réseau. Tous les conteneurs partagent l'interface réseau et la pile. On peut atteindre tous les conteneurs d'un pod localement à l'aide du nom d'hôte.

Une définition simple d'un pod au format YAML est illustrée dans les lignes de code suivantes :

```
---
apiVersion: v1
kind: Pod
metadata:
  name: tappdeployment
  labels:
    appname: tapp
    ostype: linux
spec:
  containers:
  - name: mynewcontainer
    image: "tacracr.azurecr.io/tapp:latest"
    ports:
    - containerPort: 80
      protocol: TCP
      name: http
```

La définition de pod illustrée a un nom et définit quelques étiquettes, qui peuvent être utilisées par la ressource de service pour exposition à d'autres pods, nœuds et ressources personnalisées externes. Cet exemple définit également un conteneur unique basé sur une image personnalisée stockée dans Azure Container Registry et ouvre le port **`80`** pour le conteneur.

Services

Kubernetes permet de créer des pods avec plusieurs instances. Ces pods doivent être accessibles à partir de n'importe quel pod ou nœud au sein d'un cluster. Il est possible d'utiliser l'adresse IP d'un pod directement et d'accéder au pod. Cependant, c'est loin d'être idéal. Les pods sont éphémères et peuvent obtenir une nouvelle adresse IP si le pod précédent est tombé en panne. Dans de tels cas, l'application tombera en panne facilement. Kubernetes fournit des services qui découplent les instances pod de leurs clients. Les pods peuvent être créés et supprimés, mais l'adresse IP d'un service Kubernetes reste constante et stable. Les clients peuvent se connecter à l'adresse IP du service, qui dispose à son tour d'un point de terminaison pour chaque pod auquel il peut envoyer des demandes. S'il y a plusieurs instances de pods, chacune de leurs adresses IP sera disponible pour le service en tant qu'objet de point de terminaison. Lorsqu'un pod cesse de fonctionner, les points de terminaison sont mis à jour pour refléter les instances de pods actuelles ainsi que leurs adresses IP.

Les services sont hautement découplés avec les pods. L'objectif principal des services est la mise en file d'attente pour les pods qui ont des étiquettes dans leurs définitions de sélecteur de service. Un service définit des sélecteurs d'étiquettes et, en fonction de ces sélecteurs, les adresses IP des pods sont ajoutées à la ressource du service. Les pods et les services peuvent être gérés indépendamment les uns des autres.

Un service fournit plusieurs types de schémas d'adresses IP. Il y a quatre types de services : ClusterIP, NodePort, LoadBalancer et le contrôleur Ingress qui utilise Application Gateway.

Le schéma le plus fondamental est ClusterIP. Il s'agit d'une adresse IP interne qui peut être atteinte uniquement à partir du cluster. Le schéma ClusterIP est illustré à la *figure 14.4* :

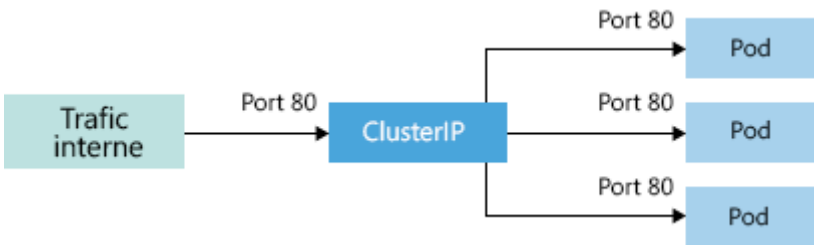


Figure 14.4 : Le fonctionnement de ClusterIP

ClusterIP permet également la création de NodePort, à l'aide duquel le cluster obtient un ClusterIP. Toutefois, il peut également ouvrir un port sur chacun des nœuds au sein d'un cluster. Les pods peuvent être atteints à l'aide d'adresses ClusterIP, ainsi qu'avec une combinaison de l'IP et du port du nœud :

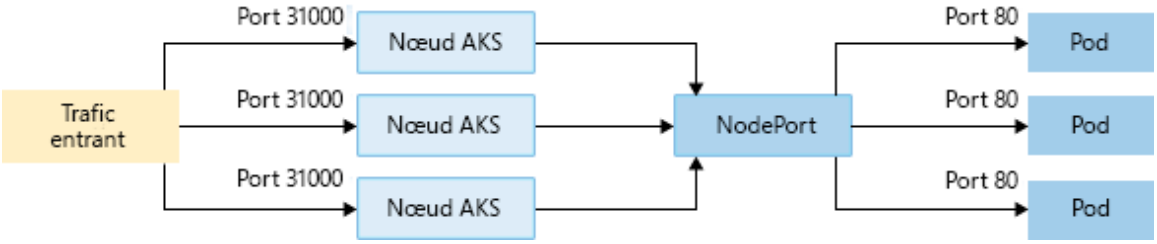


Figure 14.5 : Le fonctionnement de NodePort

Les services peuvent se référer non seulement aux pods, mais aussi aux points de terminaison externes. Enfin, les services permettent également la création d'un service basé sur un équilibreur de charge capable de recevoir des requêtes externes et de les rediriger vers une instance de pod en utilisant ClusterIP et NodePort en interne :

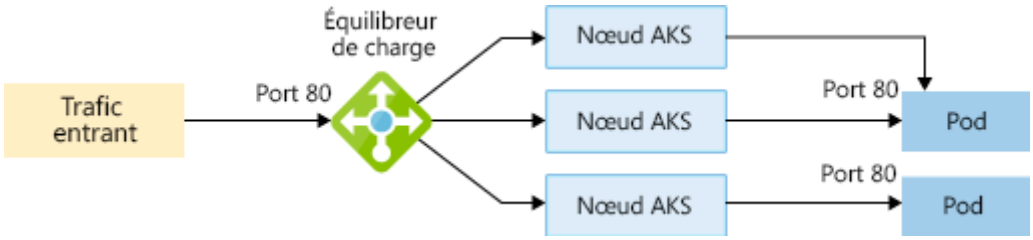


Figure 14.6 : Le fonctionnement de l'équilibreur de charge

Il existe un dernier type de service appelé contrôleur Ingress, qui offre des fonctionnalités avancées telles que l'acheminement basé sur l'URL, comme indiqué à la *figure* 14.7 :

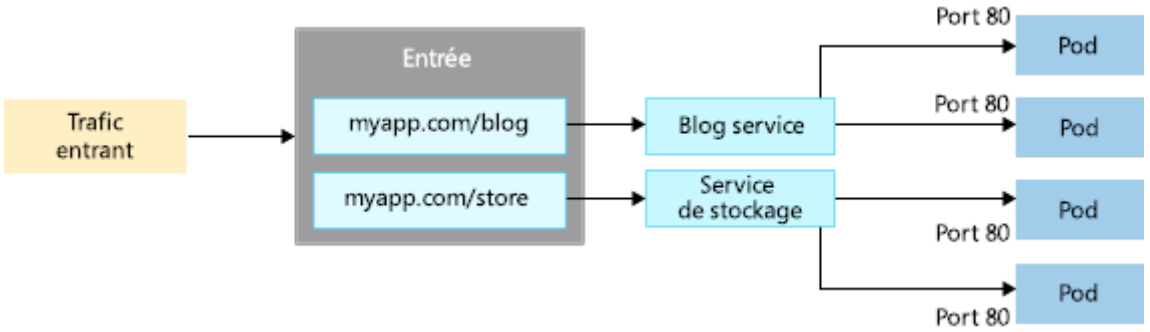


Figure 14.7 : Le fonctionnement du contrôleur Ingress

Une définition de service au format YAML est illustrée ici :

```
apiVersion: v1
kind: Service
metadata:
  name: tappservice
  labels:
    appname: tapp
    ostype: linux
spec:
  type: LoadBalancer
  selector:
    appname: myappnew
  ports:
  - name: http
    port: 8080
    targetPort: 80
    protocol: TCP
```

Cette définition de service crée un service basé sur l'équilibreur de charge à l'aide de sélecteurs d'étiquettes.

Déploiements

Les déploiements Kubernetes sont des ressources de niveau supérieur par rapport aux ReplicaSets aux pods. Les déploiements offrent des fonctionnalités en lien avec la mise à niveau et la publication d'une application. Les ressources de déploiement créent un ReplicaSet, et le ReplicaSet gère le pod. Il est important de comprendre le besoin en ressources de déploiement lorsque les ReplicaSets existent déjà.

Les déploiements jouent un rôle important dans la mise à niveau des applications. Si une application est déjà en production et qu'une nouvelle version de l'application doit être déployée, plusieurs choix s'offrent à vous :

1. Supprimer les pods existants et créer de nouveaux pods – cette méthode entraîne un temps d'arrêt pour l'application, c'est pourquoi elle ne doit être utilisée que si ce temps d'arrêt est acceptable. Vous risquez un temps d'arrêt accru si le déploiement contient des bogues et que vous devez revenir à une version précédente.

2. Déploiement bleu-vert – Avec cette méthode, les pods existants restent fonctionnels, et un nouvel ensemble de pods est créé avec la nouvelle version de l'application. Les nouveaux pods ne sont pas accessibles en externe. Une fois les tests terminés avec succès, Kubernetes commence à rediriger vers le nouvel ensemble de pods. Les anciens pods peuvent rester tels quels ou peuvent être supprimés par la suite.
3. Mises à niveau propagées – Avec cette méthode, les pods existants sont supprimés un à la fois, tandis que les nouveaux pods pour la nouvelle version de l'application sont créés un à la fois. Les nouveaux pods sont déployés de façon incrémentielle, tandis que les anciens pods sont réduits progressivement jusqu'à ce qu'il n'y en ait plus.

Toutes ces démarches doivent être effectuées manuellement sans l'aide d'une ressource de déploiement. Une ressource de déploiement automatise l'intégralité du processus de publication et de mise à niveau. Elle peut également aider à revenir automatiquement à une version précédente en cas de problème avec le déploiement en cours.

Une définition du déploiement est illustrée dans la liste de codes suivante :

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tappdeployment
  labels:
    appname: tapp
    ostype: linux
spec:
  replicas: 3
  selector:
    matchLabels:
      appname: myappnew
  strategy:
    type: RollingUpdate
    rollingUpdate:
       maxSurge: 1
       maxUnavailable: 1
  template:
```

```
    metadata:
      name: mypod
      labels:
        appname: myappnew
    spec:
      containers:
      - name: mynewcontainer
        image: "tacracr.azurecr.io/tapp:latest"
        ports:
        - containerPort: 80
          protocol: TCP
          name: http
```

Il est important de noter qu'un déploiement a une dimension **stratégique**, qui détermine si la méthode de **recréation** ou de **mise à jour propagée** est utilisée. La **recréation** supprimera tous les pods existants et créera de nouveaux pods. Le déploiement contient également des détails de configuration concernant la **mise à jour propagée** en indiquant le nombre maximal de pods pouvant être créés et supprimés en une seule exécution.

Contrôleur de réplication et ReplicaSet

La ressource du contrôleur de réplication Kubernetes garantit que le nombre souhaité d'instances pod est toujours en cours d'exécution au sein d'un cluster. Toute déviation par rapport à l'état souhaité est surveillée par le contrôleur de réplication, et ce dernier crée de nouvelles instances pod pour maintenir l'état souhaité.

Les ReplicaSets sont la nouvelle version du contrôleur de réplication. Les ReplicaSets offrent la même fonctionnalité que les contrôleurs de réplication, avec quelques fonctionnalités avancées. La principale d'entre elles est la riche capacité à définir les sélecteurs associés aux pods. Avec ReplicaSets, il est possible de définir les expressions dynamiques que n'étaient pas disponibles avec les contrôleurs de réplication.

Il est recommandé d'utiliser des ReplicaSets plutôt que des contrôleurs de réplication.

Les lignes de code suivantes montrent un exemple de définition de ressource **ReplicaSet** :

```
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
```

```
    name: tappdeployment
    labels:
      appname: tapp
      ostype: linux
  spec:
    replicas: 3
    selector:
      matchLabels:
        appname: myappnew
    template:
      metadata:
        name: mypod
        labels:
          appname: myappnew
      spec:
        containers:
        - name: mynewcontainer
          image: "tacracr.azurecr.io/tapp:latest"
          ports:
          - containerPort: 80
            protocol: TCP
            name: http
```

Il est important de noter que les ReplicaSets ont une propriété de **répliques**, qui détermine le nombre d'instances pod, une propriété **sélecteur**, qui définit les pods devant être gérés par ReplicaSet, et enfin la propriété **modèle**, qui définit le pod lui-même.

ConfigMaps et secrets

Kubernetes fournit deux ressources importantes pour stocker les données de configuration. Les ConfigMaps sont utilisées pour stocker des données de configuration générales qui ne sont pas sensibles à la sécurité. Les données génériques de configuration de l'application, comme les noms de dossiers, les noms de volumes et les noms DNS, peuvent être stockées dans ConfigMaps. D'autre part, les données sensibles, comme les informations d'identification, les certificats et les secrets, doivent être stockées dans les ressources de secrets. Ces données de secrets sont chiffrées et stockées dans le magasin de données Kubernetes etcd.

Les données de ConfigMaps et de secrets peuvent être rendues disponibles en tant que variables ou volumes d'environnement au sein des pods.

La définition du pod qui veut consommer ces ressources doit comprendre une référence à ces ressources. Nous avons couvert les primitives Kubernetes et les rôles de chacun des éléments constitutifs. Ensuite, vous découvrirez l'architecture d'AKS.

Architecture d'AKS

Dans la section précédente, nous avons vu l'architecture d'un cluster non géré. Maintenant, nous allons explorer l'architecture d'AKS. Une fois que vous aurez lu cette section, vous connaîtrez les principales différences entre l'architecture des clusters gérés et non gérés (AKS, dans ce cas).

Lorsqu'une instance AKS est créée, seuls les nœuds de travail sont créés. Les composants maîtres sont gérés par Azure. Les composants maîtres sont le serveur d'API, le planificateur, etcd et le gestionnaire de contrôleurs, dont nous avons parlé plus tôt. Les kubelets et kube-proxy sont déployés sur les nœuds de travail. La communication entre les nœuds et les composants principaux a lieu à l'aide de kubelets, qui agissent en tant qu'agents des clusters Kubernetes pour le nœud :

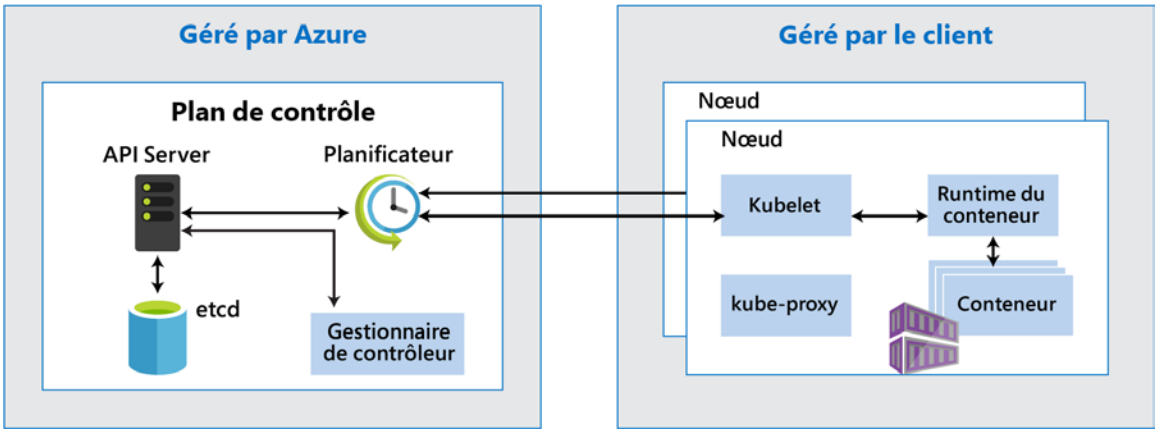


Figure 14.8 : Architecture d'AKS

Lorsqu'un utilisateur demande une instance de pod, la demande de l'utilisateur est transmise au serveur d'API. Le serveur d'API vérifie et valide les détails de la requête, et les stocke dans etcd (le magasin de données pour le cluster). Il crée également la ressource de déploiement (si la requête pod est encapsulée avec une ressource de déploiement). Le contrôleur de déploiement surveille la création de toutes les nouvelles ressources de déploiement. S'il en détecte une, il crée une ressource ReplicaSet en fonction de la définition fournie dans la demande de l'utilisateur.

Le contrôleur ReplicaSet surveille la création de nouvelles ressources ReplicaSet et lorsqu'il constate qu'une ressource a été créée, il demande au planificateur de planifier les pods. Le planificateur dispose de ses propres procédures et règles qui lui permettent de trouver un nœud approprié pour l'hébergement des pods. Le planificateur informe le kubelet de la présence du nœud et le kubelet récupère ensuite la définition pour le pod et crée les pods à l'aide du runtime de conteneur installé sur les nœuds. En dernier lieu, le pod crée les conteneurs dans sa définition.

kube-proxy permet de conserver la liste des adresses IP des informations de pod et de service sur les nœuds locaux, ainsi que de mettre à jour les règles de pare-feu et de routage locales. Pour résumer rapidement ce que nous avons vu jusqu'à présent, nous avons commencé par l'architecture Kubernetes, puis nous sommes passés aux primitives, et enfin à l'architecture d'AKS. Maintenant que vous connaissez ces concepts, nous allons créer un cluster AKS dans la section suivante.

Déploiement d'un cluster AKS

AKS peut être configuré à l'aide du portail Azure, de l'**interface de ligne de commande** Azure, des cmdlets de commande Azure PowerShell, des modèles ARM, des **SDK** (**kits de développement logiciel**) pour les langages pris en charge et même des API REST Azure ARM.

Le portail Azure est le moyen le plus simple de créer une instance AKS. Toutefois, pour activer DevOps, il est préférable de créer une instance AKS à l'aide de modèles ARM, de l'interface de ligne de commande ou de PowerShell.

Création d'un cluster AKS

Nous allons créer un groupe de ressources afin de déployer notre cluster AKS. Dans l'interface de ligne de commande Azure, utilisez la commande **`az group create`** :

```
az group create -n AzureForArchitects -l southeastasia
```

Ici, **`-n`** correspond au nom du groupe de ressources et **`-l`** correspond à l'emplacement. Si la requête a abouti, vous obtenez une réponse similaire à celle-ci :

```
rithin az group create -n AzureForArchitects -l southeastasia
{
  "id": "/subscriptions/[redacted]/resourceGroups/AzureForArchitects",
  "location": "southeastasia",
  "managedBy": null,
  "name": "AzureForArchitects",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": null
}
```

Figure 14.9 : Création d'un groupe de ressources

Maintenant que le groupe de ressources est prêt, nous allons créer le cluster AKS à l'aide de la commande **`az aks create`**. La commande suivante va créer un cluster nommé **`AzureForArchitects-AKS`** dans le groupe de ressources **`AzureForArchitects`** avec un nombre de nœuds égal à **`2`**. Le paramètre **`--generate-ssh-keys`** permettra la création de paires de clés **RSA** (**Rivest–Shamir–Adleman**), un cryptosystème de clé publique :

```
az aks create --resource-group AzureForArchitects \
--name AzureForArchitects-AKS \
--node-count 2 \
--generate-ssh-keys
```

Si la commande aboutit, vous devriez voir une sortie similaire à celle-ci :

```
rithin az aks create --resource-group AzureForArchitects \
> --name AzureForArchitects-AKS \
> --node-count 2 \
> --generate-ssh-keys
{
  "aadProfile": null,
  "addonProfiles": null,
  "agentPoolProfiles": [
    {
      "availabilityZones": null,
      "count": 2,
      "enableAutoScaling": null,
      "maxCount": null,
      "maxPods": 110,
      "minCount": null,
      "name": "nodepool1",
      "orchestratorVersion": "1.15.11",
      "osDiskSizeGb": 100,
      "osType": "Linux",
      "provisioningState": "Succeeded",
      "type": "AvailabilitySet",
      "vmSize": "Standard_DS2_v2",
      "vnetSubnetId": null
    }
  ],
  "apiServerAuthorizedIpRanges": null,
  "dnsPrefix": "AzureForAr-AzureForArchitec-1b2287",
  "enablePodSecurityPolicy": null,
  "enableRbac": true,
  "fqdn": "azureforar-azureforarchitec-1b2287-72aecee5.hcp.southeastasia.azmk8s.io",
```

Figure 14.10 : Création du cluster

Si vous passez en revue le cluster, vous verrez un élément de ligne indiquant **« `nodeResourceGroup": "MC_AzureForArchitects_AzureForArchitects-AKS_southeastasia` »**. Lors de la création d'un cluster AKS, une deuxième ressource est automatiquement créée pour stocker les ressources du nœud.

Notre cluster est mis en service. Maintenant, nous devons nous connecter au cluster et interagir avec celui-ci. Pour contrôler le gestionnaire de cluster Kubernetes, nous allons utiliser kubectl. Dans la section suivante, nous allons examiner rapidement kubectl.

Kubectl

Kubectl est le composant principal avec lequel les développeurs et les consultants en infrastructure peuvent interagir avec AKS. Kubectl permet de créer une requête REST contenant l'en-tête et le corps HTTP, et de la soumettre au serveur de l'API. L'en-tête contient les détails d'authentification, tels qu'un jeton ou une combinaison nom d'utilisateur/mot de passe. Le corps contient la charge utile réelle au format JSON.

La commande kubectl fournit des informations détaillées sur les journaux lorsqu'elle est utilisée avec le commutateur verbose. Le commutateur prend une entrée entière pouvant aller de 0 à 9, qui peut être visualisée dans les journaux de détails.

Connexion au cluster

Pour permettre la connexion locale au cluster, nous devons installer kubectl. Si vous utilisez Azure Cloud Shell, kubectl est déjà installé. Si vous souhaitez vous connecter localement, utilisez **`az aks install-cli`** pour installer kubectl.

Afin de configurer kubectl pour nous connecter à notre cluster Kubernetes, nous devons télécharger les informations d'identification et configurer l'interface de ligne de commande avec celles-ci. Cela peut être fait à l'aide de la commande **`az aks get-credentials`**. Utilisez la commande comme illustré ici :

```
az aks get-credentials \
--resource-group AzureForArchitects \
--name AzureForArchitects-AKS
```

Maintenant, nous devons vérifier si nous sommes connectés au cluster. Comme nous l'avons mentionné précédemment, nous allons utiliser kubectl pour communiquer avec le cluster, et **`kubectl get nodes`** affiche une liste de nœuds dans le cluster. Lors de la création, nous définissons le nombre de nœuds sur **`2`**, de sorte que la sortie doit avoir deux nœuds. En outre, nous devons nous assurer que l'état du nœud est **`Prêt`**. La sortie doit ressembler à celle illustrée à la *figure 14.11* :

```
rithin kubectl get nodes
NAME                       STATUS   ROLES   AGE   VERSION
aks-nodepool1-31051128-0   Ready    agent   64m   v1.15.11
aks-nodepool1-31051128-1   Ready    agent   64m   v1.15.11
```

Figure 14.11 : Obtenir la liste de nœuds

Maintenant que notre nœud est à l'état **Prêt**, nous allons créer un pod. Il existe deux façons de créer des ressources dans Kubernetes. Parmi ceux-ci figurent :

- **Impératif** : dans cette méthode, nous utilisons les commandes **kubectl run**et **kubectl expose** pour créer les ressources.
- **Déclarative** : nous décrivons l'état de la ressource via JSON ou un fichier YAML. Lorsque nous avons abordé les primitives Kubernetes, nous avons vu beaucoup de fichiers YAML pour chacun des blocs de construction. Nous allons transmettre le fichier à la commande **kubectl apply** pour créer les ressources, et les ressources déclarées dans le fichier seront créées.

Prenons d'abord l'approche impérative pour créer un pod avec le nom **webserver**, qui exécute un conteneur NGINX avec le port **80** exposé :

```
kubectl run webserver --restart=Never --image nginx --port 80
```

Une fois la commande terminée, l'interface de ligne de commande vous permet de connaître l'état :

```
rithin kubectl run webserver --restart=Never --image nginx --port 80
pod/webserver created
```

Figure 14.12 : Création d'un pod

Maintenant que nous avons essayé la méthode impérative, nous allons suivre la méthode déclarative. Vous pouvez utiliser la structure du fichier YAML, que nous avons abordée dans la sous-section *Pod* de la section *Primitives Kubernetes*, et la modifier selon vos besoins.

Nous allons utiliser l'image NGINX et le pod sera nommé **webserver-2**.

Vous pouvez utiliser n'importe quel éditeur de texte et créer le fichier. Le fichier final doit être similaire à celui-ci :

```
apiVersion: v1
kind: Pod
metadata:
  name: webserver-2
  labels:
    appname: nginx
    ostype: linux
spec:
```

```
containers:
- name: wenserver-2-container
  image: nginx
  ports:
  - containerPort: 80
    protocol: TCP
    name: http
```

Dans la commande **`kubectl apply`**, nous allons passer le nom du fichier au paramètre **`-f`** , comme illustré à la *figure* 14.13, et vous pouvez voir que le pod a été créé :

```
 rithin kubectl apply -f nginx.yaml
pod/webserver-2 created
```

Figure 14.13 : Création d'un pod à l'aide de la méthode déclarative.

Étant donné que nous avons créé les pods, nous pouvons utiliser la commande **`kubectl get pods`** pour répertorier tous les pods. Kubernetes utilise le concept des espaces de noms pour l'isolation logique des ressources. Par défaut, toutes les commandes pointent vers l'espace de noms **`par défaut`** . Si vous souhaitez effectuer une action sur un espace de noms spécifique, vous pouvez passer le nom de l'espace de noms via le paramètre **`-n`**. Sur la *figure* 14.14, vous pouvez voir que **`kubectl get pods`** renvoie les pods que nous avons créés dans l'exemple précédent, qui se trouvent dans l'espace de noms par défaut. En outre, lorsque nous utilisons **`--all-namespaces`**, la sortie renvoie des pods dans tous les espaces de noms :

```
 rithin kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
webserver     1/1     Running   0          70m
webserver-2   1/1     Running   0          58m
 rithin kubectl get pods --all-namespaces
NAMESPACE     NAME                                    READY   STATUS    RESTARTS   AGE
default       webserver                               1/1     Running   0          70m
default       webserver-2                             1/1     Running   0          58m
kube-system   coredns-698c77c5d7-l2wbd                1/1     Running   0          145m
kube-system   coredns-698c77c5d7-v96gq                1/1     Running   0          142m
kube-system   coredns-autoscaler-5bd7c6759b-7kpzq     1/1     Running   0          145m
kube-system   kube-proxy-95jmg                        1/1     Running   0          142m
kube-system   kube-proxy-qsfwq                        1/1     Running   0          143m
kube-system   kubernetes-dashboard-74d8c675bc-jzhcf   1/1     Running   1          145m
kube-system   metrics-server-7d654ddc8b-txbt9         1/1     Running   1          145m
kube-system   tunnelfront-79fc68b7f-qwzvj             1/1     Running   0          145m
```

Figure 14.14 : Répertorier tous les pods

Nous allons maintenant créer un déploiement simple qui exécute NGINX et avec un équilibreur de charge qui l'expose à Internet. Le fichier **YAML** se présente comme suit :

```
#Creating a deployment that runs six replicas of nginx
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-server
spec:
  replicas: 6
  selector:
    matchLabels:
      app: nginx-server
  template:
    metadata:
      labels:
        app: nginx-server
    spec:
      containers:
      - name: nginx-server
        image: nginx
        ports:
        - containerPort: 80
          name: http
---
#Creating Service
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
```

```
spec:
  ports:
  - port: 80
  selector:
    app: nginx-server
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-lb
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: nginx-server
```

Nous allons utiliser la commande **`kubectl apply`** et passer le fichier YAML au paramètre **`-f`**.

Les trois services sont alors créés, et si vous exécutez la commande **`kubectl get deployment nginx-server`** , six réplicas s'exécutent, comme illustré à la *figure 14.15*, pour rendre le service hautement disponible :

```
 rithin kubectl get deployment nginx-server
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
nginx-server   6/6     6            6           8m17s
 rithin kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
nginx-server-777ff65664-j7lgw   1/1     Running   0          8m25s
nginx-server-777ff65664-kxtzx   1/1     Running   0          8m25s
nginx-server-777ff65664-mh8hj   1/1     Running   0          8m25s
nginx-server-777ff65664-wn79x   1/1     Running   0          8m25s
nginx-server-777ff65664-xc7kc   1/1     Running   0          8m25s
nginx-server-777ff65664-xl4nk   1/1     Running   0          8m25s
```

Figure 14.15 : Vérification du déploiement

Étant donné que notre déploiement est mis en service, nous devons vérifier l'adresse IP publique de l'équilibreur de charge que nous avons créé. Nous pouvons utiliser la commande **`kubectl get service nginx-lb --watch`**. Lorsque l'équilibreur de charge est en cours d'initialisation, **`EXTERNAL-IP`** s'affiche comme **`<pending>`**, le paramètre **`--wait`** permet à la commande d'être exécutée au premier plan, et lorsque l'IP publique est allouée, nous pouvons voir une nouvelle ligne, comme illustré ici :

```
rithin kubectl get service nginx-lb --watch
NAME       TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
nginx-lb   LoadBalancer   10.0.140.48   <pending>     80:30686/TCP   68s
nginx-lb   LoadBalancer   10.0.140.48   52.187.70.177   80:30686/TCP   115s
```

Figure 14.16 : Recherche de l'adresse IP publique de l'équilibreur de charge

Maintenant que nous disposons de l'adresse IP publique, nous pouvons accéder au navigateur et voir la page de destination NGINX, comme illustré à la *figure 14.17* :

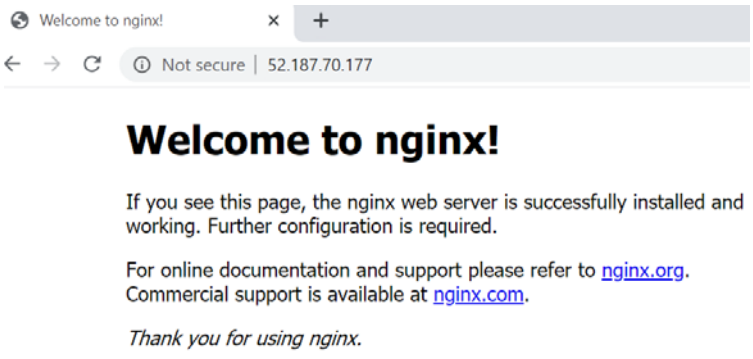


Figure 14.17 : Page de destination NGINX

De même, vous pouvez utiliser les fichiers **`YAML`** que nous avons abordés dans la section *Primitives Kubernetes* pour créer différents types de ressources.

Il existe de nombreuses commandes, telles que **`logs`**, **`describe`**, **`exec`** et **`delete`**, que les administrateurs doivent utiliser avec la commande **`kubectl`**. L'objectif de cette section était de vous permettre de créer un cluster AKS, de vous connecter au cluster et de déployer une application web simple.

Au cours de la section suivante, nous aborderons la mise en réseau AKS.

Mise en réseau AKS

La mise en réseau constitue un composant principal au sein d'un cluster Kubernetes. Les composants maîtres doivent être en mesure d'atteindre les nœuds esclaves et les pods qui s'exécutent sur eux, tandis que les nœuds de travail doivent être en mesure de communiquer entre eux, ainsi qu'avec les composants principaux.

Il peut être surprenant que le Kubernetes de base ne gère pas du tout la pile de mise en réseau. Il s'agit de la tâche du runtime de conteneur sur les nœuds.

Kubernetes a prescrit trois principes importants que tout runtime de conteneur doit respecter. Il s'agit des autorisations suivantes :

- Les pods doivent être en mesure de communiquer avec d'autres pods sans aucune transformation dans leurs adresses source ou de destination, ce qui est effectué à l'aide de la **traduction d'adresses réseau** (**NAT**).
- Les agents tels que kubelets doivent être en mesure de communiquer avec des pods directement sur les nœuds.
- Les pods hébergés directement sur le réseau hôte doivent toujours être en mesure de communiquer avec tous les pods du cluster.

Chaque pod obtient une adresse IP unique au sein du cluster Kubernetes, ainsi qu'une pile réseau complète, similaire aux machines virtuelles. Ils sont tous connectés au réseau de pont local créé par le **composant de l'interface de réseau de conteneurs** (**CNI**). Le composant CNI crée également la pile de mise en réseau du pod. Le réseau de pont s'adresse ensuite au réseau hôte et devient le vecteur du flux de trafic entre les pods et le réseau, et vice-versa.

CNI est une norme gérée et mise à jour par la **Cloud Native Computing Foundation** (**CNCF**), et de nombreux fournisseurs proposent leur propre implémentation de l'interface. Docker est l'un de ces fournisseurs. Il y en a d'autres, comme rkt (qui se prononce rocket), weave, calico, etc. Chacun a ses propres capacités et décide indépendamment des fonctionnalités du réseau, tout en veillant à ce que les principes les plus importants de la mise en réseau Kubernetes soient totalement respectés.

AKS fournit deux modèles de mise en réseau distincts :

- Kubenet
- Azure CNI

Kubenet

Kubenet est l'infrastructure de réseau par défaut dans AKS. Sous Kubenet, chaque nœud obtient une adresse IP à partir du sous-réseau du réseau virtuel auquel il est connecté. Les pods ne reçoivent pas d'adresses IP du sous-réseau. Au lieu de cela, un schéma d'adressage distinct est utilisé pour fournir des adresses IP aux services pods et Kubernetes. Lors de la création d'une instance AKS, il est important de définir la plage d'adresses IP pour les pods et les services. Étant donné que les pods ne sont pas sur le même réseau que les nœuds, les requêtes depuis et vers les pods sont toujours de type NAT/routées pour remplacer l'IP de pod source par l'adresse IP du nœud et inversement.

Dans le routage défini par l'utilisateur, Azure peut prendre en charge jusqu'à 400 itinéraires, et vous ne pouvez pas avoir de cluster supérieur à 400 nœuds. La *figure 14.18* montre comment le nœud AKS reçoit une adresse IP à partir du réseau virtuel, mais pas les pods créés dans le nœud :

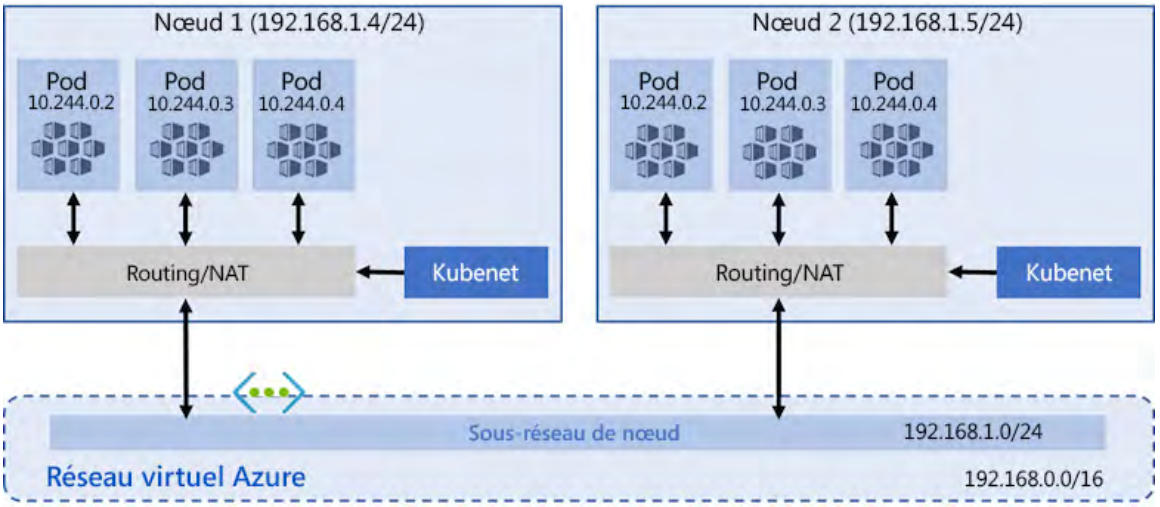


Figure 14.18 : Mise en réseau dans AKS

Par défaut, ce kubenet est configuré avec 110 pods par nœud. Cela signifie qu'il peut y avoir un maximum de 110 * 400 pods dans un cluster Kubernetes par défaut. Le nombre maximum de pods par nœud est de 250.

Ce schéma doit être utilisé lorsque la disponibilité de l'adresse IP et le routage défini par l'utilisateur ne sont pas une contrainte.

Dans l'interface de ligne de commande Azure, vous pouvez exécuter la commande suivante pour créer une instance AKS à l'aide de cette pile de mise en réseau :

```
az aks create \
    --resource-group myResourceGroup \
    --name myAKSCluster \
    --node-count 3 \
    --network-plugin kubenet \
    --service-cidr 10.0.0.0/16 \
    --dns-service-ip 10.0.0.10 \
    --pod-cidr 10.244.0.0/16 \
    --docker-bridge-address 172.17.0.1/16 \
    --vnet-subnet-id $SUBNET_ID \
    --service-principal <appId> \
    --client-secret <password>
```

Notez que toutes les adresses IP sont explicitement fournies pour les ressources de service, les pods, les nœuds et les ponts Docker. Il s'agit de plages d'adresses IP qui ne se chevauchent pas. Notez également que Kubenet est utilisé comme plug-in de réseau.

Azure CNI (mise en réseau avancée)

Avec Azure CNI, chaque nœud et pod obtient directement une adresse IP attribuée à partir du sous-réseau de réseau. Cela signifie qu'il peut y avoir autant de pods qu'il existe d'adresses IP uniques disponibles sur un sous-réseau. Cela rend la planification de la plage d'adresses IP beaucoup plus importante dans le cadre de cette stratégie de mise en réseau.

Il est important de noter que l'hébergement Windows n'est possible qu'à l'aide de la pile de mise en réseau Azure CNI. En outre, certains composants AKS, tels que les nœuds virtuels et les kubelets virtuels, dépendent également de la pile Azure CNI. Il est nécessaire de réserver des adresses IP à l'avance, en fonction du nombre de pods qui seront créés. Il doit toujours y avoir des adresses IP supplémentaires disponibles sur le sous-réseau, afin d'éviter l'épuisement des adresses IP ou pour éviter de devoir reconstruire le cluster pour un sous-réseau plus important en raison de la demande de l'application.

Par défaut, cette pile de mise en réseau est configurée pour 30 pods par nœud et peut être configurée avec 250 pods maximum par nœud.

La commande à exécuter pour créer une instance AKS à l'aide de cette pile de mise en réseau est illustrée ici :

```
az aks create \
    --resource-group myResourceGroup \
    --name myAKSCluster \
    --network-plugin azure \
    --vnet-subnet-id <subnet-id> \
    --docker-bridge-address 172.17.0.1/16 \
    --dns-service-ip 10.2.0.10 \
    --service-cidr 10.2.0.0/24 \
    --generate-ssh-keys
```

Notez que toutes les adresses IP sont explicitement fournies pour les ressources de service, les pods, les nœuds et les ponts Docker. Il s'agit de plages d'adresses IP qui ne se chevauchent pas. Notez également qu'Azure est utilisé comme plug-in réseau.

Jusqu'à présent, vous avez appris à déployer une solution et à gérer la mise en réseau d'un cluster AKS. La sécurité est un autre facteur important qui doit être abordé. Dans la section suivante, nous allons nous concentrer sur les options d'accès et d'identité pour AKS.

Accès et identité pour AKS

Les clusters Kubernetes peuvent être sécurisés de plusieurs façons.

Le compte de service est l'un des principaux types d'utilisateurs dans Kubernetes. L'API Kubernetes gère le compte de service. Les pods autorisés peuvent communiquer avec le serveur d'API à l'aide des informations d'identification des comptes de service, qui sont stockées sous la forme de secrets Kubernetes. Kubernetes ne dispose pas d'un magasin de données ou d'un fournisseur d'identité propre. Il délègue la responsabilité de l'authentification à des logiciels externes. Il fournit un plug-in d'authentification qui vérifie les informations d'identification données et les mappe aux groupes disponibles. Si l'authentification est réussie, la demande passe à un autre ensemble de plug-ins d'autorisation pour vérifier les niveaux d'autorisation de l'utilisateur sur le cluster, ainsi que les ressources étendues à l'espace de noms.

Pour Azure, la meilleure intégration de la sécurité consisterait à utiliser Azure AD. À l'aide d'Azure AD, vous pouvez également apporter vos identités sur site à AKS afin de garantir une gestion centralisée des comptes et de la sécurité. Le workflow de base de l'intégration d'Azure AD est illustré à la *figure 14.19* :

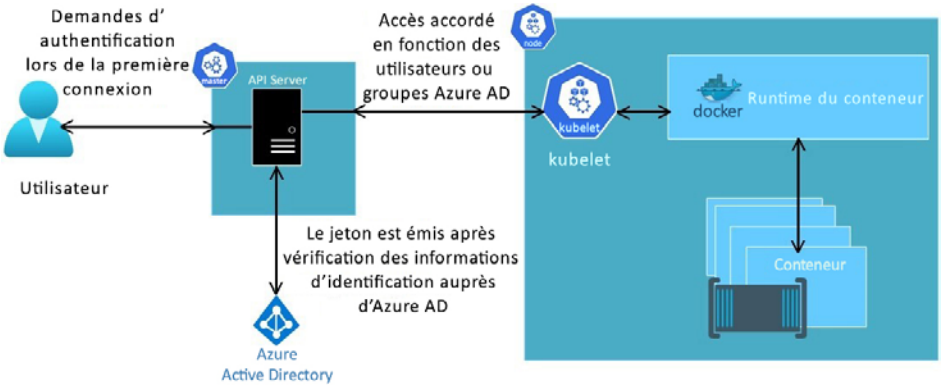


Figure 14.19 : Workflow de base de l'intégration d'Azure AD

Les utilisateurs ou les groupes peuvent être autorisés à accéder aux ressources au sein d'un espace de noms ou sur un cluster. Dans la section précédente, nous avons utilisé la commande **`az aks get-credential`** pour obtenir les informations d'identification et le contexte de configuration de kubectl. Lorsque l'utilisateur tente d'interagir avec kubectl, il est invité à se connecter à l'aide de ses informations d'identification Azure AD. Azure AD valide les informations d'identification et un jeton est émis pour l'utilisateur. En fonction du niveau d'accès dont ils disposent, ils peuvent accéder aux ressources du cluster ou de l'espace de noms.

En outre, vous pouvez tirer parti du **contrôle d'accès basé sur le rôle** (**RBAC**) Azure pour limiter l'accès aux ressources du groupe de ressources.

Dans la section suivante, nous allons aborder le kubelet virtuel, qui est l'une des façons les plus rapides de mettre à l'échelle un cluster.

Kubelet virtuel

Le kubelet virtuel est actuellement en version préliminaire et est géré par l'organisation CNCF. Il s'agit d'une approche tout à fait innovante qu'AKS utilise à des fins d'évolutivité. Le kubelet virtuel est déployé sur le cluster Kubernetes en tant que pod. Le conteneur exécuté dans le pod utilise le kit de développement logiciel Kubernetes pour créer une nouvelle ressource de nœud et se représente pour l'ensemble du cluster en tant que nœud. Les composants du cluster, y compris le serveur d'API, le planificateur et les contrôleurs, le considèrent et le traitent comme un nœud et planifient des pods sur celui-ci.

Toutefois, lorsqu'un pod est planifié sur ce nœud qui se déguise en tant que nœud, il communique avec ses composants back-end, appelés fournisseurs, pour créer, supprimer et mettre à jour les pods. Les instances de conteneur Azure sont l'un des principaux fournisseurs d'Azure. Azure Batch peut également être utilisé en tant que fournisseur. Cela signifie que les conteneurs sont réellement créés sur des instances de conteneur ou sur un lot Azure plutôt que sur le cluster lui-même ; toutefois, elles sont gérées par le cluster. L'architecture du kubelet virtuel est illustrée à la *figure 14.20* :

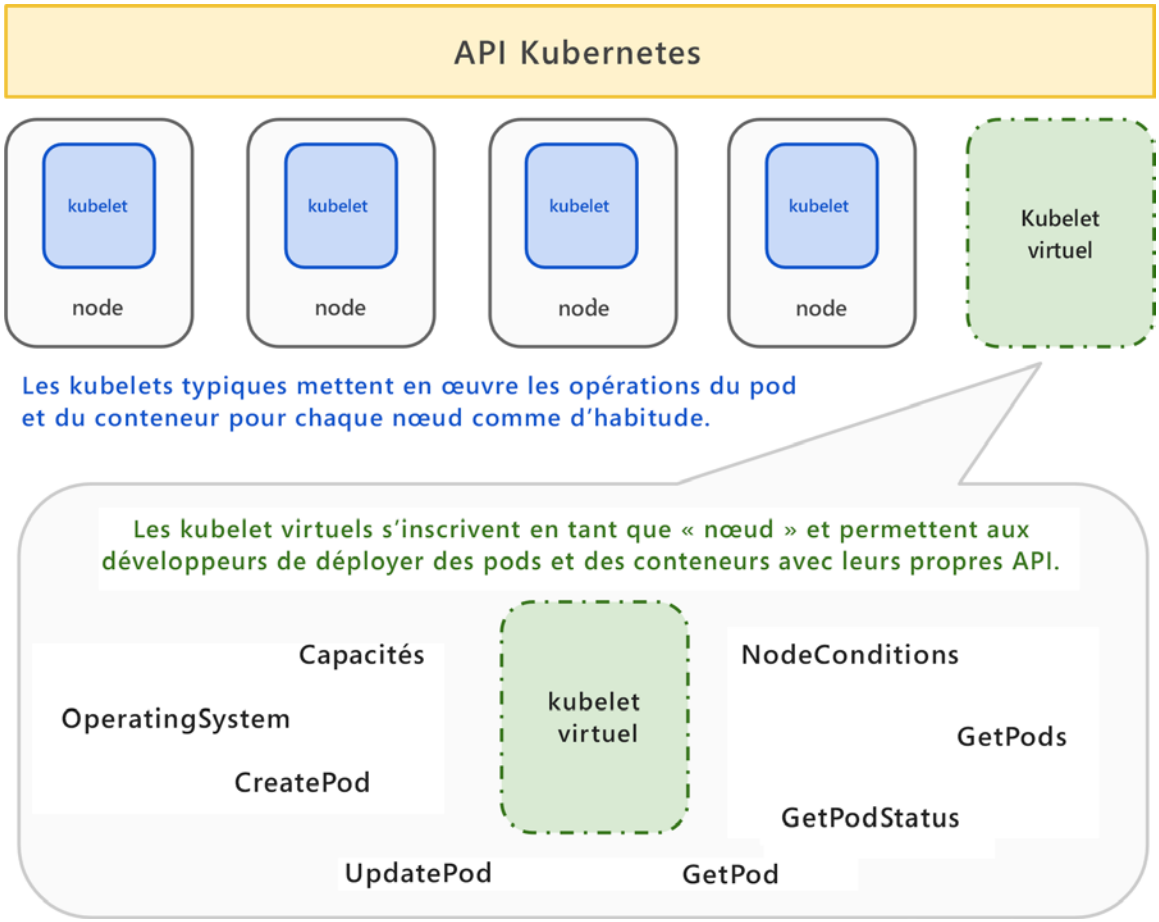


Figure 14.20 : Architecture du kubelet virtuel

Notez que le kubelet virtuel est représenté comme un nœud au sein du cluster et peut vous aider à héberger et à gérer des pods, comme un kubelet normal le ferait. Toutefois, le kubelet virtuel a une limite ; c'est ce que nous allons aborder dans la section suivante.

Nœuds virtuels

L'une des limitations du kubelet virtuel réside dans le fait que les pods déployés sur les fournisseurs de kubelet virtuel sont isolés et ne communiquent pas avec d'autres pods du cluster. S'il est nécessaire que les pods de ces fournisseurs s'adressent à d'autres pods et nœuds du cluster, et inversement, des nœuds virtuels doivent être créés. Les nœuds virtuels sont créés sur un sous-réseau différent sur le même réseau virtuel que celui qui héberge des nœuds de cluster Kubernetes, ce qui peut permettre la communication entre les pods. Seul le système d'exploitation Linux est pris en charge à l'heure actuelle pour l'utilisation des nœuds virtuels.

Les nœuds virtuels donnent une perception d'un nœud ; toutefois, le nœud n'existe pas. Tout ce qui est prévu sur ce nœud est réellement créé dans les instances de conteneur Azure. Les nœuds virtuels sont basés sur des kubelets virtuels, mais disposent d'une fonctionnalité supplémentaire de communication sans effort entre le cluster et les instances de conteneur Azure.

Lors du déploiement de pods sur des nœuds virtuels, la définition du pod doit contenir un sélecteur de nœud approprié pour faire référence aux nœuds virtuels et également aux tolérances, comme illustré dans l'extrait de code suivant :

```
nodeSelector:
  kubernetes.io/role: agent
  beta.kubernetes.io/os: linux
  type: virtual-kubelet
tolerations:
- key: virtual-kubelet.io/provider
  operator: Exists
- key: azure.com/aci
  effect: NoSchedule
```

Ici, le sélecteur de nœud utilise la propriété **`type`** pour faire référence au kubelet virtuel et la propriété **`tolerations`** pour informer Kubernetes que les nœuds contenant des problèmes, **`virtual-kubelet.io/provider`**, doivent permettre le déploiement de ces pods.

Résumé

Kubernetes est l'orchestrateur de conteneurs le plus largement utilisé. Il fonctionne avec différents runtimes de conteneur et de réseau. Dans ce chapitre, vous avez appris les notions de base de Kubernetes, de son architecture et de certains des composants d'infrastructure importants, tels qu'ETCD, le serveur d'API, les gestionnaires de contrôleur et le planificateur, ainsi que leur objectif. De plus, nous avons examiné les ressources importantes qui peuvent être déployées pour gérer les applications, telles que les pods, les contrôleurs de réplication, les ReplicaSets, les déploiements et les services.

AKS fournit deux ou trois piles de mise en réseau différentes : Azure CNI et Kubenet. Elles fournissent différentes stratégies pour attribuer des adresses IP aux pods. Tandis qu'Azure CNI fournit des adresses IP aux pods depuis le sous-réseau sous-jacent, Kubenet utilise uniquement des adresses IP virtuelles.

Nous avons également abordé certaines des fonctionnalités fournies exclusivement par Azure, telles que les nœuds virtuels et les concepts entourant le kubelet virtuel. Dans le chapitre suivant, nous allons découvrir le provisionnement et la configuration des ressources avec des modèles ARM.

15

Déploiements inter-abonnements à l'aide de modèles ARM

Les modèles Azure Resource Manager (**ARM**) sont le mécanisme privilégié pour le provisionnement des ressources et leur configuration sur Azure.

Les modèles ARM permettent d'implémenter un paradigme relativement nouveau, dénommé **infrastructure en tant que code** (**IaC**). Les modèles ARM convertissent l'infrastructure et sa configuration en code, ce qui offre de nombreux avantages. L'IaC apporte un niveau élevé de cohérence et de prévisibilité aux déploiements dans les environnements. Il garantit également que les environnements peuvent être testés avant de passer à la production, et, enfin, il offre un niveau élevé de confiance dans le processus de déploiement, la maintenance et la gouvernance.

Dans ce chapitre, nous allons aborder les thèmes suivants :

- Modèles ARM
- Déploiement de groupes de ressources avec des modèles ARM
- Déploiement de ressources dans les abonnements et les groupes de ressources
- Déploiement inter-abonnements et de groupes de ressources à l'aide de modèles liés
- Création de modèles ARM pour les solutions PaaS, de données et IaaS

Modèles ARM

L'IaC a l'avantage de pouvoir être contrôlé par la version. Il peut également être réutilisé dans les environnements, ce qui offre un degré élevé de cohérence et de prévisibilité dans les déploiements, et garantit que l'impact et le résultat du déploiement d'un modèle ARM sont identiques, quel que soit le nombre de fois où le modèle est déployé. Cette fonction est également nommée **idempotence**.

Les modèles ARM ont débuté avec l'introduction de la spécification ARM et sont désormais plus riches en fonctions et plus matures. Il est important de comprendre qu'en ce qui concerne les fonctions, il y a généralement un intervalle de quelques semaines à quelques mois entre la configuration réelle des ressources et la disponibilité de la configuration dans les modèles ARM.

Chaque ressource a sa propre configuration. Cette configuration peut être affectée d'une multitude de façons, notamment à l'aide d'Azure PowerShell, de l'interface de ligne de commande Azure, des kits SDK Azure, des API REST et des modèles ARM.

Chacune de ces techniques dispose de son propre cycle de développement et de publication, qui est différent du développement réel des ressources. Essayons de comprendre cela à l'aide d'un exemple.

La ressource Azure Databricks a sa propre cadence et son propre cycle de vie de développement. Les consommateurs de cette ressource ont leur propre cycle de vie de développement, qui est de son côté différent du développement réel des ressources. Si Databricks obtient sa première version le 31 décembre, ses cmdlets de commande Azure PowerShell peuvent ne pas être disponibles à la même date et seront peut-être publiées le 31 janvier de l'année suivante ; de même, ces fonctions peuvent être disponibles dans les modèles d'API REST et ARM vers le 15 janvier.

Les modèles ARM sont des documents basés sur JSON qui, une fois exécutés, appellent une API REST sur le plan de gestion Azure et lui soumettent l'intégralité du document. L'API REST dispose de son propre cycle de vie de développement, et le schéma JSON pour la ressource a son propre cycle de vie également.

Cela signifie que le développement Azure d'une fonction au sein d'une ressource doit se produire dans au moins trois composants différents avant de pouvoir être consommé à partir de modèles ARM. Il s'agit notamment des composants suivants :

- La ressource elle-même
- L'API REST pour la ressource
- Le schéma de ressource de modèle ARM

Chaque ressource du modèle ARM possède la propriété **`apiVersion`**. Cette propriété permet de décider de la version de l'API REST qui doit être utilisée pour mettre en service et déployer la ressource. La *figure 15.1* montre le flux des demandes depuis le modèle ARM vers les API de ressources qui sont responsables de la création, de la mise à jour et de la suppression des ressources :

Figure 15.1 : Flux de requête

Une configuration de ressource, telle qu'un compte de stockage dans un modèle ARM, se présente comme suit :

```
{
 "type": "Microsoft.Storage/storageAccounts",
  "apiVersion": "2019-04-01",
   "name": "[variables('storage2')]",
   "location": "[resourceGroup().location]",
   "kind": "Storage",
   "sku": {
                "name": "Standard_LRS"
          }
}
```

Dans le code précédent, la disponibilité de ce schéma pour la définition de **`SKU`** est basée sur le développement du schéma du modèle ARM. La disponibilité de l'API REST et son numéro de version sont déterminés par **`apiVersion`**, qui se trouve être **`2019-04-01`**. La ressource réelle est déterminée par la propriété **`type`**, qui comporte les deux parties suivantes :

- **Espace de noms de fournisseur de ressources** : les ressources dans Azure sont hébergées dans des espaces de noms et les ressources associées sont hébergées dans le même espace de noms.
- **Type de ressource** : les ressources sont référencées à l'aide du nom de leur type.

Dans ce cas, la ressource est identifiée par son nom de fournisseur et son type, qui se trouve être **`Microsoft.Storage/storageaccounts`**.

Auparavant, les modèles ARM attendaient que les groupes de ressources soient disponibles avant leur déploiement. Ils se limitaient également au déploiement dans un seul groupe de ressources au sein d'un seul abonnement.

Cela signifiait que, jusqu'à récemment, un modèle ARM pouvait déployer toutes les ressources au sein d'un seul groupe de ressources. Les modèles Azure ARM disposent désormais d'une fonctionnalité pour le déploiement de ressources sur plusieurs groupes de ressources au sein du même abonnement ou de plusieurs abonnements simultanément. Il est désormais possible de créer des groupes de ressources dans le cadre de modèles ARM, ce qui signifie qu'il est désormais possible de déployer des ressources dans plusieurs régions dans différents groupes de ressources.

Pourquoi aurions-nous besoin de créer des groupes de ressources à partir de modèles ARM, et pourquoi aurions-nous besoin d'avoir des déploiements d'inter-abonnements et de groupe de ressources simultanément ?

Pour apprécier la valeur de la création d'un groupe de ressources et de déploiements inter-abonnements, nous devons comprendre comment les déploiements ont été effectués avant que ces fonctions ne soient disponibles.

Pour déployer un modèle ARM, un groupe de ressources est une condition préalable. Les groupes de ressources doivent être créés avant le déploiement d'un modèle. Les développeurs utilisent PowerShell, l'interface de ligne de commande Azure ou l'API REST pour créer des groupes de ressources, puis initient le déploiement de modèles ARM. Cela signifie que tout déploiement end-to-end se compose de plusieurs étapes. La première étape consiste à provisionner le groupe de ressources et l'étape suivante est le déploiement du modèle ARM dans ce groupe de ressources nouvellement créé. Ces étapes peuvent être exécutées à l'aide d'un seul script PowerShell ou d'étapes individuelles à partir de la ligne de commande PowerShell. Le script PowerShell doit être complet en ce qui concerne le code associé à la gestion des exceptions, en prenant en charge les cas de périphérie et en veillant à ce qu'il n'y ait aucun bogue avant qu'il ne soit considéré comme prêt pour l'entreprise. Il est important de noter que les groupes de ressources peuvent être supprimés d'Azure, et la prochaine fois que le script s'exécute, ils devraient être disponibles. Ce script échouerait parce qu'il pourrait supposer que le groupe de ressources existe. En bref, le déploiement du modèle ARM dans un groupe de ressources doit être une étape atomique plutôt que plusieurs étapes.

Comparez cela avec la possibilité de créer ensemble des groupes de ressources et ses ressources constitutives dans les mêmes modèles ARM. Chaque fois que vous déployez le modèle, cela garantit que les groupes de ressources sont créés s'ils n'existent pas encore et que le système continue de déployer des ressources pour ces derniers après leur création.

Examinons également comment ces nouvelles fonctions peuvent aider à supprimer certaines des contraintes techniques liées aux sites de récupération d'urgence.

Avant ces fonctions, si vous aviez à déployer une solution qui avait été conçue en tenant compte de la reprise après sinistre, deux déploiements distincts étaient disponibles : un déploiement pour la région principale et un autre déploiement pour la région secondaire. Par exemple, si vous déployiez une application MVC ASP.NET à l'aide d'App Services, vous deviez créer un service d'application et le configurer pour la région principale, puis effectuer un autre déploiement avec le même modèle dans une autre région en utilisant un autre fichier **`parameters`**. Lors du déploiement d'un autre ensemble de ressources dans une autre région, comme mentionné précédemment, les paramètres utilisés avec le modèle doivent être différents pour refléter les différences entre les deux environnements. Les paramètres incluent des modifications telles que la chaîne de connexion SQL, le domaine et les adresses IP, ainsi que d'autres éléments de configuration propres à un environnement.

Avec la disponibilité inter-abonnement et le déploiement de groupe de ressources, il est possible de créer le site de récupération d'urgence en même temps que le site principal. Cela élimine deux déploiements et garantit que la même configuration peut être utilisée sur plusieurs sites.

Déploiement de groupes de ressources avec des modèles ARM

Dans cette section, un modèle ARM sera créé et déployé, ce qui créera deux groupes de ressources au sein du même abonnement.

Pour utiliser PowerShell afin de déployer des modèles qui contiennent des groupes de ressources et des ressources inter-abonnements, la dernière version de PowerShell doit être utilisée. Au moment de l'écriture de ce document, la version du module Azure 3.3.0 est utilisée :

```
PS C:\WINDOWS\system32> get-module -Name az

ModuleType Version    Name
---------- -------    ----
Script     3.3.0      az
```

Figure 15.2 : vérification de la dernière version du module Azure

Si le dernier module Azure n'est pas installé, il peut être installé à l'aide de la commande suivante :

```
install-module  -Name az  -Force
```

Il est temps de créer un modèle ARM qui créera plusieurs groupes de ressources dans le même abonnement. Le code du modèle ARM est comme suit :

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "resourceGroupInfo": {
      "type": "array"    },
    "multiLocation": {
      "type": "array"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Resources/resourceGroups",
      "location": "[parameters('multiLocation')[copyIndex()]]",
      "name": "[parameters('resourceGroupInfo')[copyIndex()]]",
      "apiVersion": "2019-10-01",
      "copy": {
        "name": "allResourceGroups",
        "count": "[length(parameters('resourceGroupInfo'))]"
      },
      "properties": {}
    }
  ],
  "outputs": {}
}
```

La première section du code concerne les paramètres que les modèles ARM attendent. Ce sont des paramètres obligatoires, et toute personne déployant ces modèles doit fournir des valeurs pour eux. Les valeurs de tableau doivent être fournies pour les deux paramètres.

La deuxième section majeure est le tableau JSON de **ressources**, qui peut contenir plusieurs ressources. Dans cet exemple, nous créons des groupes de ressources, afin de les déclarer dans la section **ressources**. Les groupes de ressources sont mis en service dans une boucle en utilisant l'élément **copy**. L'élément **copy** garantit que la ressource est exécutée un certain nombre de fois spécifié et crée une nouvelle ressource dans chaque itération. Si nous envoyons deux valeurs pour le paramètre de tableau **resourceGroupInfo**, la longueur du tableau sera deux et l'élément **copy** veillera à ce que la ressource **resourceGroup** soit exécutée deux fois.

Tous les noms de ressources d'un modèle doivent être uniques pour un type de ressource. La fonction **copyIndex** est utilisée pour attribuer le numéro d'itération actuel au nom global de la ressource et le rendre unique. En outre, il est souhaitable que les groupes de ressources soient créés dans différentes régions à l'aide de noms de régions distincts envoyés en tant que paramètres. L'affectation d'un nom et d'un emplacement pour chaque groupe de ressources est effectuée à l'aide de la fonction **copyIndex**.

Le code du fichier **paramètres** s'affiche ensuite. Ce code est assez simple et fournit des valeurs de tableau aux deux paramètres attendus par le modèle précédent. Les valeurs de ce fichier doivent être modifiées pour tous les paramètres en fonction de votre environnement :

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "resourceGroupInfo": {
      "value": [ "firstResourceGroup", "SeocndResourceGroup" ]
    },
    "multiLocation": {
      "value": [
        "West Europe",
        "East US"
      ]
    }
  }
}
```

Déploiement des modèles ARM

Pour déployer ce modèle à l'aide de PowerShell, connectez-vous à Azure avec des informations d'identification valides à l'aide de la commande suivante :

```
Login-AzAccount
```

Les informations d'identification valides peuvent être un compte d'utilisateur ou un principal de service. Ensuite, utilisez une nouvelle cmdlet de commande **`New-AzDeployment`** pour déployer le modèle. Le script de déploiement est disponible dans le fichier **`multipleResourceGroups.ps1`** :

```
New-AzDeployment  -Location "West Europe"  -TemplateFile "c:\users\rites\
source\repos\CrossSubscription\CrossSubscription\multipleResourceGroups.
json" -TemplateParameterFile "c:\users\rites\source\repos\CrossSubscription\
CrossSubscription\multipleResourceGroups.parameters.json"  -Verbose
```

Il est important de comprendre que la cmdlet de commande **`New-AzResourceGroupDeployment`** ne peut pas être utilisée ici car la portée de la cmdlet de commande **`New-AzResourceGroupDeployment`** est un groupe de ressources et elle attend qu'un groupe de ressources soit disponible en tant que condition préalable. Pour déployer des ressources au niveau de l'abonnement, Azure avait publié une nouvelle cmdlet de commande qui peut fonctionner au-delà de la portée du groupe de ressources. La nouvelle cmdlet de commande **`new-AzDeployment`**, fonctionne au niveau de l'abonnement. Il est également possible d'avoir un déploiement au niveau du groupe de gestion. Les groupes de gestion sont à un niveau plus élevé que les abonnements utilisant la cmdlet de commande **`New-AzManagementGroupDeployment`**.

Déploiement de modèles à l'aide de l'interface de ligne de commande Azure

Le même modèle peut également être déployé à l'aide de l'interface de ligne de commande Azure. Voici les étapes pour le déployer à l'aide de l'interface de ligne de commande Azure :

1. Utilisez la dernière version de l'interface de ligne de commande Azure pour créer des groupes de ressources à l'aide du modèle ARM. Au moment de l'écriture, la version 2.0.75 a été utilisée pour le déploiement, comme illustré ici :

```
C:\Users\Ritesh>az --version
azure-cli                         2.0.75 *

command-modules-nspkg              2.0.3
core                              2.0.75 *
nspkg                              3.0.4
telemetry                          1.0.4

Extensions:
aks-preview                       0.4.18
azure-devops                      0.16.0

Python location 'C:\Program Files (x86)\Microsoft SDKs\Azure\CLI2\python.exe'
Extensions directory 'C:\Users\Ritesh\.azure\cliextensions'

Python (Windows) 3.6.6 (v3.6.6:4cf1f54eb7, Jun 27 2018, 02:47:15) [MSC v.1900 32 bit (Intel)]

Legal docs and information: aka.ms/AzureCliLegal
```

Figure 15.3 : Vérification de la version de l'interface de ligne de commande Azure

2. Connectez-vous à Azure à l'aide de la commande suivante et sélectionnez l'abonnement approprié à utiliser :

```
az login
```

3. Si la connexion a accès à plusieurs abonnements, sélectionnez l'abonnement approprié à l'aide de la commande suivante :

```
az account set -subscription xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

4. Exécutez le déploiement à l'aide de la commande suivante. Le script de déploiement est disponible dans le fichier **`multipleResourceGroupsCLI.txt`** :

```
C:\Users\Ritesh>az deployment create—location westus—template-file "C:\
users\rites\source\repos\CrossSubscription\CrossSubscription\azuredeploy.
json—parameters @"C:\users\rites\source\repos\CrossSubscription\
CrossSubscription\azuredeploy.parameters.json"—verbose
```

Une fois la commande exécutée, les ressources définies dans le modèle ARM doivent être reflétées sur le portail Azure.

Déploiement de ressources dans les abonnements et les groupes de ressources

Dans la dernière section, les groupes de ressources ont été créés dans le cadre de modèles ARM. Une autre fonction Azure consiste à fournir des ressources dans plusieurs abonnements simultanément à partir d'un seul déploiement à l'aide d'un modèle ARM unique. Dans cette section, nous fournirons un nouveau compte de stockage dans deux abonnements et groupes de ressources différents. La personne déployant le modèle ARM sélectionnerait l'un des abonnements en tant qu'abonnement de base, à l'aide duquel elle initierait le déploiement et provisionnerait le compte de stockage dans l'abonnement actuel et dans un autre. La condition préalable au déploiement de ce modèle est que la personne effectuant le déploiement doit avoir accès à au moins deux abonnements et qu'elle ait des droits de contributeur sur ces abonnements. La liste de codes est affichée ici et est disponible dans le fichier **`CrossSubscriptionStorageAccount.json`** dans le code d'accompagnement :

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
      "storagePrefix1": {
        "type": "string",
        "defaultValue": "st01"
    ...
        "type": "string",
        "defaultValue": "rg01"
      },
      "remoteSub": {
        "type": "string",
        "defaultValue": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
      }
   ...
                            }
                        }
                    ],
                    "outputs": {}
                }
            }
        }
    ],
    "outputs": {}
}
```

Il est important de noter que les noms du groupe de ressources utilisé dans le code doivent déjà être disponibles dans les abonnements respectifs. Le code lèvera une erreur si les groupes de ressources ne sont pas disponibles. En outre, les noms du groupe de ressources doivent correspondre exactement à ceux du modèle ARM.

Le code du déploiement de ce modèle s'affiche ensuite. Dans ce cas, nous utilisons **`New-AzResourceGroupDeployment`**, car la portée du déploiement est un groupe de ressources. Le script de déploiement est disponible dans le fichier **`CrossSubscriptionStorageAccount.ps1`**, dans le bundle de code :

```
New-AzResourceGroupDeployment  -TemplateFile "<< path to your
CrossSubscriptionStorageAccount.json file >>" -ResourceGroupName "<<provide
your base subscription resource group name>>" -storagePrefix1 <<provide prefix
for first storage account>> -storagePrefix2 <<provide prefix for first storage
account>> -verbose
```

Une fois la commande exécutée, les ressources définies dans le modèle ARM doivent être reflétées dans le portail Azure.

Un autre exemple de déploiements inter-abonnements et de groupes de ressources

Dans cette section, nous créons deux comptes de stockage dans deux abonnements, groupes de ressources et régions différents à partir d'un modèle ARM et d'un déploiement unique. Celui-ci utilise l'approche de modèles imbriqués avec l'élément **copy** pour fournir des noms et des emplacements différents à ces groupes de ressources dans différents abonnements.

Toutefois, avant de pouvoir exécuter l'ensemble suivant de modèles ARM, une instance Azure Key Vault doit être provisionnée en tant que condition préalable et un secret doit y être ajouté. En effet, les noms des comptes de stockage sont récupérés à partir d'Azure Key Vault et transmis en tant que paramètres aux modèles ARM pour le provisionnement du compte de stockage.

Pour provisionner Azure Key Vault à l'aide d'Azure PowerShell, l'ensemble de commandes suivant peut être exécuté. Le code des commandes suivantes est disponible dans le fichier **CreateKeyVaultandSetSecret.ps1** :

```
New-AzResourceGroup -Location <<replace with location of your key vault>>
-Name <<replace with name of your resource group for key vault>> -verbose
New-AzureRmKeyVault -Name <<replace with name of your key vault>>
-ResourceGroupName <<replace with name of your resource group for
key vault>>  -Location <<replace with location of your key vault>>
-EnabledForDeployment -EnabledForTemplateDeployment -EnabledForDiskEncryption
-EnableSoftDelete -EnablePurgeProtection -Sku Standard -Verbose
```

Vous devez noter que la valeur **ResourceID** doit être notée à partir du résultat de la cmdlet de commande **New-AzKeyVault**. Cette valeur devra être remplacée dans le fichier **paramètres**. Pour plus de détails, consultez la *figure 15.4* :

Figure 15.4 : Création d'une instance Key Vault

Exécutez la commande suivante pour ajouter un nouveau secret à l'instance Azure Key Vault nouvellement créée :

```
Set-AzKeyVaultSecret -VaultName <<replace with name of your key vault>> -Name
<<replace with name of yoursecret>> -SecretValue $(ConvertTo-SecureString
-String <<replace with value of your secret>> -AsPlainText -Force ) -Verbose
```

La liste de codes est disponible dans le fichier **CrossSubscriptionNestedStorageAccount.json** dans le bundle de code :

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "hostingPlanNames": {
      "type": "array",
      "minLength": 1
    },
  ...
      "type": "Microsoft.Resources/deployments",
      "name": "deployment01",
      "apiVersion": "2019-10-01",
      "subscriptionId": "[parameters('subscriptions')[copyIndex()]]",
      "resourceGroup": "[parameters('resourceGroups')[copyIndex()]]",
      "copy": {
        "count": "[length(parameters('hostingPlanNames'))]",
        "name": "mywebsites",        "mode": "Parallel"
      },
    ...
              "kind": "Storage",
              "properties": {
              }
            }
          ]
...
```

Voici le code du fichier **paramètres**. Il est disponible dans le fichier **CrossSubscriptionNestedStorageAccount.parameters.json** :

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "hostingPlanNames": {
   ...
    "storageKey": {
      "reference": {
        "keyVault": { "id": "<<replace it with the value of Key vault
ResourceId noted before>>" },
        "secretName": "<<replace with the name of the secret available
in Key vault>>"
      }
    }
  }
}
```

Voici le code PowerShell pour le déploiement du modèle précédent. Le script de déploiement est disponible dans le fichier **CrossSubscriptionNestedStorageAccount.ps1** :

```
New-AzResourceGroupDeployment  -TemplateFile "c:\users\rites\source\repos\
CrossSubscription\CrossSubscription\CrossSubscriptionNestedStorageAccount.
json" -ResourceGroupName rg01 -TemplateParameterFile "c:\
users\rites\source\repos\CrossSubscription\CrossSubscription\
CrossSubscriptionNestedStorageAccount.parameters.json" -Verbose
```

Une fois la commande exécutée, les ressources définies dans le modèle ARM doivent être reflétées dans le portail Azure.

Déploiement inter-abonnements et de groupes de ressources à l'aide de modèles liés

L'exemple précédent utilisait des modèles imbriqués à déployer sur plusieurs abonnements et groupes de ressources. Dans l'exemple suivant, nous déploierons plusieurs plans App Service dans des abonnements et des groupes de ressources distincts à l'aide de modèles liés. Les modèles liés sont stockés dans le stockage Azure Blob, qui est protégé à l'aide de stratégies. Cela signifie que seul le détenteur de la clé de compte de stockage ou d'une signature d'accès partagé valide peut accéder à ce modèle. La clé d'accès est stockée dans Azure Key Vault et est accessible à partir du fichier **`paramètres`** à l'aide des références sous l'élément **`storageKey`**. Vous devez charger le fichier **`website.json`** dans un conteneur du stockage Azure Blob. Le fichier **`website.json`** est un modèle lié responsable du provisionnement d'un plan App service et d'un service d'application. Le fichier est protégé à l'aide de la politique **Privé (pas d'accès anonyme)**, comme illustré à la *figure* 15.5. Une politique de confidentialité garantit que l'accès anonyme n'est pas autorisé. Dans le cas présent, nous avons créé un conteneur nommé **`armtemplates`** et nous l'avons défini avec une stratégie privée :

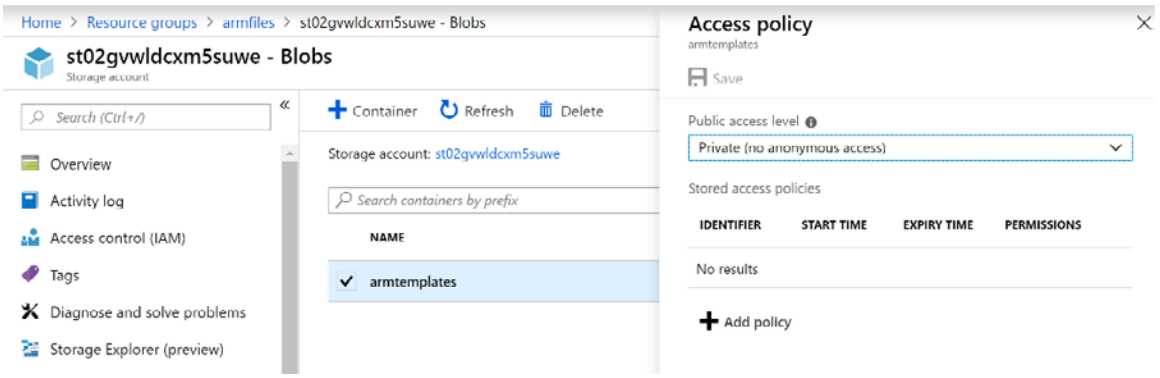


Figure 15.5 : Définition d'une stratégie privée pour le conteneur

Ce fichier est accessible uniquement à l'aide des clés **Shared Access Signature (SAS)**. Les clés SAS peuvent être générées à partir du portail Azure pour un compte de stockage à l'aide de l'élément de **signature d'accès partagé** dans le menu de gauche illustré à la *figure* 15.6. Vous devez cliquer sur le bouton **Générer les SAS et la chaîne de connexion** pour générer le jeton SAS. Notez qu'un jeton SAS ne s'affiche qu'une fois et n'est pas stocké dans Azure. Copiez-le et stockez-le quelque part afin qu'il puisse être chargé dans Azure Key Vault. La *figure* 15.6 montre la génération du jeton SAS :

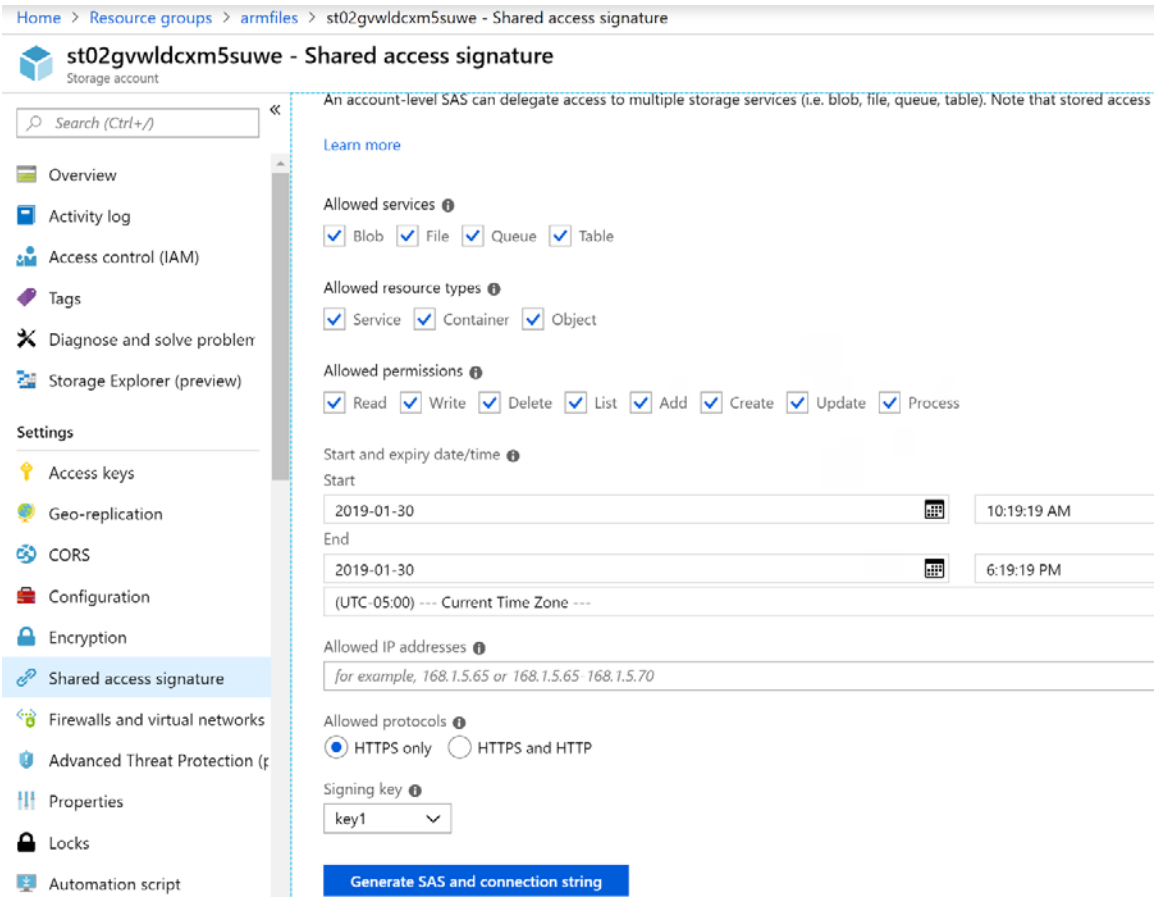


Figure 15.6 : Génération d'un jeton SAS dans le portail Azure

Nous utiliserons la même instance Key Vault que celle qui a été créé à la section précédente. Nous devons juste nous assurer qu'il existe deux secrets disponibles dans l'instance Key Vault. Le premier secret est **`StorageName`** et l'autre est **`StorageKey`**. Les commandes pour créer ces secrets dans l'instance Key Vault sont les suivantes :

```
Set-AzKeyVaultSecret -VaultName "testkeyvaultbook" -Name "storageName"
-SecretValue $(ConvertTo-SecureString -String "uniquename" -AsPlainText
-Force ) -Verbose

Set-AzKeyVaultSecret -VaultName "testkeyvaultbook" -Name "storageKey"
-SecretValue $(ConvertTo-SecureString -String "?sv=2020-03-28&ss=bfqt&srt=sc
o&sp=rwdlacup&se=2020-03-30T21:51:03Z&st=2020-03-30T14:51:03Z&spr=https&sig=
gTynGhj20er6pDl7Ab%2Bpc29WO3%2BJhvi%2BfF%2F6rHYWp4g%3D" -AsPlainText -Force
) -Verbose
```

Il vous est conseillé de modifier les noms de l'instance Key Vault et la valeur de clé secrète en fonction de votre compte de stockage.

Après avoir fait en sorte que l'instance Key Vault possède les secrets nécessaires, le code de fichier de modèle ARM peut être utilisé pour déployer les modèles imbriqués sur les abonnements et les groupes de ressources.

Le code de modèle ARM est disponible dans le fichier **CrossSubscriptionLinkedStorageAccount.json** et est également affiché ici. Nous vous recommandons de modifier la valeur de la variable **templateUrl** dans ce fichier. Elle doit être mise à jour avec un emplacement de fichier de stockage Azure Blob valide :

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "hostingPlanNames": {
      "type": "array",
      "minLength": 1
  ...
      "type": "Microsoft.Resources/deployments",
      "name": "fsdfsdf",
      "apiVersion": "2019-10-01",
      "subscriptionId": "[parameters('subscriptions')[copyIndex()]]",
      "resourceGroup": "[parameters('resourceGroups')[copyIndex()]]",
      "copy": {
        "count": "[length(parameters('hostingPlanNames'))]",
        "name": "mywebsites",
        "mode": "Parallel"
    ...
  ]
}
```

Le code du fichier **paramètres** s'affiche ensuite. Nous vous recommandons de modifier les valeurs des paramètres, y compris **resourceid** de l'instance Key Vault et le nom du secret. Les noms des services d'applications doivent être uniques, sinon le modèle ne sera pas déployé. Le code du fichier **paramètres** est disponible dans le fichier de code **CrossSubscriptionLinkedStorageAccount.parameters.json** :

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "hostingPlanNames": {
      "value": [ "firstappservice", "secondappservice" ]
  ...
    "storageKey": {
      "reference": {
        "keyVault": { "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/keyvaluedemo/providers/Microsoft.KeyVault/
vaults/forsqlvault1" },
        "secretName": "storageKey"
      }
    }
  }
}
```

Voici la commande pour déployer le modèle. Le script de déploiement est disponible dans le fichier **`CrossSubscriptionLinkedStorageAccount.ps1`** :

```
New-AzureRmResourceGroupDeployment  -TemplateFile "c:\users\
rites\source\repos\CrossSubscription\CrossSubscription\
CrossSubscriptionLinkedStorageAccount.json" -ResourceGroupName <<replace
with the base subscription resource group name >> -TemplateParameterFile
"c:\users\rites\source\repos\CrossSubscription\CrossSubscription\
CrossSubscriptionLinkedStorageAccount.parameters.json" -Verbose
```

Une fois la commande exécutée, les ressources définies dans le modèle ARM doivent être reflétées dans le portail Azure.

Maintenant que vous savez comment configurer des ressources sur les groupes de ressources et les abonnements, nous allons examiner certaines des solutions qui peuvent être créées à l'aide de modèles ARM.

Solutions de machine virtuelle à l'aide de modèles ARM

Les ressources et les solutions de l'infrastructure en tant que service (**IaaS**) peuvent être déployées et configurées à l'aide de modèles ARM. Les ressources principales associées à IaaS sont des ressources de machine virtuelle.

La création d'une ressource de machine virtuelle dépend de plusieurs autres ressources dans Azure. Voici quelques-unes des ressources nécessaires à la création d'une machine virtuelle :

- Un compte de stockage ou un disque géré pour héberger le système d'exploitation et le disque de données
- Un réseau virtuel avec des sous-réseaux
- Une carte d'interface réseau

Il existe d'autres ressources facultatives, notamment :

- Azure Load Balancer
- Groupes de sécurité réseau
- Adresse IP publique
- Tables de routage et plus encore

Cette section aborde le processus de création de machines virtuelles à l'aide de modèles ARM. Comme mentionné précédemment dans cette section, nous devons créer quelques ressources, dont la ressource de machine virtuelle dépendra, avant de créer la ressource de machine virtuelle elle-même.

Il est important de noter qu'il n'est pas toujours nécessaire de créer les ressources dépendantes. Elles ne doivent être créées que si elles n'existent pas déjà. Si elles sont déjà disponibles dans l'abonnement Azure, la ressource de machine virtuelle peut être configurée en référençant ces ressources dépendantes.

Le modèle dépend de quelques paramètres qui doivent lui être fournis au moment de l'exécution du modèle. Ces variables se rapportent à l'emplacement des ressources et à certaines de leurs valeurs de configuration. Ces valeurs sont extraites des paramètres car elles peuvent changer d'un déploiement à un autre. C'est pourquoi l'utilisation de paramètres permet d'avoir un modèle générique.

La première étape consiste à créer un compte de stockage, comme illustré dans le code suivant :

```
{
       "type": "Microsoft.Storage/storageAccounts",
       "name": "[variables('storageAccountName')]",
       "apiVersion": "2019-04-01",
       "location": "[parameters('location')]",
       "sku": {
         "name": "Standard_LRS"
       },
       "kind": "Storage",
       "properties": {}
       },
```

Une fois que vous avez créé un compte de stockage, un réseau virtuel doit être défini dans le modèle ARM. Il est important de noter qu'il n'existe aucune dépendance entre un compte de stockage et un réseau virtuel. Ils peuvent être créés en parallèle. La ressource de réseau virtuel dispose d'un sous-réseau en tant que sous-ressource. Les deux sont configurées avec leurs plages IP ; le sous-réseau a généralement une plage inférieure à la plage d'adresses IP du réseau virtuel :

```
     {
       "apiVersion": "2019-09-01",
       "type": "Microsoft.Network/virtualNetworks",
       "name": "[variables('virtualNetworkName')]",
       "location": "[parameters('location')]",
       "properties": {
         "addressSpace": {
           "addressPrefixes": [
             "[variables('addressPrefix')]"
           ]
         },
         "subnets": [
           {
             "name": "[variables('subnetName')]",
             "properties": {
               "addressPrefix": "[variables('subnetPrefix')]"
             }
```

```
        }
      ]
    }
  },
```

Si la machine virtuelle doit être accessible via l'Internet public, une adresse IP publique peut également être créée, comme illustré dans le code suivant. Comme nous l'avons déjà mentionné, il s'agit d'une ressource entièrement indépendante qui peut être créée en parallèle avec le compte de stockage et le réseau virtuel :

```
  {
    "apiVersion": "2019-11-01",
    "type": "Microsoft.Network/publicIPAddresses",
    "name": "[variables('publicIPAddressName')]",
    "location": "[parameters('location')]",
    "properties": {
      "publicIPAllocationMethod": "Dynamic",
      "dnsSettings": {
        "domainNameLabel": "[parameters('dnsLabelPrefix')]"
      }
    }
  },
```

Une fois que vous avez créé le réseau virtuel, le compte de stockage et l'adresse IP publique, une interface réseau peut être créée. Une interface réseau est dépendante d'un réseau virtuel et d'une ressource de sous-réseau. Elle peut également être associée à une adresse IP publique. Ceci est illustré dans le code suivant :

```
{
    "apiVersion": "2019-11-01",
    "type": "Microsoft.Network/networkInterfaces",
    "name": "[variables('nicName')]",
    "location": "[parameters('location')]",
    "dependsOn": [
```

```
        "[resourceId('Microsoft.Network/publicIPAddresses/',
variables('publicIPAddressName'))]",
        "[resourceId('Microsoft.Network/virtualNetworks/',
variables('virtualNetworkName'))]"
      ],
      "properties": {
        "ipConfigurations": [
          {
            "name": "ipconfig1",
            "properties": {
              "privateIPAllocationMethod": "Dynamic",
              "publicIPAddress": {
"id": "[resourceId('Microsoft.Network/
publicIPAddresses',variables('publicIPAddressName'))]"
              },
"subnet": {
"id": "[variables('subnetRef')]"
              }
            }
          }
        ]
      }
    },
```

Il est important de noter que l'adresse IP publique et le sous-réseau sont référencés par leurs identifiants Azure uniques.

Maintenant que l'interface réseau a été créée, nous disposons de toutes les ressources nécessaires pour créer une machine virtuelle. Le bloc de code suivant montre comment créer une machine virtuelle à l'aide d'un modèle ARM. Il a une dépendance à l'égard de la carte réseau et du compte de stockage. Cela crée indirectement des dépendances sur le réseau virtuel, le sous-réseau et l'adresse IP publique.

Pour la machine virtuelle, nous configurons la configuration de ressource obligatoire, ce qui inclut le **type**, l'**apiVersion, l'emplacement** et le **nom**, ainsi que toutes les dépendances, comme illustré dans le code suivant :

```
{
"apiVersion": "2019-07-01",
"type": "Microsoft.Compute/virtualMachines",
"name": "[variables('vmName')]",
"location": "[resourceGroup().location]",
"tags": {
"displayName": "VirtualMachine"
      },
"dependsOn": [
"[concat('Microsoft.Storage/storageAccounts/',
variables('storageAccountName'))]",
"[concat('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
      ],
"properties": {
"hardwareProfile": { "vmSize": "[variables('vmSize')]" },
"availabilitySet": {
"id": "[resourceId('Microsoft.Compute/availabilitySets',
parameters('adAvailabilitySetName'))]"
        },
"osProfile": {
"computerName": "[variables('vmName')]",
"adminUsername": "[parameters('adminUsername')]",
"adminPassword": "[parameters('adminPassword')]"
        },
"storageProfile": {
"imageReference": {
```

```
"publisher": "[variables('imagePublisher')]",
"offer": "[variables('imageOffer')]",
"sku": "[parameters('windowsOSVersion')]",
"version": "latest"
          },
"osDisk": { "createOption": "FromImage" },
"copy": [
            {
"name": "dataDisks",
"count": 3,
"input": {
"lun": "[copyIndex('dataDisks')]",
"createOption": "Empty",
"diskSizeGB": "1023",
"name": "[concat(variables('vmName'), '-datadisk', copyIndex('dataDisks'))]"
              }
            }
          ]
        },
"networkProfile": {
"networkInterfaces": [
            {
"id": "[resourceId('Microsoft.Network/networkInterfaces',
variables('nicName'))]"
            }
          ]
        }
      }
    }
```

Dans le code précédent, la machine virtuelle est configurée avec :

- Un profil matériel : la taille de la machine virtuelle.
- Un profil de système d'exploitation : le nom et les informations d'identification pour se connecter à la machine virtuelle.
- Un profil de stockage : le compte de stockage sur lequel stocker le fichier de **disque dur virtuel** (**VHD**) pour la machine virtuelle, y compris les disques de données.
- Un profil réseau : la référence à la carte d'interface réseau.

La section suivante présente un exemple d'utilisation de modèles ARM pour configurer une plateforme en tant que solution de service.

Solutions PaaS utilisant les modèles ARM

Les ressources et les solutions de la plateforme en tant que service (**PaaS**) peuvent être déployées à l'aide de modèles ARM. L'une des principales ressources relatives à PaaS est Azure Web Apps. Dans cette section, nous nous concentrerons sur la création d'applications web sur Azure à l'aide de modèles ARM.

Le modèle s'attend à ce que quelques paramètres soient fournis lors de son exécution. Les paramètres nécessaires sont la référence pour le plan App service, la région Azure qui héberge les ressources et la capacité de référence du plan App Service.

Plusieurs variables sont déclarées dans le modèle afin de le rendre générique et facilement gérable. La première, **`hostingPlanName`**, est destinée au nom du plan App Service, tandis que la deuxième, **`webSiteName`**, est pour le service d'application lui-même.

Il existe au minimum deux ressources qui doivent être déclarées et configurées pour une application web opérationnelle dans Azure. Les voici :

- Le plan Azure App Service
- Azure App Service

La première étape de création d'une application web sur Azure consiste à définir la configuration d'un plan Azure App Service. Le code suivant définit un nouveau plan App Service. Il est important de noter que le type de ressource est **`Microsoft.Web/serverfarms`**. La plupart des valeurs de configuration du plan, telles que l'**`emplacement`**, le **`nom`** et la **`capacité`**, sont des paramètres du modèle ARM :

```
  {
"apiVersion": "2019-08-01",
"name": "[variables('hostingPlanName')]",
"type": "Microsoft.Web/serverfarms",
"location": "[parameters('location')]",
"tags": {
"displayName": "HostingPlan"
      },
"sku": {
"name": "[parameters('skuName')]",
"capacity": "[parameters('skuCapacity')]"
      },
"properties": {
"name": "[variables('hostingPlanName')]"
      }
    },
```

La ressource suivante qui doit être provisionnée après un plan est le service d'application lui-même. Il est important de créer une dépendance entre ces deux ressources afin qu'un plan soit déjà créé avant que le service d'application lui-même soit créé :

```
  {
"apiVersion": "2019-08-01",
"name": "[variables('webSiteName')]",
"type": "Microsoft.Web/sites",
"location": "[parameters('location')]",
"dependsOn": [
"[variables('hostingPlanName')]"
      ],
"properties": {
"name": "[variables('webSiteName')]",
"serverFarmId": "[resourceId('Microsoft.Web/serverfarms',
variables('hostingPlanName'))]"
      },
"resources": [
```

```
        {
"apiVersion": "2019-08-01",
"type": "config",
"name": "connectionstrings",
"dependsOn": [
"[variables('webSiteName')]"
          ],
"properties": {
"DefaultConnection": {
"value": "[concat( 'sql connection string here')]",
"type": "SQLAzure"
            }
          }
        }
      ]
    }
```

Dans le code précédent, une ressource de type **`Microsoft.Web/sites`** est définie et elle a une dépendance envers le plan. Elle utilise également le plan App Service et est associée à celui-ci à l'aide de **`serverFarmId`**. Elle déclare également une chaîne de connexion qui peut être utilisée pour se connecter à SQL Server.

Cette section a montré un exemple de création d'une solution PaaS sur Azure à l'aide d'un modèle ARM. De même, d'autres solutions PaaS, notamment les applications Azure Function, Kubernetes Service et Service Fabric, peuvent être créées à l'aide de modèles ARM.

Solutions associées aux données à l'aide de modèles ARM

Il existe de nombreuses ressources dans Azure relatives à la gestion et au stockage des données. Parmi les ressources importantes relatives aux données, citons Azure SQL, Azure Cosmos DB, Azure Data Lake Storage, Data Lake Analytics, Azure Synapsis, Databricks et Data Factory.

Toutes ces ressources peuvent être provisionnées et configurées à l'aide d'un modèle ARM. Dans cette section, nous allons créer un modèle ARM pour provisionner une ressource Data Factory responsable de la migration des données du stockage Azure Blob vers Azure SQL Database à l'aide de procédures stockées.

Vous trouverez le fichier de paramètres ainsi que le modèle. Ces valeurs peuvent varier d'un déploiement à l'autre. Nous conserverons le modèle générique afin que vous puissiez le personnaliser et l'utiliser facilement avec d'autres déploiements également.

La totalité du code de cette section est disponible sur https://github.com/Azure/azure-quickstart-templates/blob/master/101-data-factory-blob-to-sql-copy-stored-proc.

La première étape consiste à déclarer la configuration de la fabrique de données dans le modèle ARM, comme illustré dans le code suivant :

```
"name": "[variables('dataFactoryName')]",
"apiVersion": "2018-06-01",
"type": "Microsoft.DataFactory/datafactories",
"location": "[parameters('location')]",
```

Chaque fabrique de données dispose de plusieurs services liés. Ces services liés servent de connecteurs pour obtenir des données dans la fabrique de données, ou la fabrique de données peut y envoyer des données. La liste de codes suivante crée un service lié pour le compte de stockage Azure à partir duquel les objets blob seront lus dans la fabrique de données, et un autre service lié pour Azure SQL Database :

```
  {
"type": "linkedservices",
"name": "[variables('storageLinkedServiceName')]",
"apiVersion": "2018-06-01",
"dependsOn": [
"[variables('dataFactoryName')]"
          ],
"properties": {
"type": "AzureStorage",
"description": "Azure Storage Linked Service",
"typeProperties": {
"connectionString":
"[concat('DefaultEndpointsProtocol=https;
AccountName=',parameters('storageAccountName'),';
AccountKey=',parameters('storageAccountKey'))]"
            }
          }
        },
        {
"type": "linkedservices",
```

```
"name": "[variables('sqlLinkedServiceName')]",
"apiVersion": "2018-06-01",
"dependsOn": [
"[variables('dataFactoryName')]"
          ],
"properties": {
"type": "AzureSqlDatabase",
"description": "Azure SQL linked service",
"typeProperties": {
"connectionString": "[concat('Data Source=tcp:', parameters('sqlServerName'),
'.database.windows.net,1433;Initial Catalog=', parameters('sqlDatabaseName'),
';Integrated Security=False;User ID=', parameters('sqlUserId'), ';Password=',
parameters('sqlPassword'), ';Connect Timeout=30;Encrypt=True')]"
            }
          }
        },
```

Après les services liés, il est temps de définir les jeux de données pour Azure Data Factory. Les jeux de données permettent d'identifier les données qui doivent être lues et placées dans la fabrique de données. Ils peuvent également représenter les données temporaires qui doivent être stockées par la fabrique de données pendant la transformation, ou même l'emplacement de destination où les données seront écrites. Le bloc de code suivant crée trois jeux de données, un pour chacun des aspects des jeux de données qui ont été mentionnés.

Le jeu de données de lecture est illustré dans le bloc de code suivant :

```
  {
"type": "datasets",
"name": "[variables('storageDataset')]",
"dependsOn": [
"[variables('dataFactoryName')]",
"[variables('storageLinkedServiceName')]"
          ],
"apiVersion": "2018-06-01",
"properties": {
"type": "AzureBlob",
"linkedServiceName": "[variables('storageLinkedServiceName')]",
```

```
"typeProperties": {
"folderPath": "[concat(parameters('sourceBlobContainer'), '/')]",
"fileName": "[parameters('sourceBlobName')]",
"format": {
"type": "TextFormat"
            }
          },
"availability": {
"frequency": "Hour",
"interval": 1
          },
"external": true
        }
      },
```

Le jeu de données intermédiaire est illustré dans les lignes de code suivantes :

```
 {
"type": "datasets",
"name": "[variables('intermediateDataset')]",
"dependsOn": [
"[variables('dataFactoryName')]",
"[variables('sqlLinkedServiceName')]"
        ],
"apiVersion": "2018-06-01",
"properties": {
"type": "AzureSqlTable",
"linkedServiceName": "[variables('sqlLinkedServiceName')]",
"typeProperties": {
"tableName": "[variables('intermediateDataset')]"
          },
"availability": {
"frequency": "Hour",
"interval": 1
```

```
            }
          }
        },
```

En dernier lieu, le jeu de données utilisé pour la destination est illustré ici :

```
 {
"type": "datasets",
"name": "[variables('sqlDataset')]",
"dependsOn": [
"[variables('dataFactoryName')]",
"[variables('sqlLinkedServiceName')]"
          ],
"apiVersion": "2018-06-01",
"properties": {
"type": "AzureSqlTable",
"linkedServiceName": "[variables('sqlLinkedServiceName')]",
"typeProperties": {
"tableName": "[parameters('sqlTargetTable')]"
            },
"availability": {
"frequency": "Hour",
"interval": 1
            }
          }
        },
```

En dernier lieu, nous avons besoin d'un pipeline dans Data Factory qui peut réunir tous les jeux de données et les services liés, et faciliter la création des solutions de données d'extraction-transformation-charge. Un pipeline se compose de plusieurs activités, chacune répondant à une tâche particulière. Toutes ces activités peuvent être définies dans le modèle ARM, comme nous allons le voir maintenant. La première activité copie les objets blobs du compte de stockage vers un serveur SQL intermédiaire, comme illustré dans le code suivant :

```
{
"type": "dataPipelines",
"name": "[variables('pipelineName')]",
"dependsOn": [
```

```
"[variables('dataFactoryName')]",
"[variables('storageLinkedServiceName')]",
"[variables('sqlLinkedServiceName')]",
"[variables('storageDataset')]",
"[variables('sqlDataset')]"
          ],
"apiVersion": "2018-06-01",
"properties": {
"description": "Copies data from Azure Blob to Sql DB while invoking stored
procedure",
"activities": [
              {
"name": "BlobtoSqlTableCopyActivity",
"type": "Copy",
"typeProperties": {
"source": {
"type": "BlobSource"
                  },
"sink": {
"type": "SqlSink",
"writeBatchSize": 0,
"writeBatchTimeout": "00:00:00"
                  }
                },
"inputs": [
                  {
"name": "[variables('storageDataset')]"
                  }
                ],
"outputs": [
                  {
"name": "[variables('intermediateDataset')]"
```

```
                }
              ]
            },
            {
"name": "SqlTabletoSqlDbSprocActivity",
"type": "SqlServerStoredProcedure",
"inputs": [
                {
"name": "[variables('intermediateDataset')]"
                }
              ],
"outputs": [
                {
"name": "[variables('sqlDataset')]"
                }
              ],
"typeProperties": {
"storedProcedureName": "[parameters('sqlWriterStoredProcedureName')]"
              },
"scheduler": {
"frequency": "Hour",
"interval": 1
              },
"policy": {
"timeout": "02:00:00",
"concurrency": 1,
"executionPriorityOrder": "NewestFirst",
"retry": 3
              }
            }
          ],
```

```
"start": "2020-10-01T00:00:00Z",
"end": "2020-10-02T00:00:00Z"
          }
        }
      ]
    }
```

La dernière activité copie les données du jeu de données intermédiaire vers le jeu de données de destination final.

Il y a également des temps de début et de fin au cours desquels le pipeline doit être exécuté.

Cette section se concentre sur la création d'un modèle ARM pour une solution relative aux données. Dans la section suivante, nous allons traiter des modèles ARM pour la création de datacenters sur Azure avec Active Directory et DNS.

Création d'une solution IaaS sur Azure avec Active Directory et DNS

La création d'une solution IaaS sur Azure implique la création de plusieurs machines virtuelles, la promotion d'une machine virtuelle comme contrôleur de domaine, et la mise en place d'autres machines virtuelles dans le contrôleur de domaine en tant que nœuds joints au domaine. Cela implique également l'installation d'un serveur DNS pour la résolution de noms et, éventuellement, d'un serveur intermédiaire pour accéder à ces machines virtuelles en toute sécurité.

Le modèle crée une forêt Active Directory sur les machines virtuelles. Il crée plusieurs machines virtuelles en fonction des paramètres fournis.

Le modèle crée :

- Quelques jeux de disponibilité
- Un réseau virtuel
- Des groupes de sécurité réseau pour définir les ports et les adresses IP autorisés et refusés

Le modèle effectue ensuite les opérations suivantes :

- Il met en service un ou deux domaines. Le domaine racine est créé par défaut, tandis que le domaine enfant est facultatif.
- Il met en service deux contrôleurs de domaine par domaine.
- Il exécute les scripts de configuration d'état souhaité pour promouvoir une machine virtuelle en tant que contrôleur de domaine.

Nous pouvons créer plusieurs machines virtuelles à l'aide de l'approche abordée dans la section *Solutions de machines virtuelles utilisant des modèles* ARM. Toutefois, ces machines virtuelles doivent faire partie d'un ensemble de disponibilité s'ils doivent être hautement disponibles. Il convient de noter que les groupes de disponibilité offrent une disponibilité de 99,95 % pour les applications déployées sur ces machines virtuelles, tandis que les zones de disponibilité offrent une disponibilité de 99,99 %.

Un ensemble de disponibilité peut être configuré comme illustré dans le code suivant :

```
   {
 "name": "[variables('adAvailabilitySetNameRoot')]",
 "type": "Microsoft.Compute/availabilitySets",
 "apiVersion": "2019-07-01",
 "location": "[parameters('location')]",
 "sku": {
 "name": "Aligned"
      },
 "properties": {
 "PlatformUpdateDomainCount": 3,
 "PlatformFaultDomainCount": 2
      }
    },
```

Une fois que l'ensemble de disponibilité est créé, un profil supplémentaire doit être ajouté à la configuration de la machine virtuelle pour associer la machine virtuelle à l'ensemble de disponibilité, comme illustré dans le code suivant :

```
 "availabilitySet" : {
 "id": "[resourceId('Microsoft.Compute/availabilitySets',
 parameters('adAvailabilitySetName'))]"
      }
```

Notez que les ensembles de disponibilité sont obligatoires afin d'utiliser des équilibreurs de charge avec des machines virtuelles.

Une autre modification nécessaire à la configuration du réseau virtuel consiste à ajouter des informations DNS, comme illustré dans le code suivant :

```
  {
"name": "[parameters('virtualNetworkName')]",
"type": "Microsoft.Network/virtualNetworks",
"location": "[parameters('location')]",
"apiVersion": "2019-09-01",
"properties": {
"addressSpace": {
"addressPrefixes": [
"[parameters('virtualNetworkAddressRange')]"
          ]
        },
"dhcpOptions": {
"dnsServers": "[parameters('DNSServerAddress')]"
        },
"subnets": [
          {
"name": "[parameters('subnetName')]",
"properties": {
"addressPrefix": "[parameters('subnetRange')]"
            }
          }
        ]
      }
    },
```

Enfin, pour convertir une machine virtuelle en Active Directory, un script PowerShell ou un script de configuration d'état souhaité (**DSC**) doit être exécuté sur la machine virtuelle. Même pour joindre d'autres machines virtuelles au domaine, un autre ensemble de scripts doit être exécuté sur ces machines virtuelles.

Les scripts peuvent être exécutés sur la machine virtuelle à l'aide de la ressource **CustomScriptExtension**, comme illustré dans le code suivant :

```
 {
"type": "Microsoft.Compute/virtualMachines/extensions",
"name": "[concat(parameters('adNextDCVMName'),'/PrepareNextDC')]",
"apiVersion": "2018-06-01",
"location": "[parameters('location')]",
"properties": {
"publisher": "Microsoft.Powershell",
"type": "DSC",
"typeHandlerVersion": "2.21",
"autoUpgradeMinorVersion": true,
"settings": {
"modulesURL": "[parameters('adNextDCConfigurationModulesURL')]",
"configurationFunction": "[parameters('adNextDCConfigurationFunction')]",
"properties": {
"domainName": "[parameters('domainName')]",
"DNSServer": "[parameters('DNSServer')]",
"DNSForwarder": "[parameters('DNSServer')]",
"adminCreds": {
"userName": "[parameters('adminUserName')]",
"password": "privateSettingsRef:adminPassword"
            }
          }
        },
"protectedSettings": {
"items": {
"adminPassword": "[parameters('adminPassword')]"
          }
        }
      }
    },
```

Dans cette section, nous avons créé un datacenter sur Azure à l'aide du paradigme IaaS. Nous avons créé plusieurs machines virtuelles et nous avons converti l'une d'entre elles en contrôleur de domaine, en installant DNS et en lui attribuant un domaine. Maintenant, d'autres machines virtuelles sur le réseau peuvent être jointes à ce domaine et elles peuvent former un datacenter complet sur Azure.

Veuillez consulter https://github.com/Azure/azure-quickstart-templates/tree/master/301-create-ad-forest-with-subdomain si vous souhaitez obtenir une liste complète des codes pour la création d'un datacenter sur Azure.

Résumé

L'option de déploiement des ressources à l'aide d'un déploiement unique vers plusieurs abonnements, groupes de ressources et régions offre des capacités améliorées pour déployer, réduire les bogues lors du déploiement et accéder à des avantages avancés, tels que la création de sites de récupération d'urgence et la mise en place de la haute disponibilité.

Au cours de ce chapitre, vous avez vu comment créer différents types de solutions à l'aide de modèles ARM. Nous avons notamment appris à créer une solution basée sur l'infrastructure comprenant des machines virtuelles ; une solution basée sur une plateforme à l'aide d'Azure App Service ; une solution relative aux données à l'aide de la ressource Data Factory (y compris sa configuration) ; et un datacenter sur Azure avec des machines virtuelles, Active Directory et DNS installés au-dessus de la machine virtuelle.

Dans le chapitre suivant, nous nous concentrerons sur la création de modèles ARM modulaires, une compétence essentielle pour les architectes qui veulent vraiment perfectionner leurs modèles ARM. Ce chapitre présentera également différentes façons de concevoir des modèles ARM et de créer des modèles ARM réutilisables et modulaires.

16

Conception et application de modèles ARM modulaires

Nous savons qu'il existe plusieurs façons de créer un modèle **Azure Resource Manager (ARM)**. Il est assez facile de créer un modèle pour mettre en service toutes les ressources nécessaires dans Azure à l'aide de Visual Studio et Visual Studio Code. Un modèle ARM unique peut consister en toutes les ressources requises pour une solution sur Azure. Ce modèle ARM unique peut contenir à peine quelques ressources ou encore des ressources en très grand quantité.

Bien qu'il soit tentant de créer un modèle unique composé de l'ensemble des ressources, il est conseillé de planifier une implémentation de modèle ARM divisée en plusieurs modèles ARM plus petits à l'avance, de sorte que les problèmes futurs liés à ces derniers puissent être évités.

Dans ce chapitre, nous verrons comment créer des modèles ARM de façon modulaire pour leur permettre d'évoluer au fil du temps avec une intervention minimale, c'est-à-dire avec peu de modifications et peu d'efforts à mettre dans les tests et le déploiement.

Cependant, avant d'écrire des modèles modulaires, il est préférable de comprendre les problèmes résolus en les écrivant de façon modulaire.

Dans ce chapitre, nous allons aborder les thèmes suivants :

- Problèmes avec un seul modèle
- Présentation du déploiement imbriqué et lié
- Modèles liés
- Modèles imbriqués
- Configurations à flux libre
- Configurations connues

Maintenant, nous allons explorer les sujets mentionnés ci-dessus en détail, ce qui vous aidera à écrire des modèles modulaires à l'aide des meilleures pratiques de l'industrie.

Problèmes associés à l'approche de modèle unique

À première vue, il peut sembler peu probable qu'un seul grand modèle comprenant toutes les ressources soit plus tard problématique. Or c'est pourtant ce qui risquerait de se produire. Nous allons examiner les problèmes qui pourraient survenir avec de grands modèles simples.

Flexibilité réduite au fil des changements de modèle

Utiliser un seul grand modèle avec toutes les ressources, c'est le rendre plus difficile à modifier plus tard. Compte tenu du fait que toutes les dépendances, paramètres et variables se trouvent dans un modèle unique, toute modification du modèle peut prendre un temps considérable par rapport aux modèles plus petits. Les modifications peuvent avoir un impact sur d'autres sections du modèle qui pourraient passer inaperçues tout comme elles peuvent introduire des bogues.

Dépannage de grands modèles

Les grands modèles sont difficiles à dépanner. C'est bien connu. Plus le nombre de ressources dans un modèle est important, plus il est difficile de dépanner le modèle. Un modèle déploie toutes les ressources qu'il contient. Remonter jusqu'à un bogue implique le déploiement assez fréquent du modèle. La productivité des développeurs est réduite pendant qu'ils attendent l'achèvement du déploiement de modèle.

De plus, le déploiement d'un seul modèle prend plus de temps que celui de modèles plus petits. Les développeurs doivent attendre que les ressources contenant des erreurs soient déployées avant d'entreprendre toute action.

Surdépendance

Les dépendances entre les ressources tendent aussi à se complexifier dans les plus grands modèles. Il est assez facile d'abuser de l'utilisation de la fonction **dependsOn** dans les modèles ARM en raison de leur fonctionnement. Chaque ressource d'un modèle peut faire référence à toutes ses ressources antérieures plutôt qu'à la création d'un arbre de dépendances. Les modèles ARM ne posent pas problème si une ressource unique dépend de toutes les autres ressources du modèle ARM, même si ces autres ressources peuvent avoir des interdépendances au sein d'elles-mêmes. Cela rend les modèles ARM changeants sujets à des bogues et, parfois, il n'est même pas possible de les modifier.

Agilité réduite

En général, il y a dans un projet plusieurs équipes, chacune possédant ses propres ressources dans Azure. Ces équipes auront du mal à travailler avec un seul modèle ARM parce qu'il n'y a qu'un seul développeur à pouvoir le mettre à jour. Un même modèle mis à jour par plusieurs équipes peut engendrer des versions conflictuelles ou difficiles à bien fusionner. Le fait d'opter pour plusieurs modèles plus petits peut donner à chaque équipe la possibilité de créer sa propre partie de modèle ARM.

Réutilisation impossible

Si vous n'avez qu'un modèle unique, c'est alors tout ce que vous avez : utiliser ce modèle veut dire déployer toutes les ressources. La solution ne dispose d'aucune option préconçue pour sélectionner des ressources individuelles sans effectuer quelques manœuvres, telles que l'ajout de ressources conditionnelles. Un seul grand modèle n'est pas réutilisable parce qu'il faut soit prendre toutes les ressources soit aucune.

Constatant les nombreux problèmes que posent les grands modèles uniques, vous avez intérêt à créer des modèles modulaires pour profiter des avantages suivants :

- Plusieurs équipes peuvent travailler sur leurs modèles isolément.
- Les modèles peuvent être réutilisés dans d'autres projets et solutions.
- Les modèles sont faciles à déboguer et à dépanner.

Maintenant que nous avons abordé certains des problèmes avec de grands modèles simples, dans la section suivante, nous allons examiner le nœud des modèles modulaires et comment ils peuvent aider les développeurs à mettre en œuvre des déploiements efficaces.

Explication du principe de la responsabilité unique

Le **principe de responsabilité unique** est l'un des principes fondamentaux des principes de conception SOLID. Selon ce principe, une classe ou un segment de code doit être responsable d'une seule fonction et doit posséder cette fonctionnalité entièrement. Le code doit changer ou évoluer uniquement s'il existe un changement fonctionnel ou un bogue dans la fonction actuelle, et non pas autrement. Ce code ne doit pas changer à cause d'une modification dans un autre composant ou d'un code qui ne fait pas partie du composant actuel.

Appliqué aux modèles ARM, ce principe permet de créer des modèles qui ont pour unique responsabilité de déployer une seule ressource ou fonctionnalité au lieu de déployer toutes les ressources et une solution complète.

Ce principe vous aidera à créer plusieurs modèles, chacun étant responsable d'une ressource unique ou d'un petit groupe de ressources plutôt que de toutes les ressources.

Dépannage et débogage plus rapides

Chaque déploiement de modèle est en fait une activité distincte dans Azure et une instance distincte composée d'entrées, de sorties et de journaux. Quand plusieurs modèles sont déployés en vue de mettre en service une solution, chaque déploiement de modèle possède ses entrées de journal distinctes ainsi que ses descriptions d'entrée et de sortie. Il est beaucoup plus facile d'isoler les bogues et de résoudre les problèmes en utilisant les journaux indépendants de plusieurs déploiements contrairement à ceux d'un seul grand modèle.

Modèles modulaires

Quand un seul grand modèle est subdivisé en plusieurs modèles dans lesquels chaque modèle plus petit prend en charge les ressources qu'il contient (ressources détenues et maintenues exclusivement) et la responsabilité du modèle qui le contient, on peut dire qu'il s'agit de modèles modulaires. Chaque modèle au sein de ces modèles suit le principe de responsabilité unique.

Avant d'apprendre comment diviser un grand modèle en plusieurs modèles réutilisables plus petits, il est important de comprendre la technologie derrière la création de ces modèles plus petits et le moyen de les créer pour déployer des solutions complètes.

Ressources de déploiement

ARM fournit une fonctionnalité permettant de lier des modèles. Même si nous avons déjà couvert en détail les modèles liés, revenons-y ici pour vous aider à comprendre l'utilité de lier les modèles pour atteindre la modularité, la composition et la réutilisation visées.

Les modèles ARM fournissent ce qu'on appelle des **déploiements**. Il s'agit de ressources spécialisées disponibles à partir de l'espace de noms **`Microsoft.Resources`**. Voici ce à quoi ressemble assez fidèlement une ressource de déploiements dans un modèle ARM :

```
"resources": [
  {
      "apiVersion": "2019-10-01",
      "name": "linkedTemplate",
      "type": "Microsoft.Resources/deployments",
      "properties": {
          "mode": "Incremental",
          <nested-template-or-external-template>
      }
  }
]
```

Ce modèle est explicite, et les deux configurations les plus importantes dans la ressource de modèle sont le type et les propriétés. Le type ici renvoie à la ressource de déploiement plutôt qu'à n'importe quelle ressource Azure en particulier (stockage, machine virtuelle, etc.) et les propriétés spécifient la configuration de déploiement, y compris un déploiement de modèle lié ou un déploiement de modèle imbriqué.

Au fond, à quoi sert la ressource de déploiements? Une ressource de déploiements a pour fonction de déployer un autre modèle. Un autre modèle peut être un modèle externe dans un fichier de modèle ARM distinct, ou il peut s'agir d'un modèle imbriqué. Cela signifie qu'il est possible d'invoquer d'autres modèles à partir d'un modèle, tout comme un appel de fonction.

Il peut y avoir des niveaux imbriqués de déploiements dans les modèles ARM. Cela signifie en fait qu'un modèle unique peut en appeler un autre, et que le modèle appelé peut en appeler lui aussi un autre, et ainsi de suite jusqu'à cinq niveaux d'appels imbriqués.

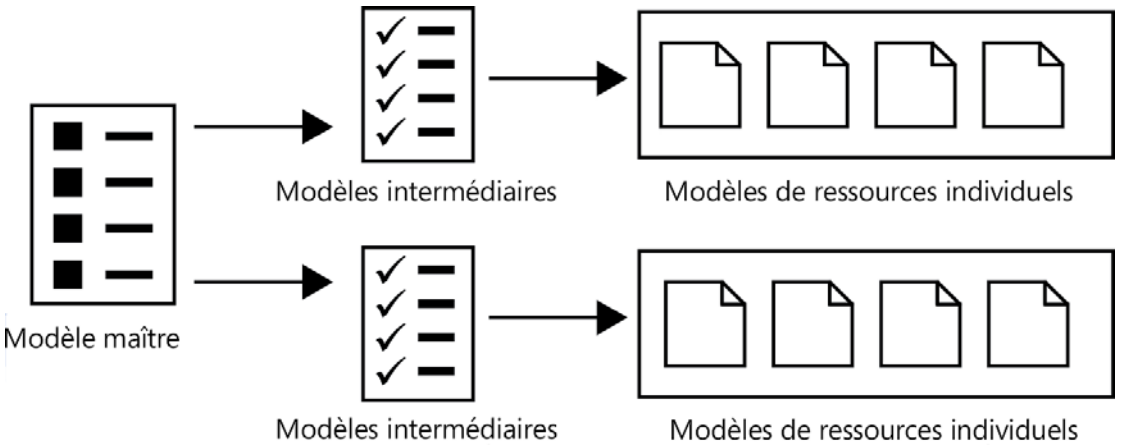


Figure 16.1 : Décomposition du modèle en modèles plus petits

Maintenant que nous comprenons que les grands modèles peuvent être modulaires avec des ressources distinctes dans des modèles distincts, nous devons les relier et les réunir pour déployer des ressources sur Azure. Les modèles liés et imbriqués permettent de composer plusieurs modèles ensemble.

Modèles liés

Les modèles liés sont des modèles qui appellent des modèles externes. Les modèles externes sont stockés dans différents fichiers de modèle ARM. Voici un exemple de modèles liés :

```
"resources": [
  {
     "apiVersion": "2019-10-01",
     "name": "linkedTemplate",
     "type": "Microsoft.Resources/deployments",
     "properties": {
       "mode": "Incremental",
       "templateLink": {
         "uri":"https://mystorageaccount.blob.core.windows.net/
AzureTemplates/newStorageAccount.json",
          "contentVersion":"1.0.0.0"
       },
       "parametersLink": {
```

```
          "uri":"https://mystorageaccount.blob.core.windows.net/
AzureTemplates/newStorageAccount.parameters.json",
           "contentVersion":"1.0.0.0"
       }
     }
  }
]
```

Comparativement au modèle précédent, ce modèle contient deux propriétés supplémentaires importantes, à savoir **`templateLink`** et **`parametersLink`**. La propriété **`templateLink`** renvoie à l'URL réelle de l'emplacement du fichier de modèle externe et **`parametersLink`** correspond à l'emplacement URL du fichier de **`parameters`** correspondant. Notez ici que le modèle appelant doit disposer des droits d'accès à l'emplacement du modèle appelé. Par exemple, si les modèles externes sont stockés dans le stockage Blob Azure, qui est protégé par des clés, les clés **Secure Access Signature** (**SAS**) appropriées doivent être disponibles pour que le modèle d'appelant puisse accéder aux modèles liés.

Vous pouvez aussi fournir des paramètres intégrés explicites au lieu d'une valeur pour **`parametersLink`**, comme le montre l'exemple suivant :

```
"resources": [
  {
     "apiVersion": "2019-10-01",
     "name": "linkedTemplate",
     "type": "Microsoft.Resources/deployments",
     "properties": {
       "mode": "Incremental",
       "templateLink": {
         "uri":"https://mystorageaccount.blob.core.windows.net/
AzureTemplates/newStorageAccount.json",
           "contentVersion":"1.0.0.0"
       },
       "parameters": {
          "StorageAccountName":{"value": "
                           [parameters('StorageAccountName')]"}
        }
     }
  }
]
```

Vous avez maintenant une bonne compréhension des modèles liés. Un sujet étroitement lié est les modèles imbriqués, que la section suivante abordera en détail.

Modèles imbriqués

Les modèles imbriqués sont une fonction relativement nouvelle dans les modèles ARM par rapport aux modèles liés externes.

Les modèles imbriqués ne définissent pas les ressources dans les fichiers externes. Les ressources sont définies dans le modèle appelant lui-même et dans la ressource de déploiements, comme le montre l'exemple suivant :

```
"resources": [
  {
    "apiVersion": "2019-10-01",
    "name": "nestedTemplate",
    "type": "Microsoft.Resources/deployments",
    "properties": {
      "mode": "Incremental",
      "template": {
        "$schema": "https://schema.management.azure.com/schemas/2015-
              01-01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "resources": [
          {
            "type": "Microsoft.Storage/storageAccounts",
            "name": "[variables('storageName')]",
            "apiVersion": "2019-04-01",
            "location": "West US",
            "properties": {
              "accountType": "Standard_LRS"
            }
          }
        ]
      }
    }
  }
]
```

Dans ce segment de code, nous pouvons voir que la ressource de compte de stockage est imbriquée dans le modèle d'origine dans le cadre de la ressource de déploiement. Au lieu d'utiliser les attributs **`templateLink`** et **`parametersLink`** un tableau de **`ressources`** est utilisé pour créer plusieurs ressources dans le cadre d'un déploiement unique. Un déploiement imbriqué a pour avantage que les ressources au sein d'un parent peuvent être utilisées pour les reconfigurer à l'aide de leurs noms. Habituellement, une ressource avec un nom ne peut exister qu'une seule fois dans un modèle. Les modèles imbriqués nous permettent de les utiliser dans le même modèle et de voir à l'autosuffisance de tous les modèles plutôt qu'à leur stockage séparé. Ces modèles peuvent être accessibles aux fichiers externes ou non.

Maintenant que nous avons compris la technologie derrière les modèles ARM modulaires, comment diviser un grand modèle en des modèles plus petits?

Il existe plusieurs façons dont un grand modèle peut être décomposé en plus petits modèles. Microsoft recommande le modèle suivant pour la décomposition des modèles ARM :

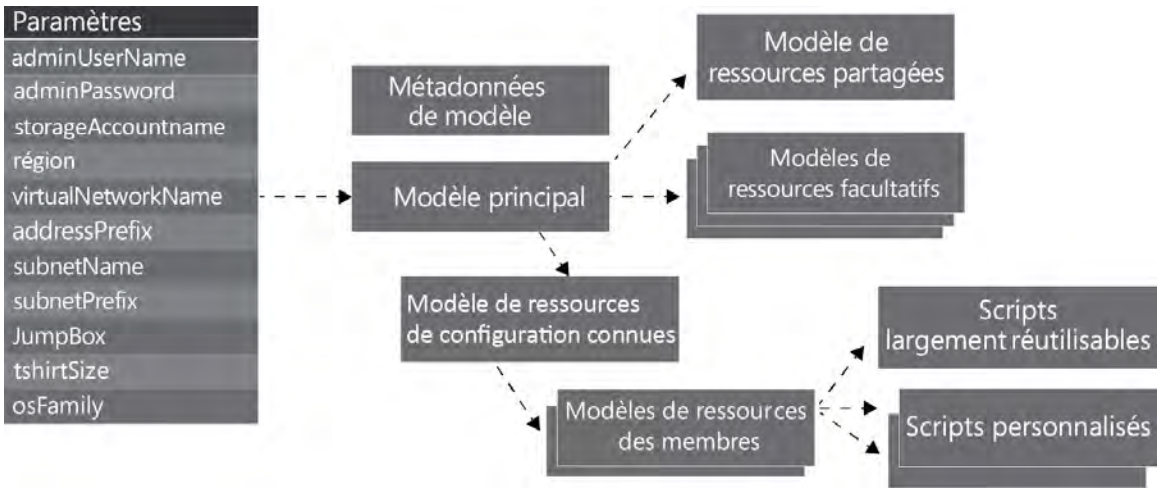


Figure 16.2 : Stratégie de décomposition du modèle

La subdivision d'un grand modèle en modèles plus petits passe toujours par un modèle principal, qui sert à déployer la solution. Ce modèle principal ou modèle maître appelle en interne d'autres modèles liés ou imbriqués, qui appellent à leur tour d'autres modèles. Les modèles contenant des ressources Azure sont ensuite déployés.

Le modèle principal peut invoquer un modèle de ressource de configuration connu, qui, à son tour, appellera des modèles comprenant des ressources Azure. Le modèle de ressource de configuration connu est spécifique à un projet ou une solution et il n'est pas associé à de nombreux facteurs réutilisables. Les modèles de ressources membres sont des modèles réutilisables appelés par le modèle de ressource de configuration connu.

Le modèle maître peut éventuellement appeler des modèles de ressources partagées et d'autres modèles de ressources s'ils existent.

Il est important de comprendre les configurations connues. Les modèles peuvent être créés en tant que configurations connues ou en tant que configurations à flux libre.

Configurations à flux libre

Les modèles ARM peuvent être créés en tant que modèles génériques où la plupart, sinon la totalité, des valeurs attribuées aux variables sont obtenues en tant que paramètres. Cela permet à la personne qui utilise le modèle de transmettre toute valeur qu'elle juge nécessaire pour déployer des ressources dans Azure. Par exemple, la personne qui déploie le modèle peut choisir une machine virtuelle de n'importe quelle taille, n'importe quel nombre de machines virtuelles et toute configuration pour son stockage et ses réseaux. Ceci est connu comme étant une configuration à flux libre, où la plupart de la configuration est autorisée et les modèles proviennent de l'utilisateur au lieu d'être déclarés dans le modèle.

Ce type de configuration présente un certain nombre de problématiques. Certaines configurations ne sont pas prises en charge dans chaque région Azure et ni dans chaque datacenter Azure, et c'est là le plus grand problème. Les modèles échoueront à créer des ressources si ces dernières ne sont pas autorisées à être créées dans des emplacements ou des régions spécifiques. La configuration à flux libre pose également le problème que les utilisateurs peuvent fournir n'importe quelle valeur qu'ils jugent nécessaire et un modèle les honorera, augmentant ainsi l'empreinte de coût et de déploiement même si elles ne sont pas complètement exigées.

Configurations connues

Les configurations connues, quant à elles, sont des configurations prédéterminées spécifiques au déploiement d'un environnement à l'aide de modèles ARM. Ces configurations prédéterminées, on les appelle des **configurations par format**. Un peu à la manière des tailles de t-shirt, les configurations prédéterminées peuvent être de petite, moyenne et grande tailles. De même, les modèles peuvent être préconfigurés pour permettre le déploiement d'un environnement de petite, moyenne ou grande tailles selon les exigences. Cela signifie que les utilisateurs ne peuvent pas déterminer une taille personnalisée aléatoire pour l'environnement, mais ils peuvent choisir parmi diverses options fournies, et les modèles ARM exécutés pendant l'exécution assureront qu'une configuration appropriée de l'environnement est fournie.

Ainsi, la première étape de la création d'un modèle ARM modulaire consiste à décider des configurations connues pour un environnement.

Par exemple, voici la configuration d'un déploiement de datacenter sur Azure :

Taille de T-Shirt	Configuration de modèle ARM
Petit	Quatre machines virtuelles avec 7 Go de mémoire ainsi que quatre cœurs de CPU
Moyenne	Huit machines virtuelles avec 14 Go de mémoire ainsi que huit cœurs de CPU
Large	16 machines virtuelles avec 28 Go de mémoire ainsi que huit cœurs de CPU

Tableau 16.1 : Configuration d'un déploiement de datacenter sur Azure

Maintenant que nous connaissons les configurations, nous pouvons créer des modèles ARM modulaires.

Il existe deux façons d'écrire des modèles ARM modulaires :

- **Modèles composés** : les modèles composés sont liés à d'autres modèles. Les modèles maîtres et intermédiaires sont des exemples de modèles composés.
- **Modèles foliaires** : les modèles foliaires sont des modèles qui contiennent une ressource Azure unique.

Les modèles ARM peuvent être divisés en modèles modulaires en fonction des éléments suivants :

- Technologies
- Fonctionnalité

Voici le moyen idéal de décider de la méthode modulaire à adopter pour créer un modèle ARM :

- Définissez les modèles foliaires (ou modèles à l'échelle des ressources) constitués de ressources uniques. Dans le diagramme plus loin, l'extrême droite regroupe les modèles foliaires. Dans le diagramme, les machines virtuelles, le réseau virtuel, le stockage et les autres éléments dans la même colonne représentent les modèles foliaires.
- Composez des modèles pour chaque environnement à l'aide des modèles foliaires. Ces modèles propres à chaque environnement génèrent un environnement Azure, comme un environnement SQL Server, un environnement App Service ou un environnement de datacenter. Nous allons aborder ce sujet plus en détail. Prenons l'exemple d'un environnement SQL Azure. Pour créer un environnement Azure SQL, plusieurs ressources sont nécessaires. Au strict minimum, un SQL Server logique, une base de données SQL et quelques ressources de pare-feu SQL doivent être provisionnés. Toutes ces ressources sont définies dans des modèles individuels de type foliaire. Ces ressources peuvent être composées ensemble dans un modèle unique qui a la capacité de créer un environnement Azure SQL. Toute personne souhaitant créer un environnement SQL peut utiliser ce modèle composé. La *figure 16.3* montre le **datacenter**, la **messagerie** et **App Service** en tan que modèles propres à un environnement donné.
- Créez des modèles avec une abstraction supérieure qui compose plusieurs modèles spécifiques à l'environnement en solutions. Ces modèles sont composés de modèles spécifiques à l'environnement qui ont été créés à l'étape précédente. Par exemple, pour créer une solution d'inventaire e-commerce qui a besoin d'un environnement App Service et d'un environnement SQL, deux modèles d'environnement, App Service et SQL Server, peuvent être composés ensemble. La *figure 16.3* comprend des modèles **fonctionnels 1** et **fonctionnels 2**, qui sont composés de modèles enfants.
- Enfin, un modèle maître doit être créé, qui doit être composé de plusieurs modèles où chaque modèle est capable de déployer une solution.

Les étapes précédentes pour créer un modèle modulaire conçu peuvent être facilement comprises à l'aide de la *figure 16.3* :

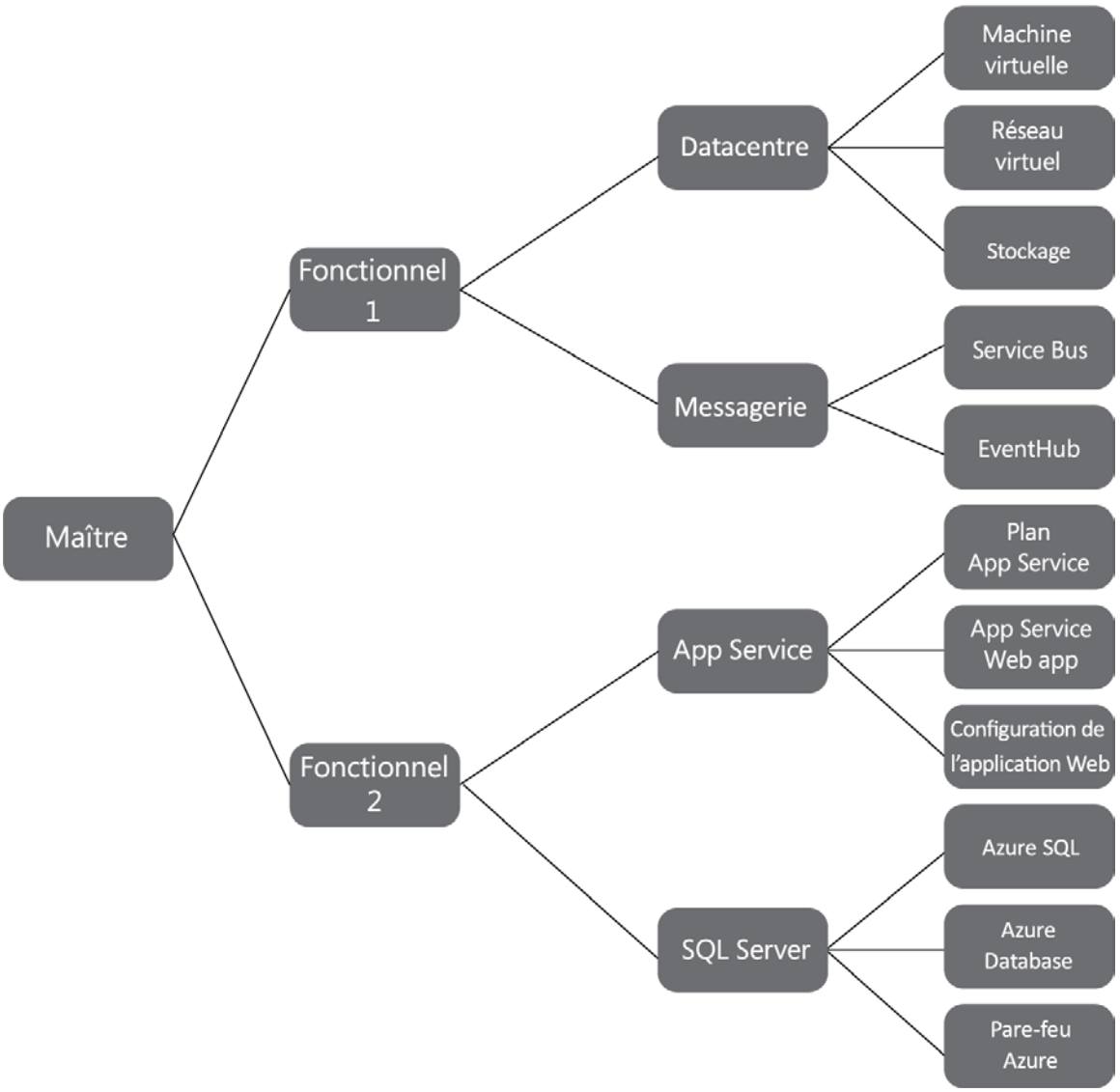


Figure 16.3 : mappage des modèles et des ressources

Maintenant, mettons en œuvre une partie de la fonction illustrée dans le schéma précédent. Dans cette implémentation, nous fournirons une machine virtuelle avec une extension de script en utilisant une approche modulaire. L'extension de script personnalisée déploie des fichiers binaires Docker et prépare un environnement de conteneur sur une machine virtuelle Windows Server 2016.

Maintenant, nous allons créer une solution à l'aide de modèles ARM à l'aide d'une approche modulaire. Comme mentionné précédemment, la première étape consiste à créer des modèles de ressources individuels. Ces modèles de ressources individuels seront utilisés pour composer des modèles supplémentaires capables de créer un environnement. Ces modèles seront nécessaires pour créer une machine virtuelle. Tous les modèles ARM présentés ici sont disponibles dans le code de chapitre ci-joint. Les noms et le code de ces modèles sont les suivants :

- **Storage.json**
- **virtualNetwork.json**
- **PublicIPAddress.json**
- **NIC.json**
- **VirtualMachine.json**
- **CustomScriptExtension.json**

Tout d'abord, examinons le code du modèle, plus particulièrement **Storage.json**. Ce modèle fournit un compte de stockage, dont chaque machine virtuelle a besoin pour stocker ses fichiers de système d'exploitation et ses fichiers de données sur disque :

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string",
      "minLength": 1
    },
    "storageType": {
      "type": "string",
      "minLength": 1
    },
   ...
  "outputs": {
    "resourceDetails": {
      "type": "object",
      "value": "[reference(parameters('storageAccountName'))]"
    }
  }
}
```

Examinons ensuite le code du modèle d'adresse IP publique. Une machine virtuelle qui doit être accessible sur Internet a besoin d'une ressource d'adresse IP publique affectée à sa carte d'interface réseau. Bien que l'exposition d'une machine virtuelle à Internet soit facultative, cette ressource peut être utilisée pour créer une machine virtuelle. Le code suivant se trouve dans le fichier **`publicipaddress.json`** :

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
  "parameters": {
    "publicIPAddressName": {
      "type": "string",
      "minLength": 1
    },
    "publicIPAddressType": {
      "type": "string",
      "minLength": 1
   ...
      }
    }
  ],
  "outputs": {
    "resourceDetails": {
      "type": "object",
      "value": "[reference(parameters('publicIPAddressName'))]"
    }
  }
}
```

Voyons ensuite le code du réseau virtuel. Les machines virtuelles sur Azure ont besoin d'un réseau virtuel pour la communication. Ce modèle sera utilisé pour créer un réseau virtuel sur Azure avec une plage d'adresses prédéfinie et des sous-réseaux. Le code suivant se trouve dans le fichier **`virtualNetwork.json`** :

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
  "parameters": {
    "virtualNetworkName": {
      "type": "string",
      "minLength": 1
 ...
    },
    "subnetPrefix": {
      "type": "string",
      "minLength": 1
    },
    "resourceLocation": {
      "type": "string",
      "minLength": 1
    }
 ...
       "subnets": [
         {
           "name": "[parameters('subnetName')]",
           "properties": {
             "addressPrefix": "[parameters('subnetPrefix')]"
           }
         }
        ]
      }
    }
  ],
  "outputs": {
    "resourceDetails": {
      "type": "object",
      "value": "[reference(parameters('virtualNetworkName'))]"
    }
  }
}
```

Examinons ensuite le code de la carte d'interface réseau. Pour se connecter à un réseau virtuel et accepter et envoyer des demandes vers et depuis Internet, il faut une carte réseau virtuelle à une machine virtuelle. Le code suivant se trouve dans le fichier **`NIC.json`** :

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
  "parameters": {
    "nicName": {
      "type": "string",
      "minLength": 1
    },
    "publicIpReference": {
      "type": "string",
      "minLength": 1
 ...
[resourceId(subscription().subscriptionId,resourceGroup().name, 'Microsoft.
Network/publicIPAddresses', parameters('publicIpReference'))]",
    "vnetRef": "[resourceId(subscription().
subscriptionId,resourceGroup().name, 'Microsoft.Network/virtualNetworks',
parameters('virtualNetworkReference'))]",
    "subnet1Ref": "[concat(variables('vnetRef'),'/subnets/',
parameters('subnetReference'))]"
  },
 ...
                "id": "[variables('subnet1Ref')]"
              }
            }
          }
        ]
      }
    }
  ],
  "outputs": {
    "resourceDetails": {
      "type": "object",
      "value": "[reference(parameters('nicName'))]"
    }
  }
}
```

Voyons ensuite le code servant à créer une machine virtuelle. Chaque machine virtuelle est une ressource dans Azure et notez que ce modèle n'a aucune référence au stockage, au réseau, aux adresses IP publiques ou à d'autres ressources créées précédemment. Cette référence et cette composition se produiront plus loin dans cette section à l'aide d'un autre modèle. Le code suivant se trouve dans le fichier **`VirtualMachine.json`** :

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01
01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vmName": {
      "type": "string",
      "minLength": 1
   ...
    },
    "imageOffer": {
      "type": "string",
      "minLength": 1
    },
    "windowsOSVersion": {
      "type": "string",
      "minLength": 1
    },
   ...
  "outputs": {
    "resourceDetails": {
      "type": "object",
      "value": "[reference(parameters('vmName'))]"
    }
  }
}
```

Examinons ensuite le code de création d'une extension de script personnalisée. Une fois mise en service, cette ressource exécute un script PowerShell sur une machine virtuelle. Cette ressource offre la possibilité d'exécuter des tâches après la mise en service dans des machines virtuelles Azure. Le code suivant se trouve dans le fichier **`CustomScriptExtension.json`** :

```
{
    "$schema": "http://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
      "VMName": {
        "type": "string",
        "defaultValue": "sqldock",
        "metadata": {
...
            "commandToExecute": "[concat('powershell -ExecutionPolicy
Unrestricted -file docker.ps1')]"
          },
          "protectedSettings": {
          }
        }
      }
    ],
    "outputs": {
    }
}
```

Ensuite, nous allons examiner le code PowerShell d'extension de script personnalisé qui prépare l'environnement Docker. Veuillez noter qu'un redémarrage de la machine virtuelle peut se produire lors de l'exécution du script PowerShell, selon que la fonction de conteneurs Windows est déjà installée ou non. Le script suivant installe l'ensemble NuGet, le fournisseur **`DockerMsftProvider`** et l'exécutable Docker. Le fichier **`docker.ps1`** se trouve dans le code accompagnant le chapitre :

```
#
# docker.ps1
#
Install-PackageProvider -Name Nuget -Force -ForceBootstrap -Confirm:$false
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
-Confirm:$false -verboseInstall-Package -Name docker -ProviderName
DockerMsftProvider -Force -ForceBootstrap -Confirm:$false
```

Tous les modèles liés et vus précédemment doivent être chargés vers un conteneur dans un compte de stockage Blob Azure. Ce conteneur peut avoir une stratégie d'accès privé appliquée comme nous l'avons vu dans le chapitre précédent ; toutefois, pour cet exemple, nous allons définir la stratégie d'accès en tant que **`conteneur`**. Cela veut dire qu'il est possible d'accéder aux modèles liés sans jeton SAS.

Enfin, passons à l'écriture du modèle maître. Dans le modèle maître, tous les modèles liés sont composés ensemble pour créer une solution, c'est-à-dire déployer une machine virtuelle et exécuter un script à l'intérieur de celle-ci. La même approche peut être appliquée à la création d'autres solutions comme fournir un datacenter composé de plusieurs machines virtuelles interconnectées. Le code suivant se trouve dans le fichier **`Master.json`** :

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName": {
      "type": "string",
      "minLength": 1
   ...
    },
    "subnetName": {
      "type": "string",
      "minLength": 1
    },
    "subnetPrefix": {
      "type": "string",
      "minLength": 1
  ...
    "windowsOSVersion": {
      "type": "string",
      "minLength": 1
    },
    "vhdStorageName": {
      "type": "string",
      "minLength": 1
    },
    "vhdStorageContainerName": {
      "type": "string",
      "minLength": 1
   ...[concat('https://',parameters('storageAccountName'),'armtfiles.blob.
```

```
core.windows.net/',variables('containerName'),'/Storage.json')]",
          "contentVersion": "1.0.0.0"
        },
        "parameters": {
          "storageAccountName": {
            "value": "[parameters('storageAccountName')]"
          },
          "storageType": {
            "value": "[parameters('storageType')]"
          },
          "resourceLocation": {
            "value": "[resourceGroup().location]"
      ...
  "outputs": {
    "resourceDetails": {
      "type": "object",
      "value": "[reference('GetVM').outputs.resourceDetails.value]"
 }
 }
}
```

Les modèles maîtres appellent les modèles externes et coordonnent également les interdépendances entre eux.

Les modèles externes doivent être disponibles dans un emplacement bien connu pour que le modèle maître puisse y accéder et les appeler. Dans cet exemple, les modèles externes sont stockés dans le conteneur Stockage de grands objets binaires Azure et ces informations ont été transmises au modèle ARM sous la forme de paramètres.

Les modèles externes dans le Stockage de grands objets binaires Azure peuvent être protégés par la configuration des stratégies d'accès. La commande utilisée pour déployer le modèle principal s'affiche ensuite. La commande peut paraître complexe, mais la majorité des valeurs est utilisée comme paramètre. Nous vous conseillons de modifier la valeur de ces paramètres avant de l'exécuter. Les modèles liés ont été téléversés vers un compte de stockage nommé **st02gvwldcxm5suwe** dans le conteneur **armtemplates**. Le groupe de ressources doit être créé s'il n'existe pas déjà. La première commande sert à créer un nouveau groupe de ressources dans la région **Europe de l'Ouest** :

```
New-AzResourceGroup -Name "testvmrg" -Location "West Europe" -Verbose
```

Les autres valeurs de paramètre servent à configurer chaque ressource. Le nom du compte de stockage et la valeur **dnsNameForPublicIP** doivent être uniques dans Azure :

```
New-AzResourceGroupDeployment -Name "testdeploy1" -ResourceGroupName
testvmrg -Mode Incremental -TemplateFile "C:\chapter 05\Master.json"
-storageAccountName "st02gvwldcxm5suwe" -storageType "Standard_LRS"
-publicIPAddressName "uniipaddname" -publicIPAddressType "Dynamic"
-dnsNameForPublicIP "azureforarchitectsbook" -virtualNetworkName
vnetwork01 -addressPrefix "10.0.1.0/16" -subnetName "subnet01" -subnetPrefix
"10.0.1.0/24" -nicName nic02 -vmSize "Standard_DS1" -adminUsername "sysadmin"
-adminPassword $(ConvertTo-SecureString -String sysadmin@123 -AsPlainText
-Force) -vhdStorageName oddnewuniqueacc -vhdStorageContainerName vhds
-OSDiskName mynewvm -vmName vm10 -windowsOSVersion 2012-R2-Datacenter
-imagePublisher MicrosoftWindowsServer -imageOffer WindowsServer
-containerName armtemplates -Verbose
```

Dans cette section, nous avons abordé les bonnes pratiques en matière de décomposition de grands modèles en modèles réutilisables plus petits afin de les combiner au moment de l'exécution pour déployer des solutions complètes sur Azure. Au fur et à mesure que nous progresserons dans le livre, nous allons modifier le modèle ARM étape par étape jusqu'à ce que nous ayons exploré ses parties essentielles. Nous avons utilisé les cmdlets de commande Azure PowerShell pour lancer le déploiement de modèles sur Azure.

Passons maintenant à **copy** et **copyIndex**.

Présentation des fonctions copy et copyIndex

Plusieurs instances d'une ressource particulière ou d'un groupe de ressources sont nécessaires à plusieurs reprises. Par exemple, vous devrez peut-être configurer 10 machines virtuelles du même type. Dans ces cas de figure, il n'est pas prudent de déployer des modèles 10 fois pour créer ces instances. Une meilleure approche consiste à utiliser les fonctions **copy** et **copyIndex** des modèles ARM.

copy est un attribut de chaque définition de ressource. Cela signifie qu'il peut être utilisé pour créer plusieurs instances de n'importe quel type de ressource.

Nous allons voir ce concept en détail en nous appuyant sur un exemple de création de plusieurs comptes de stockage au sein d'un déploiement de modèle ARM unique.

L'extrait de code suivant crée 10 comptes de stockage en série. Ils auraient pu être créés en parallèle à l'aide de **`Parallel`** au lieu de **`Serial`** pour la propriété **`mode`** :

```
"resources": [
      {
          "apiVersion": "2019-06-01",
          "type": "Microsoft.Storage/storageAccounts",
          "location": "[resourceGroup().location]",
          "name": "[concat(variables('storageAccountName'), copyIndex())]",
          "tags":{
              "displayName": "[variables('storageAccountName')]"
          },
          "sku":{
              "name":"Premium_ZRS"
          },
          "kind": "StorageV2",
          "copy":{
              "name": "storageInstances",
              "count": 10,
              "mode": "Serial"
          }
       }
     ],
```

Dans le code précédent, **`copy`** a été utilisé pour provisionner 10 instances du compte de stockage en série, c'est-à-dire l'une après l'autre. Les noms de compte de stockage doivent être uniques pour les 10 instances, et **`copyIndex`** a été utilisée pour les rendre uniques en concaténant le nom de stockage d'origine avec la valeur d'index. La valeur renvoyée par la fonction **`copyIndex`** change dans chaque itération ; elle commencera à 0 et continuera pendant 10 itérations. Cela signifie qu'elle retournera **`9`** pour la dernière itération.

Maintenant que nous avons appris à créer plusieurs instances d'un modèle ARM, nous allons explorer la sécurisation de ces modèles à partir de vulnérabilités connues.

Sécurisation des modèles ARM

Un autre aspect important lié à la création de modèles ARM d'entreprise consiste à les sécuriser de manière appropriée. Les modèles ARM contiennent la configuration des ressources et des informations essentielles sur l'infrastructure, de sorte qu'ils ne doivent pas être compromis ou accessibles à des personnes non autorisées.

La première étape de la sécurisation des modèles ARM consiste à les stocker dans des comptes de stockage et à arrêter tout accès anonyme au conteneur de compte de stockage. En outre, les jetons SAS doivent être générés pour les comptes de stockage et utilisés dans les modèles ARM pour consommer des modèles liés. Cela garantit que seuls les détenteurs de jetons SAS peuvent accéder aux modèles. En outre, ces jetons SAS doivent être stockés dans Azure Key Vault au lieu d'être codés en dur dans des modèles ARM. Cela permet de s'assurer que même les personnes responsables du déploiement n'ont pas accès au jeton SAS.

Une autre mesure dans la sécurisation des modèles ARM garantit que les informations et les secrets sensibles, tels que les chaînes de connexion de base de données, les identificateurs d'abonnement et de locataire Azure, les identificateurs principaux de service, les adresses IP, etc., ne doivent pas être codés en dur dans des modèles ARM. Ils doivent tous être paramétrées, et les valeurs doivent être extraites au moment de l'exécution à partir d'Azure Key Vault. Toutefois, avant d'utiliser cette approche, il est important que ces secrets soient stockés dans le Key Vault avant d'exécuter tous les modèles ARM.

Le code suivant illustre l'une des façons dont les valeurs peuvent être extraites d'Azure Key Vault au moment de l'exécution à l'aide du fichier de paramètres :

```
{
    "$schema": https://schema.management.azure.com/schemas/2016-01-01/
deploymentParameters.json#,
    "contentVersion": "1.0.0.0",
    "parameters": {
        "storageAccountName": {
            "reference": {
                "keyVault": {
                    "id": "/subscriptions/--subscription id --/
resourceGroups/rgname/providers/Microsoft.KeyVault/vaults/keyvaultbook"),
                "secretName": "StorageAccountName"
                }
            }
        }
    }
}
```

Dans cette liste de codes, un paramètre faisant référence à Azure Key Vault est défini pour extraire des valeurs lors de l'exécution pendant le déploiement. L'identificateur Azure Key Vault et le nom secret ont été fournis en tant que valeurs d'entrée.

Maintenant que vous avez appris à sécuriser les modèles ARM, examinons l'identification des différentes dépendances entre elles et la façon dont nous pouvons activer la communication entre plusieurs modèles.

Utilisation des sorties entre les modèles ARM

L'un des aspects importants qui peuvent être facilement négligés lors de l'utilisation de modèles liés réside dans le fait qu'il peut y avoir des dépendances de ressources au sein de modèles liés. Par exemple, une ressource SQL Server peut se trouver dans un modèle lié différent de celui d'une ressource de machine virtuelle. Si nous souhaitons ouvrir le pare-feu SQL Server pour l'adresse IP de la machine virtuelle, nous devons être en mesure de transmettre dynamiquement ces informations à la ressource du pare-feu SQL Server après la mise en service de la machine virtuelle.

Cela peut être fait à l'aide de la méthode simple de référence à la ressource d'adresse IP à l'aide de la fonction **`REFERENCES`** si les ressources SQL Server et machine virtuelle se trouvent dans le même modèle.

Cela devient un peu plus complexe dans le cas de modèles liés si nous voulons partager des valeurs de propriété d'exécution d'une ressource à une autre lorsqu'ils se trouvent dans des modèles différents.

Les modèles ARM fournissent une configuration des **`sorties`**, qui est chargée de générer des sorties à partir du déploiement du modèle actuel et de les renvoyer à l'utilisateur. Par exemple, nous pouvons générer un objet complet, comme illustré dans la liste de codes suivante, à l'aide de la fonction **`référence`**, ou nous pouvons seulement afficher une adresse IP sous la forme d'une valeur de chaîne :

```
"outputs": {
    "storageAccountDetails": {
        "type": "object",
        "value": "[reference(resourceid
            ('Microsoft.Storage/storageAccounts',
            variables('storageAccountName')))]",
    "virtualMachineIPAddress": {
        "type": "string",
        "value": "[reference(variables
            ('publicIPAddressName')).properties.ipAddress]"
        }
    }
}
```

Les paramètres au sein d'un modèle lié peuvent être utilisés par le modèle principal. Lorsqu'un modèle lié est appelé, la sortie est disponible pour le modèle maître qui peut être fourni en tant que paramètre au modèle associé ou imbriqué suivant. Ainsi, il est possible d'envoyer les valeurs de configuration du runtime des ressources d'un modèle à un autre.

Le code du modèle maître est similaire à ce qui est illustré ici. Il s'agit du code utilisé pour appeler le premier modèle :

```
{
    "type": "Microsoft.Resources/deployments",
    "apiVersion": "2017-05-10",
    "name": "createvm",
    "resoureceGroup": "myrg",
    "dependsOn": [
        "allResourceGroups"
    ],
    "properties":{
        "mode": "Incremental",
        "templateLink":{
            "uri": "[variables(
                'templateRefSharedServicesTemplateUri')]",
            "contentVersion": "1.0.0.0"
        },
        "parameters": {
            "VMName": {
                "value": "[variables('VmName')]"
            }
        }
    }
}
```

L'extrait de code précédent du modèle maître appelle un modèle imbriqué chargé du provisionnement d'une machine virtuelle. Le modèle imbriqué comporte une section de sortie qui fournit l'adresse IP de la machine virtuelle. Le modèle principal aura une autre ressource de déploiement dans son modèle qui prendra la valeur de sortie et l'enverra en tant que paramètre au modèle imbriqué suivant, en passant l'adresse IP au moment de l'exécution. Ceci est illustré dans le code suivant :

```
{
    "type": "Microsoft,Resources/deployments",
    "apiVersion": "2017-05-10",
    "name": "createSQLServer",
    "resourceGroup": "myrg",
    "dependsOn": [
        "createvm"
    ],
    "properties": {
        "mode": "Incremental",
        "templateLink": {
            "uri": "[variables('templateRefsql')]",
            "contentVersion": "1.0.0.0"
        },
        "parameters": {
            "VMName": {
                "value": "[reference
('createvm').outputs.virtualMachineIPAddress.value]"
            }
        }
    }
}
```

Dans la liste de codes précédente, un modèle imbriqué est appelé et un paramètre lui est transmis. La valeur du paramètre est déduite de la sortie du modèle lié précédent, nommée **`virtualMachineIPAddress`**. Maintenant, le modèle imbriqué obtiendra l'adresse IP de la machine virtuelle de manière dynamique et peut l'utiliser comme adresse IP figurant sur la liste blanche.

À l'aide de cette approche, nous pouvons passer des valeurs d'exécution d'un modèle imbriqué à un autre.

Résumé

Les modèles ARM constituent le moyen privilégié pour mettre en service des ressources dans Azure. Ils sont idempotents de par leur nature, apportant la cohérence, la prévisibilité, et la réutilisation à la création de l'environnement. Dans ce chapitre, nous avons appris comment créer un modèle ARM modulaire. Il est important que les équipes consacrent du temps à concevoir des modèles ARM correctement, afin que plusieurs équipes puissent travailler ensemble. Ils sont hautement réutilisables et nécessitent des changements minimes pour évoluer. Dans ce chapitre, nous avons appris à créer des modèles sécurisés dès la conception, à configurer plusieurs instances de ressources dans un déploiement unique et à passer des sorties d'un modèle imbriqué à un autre à l'aide de la section des sorties des modèles ARM.

Le prochain chapitre abordera une technologie différente et très utilisée, connue sous le nom de technologie sans serveur dans Azure. Azure Functions est l'une des principales ressources sans serveur d'Azure et nous l'aborderons en détail, ainsi que les fonctions durables.

17

Concevoir des solutions IoT

Au cours du chapitre précédent, vous avez découvert les modèles ARM et jusqu'ici, nous nous sommes intéressés aux préoccupations architecturales et à leurs solutions dans Azure en général. Toutefois, ce chapitre ne se base pas sur l'architecture d'une manière générale. Il traite de l'une des technologies les plus perturbatrices de ce siècle. Ce chapitre aborde les détails de l'**Internet des objets** (**IoT**) et d'Azure.

Azure IoT fait référence à une collection de services Cloud gérés par Microsoft qui peuvent se connecter, surveiller et contrôler des milliards d'actifs de l'IoT. En d'autres termes, une solution de l'Internet des objets est constituée d'un ou de plusieurs appareils IoT qui communiquent constamment avec un ou plusieurs serveurs back-end dans le Cloud.

Les thèmes suivants vont être abordés dans ce chapitre :

- Azure et IoT
- Présentation de l'IoT d'Azure
- Gestion des appareils
- Enregistrer les appareils
- Communication entre l'appareil et IoT Hub
- Mise à l'échelle des solutions IoT
- Haute disponibilité pour les solutions IoT
- Protocoles IoT
- Utiliser les propriétés des messages pour acheminer les messages

IoT

Internet a été inventé dans les années 1980 et est aujourd'hui disponible pour tous. Presque tout le monde a opté pour une présence sur Internet et a commencé à créer ses propres pages web statiques. Le contenu statique est devenu à terme dynamique et pouvait être généré à la volée, selon le contexte. Dans la plupart des cas, un navigateur était nécessaire pour accéder à Internet. Une multitude de navigateurs était à disposition, sans laquelle toute utilisation d'Internet relevait d'un véritable défi.

Durant la première décennie de ce siècle, des dispositifs intéressants ont commencé à faire leur apparition, à savoir des appareils qui se présentaient sous la forme d'un téléphone mobile ou d'une tablette. Les téléphones mobiles sont devenus toujours plus abordables et disponibles partout. Les fonctions matérielles et logicielles de ces appareils portatifs se sont considérablement améliorées, tant et si bien que les gens ont commencé à utiliser les navigateurs sur leurs appareils mobiles plutôt que sur leurs ordinateurs de bureau. Toutefois, une autre innovation perceptible a fait son apparition, à savoir les applications mobiles. Ces applications mobiles étaient téléchargées depuis des boutiques dédiées et se connectaient à Internet afin de communiquer avec les systèmes back-end. Vers la fin de la dernière décennie, des millions d'applications étaient disponibles, lesquelles comportaient quasiment toutes les fonctions qu'il était possible de leur attribuer. Le système back-end de ces applications était bâti sur le Cloud, de sorte à le rendre rapidement évolutif. C'était l'ère de la connexion des applications et des serveurs.

Mais, était-ce le summum de l'innovation ? Quelle a été l'évolution suivante de l'Internet ? Eh bien, un autre paradigme occupe désormais une place centrale, à savoir l'IdO. Au lieu de se contenter de connecter uniquement les appareils mobiles et les tablettes, pourquoi ne pas connecter également d'autres appareils à Internet ? Ces appareils étaient auparavant disponibles sur certains marchés, étaient coûteux, inaccessibles au grand public et dotés de capacités matérielles et logicielles limitées. Cependant, la commercialisation de ces dispositifs connaît une croissance à grande échelle depuis le début de notre décennie. Ces dispositifs sont de plus en plus petits, performants du point de vue matériel et logiciel, disposent de capacités de stockage et de traitement accrues et sont capables de se connecter à Internet sur plusieurs protocoles, tout en offrant la possibilité de les fixer à n'importe quel support. Il s'agit de l'ère de la connexion des appareils au serveur, aux applications et à d'autres appareils.

Il en a découlé l'idée selon laquelle ces applications IoT étaient capables de révolutionner les industries. Des nouvelles solutions auparavant inédites commencent à se concrétiser. Désormais, ces appareils pouvaient être fixés à n'importe quel support et pouvaient recueillir et transmettre des informations à un système back-end capable d'assimiler ces données en provenance de toutes sortes de dispositifs, puis de prendre une action ou d'établir un rapport sur la situation en présence.

Les capteurs et les contrôles IoT peuvent être exploités dans de nombreux cas d'utilisation commerciale. Par exemple, ils peuvent être utilisés dans les systèmes de suivi des véhicules, qui peuvent suivre tous les paramètres vitaux d'un véhicule et envoyer ces détails à un magasin de données centralisé pour analyse. Les initiatives urbaines intelligentes peuvent également faire appel à divers capteurs pour suivre les niveaux de pollution, la température et la congestion des rues. L'IoT a également fait son entrée dans des activités liées à l'agriculture, telles que la mesure de la fertilité du sol, de l'humidité, etc. Vous pouvez consulter les études de cas techniques de Microsoft pour l'IoT, https://microsoft.github.io/techcasestudies/#technology=IoT&sortBy=featured, afin de découvrir des exemples concrets de la façon dont les entreprises exploitent Azure IoT.

Avant d'explorer les outils et les services relatifs à l'IoT, nous allons examiner en détail l'architecture de l'IoT.

Architecture IoT

Avant de présenter Azure ainsi que son offre en termes de fonctions et de services IoT, il est essentiel de comprendre les différents composants nécessaires à la création des solutions IoT end-to-end.

Imaginez des dispositifs IoT à travers le monde qui envoient des millions de messages chaque seconde à une base de données centralisée. Pourquoi ces données sont-elles collectées ? Eh bien, pour extraire des informations riches sur les événements, les anomalies et les valeurs aberrantes qui ont trait aux éléments que ces appareils surveillent.

Essayons de comprendre cela plus en détail.

L'architecture IoT peut être divisée en plusieurs phases distinctes, comme suit :

1. **Connectivité :** cette phase implique une connexion entre un appareil et le service IoT.
2. **Identité :** une fois le service IoT connecté, l'appareil est d'abord identifié et le système vérifie qu'il est autorisé à envoyer la télémétrie de l'appareil au service IoT. Cela se fait à l'aide d'un processus d'authentification.
3. **Capture :** au cours de cette phase, la télémétrie de l'appareil est capturée et reçue par le service IoT.
4. **Ingestion :** au cours de cette phase, le service IoT ingère la télémétrie de l'appareil.
5. **Stockage :** la télémétrie de l'appareil est stockée. Il peut s'agir d'un magasin temporaire ou permanent.
6. **Transformation :** les données de télémétrie sont transformées en vue d'un traitement ultérieur. Il s'agit notamment d'augmenter les données existantes et de déduire des données.
7. **Analyse :** les données transformées sont utilisées pour trouver des modèles, des anomalies et des informations.
8. **Présentation :** les informations sont affichées sous forme de tableaux de bord et de rapports. En outre, de nouvelles alertes peuvent être générées afin d'appeler des scripts et des processus d'automatisation.

La figure 17.1 illustre une architecture générique basée sur l'IoT. Les données sont générées ou recueillies par des appareils, puis envoyées à des passerelles dans le Cloud. La passerelle dans le Cloud, quant à elle, transmet les données à plusieurs services back-end en vue de leur traitement. Les passerelles Cloud sont des composants optionnels ; elles doivent être utilisées lorsque les appareils ne sont pas capables d'envoyer eux-mêmes des requêtes aux services back-end, en raison de ressources limitées ou d'un réseau trop peu fiable. Ces passerelles Cloud peuvent rassembler les données issues de plusieurs appareils, puis les envoyer aux services back-end. Les données peuvent être traitées par ces services backend et apparaissent ensuite en tant qu'informations ou sous forme de tableaux de bord, à l'attention des utilisateurs :

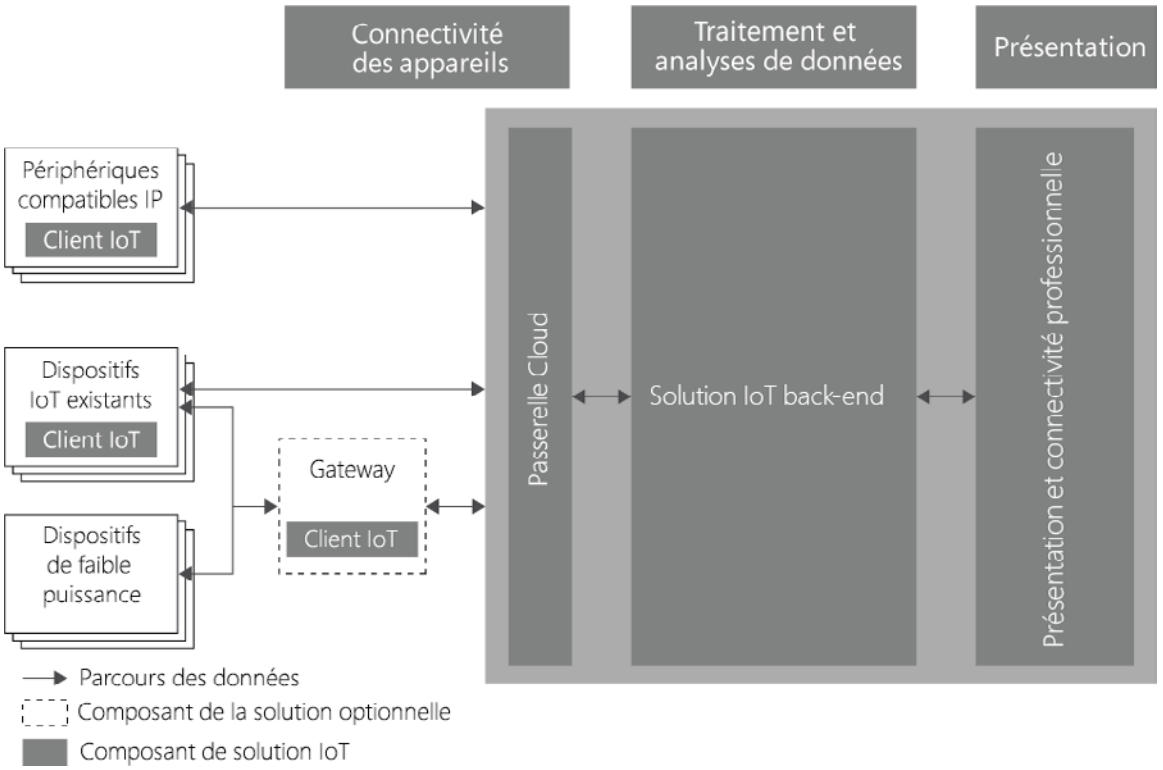


Figure 17.1 : Architecture générique basée sur l'IoT

Maintenant que nous avons une idée claire de l'architecture, nous allons chercher à comprendre comment les appareils IoT communiquent avec d'autres appareils.

Connectivité

Les appareils IoT doivent communiquer afin de se connecter à d'autres appareils. Il existe différents types de connectivité ; par exemple, la connectivité peut s'établir entre les appareils d'une région, entre les appareils et une passerelle centralisée, et entre les appareils et une plateforme IoT.

Dans tous les cas, les appareils IoT doivent être dotés d'une fonction de connectivité. Cette fonction peut établir une connexion à Internet, au Bluetooth, infra-rouge ou avec tout appareil à proximité.

Toutefois, certains appareils ne disposent pas de cette fonction de connexion à Internet. Dans ces cas, ils peuvent se connecter à une passerelle, qui disposera d'une connectivité à Internet.

Les appareils IoT utilisent des protocoles pour envoyer des messages. Les principaux protocoles sont **Advanced Message Queuing Protocol** (**AMQP**) et **Message Queue Telemetry Transport** (**MQTT**).

Les données des équipements doivent être envoyées à l'infrastructure IT. Le protocole MQTT est un protocole appareil à serveur, que les appareils peuvent utiliser pour envoyer des données de télémétrie et d'autres informations aux serveurs. Une fois que le serveur reçoit le message via le protocole MQTT, il doit transporter ce message vers d'autres serveurs à l'aide d'une technologie fiable, basée sur les messages et sur les files d'attente. Le protocole AMQP est à privilégier pour déplacer des messages entre les serveurs dans une infrastructure IT d'une manière fiable et prévisible :

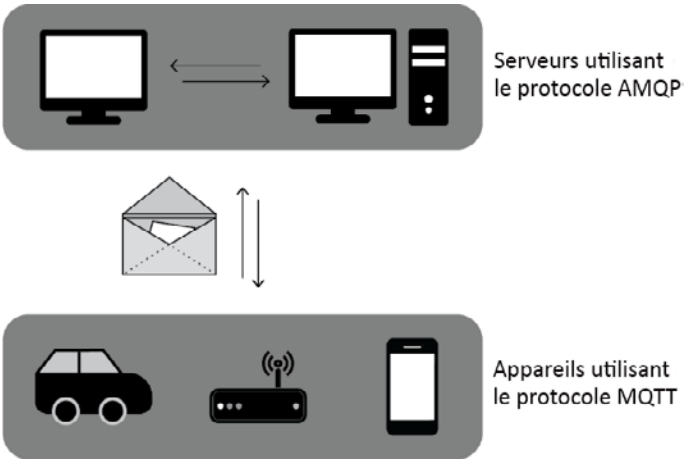


Figure 17.2 : Fonctionnement des protocoles MQTT et AMQP

Les serveurs qui reçoivent les messages initiaux de la part des appareils IoT doivent transmettre ces messages à d'autres serveurs en vue de leur traitement, par exemple pour les enregistrer dans des fichiers journaux, les évaluer, les analyser ou les présenter.

Certains appareils n'ont pas la capacité de se connecter à Internet ou ne prennent pas en charge les protocoles compatibles avec d'autres technologies de serveur. Pour permettre à ces appareils de fonctionner avec une plateforme IdO et le Cloud, des passerelles intermédiaires peuvent être utilisées. Les passerelles permettent d'intégrer les appareils dont la connectivité et les réseaux sont lents ou peu fiables, qui n'utilisent pas les protocoles standard ou encore dont les ressources et les performances sont limitées.

Si ces appareils nécessitent une infrastructure supplémentaire afin de participer et de se connecter aux services back-end, des passerelles client peuvent être déployées. Ces passerelles reçoivent des messages provenant de dispositifs à proximité et les transmettent à l'infrastructure IT et à la plateforme IoT, en vue d'un traitement plus poussé. Ces passerelles sont capables de traduire les protocoles si nécessaire.

Dans cette section, vous avez appris comment la communication est implémentée avec d'autres appareils et vous avez découvert le rôle joué par les passerelles en termes de communication. Dans la section suivante, nous allons aborder l'identité.

Identité

Les appareils IoT doivent être enregistrés dans une plateforme Cloud. Les appareils qui ne sont pas enregistrés ne doivent pas être autorisés à se connecter à une plateforme Cloud. Ces appareils doivent être enregistrés et une identité doit leur être attribuée. Un appareil doit envoyer ses informations d'identité lors de la connexion au Cloud. Si l'appareil ne parvient pas à envoyer ces informations d'identité, la connectivité doit échouer. Nous verrons plus loin dans ce chapitre comme générer une identité pour un appareil, à l'aide d'une application de simulation. Comme vous le savez déjà, les appareils IoT sont déployés pour capturer des informations et dans la section suivante, nous allons parler brièvement du processus de capture.

Capture

Les appareils IoT doivent pouvoir capturer des informations. Ils doivent avoir la capacité par exemple de lire ou de suivre le taux d'humidité de l'air ou du sol. Ces informations peuvent être capturées en fonction de la fréquence, peut-être même une fois par seconde. Une fois que les informations sont capturées, l'appareil doit être en mesure de les envoyer à la plateforme IoT en vue de leur traitement. Si un appareil ne peut pas se connecter à la plateforme IoT directement, il peut se connecter à des passerelles intermédiaires ou dans le Cloud afin de transmettre les informations capturées.

Pour l'appareil, la taille des données capturées ainsi que leur fréquence constituent les deux aspects les plus importants. Il est également important de déterminer si l'appareil dispose d'un stockage local et s'il stocke temporairement les données capturées. Par exemple l'appareil peut fonctionner en mode hors connexion si un stockage local est disponible. Même les appareils mobiles agissent parfois comme des dispositifs IoT connectés à divers instruments et ont la capacité de stocker des données. Une fois que nous avons capturé les données, nous devons les ingérer sur une plateforme IoT pour une analyse plus approfondie, et dans la section suivante, nous allons explorer l'ingestion.

Ingestion

Les données capturées et générées par les appareils doivent être envoyées à une plateforme IoT capable d'ingérer et de consommer ces données pour en extraire des informations utiles et exploitables. L'ingestion est un service important et crucial car sa disponibilité et son évolutivité ont des conséquences sur le débit des données entrantes. Si les données sont étranglées en raison de problèmes d'évolutivité ou si celles-ci ne peuvent être ingérées en raison d'un manque de disponibilité, le jeu de données peut être perdu ou biaisé. Nous avons les données capturées et nous avons besoin d'un endroit pour les stocker. Dans la section suivante, vous en apprendrez davantage sur le stockage.

Stockage

Les solutions IoT traitent généralement des millions et des milliards de documents couvrant des téraoctets ou même des pétaoctets de données. Il s'agit de données précieuses qui peuvent fournir des informations cruciales sur les opérations et sur l'état d'intégrité. Ces données doivent être stockées de sorte à permettre une analyse. Le stockage doit être disponible de sorte que les applications et services d'analyse puissent le consommer. Les solutions de stockage doivent assurer un débit et une latence appropriés du point de vue de la performance, être hautement disponible, évolutif et sécurisé. La section suivante traite de la transformation des données, qui est nécessaire pour stocker et analyser les données.

Transformation

Les solutions IoT sont généralement axées sur les données et gèrent des volumes considérables. Imaginez que chaque automobile soit équipée d'un dispositif et que chacun d'eux envoie des messages toutes les cinq secondes. Si 1 million de voitures envoient des messages, cela revient à 288 millions de messages par jour et à 8 milliards de messages par mois. Ensemble, ces données contiennent un grand nombre d'informations dissimulées, toutefois il est difficile de les exploiter d'un simple regard.

Les données capturées et stockées par les appareils IoT peuvent être consommées pour résoudre des problèmes d'entreprise, mais toutes les données capturées ne sont pas importantes. Juste un sous-ensemble de données peut être nécessaire pour résoudre un problème. En outre, les données recueillies par les appareils IdO peuvent ne pas être cohérentes non plus. Pour s'assurer de leur cohérence et de leur impartialité, des opérations de transformation appropriées doivent être exécutées, afin de les rendre exploitables en vue d'une analyse. Durant la transformation, les données sont alors filtrées, triées, extraites, enrichies et transformées vers une structure ; ces données peuvent être consommées par des composants et des applications en aval. Nous devons effectuer des analyses avec les données transformées avant de les présenter. Nous allons maintenant aborder l'analyse des données, qui constitue la prochaine étape du workflow.

Analyse des données

Les données transformées au cours de l'étape précédente deviennent les données sources et entrantes en vue d'une analyse. Selon le problème en présence, il existe différents types d'analyses qui peuvent être effectués sur ces données transformées.

Voici les différents types d'analytique qui peuvent être exécutés :

- **Analyse descriptive :** ce type d'analyse des données permet d'identifier des modèles et de recueillir des informations détaillées sur les statuts des appareils IoT et sur l'état d'intégrité global. Cette phase identifie et résume les données en vue d'analytiques avancées subséquentes. Cela permet de détecter des informations telles que des synthèses, des statistiques de probabilité, les écarts ainsi que d'autres tâches statistiques simples.

- **Analyse des diagnostics :** ce type d'analyse est plus avancé que l'analyse descriptive. Elle est réalisée suite à l'analyse descriptive et tente de comprendre les raisons pour lesquelles certains événements se sont produits. Elle tente de déterminer la cause racine des événements. Elle tente de trouver des réponses à l'aide de concepts avancés, tels que les hypothèses et les corrélations.
- **Analyse prédictive :** ce type d'analyse essaie de prédire les événements qui se produiront très probablement à l'avenir. Il génère des prédictions basées sur des données passées ; la régression est l'un des exemples basés sur les données passées. L'analytique prédictive pourrait, par exemple, prédire le prix d'une voiture, le comportement d'une action sur le marché boursier, quand un pneu de voiture va éclater, etc.
- **Analyse prescriptive :** ce type d'analyse des données est le plus avancé. Cette phase permet d'identifier les mesures les plus appropriées qui doivent être réalisées afin d'assurer l'intégrité des appareils, de veiller à ce que la solution ne se dégrade pas et de déterminer les mesures proactives préconisées. Les résultats de cette étape de l'analyse des données peuvent aider à éviter les désagréments futurs et à éliminer les problèmes au niveau de leurs causes profondes.

Au cours de l'étape finale, la sortie de l'analyse des données est présentée de manière lisible par l'homme pour un public plus large, afin de comprendre et d'interpréter. Dans la partie suivante, nous aborderons la présentation.

Présentation

L'analyse permet d'identifier des réponses, des modèles et des informations sur la base de données antérieures disponibles. Ces informations doivent également être mises à la disposition de toutes les parties prenantes, sous des formats compréhensibles. Pour ce faire, des rapports et tableaux de bord appropriés peuvent être générés, de manière statistique ou dynamique, puis présentés aux intéressés. Ces parties prenantes peuvent s'appuyer sur ces rapports pour prendre des mesures ultérieures ou pour améliorer en continu leur solution.

Pour résumer rapidement toutes les étapes précédentes, nous avons commencé cette section en examinant la connectivité, où nous avons introduit des passerelles pour envoyer les données à partir d'appareils qui ne prennent pas en charge les protocoles standard. Ensuite, nous avons parlé de l'identité et de la façon dont les données sont capturées. Les données capturées sont ensuite ingérées et stockées pour une transformation ultérieure. Après la transformation, l'analyse est effectuée sur les données avant qu'elles ne soient présentées à toutes les parties prenantes. Puisque nous travaillons sur Azure, dans la section suivante, nous aborderons la définition d'Azure IoT et nous examinerons les notions de base que nous avons apprises jusqu'à aujourd'hui du point de vue d'Azure.

Azure IoT

Vous avez maintenant appris les différentes étapes des solutions IoT end-to-end ; chacune de ces étapes est cruciale et leur mise en œuvre appropriée est essentielle à la réussite de toute solution. Azure fournit un grand nombre de services pour chacune de ces étapes. En dehors de ces services, Azure fournit Azure IoT Hub, qui est le service et la plateforme IoT de base d'Azure. Il est capable d'héberger des solutions IoT complexes, hautement disponibles et évolutives. Nous allons étudier plus en détail IoT Hub après avoir passé en revue d'autres services :

Appareils	Connectivité des appareils	Stockage	Analytique	Présentation et action
	Événements	SQL Database	Machine Learning	App Service
	Service Bus	Stockage de table/blob	Stream Analytics	Power BI
	Sources de données externes	Cosmos DB	HDInsight	Notification Hubs
		Sources de données externes	Data Factory	Mobile Services
		Time Series Insights	DataBricks	Logic Apps

Figure 17.3 : Liste des appareils et des services pour les solutions IoT

Dans la section suivante, nous allons suivre un modèle similaire à celui que nous avons fait pour notre couverture de l'architecture IoT afin d'en savoir plus sur la communication, l'identité, la capture, l'ingestion, le stockage, la transformation, l'analyse et la présentation avec Azure IoT.

Connectivité

IoT Hub fournit toutes les suites de protocole importantes pour les appareils qui se connectent aux hubs IoT.
Il offre :

- **HTTPS :** la méthode sécurisée de protocole de transport hypertexte utilise des certificats consistant en une paire de clés, connues sous le nom de clés publiques privées, qui sont utilisées pour chiffrer et déchiffrer les données entre un appareil et IoT Hub. Il fournit une communication à sens unique d'un appareil vers le Cloud.
- **AMQP :** AMQP est une norme du secteur pour l'envoi et la réception de messages entre les applications. Elle fournit une infrastructure riche pour la sécurité et la fiabilité des messages, et c'est l'une des raisons pour lesquelles elle est assez largement utilisée dans le domaine de l'IoT. Elle fournit à la fois des fonctionnalités d'appareil à Hub, ainsi que des capacités de Hub à appareil, et les appareils peuvent l'utiliser pour s'authentifier à l'aide de **la sécurité basée sur les revendications** (**CBS**) ou d'une couche **d'authentification et de sécurité simple** (**SASL**). Elle est principalement utilisée dans les scénarios où il existe des passerelles de champ,

et lorsqu'une identité unique associée à plusieurs appareils peut transmettre des données de télémétrie au Cloud.

- **MQTT :** MQTT est une norme du secteur pour l'envoi et la réception de messages entre les applications. Elle fournit à la fois des fonctionnalités d'appareil à Hub, ainsi que des capacités Hub vers l'appareil. Elle est principalement utilisée dans les scénarios où chaque appareil a sa propre identité et s'authentifie directement avec le Cloud.

Dans la section suivante, nous aborderons l'identité et la façon dont les appareils sont authentifiés.

Identité

IoT Hub fournit également des services pour les appareils d'authentification. Il offre une interface permettant de générer les hachages d'identité unique pour chaque appareil. Lorsque ces appareils envoient des messages contenant ce hachage, IoT Hub peut les identifier après avoir vérifié dans sa propre base de données qu'un tel hachage existe. Voyons maintenant comment les données sont capturées.

Capture

Azure fournit des passerelles IoT qui permettent à des appareils non conformes à IoT Hub de s'adapter et ainsi de communiquer leurs données. Des passerelles locales ou intermédiaires sont déployées à proximité des appareils, de sorte que plusieurs d'entre eux puissent se connecter à une passerelle unique afin de capture et d'envoyer leurs informations. De même, plusieurs de ces clusters d'appareils avec passerelle locale peuvent être déployés. Il peut par exemple s'agir d'une passerelle Cloud déployée sur le Cloud lui-même, capable de capturer et d'accepter des données issues de plusieurs sources et de les ingérer dans IoT Hub. Comme nous l'avons mentionné précédemment, nous devons ingérer les données que nous capturons. Dans la section suivante, vous découvrirez l'ingestion avec IoT Hub.

Ingestion

Un IoT Hub peut être un point de contact unique pour les appareils et d'autres applications. En d'autres termes, l'ingestion de messages IoT relève de la responsabilité du service IoT Hub. Il existe d'autres services, tels que les hubs d'événements et l'infrastructure de messagerie Service Bus, qui peuvent fournir des capacités d'ingestion des messages entrants, toutefois les avantages en la matière qu'offre IoT Hub sont bien plus intéressants. En effet, IoT Hub a été spécifiquement conçu pour ingérer les messages IoT au sein de l'écosystème Azure, de sorte que le reste des composants et des services puissent agir par la suite en conséquence. Les données ingérées sont stockées dans le stockage. Avant d'effectuer n'importe quel type de transformation ou d'analyse, nous allons explorer le rôle du stockage dans le workflow dans la section suivante.

Stockage

Azure fournit plusieurs solutions de stockage pour stocker les messages provenant d'appareils IoT. Celles-ci comprennent le stockage des données relationnelles, les schémas sans données NoSQL et les objets blob :

- **Base de données SQL :** une base de données SQL fournit le stockage de données relationnelles, JSON et de documents XML. Il fournit le langage de requête enrichi SQL et utilise un véritable serveur SQL en tant que service. Les données des appareils peuvent être stockées dans les bases de données SQL si ces données sont bien définies et s'il n'est pas prévu que le schéma subisse des modifications fréquentes.
- **Azure Storage :** Azure Storage offre un stockage via des tables et des objets blob. Le stockage de table contribue à stocker des données en tant qu'entités, dans le cadre desquelles le schéma n'est pas important. Les blobs permettent de stocker des fichiers dans des conteneurs en tant qu'objets blob.
- **Cosmos DB :** Cosmos DB est une base de données NoSQL à l'échelle de toute l'entreprise. Elle est disponible en tant que service capable de stocker des données sans schéma. Il s'agit d'une base de données distribuée à l'échelle mondiale qui peut couvrir des continents et assure la haute disponibilité et l'évolutivité des données.
- **Sources de données externes :** mis à part les services Azure, les clients peuvent apporter leurs propres magasins de données, tels que SQL Server sur des machines virtuelles Azure et peuvent les utiliser pour stocker des données au format relationnel.

La section suivante concerne la transformation et l'analyse des données.

Transformation et analyse de données

Azure fournit plusieurs ressources pour exécuter des tâches et des activités sur les données ingérées. Certaines d'entre elles sont présentées ici :

- **Data Factory :** Azure Data Factory est un service d'intégration de données basé sur le Cloud qui permet de créer des charges de travail pilotées par les données dans le Cloud afin d'orchestrer et d'automatiser le déplacement des données et leur transformation. Azure Data Factory permet de créer et de planifier des workflows axés sur les données (nommés pipelines) capables d'ingérer des données issues de magasins de données distincts ; de traiter et de transformer des données en utilisant des services de calcul tels que **Azure HDInsight**, **Hadoop, Spark**, **Azure Data Lake Analytics**, **Azure Synapse Analytics** et **Azure Machine Learning** ; et de publier des données de sortie dans un entrepôt de données pour des applications de **Business Intelligence** (**BI**) plutôt que sur une plateforme classique **Extraire-Transformer-Charger** (**ETL**).
- **Azure Databricks :** Databricks fournit un environnement Spark complet, géré et end-to-end. Il peut contribuer à la transformation des données à l'aide de Scala et de Python. Il fournit également une bibliothèque SQL pour manipuler les données à l'aide de la syntaxe SQL traditionnelle. Il est plus performant que les environnements Hadoop.

- **Azure HDInsight :** Microsoft et Hortonworks s'unissent pour aider les entreprises en offrant une plateforme d'analyse du Big Data avec Azure. HDInsight est un environnement de service dans le Cloud puissant et entièrement géré, alimenté par Apache Hadoop et Apache Spark à l'aide de Microsoft Azure HDInsight. Il permet d'accélérer les charges de travail en toute transparence avec le service Cloud de traitement des Big Data de Microsoft et Hortonworks le plus évolué du secteur.
- **Azure Stream Analytics :** il s'agit d'un service d'analyse des données entièrement géré en temps réel, qui permet de réaliser des opérations de calcul et de transformation sur les données en streaming. Le service Stream Analytics permet d'examiner de grandes quantités de données provenant des appareils ou des processus, d'extraire les informations issues de ce flux de données et de rechercher des modèles, des tendances et des relations.
- **Machine Learning :** il s'agit d'une technique de science de données qui permet aux ordinateurs d'utiliser des données existantes pour prévoir les comportements futurs, les résultats et les tendances. Grâce au Machine Learning, les ordinateurs apprennent les comportements en fonction du modèle que nous créons. Azure Machine Learning est un service Cloud d'analyse prédictive qui permet de créer et de déployer rapidement des modèles prédictifs en tant que solutions d'analyse. Cette solution fournit une bibliothèque d'algorithmes prêts à l'emploi qui permet de créer des modèles sur un PC connecté à Internet et de déployer rapidement des solutions prédictives.
- **Azure Synapse Analytics :** anciennement connu sous le nom d'Azure SQL Data Warehouse. Azure Synapse Analytics fournit des services d'analyse adaptés au stockage de données d'entreprise et aux analyses du Big Data. Il prend en charge l'ingestion directe en continu, qui peut être intégrée à Azure IoT Hub.

Maintenant que vous êtes familiarisé avec les outils de transformation et d'analyse utilisés dans Azure pour les données ingérées par les appareils IoT, nous allons découvrir comment ces données peuvent être présentées.

Présentation

Après avoir réalisé une analyse appropriée des données, celles-ci doivent être présentées aux intervenants sous un format qu'ils peuvent utiliser. Il existe de nombreuses manières de présenter les informations exploitables extraites de ces données. Celles-ci peuvent être présentées à l'aide d'applications web déployées via les services d'application Azure ou encore envoyées à des hubs de notification qui enverront des alertes sur des applications mobiles, etc. Toutefois, l'approche idéale pour présenter et utiliser ces informations consiste à utiliser des rapports et des tableaux de bord **Power BI**. Power BI est un outil de visualisation Microsoft qui crée des rapports et des tableaux de bord dynamiques sur Internet.

Pour conclure, Azure IoT est étroitement aligné sur les concepts de base de l'architecture IoT. Il suit le même processus ; toutefois, Azure nous donne la liberté de choisir différents services et dépendances en fonction de nos besoins. Dans la section suivante, nous allons nous concentrer sur Azure IoT Hub, un service hébergé dans le Cloud et entièrement géré par Azure.

Azure IoT Hub

Les projets IoT sont généralement d'une nature complexe. La complexité provient du volume élevé d'appareils et de données. Les appareils sont intégrés dans le monde entier ; par exemple, les appareils de surveillance et d'audit, qui sont utilisés pour stocker des données, transformer et analyser des pétaoctets de données, et enfin prendre des mesures basées sur des informations. En outre, ces projets s'assortissent de longues périodes de gestation et leurs exigences évoluent au fil de leur calendrier.

Si une entreprise veut se lancer dans un parcours impliquant un projet IoT rapidement, alors elle se rendra vite compte que les problèmes que nous avons mentionnés ne sont pas facilement résolus. Ces projets nécessitent des ressources matérielles conséquentes en termes de capacités de traitement et de stockage, ainsi que des services capables de gérer des volumes de données élevés.

IoT Hub est une plateforme conçue pour faciliter la mise en place de ces projets IoT et ainsi permettre une livraison plus simple et plus rapide. Cette solution fournit toutes les fonctions et les services nécessaires pour les points suivants :

- Inscription d'appareil
- Connectivité des appareils
- Passerelles sur site
- Passerelles dans le Cloud
- Application des protocoles de l'industrie tels qu'AMQP et le protocole MQTT
- Hub de stockage des messages entrants
- L'acheminement des messages sur la base des propriétés et des contenus de ces derniers
- Points de terminaison multiples pour différents types de traitement
- Connectivité à d'autres services sur Azure pour l'analyse en temps réel et bien plus encore

Jusqu'à présent, nous avons abordé une vue d'ensemble d'Azure IoT Hub, mais nous allons maintenant entrer dans le vif du sujet pour en savoir plus sur les protocoles et la façon dont les appareils sont enregistrés avec Azure IoT Hub.

Protocoles

Azure IoT Hub offre une prise en charge native de la communication via les protocoles MQTT, AMQP et HTTP. Parfois, certains dispositifs ou passerelles sur site ne sont pas en mesure d'utiliser l'un de ces protocoles standard et nécessiteront une adaptation du protocole. Dans de tels cas, une passerelle personnalisée peut être déployée. La passerelle de protocole Azure IoT permet l'adaptation du protocole pour les points de terminaison IoT Hub reliant ainsi le trafic entrant dans et sortant d'IoT Hub. Dans la section suivante, nous allons discuter de la façon dont les appareils sont enregistrés avec Azure IoT Hub.

Inscription d'appareil

Des dispositifs doivent être enregistrés avant de pouvoir envoyer des messages à IoT Hub. L'enregistrement de ces dispositifs peut se faire manuellement via le portail Azure ou peut être automatisé à l'aide des kits de développement logiciel IoT Hub. Azure fournit également des exemples d'applications de simulation, qui facilitent l'enregistrement des appareils virtuels à des fins de test et de développement. Il existe également un simulateur en ligne Raspberry Pi qui peut être utilisé en tant qu'appareil virtuel. En outre, il existe bien entendu des appareils physiques qui peuvent être configurés de sorte à se connecter à IoT Hub.

Si vous souhaitez simuler un appareil depuis un ordinateur local, généralement à des fins de test et de développement, des tutoriels sont disponibles dans les documents Azure, et ce pour plusieurs langages. Ceux-ci sont disponibles à l'adresse https://docs.microsoft.com/azure/iot-hub/iot-hub-get-started-simulated.

Le simulateur en ligne Raspberry Pi est disponible à l'adresse https://docs.microsoft.com/azure/iot-hub/iot-hub-raspberry-pi-web-simulator-get-started. En outre, suivez la procédure indiquée à l'adresse https://docs.microsoft.com/azure/iot-hub/iot-hub-get-started-physical pour utiliser des appareils physiques et les enregistrer dans IoT Hub.

Pour ajouter manuellement un appareil en utilisant le portail Azure, IoT Hub fournit le menu **Appareils IoT** qui permet de configurer un nouvel appareil. La sélection de l'option **Nouveau** vous permet de créer un nouvel appareil, comme illustré à la *figure 17.4* :

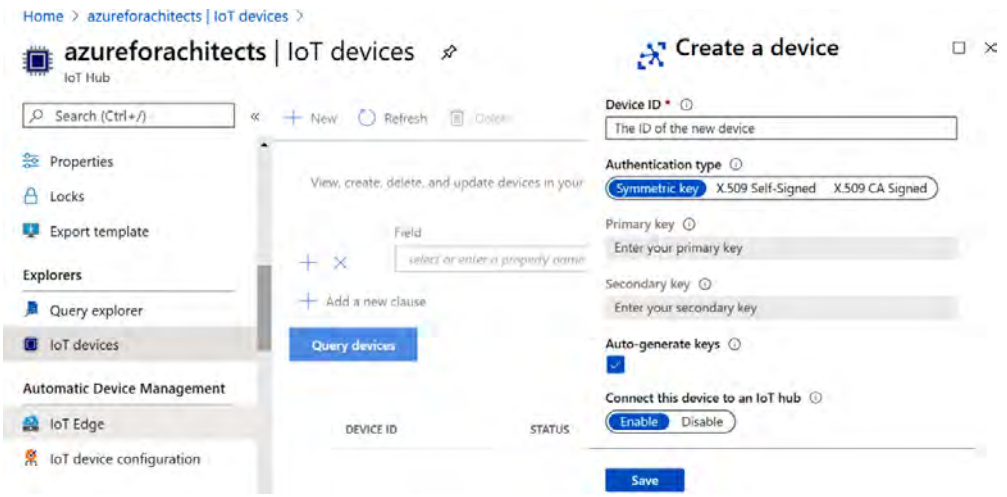


Figure 17.4 : Ajout d'un appareil via le portail Azure

Une fois l'identité de l'appareil créée, la chaîne de connexion de clé principale d'IoT Hub doit être utilisée dans chaque appareil qui est connecté à ce dernier.

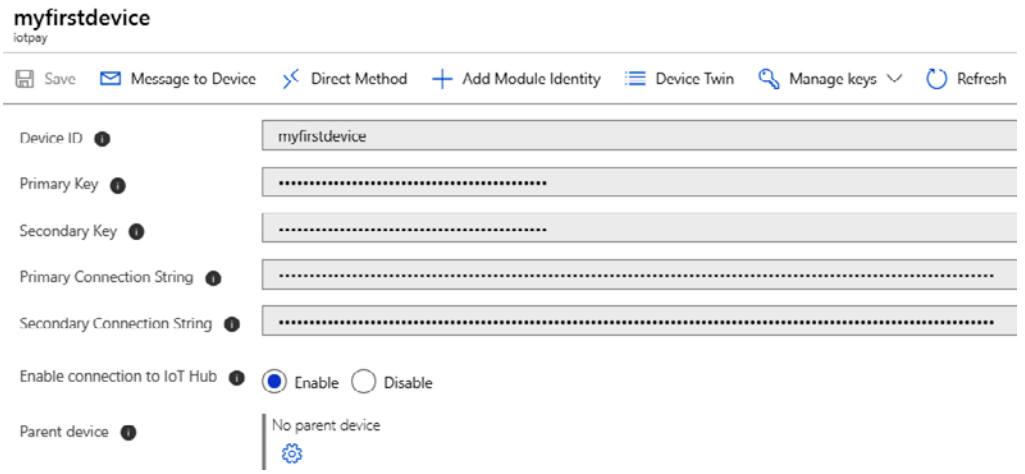


Figure 17.5 : Création de chaînes de connexion pour chaque appareil

À ce stade, l'appareil a été enregistré auprès d'IoT Hub, et notre prochaine mission consiste à faire en sorte que la communication se produise entre l'appareil et IoT Hub. La section suivante vous donnera une bonne idée de la façon dont la gestion des messages est exécutée.

Gestion des messages

Une fois les appareils enregistrés auprès d'IoT Hub, ils peuvent commencer à interagir avec celui-ci. La gestion des messages fait référence à la façon dont la communication ou l'interaction entre l'appareil IoT et IoT Hub est terminée. L'interaction peut provenir du dispositif en direction du Cloud ou vice versa.

Messages depuis l'appareil vers le Cloud

Dans le cadre de ce type de communication, il convient d'observer une bonne pratique, à savoir que, bien que l'appareil capture un grand nombre d'informations, seules les données d'une certaine importance doivent être transmises au Cloud. La taille du message est très importante dans les solutions IoT en raison de la nature intrinsèque de ces dernières, qui sont généralement très riches en volumes. Même 1 Ko de données supplémentaires peut entraîner un gigaoctet de stockage et de traitement gaspillé. Chaque message contient des propriétés et des charges utiles ; les propriétés définissent les métadonnées du message. Ces métadonnées contiennent des données sur le périphérique, l'identification, les balises ainsi que d'autres propriétés utiles au routage et à l'identification des messages.

Les appareils ou les passerelles Cloud doivent se connecter à IoT Hub pour transférer des données. IoT Hub fournit des points de terminaison publics pouvant être utilisés par les appareils afin de se connecter et d'envoyer des données. IoT Hub doit être considéré comme le premier point de contact pour le traitement backend. IoT Hub est capable de transmettre et d'acheminer davantage ces messages vers plusieurs services. Par défaut, les messages sont stockés dans le hub d'événements. Plusieurs hubs d'événements peuvent être créés pour différents messages. Les points de terminaison prédéfinis utilisés par les appareils pour envoyer et recevoir des données peuvent être consultés dans le panneau **Points de terminaison intégrés** d'IoT Hub. La *figure 17.6* montre comment vous pouvez trouver les points de terminaison prédéfinis :

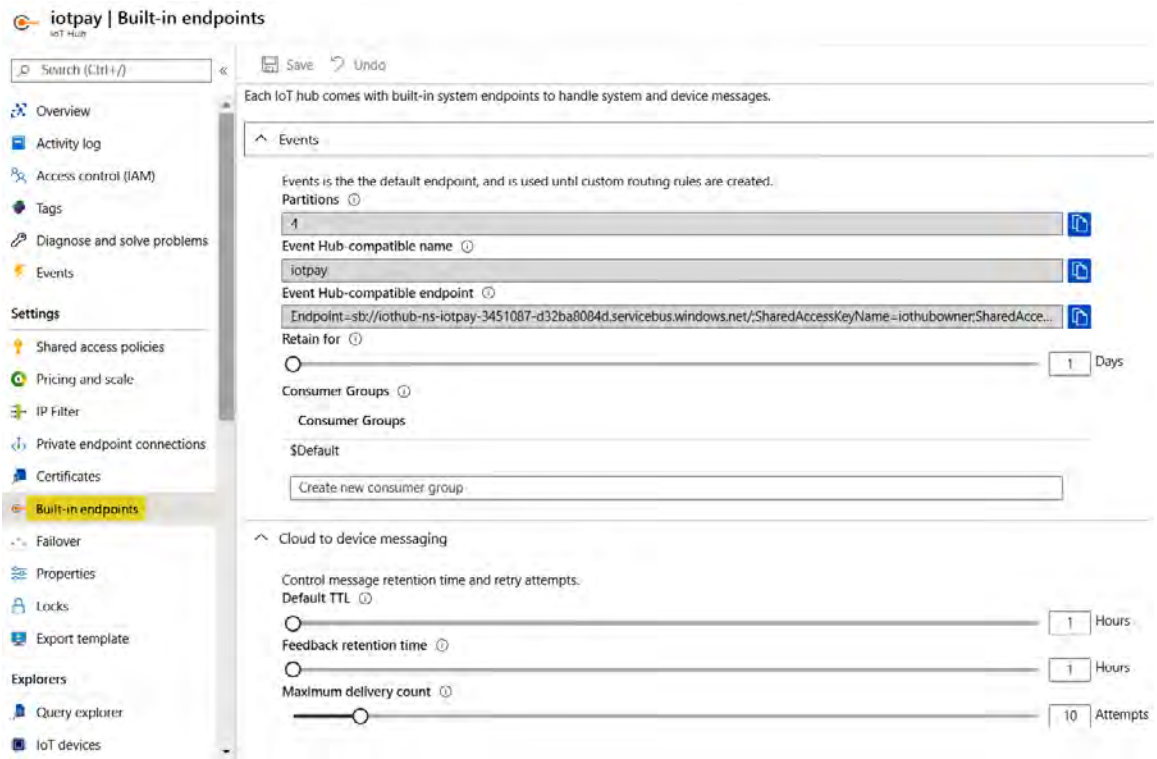


Figure 17.6 : Création de plusieurs hubs d'événements

Les messages peuvent être acheminés vers différents points de terminaison en fonction des propriétés d'en-tête et de corps du message, comme illustré à la *figure 17.7* :

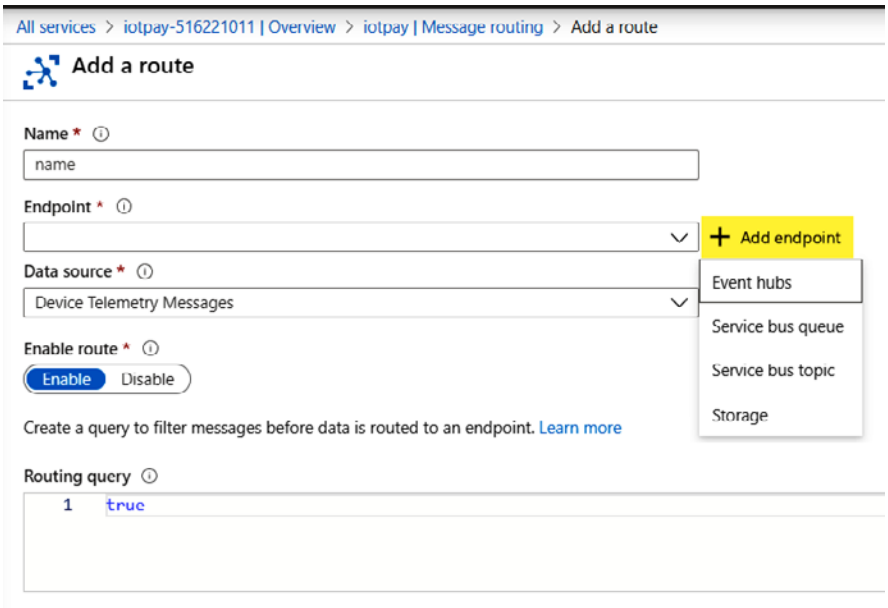


Figure 17.7 : Ajout d'un nouvel itinéraire à différents points de terminaison

Les messages contenus dans un IoT Hub y demeurent pendant sept jours par défaut, et leur taille peut aller jusqu'à 256 Ko.

Il existe un exemple de simulateur fourni par Microsoft pour simuler l'envoi de messages dans le Cloud. Il est disponible dans plusieurs langages et la version C# peut être consultée sur https://docs.microsoft.com/azure/iot-hub/iot-hub-csharp-csharp-c2d.

Messages depuis le Cloud vers l'appareil

IoT Hub est un service géré, qui fournit une infrastructure de messagerie bidirectionnelle. Les messages peuvent être envoyés depuis le Cloud vers les appareils, lesquels pourront agir en conséquence.

Il existe trois types de modèles de message Cloud vers l'appareil :

- Les méthodes directes nécessitent une confirmation immédiate des résultats. Les méthodes directes sont souvent utilisées pour instaurer le contrôle interactif des dispositifs, tel que l'ouverture et la fermeture des volets roulants d'un garage. Elles suivent le modèle requête-réponse.
- La configuration des propriétés d'appareil à l'aide d'Azure IoT fournit des propriétés de **jumeau d'appareil**. Par exemple, vous pouvez définir l'intervalle d'envoi de télémétrie à 30 minutes. Les jumeaux d'appareils sont des documents JSON qui stockent des informations sur l'état de l'appareil (métadonnées, configurations et conditions). Un hub IoT maintient un jumeau d'appareil pour chaque appareil qu'il contient.

- Les messages Cloud vers l'appareil sont utilisés pour les notifications à sens unique vers l'application de l'appareil. Ceux-ci suivent le modèle Tire et oublie.

Dans chaque organisation, la sécurité est une préoccupation majeure, et même dans le cas des appareils et des données IoT, cette inquiétude reste présente. Nous aborderons la sécurité à la section suivante.

Sécurité

La sécurité est un aspect important des applications basées sur l'IoT. Les applications IoT englobent des dispositifs qui utilisent l'Internet public pour la connectivité à des applications back-end. La sécurisation des appareils, des applications back-end et la connectivité afin de protéger les systèmes contre toute attaque malveillante ou émanant de pirates informatiques est une priorité essentielle afin de garantir la pérennité de ces applications.

La sécurité dans l'IoT

Les applications IoT sont principalement construites autour d'Internet, et la sécurité doit jouer un rôle majeur pour veiller à ce qu'une solution ne soit pas compromise. Voici quelques-unes des décisions de sécurité majeures, ayant un impact sur l'architecture de l'IoT :

- En ce qui concerne les appareils qui utilisent des points de terminaison HTTP plutôt que des points de terminaison HTTPS REST, les points de terminaison REST sécurisés via des certificats permettent de s'assurer que les messages transférés du dispositif vers le Cloud et vice versa sont bien chiffrés et signés. Les messages doivent être indéchiffrables par tout intrus et être difficiles à deviner.
- Si les appareils sont connectés à une passerelle locale, celle-ci doit se connecter au Cloud en utilisant un protocole HTTP sécurisé.
- Des dispositifs doivent être enregistrés dans IoT Hub avant de pouvoir envoyer des messages.
- Les informations transmises au Cloud doivent demeurer dans un stockage hautement sécurisé. Des jetons SAS appropriés ou des chaînes de connexion stockées dans Azure Key Vault doivent être utilisés pour la connexion.
- Azure Key Vault doit être utilisé pour stocker tous les secrets, les mots de passe et les informations d'identification, y compris les certificats.
- Azure Security Center pour l'IoT effectue la prévention et l'analyse des menaces pour tous les appareils, l'IoT Edge et l'IoT Hub, à travers vos actifs IoT. Nous pouvons créer nos propres tableaux de bord dans Azure Security Center en fonction des évaluations de sécurité. Parmi les principales fonctionnalités, citons la gestion centralisée d'Azure Security Center, la protection adaptative contre les menaces et la détection intelligente des menaces. Il est recommandé d'envisager l'utilisation d'Azure Security Center tout en mettant en œuvre des solutions IoT sécurisées.

Ensuite, nous allons examiner l'aspect de l'évolutivité d'IoT Hub.

Évolutivité

L'évolutivité d'IoT Hub diffère quelque peu de celle des autres services. Dans IoT Hub, il existe deux types de message :

- **Entrant :** messages depuis l'appareil vers le Cloud
- **Sortant :** messages depuis le Cloud vers l'appareil

Tous deux doivent être pris en compte du point de vue de l'évolutivité.

IoT Hub fournit plusieurs options de configuration durant la mise en service, afin de mettre en place l'évolutivité. Ces options sont également disponibles après la mise en service et peuvent être mises à jour afin de mieux répondre aux exigences de la solution en termes d'évolutivité.

Les options d'évolutivité disponibles pour IoT Hub sont les suivantes :

- L'édition **SKU** (**Stock Keeping Unit**), qui correspond à la taille d'IoT Hub
- Le nombre d'unités

Nous commencerons par étudier l'option d'édition SKU.

L'édition SKU

La référence d'IoT Hub détermine le nombre de messages qui peuvent être gérés par chaque unité et par jour, ce qui inclut à la fois les messages entrants et sortants. Il y a quatre niveaux, comme suit :

- **Gratuit :** celle-ci permet de gérer 8 000 messages par unité et par jour et autorise à la fois les messages entrants et sortants. 1 unité au maximum peut être mise en service. Cette édition permet de se familiariser et de tester les capacités du service IoT Hub.
- **Standard (S1) :** celle-ci permet de gérer 400 000 messages par unité et par jour et autorise à la fois les messages entrants et sortants. 200 unités au maximum peuvent être mises en service. Cette édition est destinée à un nombre de messages restreint.
- **Standard (S2) :** celle-ci permet de gérer 6 millions de messages par unité et par jour et autorise à la fois les messages entrants et sortants. 200 unités au maximum peuvent être mises en service. Cette édition est destinée à un grand nombre de messages.
- **Standard (S3) :** celle-ci permet de gérer 300 millions de messages par unité et par jour et autorise à la fois les messages entrants et sortants. 10 unités au maximum peuvent être mises en service. Cette édition est destinée à un très grand nombre de messages.

Les options de mise à niveau et de mise à l'échelle sont disponibles dans le portail Azure, sous le panneau **Tarification et évolutivité** d'IoT Hub. Les options seront présentées comme illustré à la *figure* 17.8 :

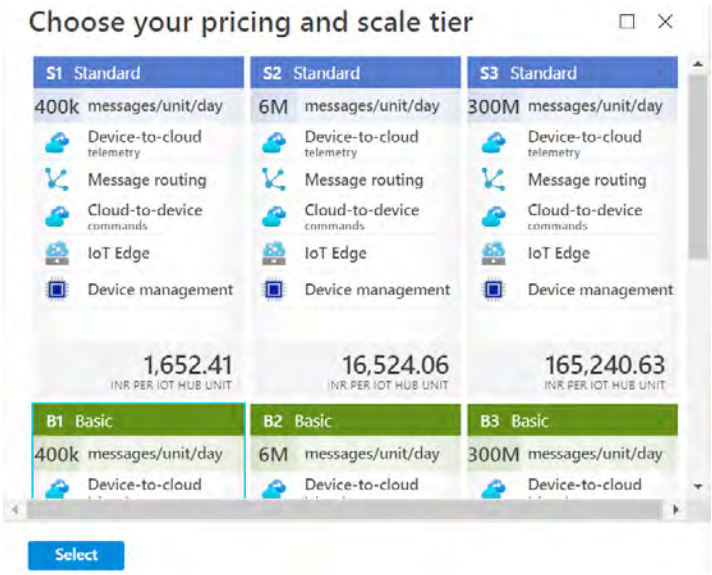


Figure 17.8 : Choix d'un niveau de tarification et de mise à l'échelle

Vous remarquerez peut-être que le niveau **Standard S3** autorise un maximum de **10 unités** seulement par rapport aux autres unités standard qui permettent l'utilisation de **200 unités** maximum. Ceci est directement lié à la taille des ressources de calcul qui sont provisionnées pour exécuter des services IoT. La taille et la capacité des machines virtuelles pour **Standard S3** sont nettement plus élevées que dans les autres niveaux, où la taille reste la même.

Unités

Les unités définissent le nombre d'instances de chaque référence exécutée à l'arrière-plan du service. Par exemple, 2 unités du niveau **Standard S1** signifient qu'IoT Hub est capable de traiter 400 000 x 2 = 800 000 messages par jour.

Les unités augmentent l'évolutivité de l'application. La *figure 17.9* provient du panneau **Tarification et évolutivité** d'IoT Hub, où vous pouvez voir le niveau de tarification actuel et le nombre d'unités :

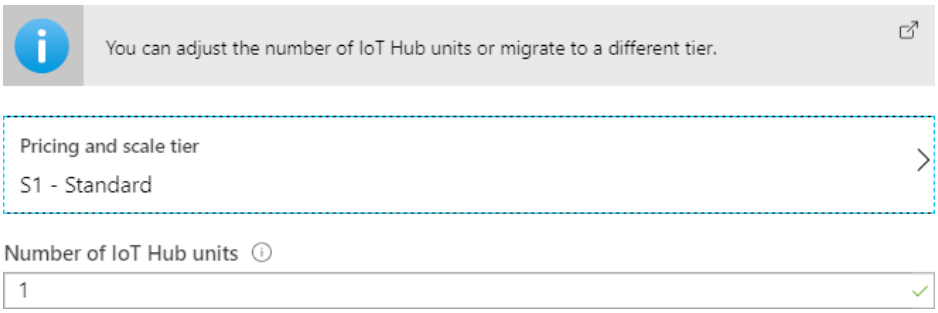


Figure 17.9 : Options de réglage ou de migration des unités IoT Hub

L'un des services en plein essor d'Azure IoT Hub est Azure IoT Edge, un service entièrement géré basé sur Azure IoT Hub. Nous allons explorer Azure IoT Edge dans la section suivante.

Azure IoT Edge

Microsoft Azure IoT Edge utilise le computing en périphérie pour mettre en œuvre des solutions IoT. Le computing en périphérie fait référence aux ressources de calcul disponibles sur votre réseau local, directement à la sortie de votre réseau, là où l'Internet public commence. Cela peut être déployé sur votre réseau principal ou sur un réseau invité avec l'isolation du pare-feu.

Azure IoT Edge est constitué du runtime IoT Edge, qui doit être installé sur un ordinateur ou un appareil. Docker sera installé sur l'ordinateur ; l'ordinateur peut exécuter Windows ou Linux. Le rôle de docker est d'exécuter les modules IoT Edge.

Azure IoT Edge repose sur le concept de Cloud hybride, qui vous permet de déployer et de gérer des solutions IoT sur le matériel local et de les intégrer facilement à Microsoft Azure.

Microsoft fournit une documentation complète pour Azure IoT Edge, avec des modèles de démarrage rapide et des conseils sur l'installation des modules. Le lien vers la documentation est https://docs.microsoft.com/azure/iot-edge.

Dans la section suivante, nous allons examiner la façon dont l'infrastructure est gérée dans le cas d'Azure IoT Hub et la façon dont la haute disponibilité est fournie aux clients.

Haute disponibilité

IoT Hub est une offre **plateforme en tant que service** (**PaaS**) d'Azure. Les clients et les utilisateurs n'interagissent pas directement avec le nombre et la taille des machines virtuelles sous-jacentes sur lesquelles le service IoT Hub s'exécute. Les utilisateurs décident de la région, de la référence du hub IoT et du nombre d'unités propres à leur application. Le reste de la configuration est déterminé et exécuté par Azure en arrière-plan. Azure veille à ce que chaque service PaaS soit hautement disponible par défaut. Pour ce faire, il veille à ce que plusieurs machines virtuelles mises en service en arrière-plan se trouvent sur des armoires distinctes du datacenter. Il place par conséquent ces machines virtuelles sur un ensemble de disponibilité et sur des domaines d'erreur et de mise à jour distincts. Cela permet d'assurer la haute disponibilité pour les tâches de maintenance planifiée et non planifiée. Les ensembles de disponibilité assurent la haute disponibilité au niveau du datacenter.

Dans la section suivante, nous aborderons Azure IoT Central.

Azure IoT Central

Azure IoT central fournit une plateforme de classe entreprise pour créer des applications IoT performantes afin de répondre aux besoins de votre entreprise de manière sécurisée, fiable et évolutive. IoT Central élimine le coût de développement, de maintenance et de gestion des solutions IoT.

IoT Central fournit une gestion centralisée qui vous permet de gérer et de surveiller les appareils, les conditions des appareils, la création de règles et les données des appareils. Sur la *figure 17.10*, vous pouvez voir quelques-uns des modèles disponibles sur le portail Azure lors de la création d'applications IoT Central :

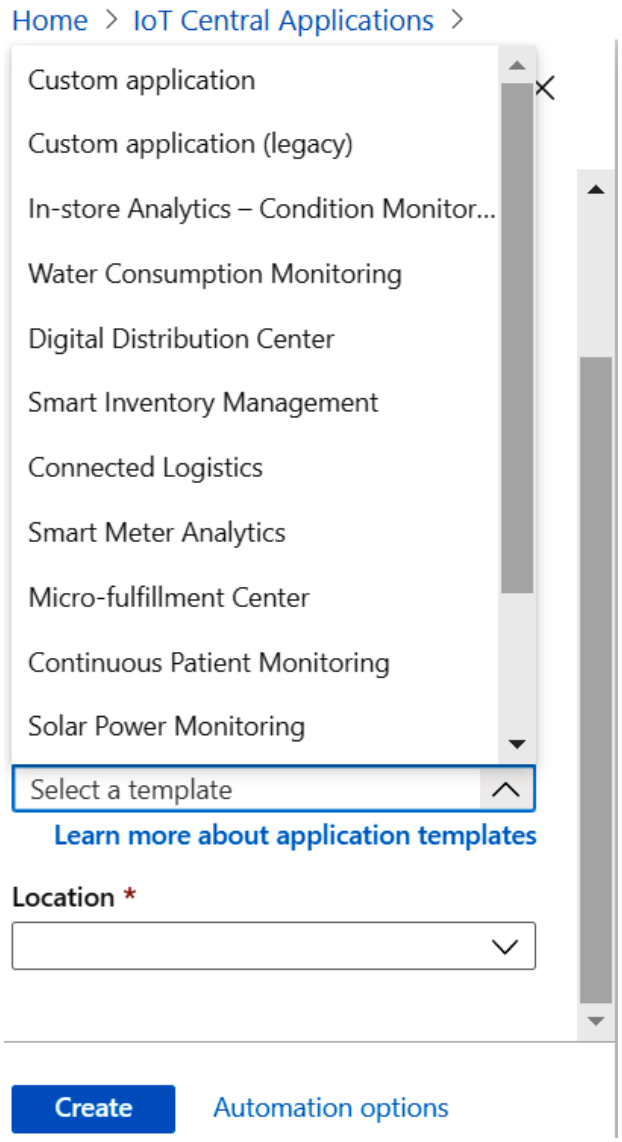


Figure 17.10 Création d'une application Azure IoT Central

Les modèles vous donneront une longueur d'avance et vous pourrez les personnaliser selon vos besoins. Cela vous fera gagner beaucoup de temps au cours de la phase de développement.

Au moment où ce document est rédigé, IoT Central propose un essai de sept jours et vous pouvez voir la tarification de ce service ici : https://azure.microsoft.com/pricing/details/iot-central/?rtc=1.

Azure IoT Central est une aubaine pour toutes les organisations qui développent des applications IoT.

Résumé

L'IoT est l'une des plus grandes technologies d'avenir de cette décennie, qui a déjà un effet perturbateur sur les secteurs. Ce qui semblait irréalisable auparavant est tout à coup devenu possible.

Au cours de ce chapitre, nous avons exploré IoT Hub et discuté de la livraison de solutions IoT au client de manière plus rapide et plus abordable qu'avec d'autres solutions. Nous avons également vu comment l'IoT permet d'accélérer l'ensemble du cycle de vie du développement et un lancement plus rapide sur le marché pour les entreprises. En dernier lieu, vous avez découvert Azure IoT Edge et Azure IoT Central.

Pour vous aider à analyser efficacement les volumes de données en constante évolution, nous aborderons Azure Synapse Analytics dans le chapitre suivant.

18

Azure Synapse Analytics pour les architectes

Azure Synapse Analytics est une évolution révolutionnaire d'Azure SQL Data Warehouse. Azure Synapse est un service d'analyse de données entièrement géré et intégré qui associe le stockage de données, l'intégration de données et le traitement des Big Data en un seul service qui permet d'accélérer délais d'extraction des informations.

Dans ce chapitre, nous explorerons Azure Synapse Analytics, en abordant les sujets suivants :

- Présentation d'Azure Synapse Analytics
- Présentation des espaces de travail Synapse et de Synapse Studio
- Migrer les systèmes hérités existants vers Azure Synapse Analytics
- Migrer les schémas d'entrepôt de données et les données existants vers Azure Synapse Analytics
- Recréer des processus ETL évolutifs à l'aide d'Azure Data Factory
- Problèmes de migration courants et solutions
- Considérations en matière de sécurité
- Outils simplifiant la migration vers Azure Synapse Analytics

Azure Synapse Analytics

Aujourd'hui, avec un stockage peu coûteux et des capacités de stockage élastique élevées, les entreprises accumulent plus de données que jamais. L'architecture d'une solution pour analyser ces volumes considérables de données afin de fournir des informations utiles sur une entreprise peut être un défi. L'un des obstacles auxquels les entreprises sont confrontées aujourd'hui est la nécessité de gérer et d'entretenir deux types de systèmes d'analyse :

- **Entrepôts de données** : ils fournissent des informations essentielles sur l'entreprise.
- **Lacs de données** : ils fournissent des informations exploitables sur les clients, les produits, les collaborateurs et les processus par l'intermédiaire de diverses méthodes d'analyse.

Ces deux systèmes d'analyse sont essentiels aux entreprises et pourtant ils fonctionnent indépendamment l'un de l'autre. Pendant ce temps, les entreprises ont besoin d'accéder à des renseignements exploitables issues de toutes leurs données organisationnelles afin de rester compétitives et d'innover dans leurs processus dans le but d'obtenir de meilleurs résultats.

Pour les architectes qui ont besoin de créer leurs propres pipelines de données end-to-end, il est nécessaire de procéder comme suit :

1. Ingérer des données provenant de diverses sources de données
2. Charger toutes ces sources de données dans un lac de données pour traitement ultérieur
3. Effectuer le nettoyage des données sur un éventail de structures et de types de données différents
4. Préparer, transformer et modéliser les données
5. Fournir les données nettoyées à des milliers d'utilisateurs à l'aide d'outils et d'applications BI

Jusqu'à présent, chacune de ces étapes avait besoin d'un outil différent. Il va sans dire que, avec un si grand nombre de services, d'applications et d'outils différents disponibles sur le marché, il peut être difficile de faire les bons choix.

Il existe de nombreux services qui ingèrent, chargent, préparent et fournissent les données. Il existe également de multiples services qui permettent de nettoyer les données en fonction du langage choisi par le développeur. De plus, certains développeurs préfèrent utiliser Spark, et d'autres SQL, tandis que d'autres encore choisissent d'utiliser des environnements sans code pour transformer les données.

Même après avoir sélectionné la collection d'outils qui semble appropriée, il existe souvent une courbe d'apprentissage abrupte pour ces outils. En outre, les architectes rencontrent parfois des défis logistiques imprévus pour gérer un pipeline de données sur des plateformes et des langages dissemblables en raison d'incompatibilités. Avec un tel éventail de problèmes, la mise en œuvre et la maintenance d'une plateforme d'analyse de données du Cloud peut s'avérer une tâche difficile.

Azure Synapse Analytics résout ces problèmes et bien d'autres. Il simplifie l'ensemble du modèle d'entrepôt de données moderne, en permettant aux architectes de se concentrer sur la création de solutions d'analyse end-to-end au sein d'un environnement unifié.

Un scénario courant pour les architectes

L'un des scénarios les plus courants auxquels un architecte est confronté est d'avoir à élaborer un plan de migration des solutions existantes d'entrepôt de données héritées vers une solution d'analyse d'entreprise moderne. Avec son évolutivité illimitée et son expérience unifiée, Azure Synapse est devenu l'un des meilleurs choix pour de nombreux architectes. Plus loin dans ce chapitre, nous aborderons également des considérations architecturales courantes pour la migration d'une solution d'entrepôt de données héritée existante vers Azure Synapse Analytics.

Dans la section suivante, nous allons fournir une présentation technique des principales fonctionnalités d'Azure Synapse Analytics. Les architectes qui découvrent l'écosystème de Azure Synapse pourront l'explorer en détail dans ce chapitre.

Présentation d'Azure Synapse Analytics

Azure Synapse Analytics permet aux professionnels des données de créer des solutions d'analyse end-to-end tout en s'appuyant sur une expérience unifiée. Il offre des fonctionnalités riches pour les développeurs SQL, des requêtes sans serveur à la demande, le support de Machine Learning, la possibilité d'intégrer Spark en mode natif, les blocs-notes collaboratifs et l'intégration de données au sein d'un seul service. Les développeurs peuvent choisir parmi une variété de langages pris en charge (par exemple, C#, SQL, Scala et Python) à l'aide de différents moteurs.

Voici quelques-unes des principales fonctionnalités d'Azure Synapse Analytics :

- SQL Analytics avec des pools (totalement provisionnés) et à la demande (sans serveur)
- Spark avec une prise en charge complète de C#, SQL, Scala et Python
- Flux de données avec une expérience de transformation du Big Data sans code
- Intégration et orchestration des données pour intégrer vos données et rendre opérationnel l'ensemble de votre développement de code
- Une version Cloud native du **traitement transactionnel/analytique hybride** (**HTAP**), fournie par le lien Azure Synapse.

Pour vous permettre d'accéder à toutes les fonctionnalités susmentionnées, Azure Synapse Studio propose une interface utilisateur web unifiée et unique.

Ce service de données intégré unique est avantageux pour les entreprises car il accélère la diffusion de la BI, de l'IA, du Machine Learning, de l'Internet des objets et des applications intelligentes.

Azure Synapse Analytics peut retrouver et fournir des informations à partir de toutes vos données stockées résidant dans l'entrepôt de données et dans les systèmes d'analyse de Big Data à des vitesses fulgurantes. Il permet aux professionnels des données d'utiliser des langages connus, comme le SQL, pour interroger des bases de données relationnelles et non relationnelles à l'échelle du pétaoctet. De plus, des fonctions avancées telles que la simultanéité illimitée, la gestion intelligente des charges de travail et l'isolation des charges de travail permettent d'optimiser les performances de toutes les requêtes pour les charges de travail stratégiques.

Qu'est-ce que l'isolation des charges de travail ?

L'une des principales caractéristiques de l'exécution des entrepôts de données d'entreprise à l'échelle est l'isolation des charges de travail. Il s'agit de la possibilité de garantir les réservations de ressources au sein d'un cluster de calcul, afin que plusieurs équipes puissent travailler sur les données sans se mettre en travers de l'autre, comme illustré à la *figure 18.1* :

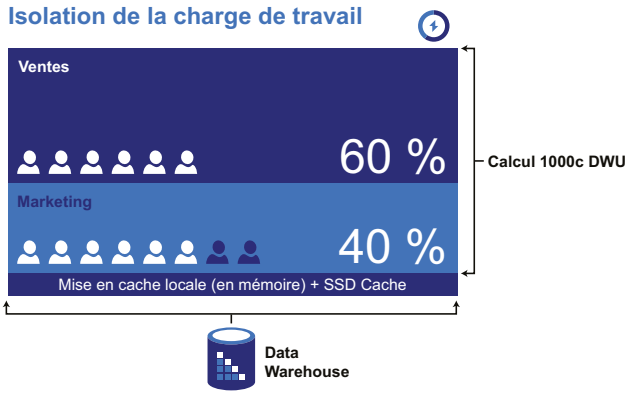


Figure 18.1 : Exemple d'isolation des charges de travail

Vous pouvez créer des groupes de charges de travail au sein d'un cluster en définissant quelques seuils simples. Ceux-ci sont ajustés automatiquement en fonction de la charge de travail et du cluster, mais ils garantissent toujours une expérience de qualité pour les utilisateurs exécutant les charges de travail. Consultez https://techcommunity.microsoft.com/t5/data-architecture-blog/configuring-workload-isolation-in-azure-synapse-analytics/ba-p/1201739 pour en savoir plus sur la configuration de l'isolation des charges de travail dans Azure Synapse Analytics.

Pour apprécier pleinement les avantages d'Azure Synapse, nous commencerons par examiner les espaces de travail Synapse et Synapse Studio.

Présentation des espaces de travail Synapse et de Synapse Studio

Au cœur d'Azure Synapse se trouve l'espace de travail. L'espace de travail est la ressource principale qui comprend votre solution d'analyse dans un entrepôt de données. L'espace de travail Synapse prend en charge à la fois le traitement relationnel et le traitement du Big Data.

Azure Synapse fournit une expérience d'interface utilisateur web unifiée pour la préparation, la gestion et l'entreposage des données, l'analyse du Big Data, la BI et les tâches d'IA. Il s'agit de Synapse Studio. Avec les espaces de travail Synapse, Synapse Studio constitue l'environnement idéal pour les ingénieurs de données et les data scientists qui peuvent y partager leurs solutions analytiques et collaborer, comme illustré à la *figure* 18.2 :

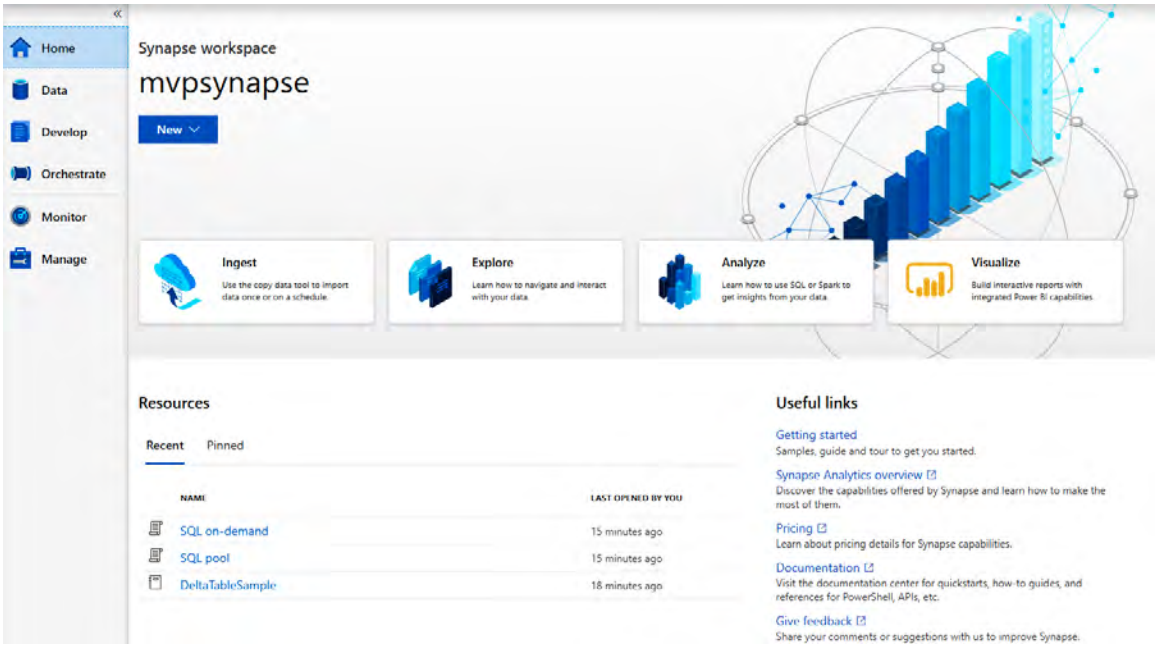


Figure 18.2 : Un espace de travail Synapse dans Azure Synapse Studio

Les sections suivantes mettront en évidence les capacités, les fonctions clés, les détails de la plateforme et les services aux utilisateurs finaux des espaces de travail Synapse et de Synapse Studio :

Fonctionnalités :

- Un entrepôt de données rapide, hautement élastique et sécurisé avec des performances et une sécurité de pointe
- Possibilité d'explorer Azure Data Lake Storage et son entrepôt de données en utilisant la syntaxe T-SQL familière avec les requêtes à la demande (sans serveur) et SQL
- Apache Spark intégré à Azure Machine Learning
- Intégration de données hybrides pour accélérer l'ingestion et la mise en œuvre de données du processus analytique (ingérer, préparer, transformer et fournir)
- Génération de rapports d'activité et service avec l'intégration de Power BI

Fonctions clés :

- Créez et exploitez des pipelines pour l'ingestion et l'organisation de données.
- Explorez directement les données dans votre lac de données Azure Data Lake Storage ou votre entrepôt de données, ainsi que toute connexion externe à l'espace de travail, en utilisant Synapse Studio.
- Écrivez du code en utilisant Notebooks et des éditeurs de requêtes T-SQL.
- Outil de transformation de données sans code, si vous préférez ne pas écrire votre propre code.
- Surveillez, sécurisez et gérez vos espaces de travail sans quitter l'environnement.
- Expérience de développement basé sur le Web pour l'ensemble de la solution d'analyse.
- La fonctionnalité de sauvegarde et de restauration du pool Azure Synapse SQL permet de créer des points de restauration pour faciliter la récupération ou la copie d'un entrepôt de données dans un état antérieur.
- Possibilité d'exécuter des requêtes T-SQL simultanées via des pools SQL sur plusieurs pétaoctets de données pour fournir des outils et applications BI.
- SQL à la demande offre des requêtes SQL sans serveur pour faciliter l'exploration et l'analyse des données dans Azure Data Lake Storage sans aucune configuration ni maintenance des infrastructures.
- Répond à l'éventail complet des besoins analytiques, de l'ingénierie des données à la science des données, en utilisant une variété de langages, tels que Python, Scala, C#, et Spark SQL.
- Les pools Spark, qui allègent la configuration et la maintenance complexes des clusters et simplifient le développement des applications Spark et l'utilisation de Spark Notebooks.

- Offre une intégration profonde entre Spark et SQL, permettant aux ingénieurs de données de préparer les données dans Spark, d'écrire les résultats traités dans SQL Pool et d'utiliser n'importe quelle combinaison de Spark avec SQL pour l'ingénierie et l'analyse des données, avec la prise en charge intégrée pour Azure Machine Learning.
- Capacité d'intégration de données hybride hautement évolutive qui accélère l'ingestion et la mise en œuvre des données grâce à des pipelines de données automatisés.
- Fournit un service intégré et fluide avec une sécurité, un déploiement, une surveillance et une facturation unifiés.

Plateforme

- Prise en charge des modèles de calcul provisionnés et sans serveur. Les exemples de calculs provisionnés comprennent les calculs SQL et les calculs Spark.
- Ces calculs provisionnés permettent aux équipes de segmenter leurs ressources informatiques afin de contrôler les coûts et l'utilisation pour mieux s'aligner sur leur structure organisationnelle.
- En revanche, les calculs sans serveur permettent aux équipes d'utiliser le service à la demande sans provisionnement ni gestion de l'infrastructure sous-jacente.
- Intégration profonde entre les moteurs Spark et SQL.

Dans la section suivante, nous allons aborder les autres fonctionnalités d'Azure Synapse, notamment Apache Spark pour Synapse, Synapse SQL, SQL à la demande, les pipelines Synapse et Azure Synapse Link pour Cosmos DB.

Apache Spark pour Synapse

Pour les clients qui veulent Apache Spark, Azure Synapse dispose d'une prise en charge de départ via Azure Databricks et est entièrement géré par Azure. La dernière version d'Apache Spark sera automatiquement mise à la disposition des utilisateurs, avec tous les correctifs de sécurité. Vous pouvez créer rapidement des blocs-notes avec le langage qui vous convient, par exemple Python, Scala, Spark SQL et .NET pour Spark.

Si vous utilisez Spark dans Azure Synapse Analytics, il est fourni en tant que solution SaaS. Par exemple, vous pouvez utiliser Spark sans configurer ni gérer vos propres services, comme un réseau virtuel. Azure Synapse Analytics s'occupera de l'infrastructure sous-jacente pour vous. Ceci vous permet d'utiliser Spark immédiatement dans votre environnement Azure Synapse Analytics.

Dans la section suivante, nous explorerons Synapse SQL.

Synapse SQL

Synapse SQL permet l'utilisation de T-SQL pour interroger et analyser les données. Vous pouvez choisir parmi deux modèles :

1. Modèle entièrement configuré
2. Modèle SQL à la demande (sans serveur)

SQL à la demande

SQL à la demande fournit des requêtes SQL sans serveur. Ceci facilite l'exploration et l'analyse des données dans Azure Data Lake Storage sans aucune installation ni maintenance de l'infrastructure :

IT classique	IaaS	PaaS	Infrastructure sans serveur	SaaS
Application	Application	Application	Application	Application
Données	Données	Données	Données	Données
Durée	Durée	Durée	Durée	Durée
Intergiciel	Intergiciel	Intergiciel	Intergiciel	Intergiciel
Système d'exploitation	Système d'exploitation	Système d'exploitation	Système d'exploitation	Système d'exploitation
Virtualisation	Virtualisation	Virtualisation	Virtualisation	Virtualisation
Serveurs	Serveurs	Serveurs	Serveurs	Serveurs
Stockage	Stockage	Stockage	Stockage	Stockage
Mise en réseau	Mise en réseau	Mise en réseau	Mise en réseau	Mise en réseau

Vous gérez

Tableau 18.1 : Comparaison entre différentes infrastructures

Fonctions clés :

- Les analystes peuvent se concentrer sur l'analyse des données sans se soucier de la gestion de l'infrastructure.
- Les clients peuvent bénéficier d'un modèle de tarification simple et flexible, car ils ne paient que ce qu'ils utilisent.

- SQL On-Demand utilise la syntaxe familière du langage T-SQL et le meilleur optimiseur de requêtes SQL du marché. L'optimiseur de requêtes SQL est le cerveau qui se cache derrière le moteur de requête.
- Vous pouvez facilement mettre à l'échelle vos calculs et votre stockage, indépendamment l'un de l'autre, au fur et à mesure que vos besoins augmentent.
- Intégration transparente avec SQL Analytics Pool et Spark via la synchronisation des métadonnées et des connecteurs natifs.

Pipelines Synapse

Les pipelines Synapse permettent aux développeurs de créer des workflows end-to-end pour le déplacement des données et les scénarios de traitement des données. Azure Synapse Analytics utilise la technologie **Azure Data Factory** (**ADF**) pour fournir des fonctions d'intégration de données. Les principales fonctions de l'ADF qui sont essentielles au pipeline de l'entrepôt de données moderne sont disponibles dans Azure Synapse Analytics. Toutes ces fonctions sont entourées d'un modèle de sécurité commun, le **contrôle d'accès en fonction du rôle** (**RBAC**), dans l'espace de travail Azure Synapse Analytics.

La *figure 18.3* montre un exemple de pipeline de données et des activités d'ADF qui sont directement intégrées dans l'environnement Azure Synapse Analytics :

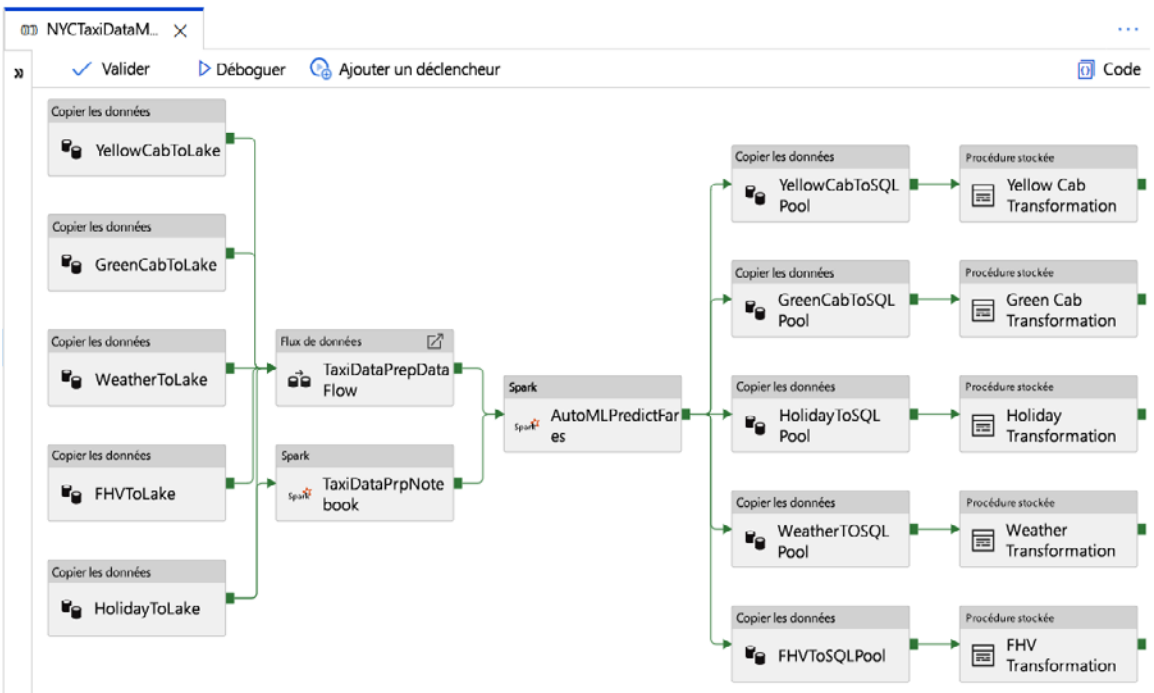


Figure 18.3 : Pipelines de données dans Azure Synapse Analytics

Fonctions clés :

- Services de plateforme intégrée pour la gestion, la sécurité, la surveillance et la gestion des métadonnées.
- Intégration native entre Spark et SQL. Utilisation d'une seule ligne de code pour lire et écrire avec Spark depuis/vers SQL Analytics.
- Possibilité de créer une table Spark et de l'interroger instantanément avec SQL Analytics sans définir de schéma.
- Environnement « sans clé ». Avec l'authentification unique et l'authentification par transmission AAD, aucune clé ni aucun login n'est nécessaire pour interagir avec **Azure Data Lake Storage** (**ADLS**)/les bases de données.

Dans la section suivante, nous allons couvrir Azure Synapse Link pour Cosmos DB.

Azure Synapse Link pour Cosmos DB

Azure Synapse Link est une version Cloud native de HTAP. C'est une extension d'Azure Synapse. Comme nous l'avons déjà vu, Azure Synapse est un service géré unique qui permet d'effectuer des analyses sur les lacs de données et les entrepôts de données, à l'aide d'un calcul sans serveur et configuré. Avec Azure Synapse Link, cette portée peut également être étendue aux sources de données opérationnelles.

Azure Synapse Link élimine le goulot d'étranglement que l'on retrouve dans les systèmes opérationnels et analytiques traditionnels. Azure Synapse rend cela possible en séparant le calcul du stockage sur l'ensemble de ses services de données. Sur le plan transactionnel, Cosmos DB est un service de base de données multi-modèles hautes performances, géo-répliqué. Du point de vue des analyses de données Azure Synapse offre une évolutivité illimitée. Vous pouvez mettre à l'échelle les ressources pour les transactions et les analyses indépendamment. Ces solutions permettent faire du HTAP natif du Cloud une réalité. Dès que l'utilisateur indique les données de Cosmos DB qu'il souhaite mettre à disposition pour l'analyse, les données deviennent disponibles dans Synapse. Il prend les données opérationnelles que vous souhaitez analyser et en gère automatiquement une version en colonnes orientée analyse. En conséquence, toutes les modifications apportées aux données opérationnelles dans Cosmos DB sont continuellement mises à jour en fonction des données de Link et de Synapse.

Azure Synapse Link présente surtout l'avantage d'atténuer le besoin en traitement par lots planifié ou la nécessité de créer et de gérer des pipelines opérationnels.

Comme nous l'avons mentionné précédemment, Azure Synapse est la plateforme la plus choisie par les architectes pour la migration des solutions d'entrepôt de données héritées existantes vers une solution d'analyse d'entreprise moderne. Dans la section suivante, nous aborderons également des considérations architecturales courantes pour la migration d'une solution d'entrepôt de données héritée existante vers Azure Synapse Analytics.

Migrer les systèmes hérités existants vers Azure Synapse Analytics

À l'heure actuelle, de nombreuses organisations migrent leurs solutions d'entrepôt de données héritées vers Azure Synapse Analytics afin de bénéficier de la haute disponibilité, de la sécurité, de la rapidité, de l'évolutivité, de la rentabilité et des performances d'Azure Synapse.

Pour les entreprises qui exécutent des systèmes d'entrepôt de données hérités tels que Netezza, la situation est encore plus désastreuse, car IBM a annoncé la fin du support de Netezza (https://www.ibm.com/support/pages/end-support-dates-netezza-5200-netezza-8x50z-series-and-netezza-10000-series-appliances).

Il y a plusieurs décennies, certaines entreprises ont choisi Netezza pour gérer et analyser les volumes de données importants. Aujourd'hui, à mesure que les technologies évoluent, les atouts d'une solution de stockage de données dans le Cloud l'emportent largement sur ceux d'une solution sur site. Azure Synapse est un service d'analyse de données basé sur le Cloud sans limite, avec un délai inégalé pour obtenir des informations qui accélèrent la livraison de BI, d'IA et d'applications intelligentes pour les entreprises. Grâce à sa capacité multicluster et à son architecture de calcul et de stockage séparée, Azure Synapse peut être redimensionné instantanément, dans une mesure qui n'était tout simplement pas envisageable auparavant avec des systèmes hérités comme Netezza.

Cette section couvre les considérations architecturales et la méthodologie globale à suivre pour planifier, préparer et effectuer une migration réussie d'un système de stockage de données hérité vers Azure Synapse Analytics. Chaque fois que cela est approprié, des exemples spécifiques et des références à Netezza seront donnés. Il ne s'agit pas d'un chapitre détaillé sur la migration, mais plutôt d'une présentation pratique facilitant la planification de l'opération et la portée du projet.

Ce chapitre répertorie également plusieurs problèmes de migration courants et des solutions pour tenter de les résoudre. Il contient des détails techniques sur les différences entre Netezza et Azure Synapse Analytics. Ils doivent être pris en compte dans le cadre de votre plan de migration.

Pourquoi migrer votre entrepôt de données hérité vers Azure Synapse Analytics ?

En migrant vers Azure Synapse Analytics, les entreprises disposant de systèmes de stockage de données hérités peuvent tirer parti des dernières innovations en matière de technologies Cloud. Elles peuvent aussi déléguer des tâches comme la maintenance de l'infrastructure et la mise à niveau de la plateforme à Azure.

Les clients qui ont migré vers Azure Synapse profitent déjà de nombreux ses avantages, dont certains sont répertoriés ci-dessous.

Performance

Azure Synapse Analytics offre des performances inégalées en termes de base de données relationnelle via l'utilisation de techniques comme le **traitement hautement parallèle** (**MPP**) et la mise en cache automatique en mémoire. Pour plus d'informations, consultez l'architecture Azure Synapse Analytics (https://docs.microsoft.com/azure/synapse-analytics/sql-data-warehouse/massively-parallel-processing-mpp-architecture).

Rapidité

Le stockage de données est très exigeant en termes de processus. Il implique l'ingestion, la transformation, le nettoyage, l'agrégation et l'intégration des données, puis la production de visualisations et de rapports. Les nombreux processus visant à transférer des données de leur source d'origine vers un entrepôt de données sont à la fois complexes et interdépendants. Le moindre goulot d'étranglement risque de ralentir le pipeline entier, et un pic inattendu du volume de données renforce le besoin de réactivité. Lorsque la vitesse se révèle cruciale, Azure Synapse Analytics répond aux exigences de traitement rapide.

Sécurité renforcée et conformité améliorée

Azure est une plateforme Cloud sécurisée et hautement évolutive, disponible dans le monde entier. Elle offre de nombreuses fonctionnalités de sécurité, y compris Azure Active Directory, RBAC, les identités gérées et les points de terminaison privés gérés. Dans la mesure où le service Azure Synapse Analytics fait partie intégrante de l'écosystème Azure, il réunit tous les avantages que nous venons d'énumérer.

Élasticité et rentabilité

Dans un entrepôt de données, les demandes de traitement des charges de travail peuvent fluctuer. Parfois, les variations sont considérables, passant de pics à des planchers. Par exemple, on peut observer des pics soudains dans le volume des données de vente en période de fêtes. De par l'élasticité du Cloud, Azure Synapse peut augmenter ou diminuer rapidement sa capacité en fonction de la demande, sans le moindre impact sur la disponibilité, la stabilité, les performances et la sécurité de l'infrastructure. Et vous serez ravi d'apprendre que vous payez uniquement pour votre utilisation réelle.

Infrastructure gérée

En éliminant la surcharge de travail liée à la gestion et au fonctionnement des datacenters au profit de l'entrepôt de données, les entreprises peuvent réaffecter des ressources précieuses là où elles apportent une valeur ajoutée. Elles peuvent aussi se concentrer sur l'utilisation de l'entrepôt de données pour fournir les informations les plus utiles. Cela réduit le coût total global de possession, et permet de mieux contrôler les dépenses d'exploitation.

Évolutivité

Le volume de données stockées dans un entrepôt augmente généralement au fil du temps, et du fait de l'historique collecté. Vous pouvez dimensionner Azure Synapse Analytics pour l'adapter à cette croissance en ajoutant progressivement des ressources à mesure que les données et les charges de travail se multiplient.

Économies

L'exécution d'un datacenter hérité sur site est coûteuse (serveurs et matériel, mise en réseau, espace physique, électricité, refroidissement et personnel). Il est possible de réduire considérablement ces dépenses grâce à Azure Synapse Analytics. Avec la séparation des couches de calcul et de stockage, Azure Synapse offre un rapport prix-performance très intéressant.

Azure Synapse Analytics vous permet vraiment de profiter de l'évolutivité du Cloud via le paiement à l'utilisation, sans vous soucier de reconfigurer quoi que ce soit lorsque le nombre de données ou de charges de travail augmente.

Maintenant que vous avez vu pourquoi il est bénéfique de migrer vers Azure Synapse Analytics, nous allons aborder le processus de migration.

Processus de migration en trois étapes

Pour mener à bien un projet de migration de données, il faut commencer par rédiger un plan précis. Un plan efficace répertorie les nombreux points à prendre en compte, en prêtant une attention particulière à l'architecture et à la préparation des données. Vous trouverez ci-après un plan de migration en trois étapes.

Préparation

- Définir la portée de la migration (composants à migrer)
- Dresser l'inventaire des données et des processus à migrer
- Définir les modifications du modèle de données (le cas échéant)
- Définir le mécanisme d'extraction des données source
- Identifier les outils et services Azure (et tiers) à utiliser
- Former le personnel bien à l'avance sur la nouvelle plateforme
- Configurer la plateforme cible Azure

Migration

- Commencer à petite échelle et simplement
- Automatiser les processus chaque fois que c'est possible
- Tirer parti des fonctions et des outils intégrés à Azure pour simplifier la migration
- Migrer les métadonnées associées aux tables et aux vues
- Migrer les données historiques à conserver
- Migrer ou refactoriser les procédures stockées et les processus métier
- Migrer ou refactoriser les processus de chargement incrémentiel ETL/ELT

Post-migration

- Piloter et documenter toutes les étapes du processus
- Mettre à profit l'expérience acquise afin d'établir un modèle pour les migrations ultérieures
- Remanier le modèle de données si nécessaire
- Tester les applications et les outils de requête
- Évaluer et optimiser les performances de requête

Nous allons maintenant aborder les deux types de stratégies de migration disponibles.

Deux types de stratégies de migration

Les architectes doivent commencer à planifier la migration en évaluant l'entrepôt de données existant pour déterminer la stratégie qui leur convient le mieux. Nous allons voir les deux types de stratégies de migration qui sont à votre disposition.

Stratégie de réplication « lift-and-shift »

Dans le cadre de la stratégie de réplication « lift-and-shift », le modèle de données existant reste intact lors de la migration vers la nouvelle plateforme Azure Synapse Analytics. Cela permet de réduire les risques et les délais de migration tout en limitant la portée des modifications au maximum.

La réplication « lift-and-shift » est parfaitement adaptée aux environnements d'entrepôt de données hérités comme Netezza, lorsque l'un des conditions suivantes s'applique :

- Un seul mini-data warehouse doit être migré.
- Les données se présentent déjà dans un schéma en étoile ou en flocon bien conçu.
- Les pressions sur les coûts et les délais s'intensifient pour passer à un environnement Cloud moderne.

Stratégie de refonte

Dans les scénarios où l'entrepôt de données hérité a évolué au fil du temps, il peut être essentiel de le réorganiser pour conserver des performances optimales ou prendre en charge de nouveaux types de données. Cela peut revenir à modifier le modèle de données sous-jacent.

Pour minimiser les risques, il est conseillé de procéder à la réplication « lift-and-shift » avant de moderniser progressivement le modèle de données de l'entrepôt de données sur Azure Synapse Analytics en suivant la stratégie de refonte. Un changement complet du modèle de données aura pour effet d'augmenter les risques en raison de son impact sur les tâches ETL de l'entrepôt de données source et des mini-data warehouses en aval.

Dans la section suivante, nous vous proposerons des recommandations qui vous permettront de réduire la complexité de votre système de stockage de données hérité avant la migration.

Réduire la complexité de votre entrepôt de données hérité avant la migration

Dans la section précédente, nous avons présenté les deux stratégies de migration. Respectez cette bonne pratique : au cours de l'évaluation initiale, prenez soin de déterminer toutes les options possibles pour simplifier votre entrepôt de données existant, puis de les documenter. L'objectif est de réduire la complexité de votre système de stockage de données hérité au préalable en vue de faciliter la migration.

Voici quelques recommandations pour réduire la complexité de votre entrepôt de données hérité :

- **Supprimez et archivez les tables inutilisées avant de migrer** : évitez de migrer les données qui ne sont plus utilisées. Cela permettra de réduire le volume global de données à migrer.
- **Convertissez les mini-data warehouses physiques en mini-data warehouses virtuels** : réduisez autant que possible ce qui doit être migré, limitez le coût total de possession et améliorez votre agilité.

Dans la section suivante, nous verrons pourquoi il est préférable d'envisager de convertir un mini-data warehouse physique en mini-data warehouse virtuel.

Convertir les mini-data warehouses physiques en mini-data warehouses virtuels

Avant de migrer votre entrepôt de données hérité, envisagez de convertir vos mini-data warehouses physiques actuels en mini-data warehouses virtuels. Grâce aux mini-data warehouses virtuels, vous pouvez éliminer des magasins de données physiques et des tâches ETL, sans perdre la moindre fonctionnalité avant la migration. Les objectifs sont multiples : réduire le nombre de magasins de données à migrer, les copies des données et le coût total de possession, tout en gagnant en agilité. Pour y parvenir, il faut transformer les mini-data warehouses physiques en mini-data warehouses virtuels avant de migrer votre entrepôt de données. Nous pouvons considérer cela comme une étape de modernisation de l'entrepôt de données préalable à la migration.

Inconvénients des mini-data warehouses physiques

- Plusieurs copies des mêmes données
- Coût total de possession élevé
- Modifications difficiles en raison de leur impact sur les tâches ETL

Avantages des mini-data warehouses virtuels

- Architecture d'entrepôt de données simplifiée
- Inutile de stocker des copies de données
- Plus d'agilité
- Faible coût total de possession
- Optimisation de type pushdown pour exploiter tout le potentiel d'Azure Synapse Analytics
- Modifications facilitées
- Masquage aisé des données sensibles

Dans la section suivante, nous aborderons la migration des schémas d'entrepôt de données existants vers Azure Synapse Analytics.

Migrer le schéma d'entrepôt de données existant vers Azure Synapse Analytics

La migration des schémas d'un entrepôt de données hérité existant consiste à migrer les tables intermédiaires existantes, l'entrepôt de données hérité et le schéma du mini-data warehouse dépendant.

Pour mieux saisir l'ampleur et la portée de la migration du schéma, nous vous conseillons de dresser l'inventaire de l'entrepôt de données hérité et du mini-data warehouse existant.

Cette check-list vous aidera à recueillir les informations nécessaires :

- Nombre de lignes
- Taille des données intermédiaires, de l'entrepôt de données et du mini-data warehouse : tables et index
- Taux de compression des données
- Configuration matérielle actuelle
- Tables (y compris les partitions) : identifiez les tables de petite dimension
- Types de données
- Vues
- Index
- Dépendances d'objets
- Utilisation des objets

- Fonctions : fonctions prêtes à l'emploi et **fonctions définies par l'utilisateur (UDF)**
- Procédures stockées
- Exigences en termes d'évolutivité
- Projections de croissance
- Exigences de la charge de travail : utilisateurs simultanés

Dès que vous aurez fini l'inventaire, vous pourrez déterminer la portée du schéma que vous souhaitez migrer. Concrètement, il existe quatre options pour évaluer l'étendue de la migration du schéma de votre entrepôt de données hérité :

1. Migrer un mini-data warehouse à la fois :

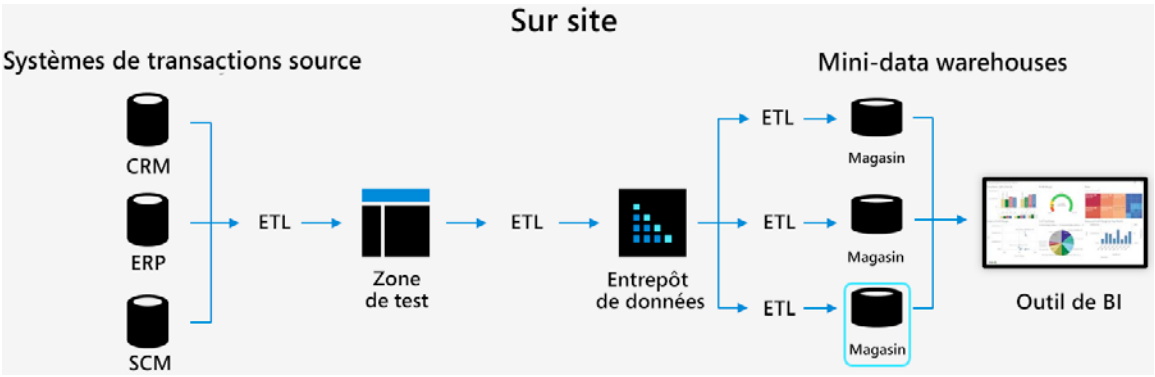


Figure 18.4 : Migration d'un mini-data warehouse à la fois

2. Migrer tous les mini-data warehouses en même temps, puis l'entrepôt de données :

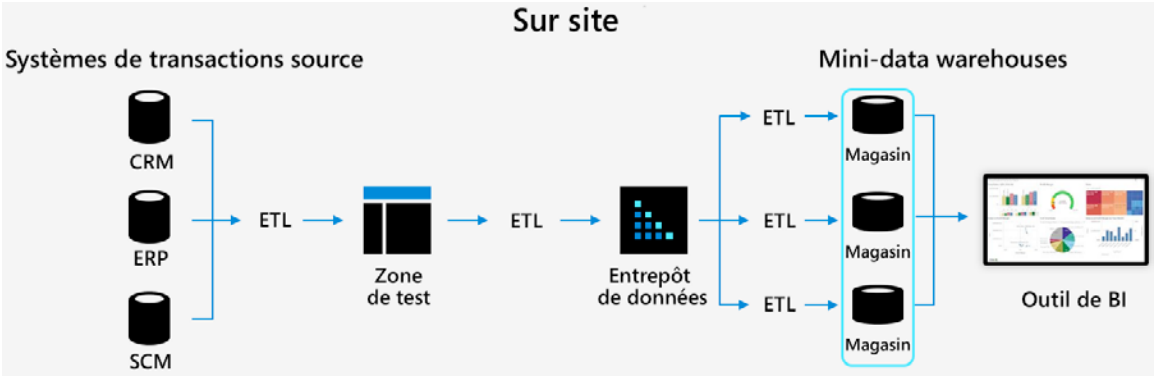


Figure 18.5 : Migration de tous les mini-data warehouses en même temps, puis l'entrepôt de données

3. Migrer simultanément l'entrepôt de données et la zone intermédiaire :

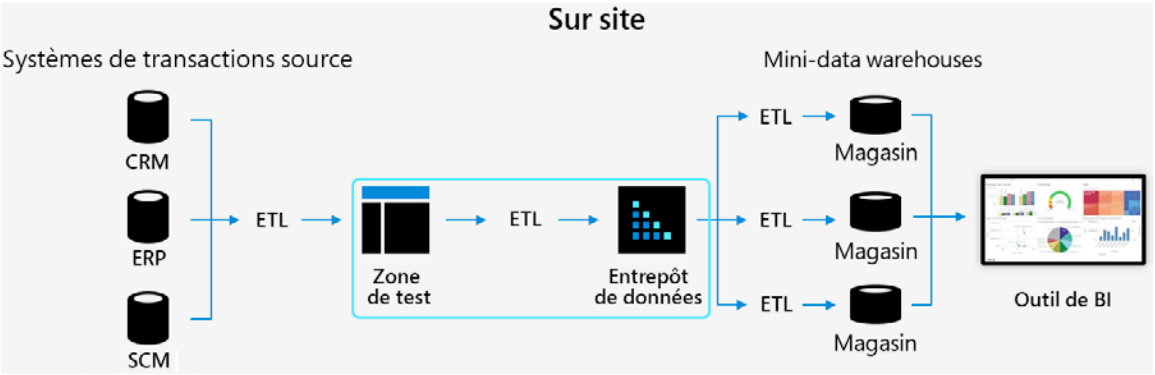


Figure 18.6 : Migration de l'entrepôt de données et de la zone intermédiaire

4. Tout migrer en même temps :

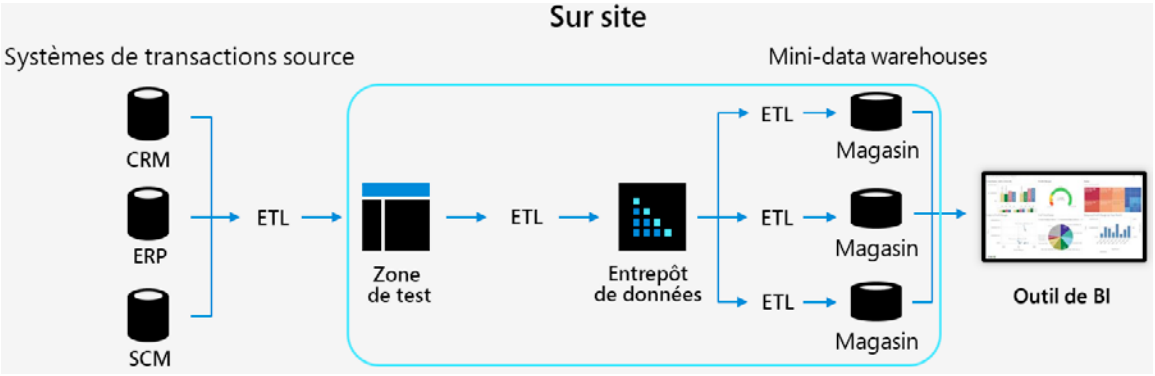


Figure 18.7 : Migration de tout en même temps

Allez à l'essentiel lorsque vous étudiez ces options. L'objectif est de choisir la conception de BD physique qui équivaut à votre système de stockage de données actuel ou le surpasse en termes de performances, tout en réduisant les coûts (dans la mesure du possible).

Pour résumer, voici quelques conseils pour migrer le schéma :

- Évitez de migrer des objets ou des processus superflus.
- Envisagez d'utiliser des mini-data warehouses virtuels pour réduire le nombre de mini-data warehouses physiques (ou les remplacer tous).
- Automatisez les processus chaque fois que cela est possible. La mise en œuvre de DataOps doit être envisagée parallèlement à la migration vers Azure synapse.
- Utilisez les métadonnées issues des tables du catalogue système dans l'entrepôt de données hérité pour générer des instructions **Data Definition Language** (**DDL**) pour Azure Synapse Analytics.
- Appliquez les modifications requises au modèle de données ou les optimisations de mappage de données sur Azure Synapse Analytics.

Dans la section suivante, nous aborderons la migration des données historiques d'un entrepôt de données hérité vers Azure Synapse Analytics.

Migrer des données historiques et des processus ETL de votre entrepôt de données hérité vers Azure Synapse Analytics

Après avoir déterminé la portée de la migration du schéma, vous pouvez décider comment migrer les données historiques.

Voici les étapes à suivre pour migrer les données historiques :

1. Créer des tables cible sur Azure Synapse Analytics
2. Migrer les données historiques existantes
3. Migrer toutes les fonctions et procédures stockées requises
4. Migrer la zone intermédiaire de chargement incrémentiel (ETL/ELT) et les processus pour les données entrantes
5. Appliquer toutes les options de réglage des performances requises

Le tableau 18.2 décrit les quatre options de migration de données, ainsi que leurs avantages et inconvénients :

Options de migration	Avantages	Inconvénients
Migrer les données des mini-data warehouses, puis les données de l'entrepôt de données	• La migration des données d'un seul mini-data warehouse à la fois est une approche progressive à faible risque. Une preuve d'analyse de rentabilté sera ainsi apportée plus rapidement aux utilisateurs des analyses départementales. • La migration ultérieure des processus ETL se limite aux données stockées dans les mini-data warehouses dépendants migrés.	• Jusqu'au terme de la migration, certaines données seront stockées à la fois sur site et sur Azure. • Pour faire passer le traitement ETL de l'entrepôt de données aux mini-data warehouses, il faudra relier le pare-feu et désigner Azure Synapse en tant que cible.
Migrer les données de l'entrepôt de données, puis les données des mini-data warehouses	• Toutes les données historiques de l'entrepôt de données sont migrées.	• Le fait de laisser les mini-data warehouses dépendants sur site n'est pas idéal, car les processus ETL devront renvoyer les données au datacenter. • Il n'y a pas d'opportunité réelle de migration progressive des données.
Migrer les données de l'entrepôt de données et des mini-data warehouses en même temps	• Toutes les données sont migrées en une seule fois.	• Les risques sont potentiellement plus élevés. • Selon toute vraisemblance, il faudra aussi migrer les processus ETL ensemble.
Convertir les mini-data warehouses physiques en mini-data warehouses virtuels, puis migrer uniquement l'entrepôt de données	• Il ne faut migrer aucun mini-data warehouse. • Inutile de migrer le moindre processus ETL de l'entrepôt de données aux mini-data warehouses. • Il faut migrer uniquement les données de l'entrepôt de données. • Il y a moins de copies de données. • On ne perd aucune fonctionnalité. • Faible coût total de possession. • Plus d'agilité. • L'architecture de données globale est plus simple. • Cette opération est possible avec des vues dans Azure Synapse.	• Si les vues imbriquées ne prennent pas en charge les mini-data warehouses virtuels, il faudra utiliser un logiciel de virtualisation des données tiers sur Azure. • Il faut convertir tous les mini-data warehouses avant de migrer les données de l'entrepôt de données. • Il faudra porter les mappages des mini-data warehouses virtuels et de l'entrepôt de données vers le serveur de virtualisation des données sur Azure, puis les rediriger vers Azure Synapse.

Tableau 18.2 : Options de migration de données avec leurs avantages et inconvénients

Dans la section suivante, nous aborderons la migration des processus ETL existants vers Azure Synapse Analytics.

Migrer des processus ETL existants vers Azure Synapse Analytics

Un certain nombre d'options s'offrent à vous pour migrer des processus ETL existants vers Azure Synapse Analytics. Le *tableau* 18.3 décrit plusieurs options de migration des tâches ETL en fonction de leur mode de création :

Mode de conception des tâches ETL	Options de migration	Motifs pour migrer et points à prendre en compte
Code et scripts 3GL personnalisés	• Prévoyez de les recréer à l'aide d'ADF.	• Il n'y a pas de traçabilité des métadonnées. • Il est difficile de les conserver si leurs auteurs ont quitté l'entreprise. • Si les tables intermédiaires se trouvent dans l'entrepôt de données hérité et que SQL est utilisé pour transformer les données, gommez les différences avec T-SQL.
Procédures stockées exécutées dans le SGBD de votre entrepôt de données hérité	• Prévoyez de les recréer à l'aide d'ADF.	• Il y aura sans doute des différences importantes entre l'entrepôt de données hérité et Azure Synapse. • Il n'y a pas de traçabilité des métadonnées. • Une évaluation minutieuse est alors nécessaire, mais le principal avantage réside peut-être dans l'approche Pipeline ou Code, qui est prise en charge par ADF.
Outil ETL graphique (Informatica ou Talend, par exemple)	• Continuez à utiliser votre outil ETL existant et désignez Azure Synapse en tant que cible. • Passez éventuellement à une version Azure de votre outil ETL existant, puis portez les métadonnées pour exécuter des tâches ELT sur Azure en veillant à activer l'accès aux sources de données sur site. • Contrôlez l'exécution des services ETL à l'aide d'ADF.	• Évite de recréer des tâches. • Cette méthode réduit les risques et accélère la migration.
Logiciel d'automatisation de l'entrepôt de données	• Continuez à utiliser votre outil ETL existant, puis désignez Azure Synapse en tant que cible et zone intermédiaire.	• Évite de recréer des tâches. • Cette méthode réduit les risques et accélère la migration.

Tableau 18.3 : Options de migration ETL

Dans la section suivante, nous verrons la recréation de processus ETL évolutifs à l'aide d'ADF.

Recréer des processus ETL évolutifs à l'aide d'ADF

Voici une autre option pour gérer vos processus ETL hérités : vous pouvez les recréer à l'aide d'ADF. ADF est un service d'intégration Azure conçu pour créer des workflows pilotés par les données (c'est-à-dire des pipelines) afin d'orchestrer et d'automatiser le transfert et la transformation des données. Vous pouvez utiliser ADF pour créer et planifier des pipelines en vue d'ingérer des données provenant de différents magasins de données. ADF peut traiter et transformer des données à l'aide de services de traitement comme Spark, Azure Machine Learning, Azure HDInsight, Hadoop et Azure Data Lake Analytics :

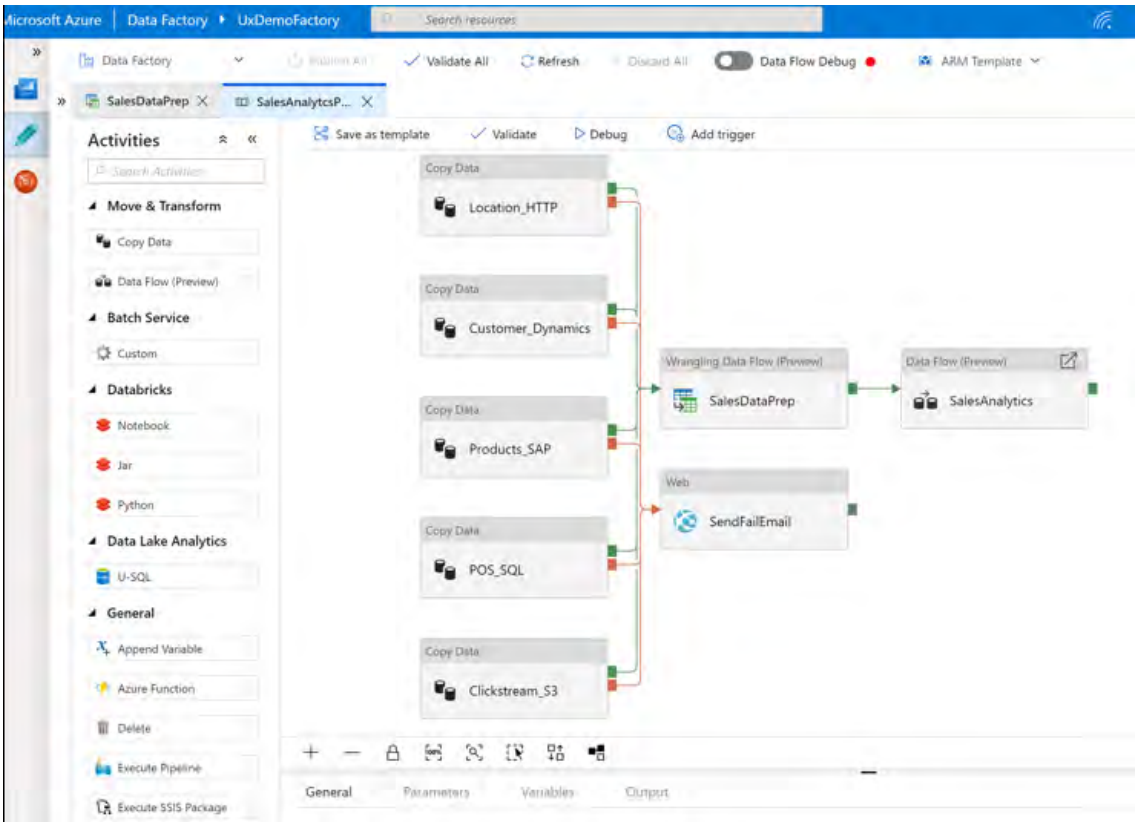


Figure 18.8 : Recréer des processus ETL évolutifs à l'aide d'ADF

La section suivante propose des recommandations pour migrer des requêtes, des rapports BI, des tableaux de bord et d'autres visualisations.

Recommandations pour migrer des requêtes, des rapports BI, des tableaux de bord et d'autres visualisations

La migration de requêtes, de rapports BI, de tableaux de bord et d'autres visualisations de votre entrepôt de données hérité vers Azure Synapse Analytics est une opération simple si votre système existant utilise le langage SQL standard.

Mais ce n'est pas toujours le cas. Dans la négative, une autre stratégie s'impose :

- Commencez par identifier les rapports à migrer en priorité.
- Fiez-vous aux statistiques d'utilisation pour identifier les rapports qui ne sont jamais exploités.
- Évitez de migrer des éléments dont vous ne vous servez plus.
- Après avoir dressé la liste des rapports à migrer (par ordre de priorité) et des rapports inutilisés à ignorer, confirmez cette liste avec les parties prenantes.
- Pour les rapports que vous souhaitez migrer, identifiez les incompatibilités au plus tôt pour évaluer la démarche de migration.
- Envisagez de virtualiser des données pour protéger les applications et les outils BI contre d'éventuelles modifications structurelles apportées au modèle de données de l'entrepôt et/ou des mini-entrepôts de données pendant la migration.

Problèmes de migration courants et solutions

Il est possible que vous rencontriez des problèmes au cours de la migration. Cette section répertorie des problèmes courants et propose des solutions pour tenter de les régler.

Problème 1 : types de données non pris en charge et solutions de contournement

Le tableau 18.4 répertorie les types de données des systèmes de stockage de données hérités qui ne sont pas pris en charge, ainsi que la solution de contournement adaptée à Azure Synapse Analytics :

Type de données non pris en charge	Solution de contournement pour Azure Synapse Analytics
geometry	varbinary
geography	varbinary
hierarchyid	nvarchar(4000)
image	varbinary
text	varchar
ntext	nvarchar
sql_variant	Fractionnez la colonne en plusieurs colonnes fortement typées
table	Convertissez les éléments en tables temporaires
timestamp	Réécrivez le code pour utiliser datetime2 et la fonction CURRENT_TIMESTAMP
xml	varchar
user-defined type	Rétablissez le type de données natif lorsque cela est possible.

Tableau 18.4 : Types de données non pris en charge et solutions de contournement utiles dans Azure Synapse Analytics

Problème 2 : Différences de types de données entre Netezza et Azure Synapse
Le tableau 18.5 mappe les types de données Netezza à leurs types de données Azure Synapse équivalents :

Type de données Netezza	Type de données Azure Synapse
BIGINT	BIGINT
BINARY VARYING(n)	VARBINARY(n)
BOOLEAN	BIT
BYTEINT	TINYINT
CHARACTER VARYING(n)	VARCHAR(n)
CHARACTER(n)	CHAR(n)
DATE	DATE
DECIMAL(p,s)	DECIMAL(p,s)
DOUBLE PRECISION	FLOAT
FLOAT(n)	FLOAT(n)
INTEGER	INT
INTERVAL	Les types de données INTERVAL ne sont actuellement pas pris en charge directement dans Azure Synapse. Ils peuvent toutefois être calculés à l'aide de fonctions temporelles, telles que DATEDIFF
MONEY	MONEY
NATIONAL CHARACTER VARYING(n)	NVARCHAR(n)
NATIONAL CHARACTER(n)	NCHAR(n)
NUMERIC(p,s)	NUMERIC(p,s)
REAL	REAL
SMALLINT	SMALLINT
ST_GEOMETRY(n)	Les types de données spatiales tels que les ST_GEOMETRY ne sont actuellement pas pris en charge dans Azure Synapse. Les données peuvent néanmoins être stockées en tant que VARCHAR ou VARBINARY
TIME	TIME
TIME WITH TIME ZONE	DATETIMEOFFSET
TIMESTAMP	DATETIME

Tableau 18.5 : Types de données Netezza et leurs équivalents Azure Synapse

Problème 3 : Différences en matière de contraintes d'intégrité

Portez une attention particulière aux différences entre l'entrepôt de données hérité ou le mini-data warehouse et Azure Synapse Analytics sur le plan des contraintes d'intégrité. Dans la *figure* 18.9, le système de stockage de données hérité se trouve à gauche, et le nouvel environnement Azure Synapse Analytics, à droite :

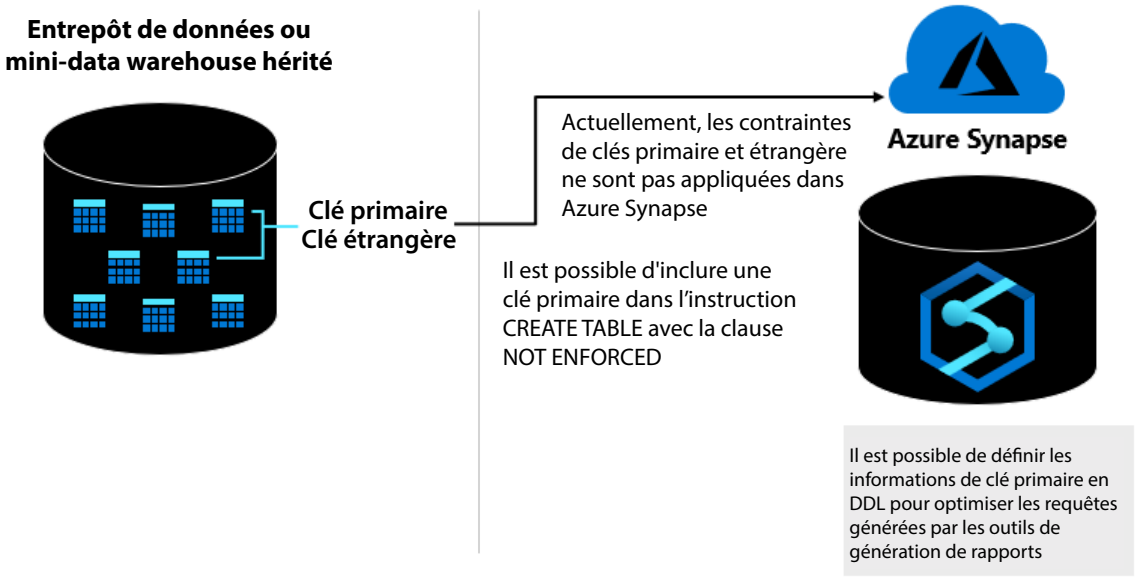


Figure 18.9 : Différences en matière de contraintes d'intégrité

Dans les sections suivantes, nous vous expliquons précisément comment résoudre d'autres incompatibilités SQL fréquentes au cours de la migration d'un entrepôt de données hérité vers Azure Synapse Analytics.

Incompatibilités SQL courantes et solutions

Cette section fournit des détails techniques sur les incompatibilités et les solutions SQL courantes entre les systèmes d'entrepôt de données hérités et Azure Synapse Analytics. La section explique et compare les différences et fournit des solutions à l'aide d'une table de référence rapide que vous pourrez consulter ultérieurement lors du lancement de votre projet de migration.

Les sujets que nous allons aborder sont les suivants :

- Différences du **langage de définition des données** (**DDL**) SQL et solutions
- Différences du **langage de manipulation des données** (**DML**) SQL et solutions
- Différences du **langage de contrôle des données** (**DCL**) SQL et solutions
- Différences SQL étendues et solutions de contournement

Différences et solutions SQL DDL

Dans cette section, nous aborderons les différences et les solutions pour SQL DDL entre les systèmes d'entrepôt de données hérités et Azure Synapse Analytics.

Problèmes	Système d'entrepôt de données hérité	Résolutions
Types de tables propriétaires	• Le cas échéant, identifiez les types de tables propriétaires utilisés sur le système hérité.	• Migrez vers des tables standard dans Azure Synapse Analytics. • Pour les séries chronologiques, basez l'index ou la partition sur la colonne date/time. • Il faut définir un filtre supplémentaire dans les requêtes temporelles appropriées.
Affichages	• Identifiez les vues des tables du catalogue et des scripts DDL.	• Les vues avec des fonctions ou des extensions SQL propriétaires doivent être réécrites. • Azure Synapse Analytics prend également en charge les vues matérialisées, qu'il conserve et actualise automatiquement.
Valeurs nulles	• Les valeurs NULL peuvent être gérées différemment dans les bases de données SQL héritées. Par exemple, dans Oracle, une chaîne vide équivaut à une valeur NULL. • Certains SGBD englobent des fonctions SQL propriétaires pour gérer les valeurs NULL, par exemple, NVL dans Oracle.	• Générez des requêtes SQL pour tester les valeurs NULL. • Testez les rapports qui contiennent des colonnes pouvant accepter la valeur Null.

Tableau 18.6 : Différences SQL DDL entre les systèmes hérités et Azure Synapse

Différences et solutions SQL DML

Dans cette section, nous aborderons les différences et les solutions pour SQL DML entre les systèmes d'entrepôt de données hérités et Azure Synapse Analytics :

Fonction	Netezza	Équivalents Azure Synapse
STRPOS	SÉLECTIONNER STRPOS('ABCDEFG', 'BCD') …	SÉLECTIONNEZ CHARINDEX('BCD', 'ABCDEFG') …
ÂGE	SÉLECTIONNER L'ÂGE ('25-12-1940', '25-12- 2020') À PARTIR DE...	SÉLECTIONNER DATEDIFF (day, '1940-12- 25', '2020-12-25')) À PARTIR DE...
NOW()	NOW()	CURRENT_TIMESTAMP
SÉQUENCE	CRÉER UNE SÉQUENCE...	Réécrivez à l'aide des colonnes IDENTITY sur Azure Synapse Analytics.
UDF	Les UDF Netezza sont écrits en nzLua ou C++.	Réécrivez le code en T-SQL sur Azure Synapse Analytics.
Procédures stockées	Les procédures stockées Netezza sont écrites en NZPLSQL (basé sur postgres PL/pgSQL).	Réécrivez le code en T-SQL sur Azure Synapse Analytics.

Tableau 18.7 : Différences SQL DML entre Netezza et Azure Synapse

Ensuite, nous verrons les différences et les solutions de SQL DCL entre les systèmes d'entrepôt de données hérités et Azure Synapse Analytics.

Différences et solutions SQL DCL

Dans cette section, nous aborderons les différences et les solutions pour SQL DCL entre les systèmes d'entrepôt de données hérités et Azure Synapse Analytics. Netezza prend en charge deux classes de droits d'accès : admin et objet. Le *tableau* 18.8 mappe les droits d'accès Netezza et leurs équivalents Azure Synapse correspondants pour une référence rapide.

Mappage des privilèges d'administrateur Netezza à leurs équivalents Azure Synapse

Le tableau 18.8 mappe les privilèges d'administrateur Netezza à leurs équivalents Azure Synapse :

Privilège d'administrateur	Description	Équivalents Azure Synapse
Sauvegarde	Permet aux utilisateurs de créer des sauvegardes et d'exécuter la commande nzbackup.	Fonction Backup and Restore dans le pool SQL Azure Synapse
[Créer] Agrégat	Permet à l'utilisateur de créer des agrégats définis par l'utilisateur (UDAs). L'autorisation d'utiliser des UDA existants est contrôlée par des privilèges d'objet.	La fonction CRÉER UNE FONCTION d'Azure Synapse intègre la fonctionnalité d'agrégation Netezza
[Créer] Base de données	Permet à l'utilisateur de créer des bases de données. L'autorisation d'utiliser des bases de données existantes est contrôlée par des privilèges d'objet.	CRÉER UNE BASE DE DONNÉES
[Créer] Table externe	Permet à l'utilisateur de créer des tables externes. L'autorisation d'utiliser des tables existantes est contrôlée par des privilèges d'objet.	CRÉER UNE TABLE
[Créer] Fonction	Permet à l'utilisateur de créer des fonctions définies par l'utilisateur. L'autorisation d'utiliser des fonctions définies par l'utilisateur existantes est contrôlée par des privilèges d'objet.	CRÉER UNE FONCTION
[Créer] Groupe	Permet à l'utilisateur de créer des groupes. L'autorisation d'utiliser des groupes existants est contrôlée par des privilèges d'objet.	CRÉER UN RÔLE
[Créer] Index	Réservé à l'usage du système. Les utilisateurs ne peuvent pas créer d'index.	CRÉER UN INDEX
[Créer] Bibliothèque	Permet à l'utilisateur de créer des bibliothèques partagées. L'autorisation d'utiliser des bibliothèques partagées est contrôlée par des privilèges d'objet.	N/A
[Créer] Vue matérialisée	Permet à l'utilisateur de créer des vues matérialisées.	CRÉER UNE VUE
[Créer] Procédure	Permet à l'utilisateur de créer des procédures stockées. L'autorisation d'utiliser des procédures existantes est contrôlée par des privilèges d'objet.	CRÉER UNE PROCÉDURE
[Créer] Schéma	Permet à l'utilisateur de créer des schémas. L'autorisation d'utiliser des schémas existants est contrôlée par des privilèges d'objet.	CRÉER UN SCHÉMA

Privilège d'administrateur	Description	Équivalents Azure Synapse
[Créer] Séquence	Permet à l'utilisateur de créer des séquences de bases de données.	N/A
[Créer] Synonyme	Permet à l'utilisateur de créer des synonymes.	CRÉER UN SYNONYME
[Créer] Table	Permet à l'utilisateur de créer des tables. L'autorisation d'utiliser des tables existantes est contrôlée par des privilèges d'objet.	CRÉER UNE TABLE
[Créer] Table temporaire	Permet à l'utilisateur de créer des tables temporaires. L'autorisation d'utiliser des tables existantes est contrôlée par des privilèges d'objet.	CRÉER UNE TABLE
[Créer] Utilisateur	Permet à l'utilisateur de créer des utilisateurs. L'autorisation d'utiliser des utilisateurs existants est contrôlée par des privilèges d'objet.	CRÉER UN UTILISATEUR
[Créer] Vue	Permet à l'utilisateur de créer des vues. L'autorisation d'utiliser des vues existantes est contrôlée par des privilèges d'objet.	CRÉER UNE VUE
[Gérer] Matériel	Permet à l'utilisateur d'afficher l'état du matériel, de gérer les SPU, de gérer la topologie et la mise en miroir et d'exécuter des tests de diagnostic.	Géré automatiquement via le portail Azure dans Azure Synapse
[Gérer] Sécurité	Permet à l'utilisateur d'exécuter des commandes et de réaliser des opérations liées à la gestion et à la configuration des bases de données d'historique ; la gestion des objets de sécurité à plusieurs niveaux, notamment en spécifiant la sécurité pour les utilisateurs et les groupes ; et la gestion des clés de base de données pour la signature numérique des données d'audit.	Géré automatiquement via le portail Azure dans Azure Synapse
[Gérer] Système	Permet à l'utilisateur d'effectuer les opérations de gestion suivantes : émarrer/arrêter/suspendre/reprendre le système, interrompre les sessions et afficher la carte de distribution, les statistiques du système et les journaux. L'utilisateur peut utiliser ces commandes : nzsystem, nzstate, nzstats et nzsession.	Géré automatiquement via le portail Azure dans Azure Synapse
Restauration	Permet à l'utilisateur de restaurer le système. L'utilisateur peut exécuter la commande nzrestore.	Géré automatiquement dans Azure Synapse
Non limité	Permet à l'utilisateur de créer ou de modifier une fonction définie par l'utilisateur ou un agrégat afin que ceux-ci s'exécutent en mode non limité.	N/A

Tableau 18.8 : Privilèges d'administrateur Netezza et leurs équivalents Azure Synapse

Mappage des privilèges d'objet Netezza à leurs équivalents Azure Synapse

Le tableau 18.9 mappe les privilèges d'objet Netezza aux équivalents Azure Synapse pour une référence rapide :

Privilège d'objet	Description	Équivalents Azure Synapse
Interrompre	Permet à l'utilisateur d'interrompre des sessions. S'applique aux groupes et aux utilisateurs.	INTERROMPRE LA CONNEXION À LA BASE DE DONNÉES
Modifier	Permet à l'utilisateur de modifier des attributs d'objet. S'applique à tous les objets.	MODIFIER
Supprimer	Permet à l'utilisateur de supprimer des lignes de table. S'applique uniquement aux tables.	SUPPRIMER
Déposer	Permet à l'utilisateur de déposer des objets. S'applique à tous les types d'objets.	DÉPOSER
Exécution	Permet à l'utilisateur d'exécuter des UDF, des UDAs ou des procédures stockées.	EXÉCUTER
GenStats	Permet à l'utilisateur de générer des statistiques sur des tables ou des bases de données. L'utilisateur peut exécuter la commande GENERATE STATISTICS.	Géré automatiquement dans Azure Synapse
Préparer	Permet à l'utilisateur de récupérer de l'espace disque pour les lignes supprimées ou obsolètes, et de réorganiser une table selon les clés d'organisation, ou de migrer des données pour les tables qui ont plusieurs versions stockées.	Géré automatiquement dans Azure Synapse
Insertion	Permet à l'utilisateur d'insérer des lignes dans une table. S'applique uniquement aux tables.	INSÉRER
Répertorier	Permet à l'utilisateur d'afficher un nom d'objet, soit dans une liste, soit d'une autre manière. S'applique à tous les objets.	RÉPERTORIER
Sélectionner	Permet à l'utilisateur de sélectionner (ou d'interroger) les lignes d'une table. S'applique aux tables et aux vues.	SÉLECTIONNER
Tronquer	Permet à l'utilisateur de supprimer toutes les lignes d'une table. S'applique uniquement aux tables.	TRONQUER
Mettre à jour	Permet à l'utilisateur de modifier des lignes de table. S'applique uniquement aux tables.	METTRE À JOUR

Tableau 18.9 : Privilèges d'objet Netezza et leurs équivalents Azure Synapse

Différences SQL étendues et solutions de contournement

Le tableau 18.10 décrit les différences SQL étendues et les solutions de contournement possibles lors de la migration vers Azure Synapse Analytics :

SQL extension	Description	Comment migrer
UDF	• Elles peuvent contenir du code arbitraire. • Elles peuvent être codées dans plusieurs langages (Lua et Java, par exemple). • Elles peuvent être appelées au sein d'une instruction SQL SELECT de la même manière que des fonctions intégrées, comme SUM() et AVG().	• Utilisez CREATE FUNCTION et réécrivez le code en T-SQL.
Stocké procédures	• Elles peuvent contenir une ou plusieurs instructions SQL, ainsi que la logique procédurale autour de ces instructions SQL. • Elles sont implémentées dans un langage standard (Lua, par exemple) ou propriétaire (Oracle PL/SQL, par exemple).	• Réécrivez le code en T-SQL. • Voici certains outils tiers qui facilitent la migration : • Datometry • WhereScape
Déclencheurs	• Azure Synapse ne les prend pas en charge.	• D'autres produits de l'écosystème Azure proposent des fonctionnalités équivalentes. Utilisez par exemple Azure Stream Analytics pour les données d'entrée en streaming.
Analyses dans la base de données	• Azure Synapse ne les prend pas en charge.	• Exécutez des analytiques avancées comme les modèles de Machine Learning à grande échelle pour utiliser Azure Databricks. • Azure Synapse permet d'exécuter les fonctions du Machine Learning avec Spark MLlib. • Vous pouvez également migrer vers Azure SQL Database et utiliser la fonction PREDICT.
Géospatial types de données	• Azure Synapse ne les prend pas en charge.	• Stockez les données géospatiales (comme latitude/longitude) et les formats populaires (comme WKT, Well-Known Text et WKB, Well-Known Binary) dans les colonnes VARCHAR ou VARBINARY et accédez-y directement à l'aide des outils client géospatiaux.

Tableau 18.10 : Différences SQL étendues et solutions de contournement

Dans cette section, nous avons vu les problèmes de migration courants que les architectes sont susceptibles de rencontrer au cours d'un projet de migration et des solutions possibles. Dans la section suivante, nous allons examiner les considérations de sécurité dont un architecte doit tenir compte.

Considérations en matière de sécurité

La protection et la sécurisation de vos actifs sont primordiales dans tout système de stockage de données. Lors de la planification du projet de migration d'un entrepôt de données, il faut également prendre en compte la sécurité, la gestion des accès utilisateur, la sauvegarde et la restauration. Par exemple, le chiffrement des données peut être rendu obligatoire par des réglementations sectorielles et administratives comme les normes HIPAA, PCI et FedRAMP, y compris dans des secteurs non réglementés.

Azure inclut en standard de nombreuses fonctions qu'il faudrait généralement intégrer dans des produits de stockage de données hérités sur mesure. Azure Synapse prend notamment en charge le chiffrement des données au repos et en transit.

Chiffrement des données au repos

- **Le chiffrement transparent des données** (**TDE**) peut être activé pour chiffrer et déchiffrer dynamiquement les données Azure Synapse, les journaux, ainsi que les sauvegardes associées.
- Azure Data Storage peut également chiffrer automatiquement les données hors base de données.

Données en transit

Par défaut, toutes les connexions à Azure Synapse Analytics sont chiffrées à l'aide de protocoles standard du secteur, comme TLS et SSH.

En outre, le **masquage dynamique des données** (**DDM**) permet de rendre des données inaccessibles à certaines classes d'utilisateurs, en fonction de règles spécifiques.

Respectez cette bonne pratique : si votre entrepôt de données hérité contient une hiérarchie complexe d'autorisations, d'utilisateurs et de rôles, optez pour des techniques d'automatisation lors de la migration. Vous pouvez utiliser des métadonnées existantes de votre système hérité dans le but de générer les instructions SQL nécessaires pour migrer des utilisateurs, des groupes et des privilèges vers Azure Synapse Analytics.

Dans la section finale de ce chapitre, nous allons examiner certains des outils que les architectes peuvent choisir pour faciliter la migration des systèmes d'entrepôt de données hérités vers Azure Synapse Analytics.

Outils simplifiant la migration vers Azure Synapse Analytics

Maintenant que nous avons parlé de la planification et de la préparation, puis présenté le processus de migration, examinons les outils à votre disposition pour migrer un entrepôt de données hérité vers Azure Synapse Analytics. Voici les outils dont nous allons parler :

- ADF
- Utilitaire de migration Azure Data Warehouse
- Services Microsoft pour le transfert de données physiques
- Services Microsoft pour l'ingestion de données

C'est parti !

ADF

Reposant sur un modèle d'abonnement selon l'utilisation, le service d'intégration de données hybride entièrement géré ADF est adapté au traitement ETL à l'échelle du Cloud. Il offre les fonctions suivantes :

- Il traite et analyse les données en mémoire et en parallèle pour adapter et optimiser le débit.
- Il Crée des pipelines de migration d'entrepôt de données qui orchestrent et automatisent le transfert, la transformation et le chargement des données dans Azure Synapse Analytics.
- Vous pouvez aussi l'utiliser pour moderniser votre entrepôt en ingérant des données dans Azure Data Lake, en les traitant et en les analysant à grande échelle, puis en les chargeant dans un entrepôt de données.
- Il prend en charge des interfaces utilisateur basées sur les rôles afin de mapper des flux de données pour les professionnels de l'informatique ou de convertir (data wrangling) des données en libre-service pour les utilisateurs professionnels.
- Il peut se connecter à plusieurs magasins de données, allant des datacenters aux Clouds en passant par des applications SaaS.
- Plus de 90 connecteurs intégrés en mode natif qui ne nécessitent aucune opération de maintenance sont disponibles (https://azure.microsoft.com/services/data-factory).
- Il est possible d'associer le data wrangling et le mappage de flux de données dans le même pipeline pour préparer des données à grande échelle.
- L'orchestration ADF peut contrôler la migration d'un entrepôt de données vers Azure Synapse Analytics.
- Peut exécuter des packages SSIS ETL

Utilitaire de migration Azure Data Warehouse

L'utilitaire Azure Data Warehouse peut migrer des données d'un entrepôt de données SQL Server sur site vers Azure Synapse. Il offre les fonctions suivantes :

- Il utilise une approche de type Assistant pour migrer (réplication « lift-and-shift ») le schéma et les données d'un entrepôt de données SQL Server sur site.
- Vous pouvez sélectionner la base de données locale qui contient les tables que vous voulez exporter vers Azure Synapse. Sélectionnez ensuite les tables souhaitées, puis migrez le schéma.
- Il génère automatiquement le code T-SQL nécessaire pour créer une base de données vide équivalente et des tables sur Azure Synapse. Après avoir fourni les informations de connexion à Azure Synapse, vous pouvez exécuter le code T-SQL généré pour migrer le schéma.
- Après la création du schéma, vous pouvez vous servir de cet utilitaire pour migrer les données. Il exporte les données de votre entrepôt SQL Server sur site, puis génère des commandes **Bulk Copy Program** (**BCP**) pour charger ces données dans Azure Synapse.

Services Microsoft pour le transfert de données physiques

Dans cette section, nous allons examiner les services Microsoft courants qui peuvent être utilisés pour le transfert de données physiques, y compris Azure ExpressRoute, AzCopy et Azure DataBox.

Azure ExpressRoute

Azure ExpressRoute vous permet d'établir des connexions privées entre vos datacenters et Azure sans passer par l'Internet public. Il offre les fonctions suivantes :

- Bande passante maximale de 100 Gbit/s
- Latence faible
- Se connecte directement à votre **réseau étendu** (**WAN**)
- Connexions privées à Azure
- Amélioration de la rapidité et de la fiabilité

AzCopy

AzCopy est un outil de ligne de commande qui permet de copier des fichiers et des objets blob vers/depuis des comptes de stockage. Il offre les fonctions suivantes :

- Capacité de copier des données vers/depuis Azure via Internet.
- Une combinaison de AzCopy avec la bande passante ExpressRoute nécessaire pourrait être une solution optimale pour le transfert de données vers Azure synapse.

Azure Data Box

Azure Data Box vous permet de transférer des volumes importants de données vers Azure rapidement, de manière fiable et rentable. Il offre les fonctions suivantes :

- Capable de transférer de grands volumes de données (de dizaines de téraoctets à des centaines de téraoctets)
- Aucune restriction de connectivité réseau
- Idéal pour la migration ponctuelle et le transfert en bloc initial

Services Microsoft pour l'ingestion de données

Dans cette section, nous allons examiner les services Microsoft courants qui peuvent être utilisés pour l'ingestion de données, notamment :

- PolyBase
- BCP
- API SqlBulkCopy
- SQL standard

PolyBase (méthode recommandée)

PolyBase fournit le chargement de données en bloc le plus rapide et évolutif dans Azure Synapse Analytics. Il offre les fonctions suivantes :

- Il utilise le chargement en parallèle pour offrir le débit le plus élevé.
- Il peut lire des données issues de fichiers plats dans Azure Blob Storage ou de sources de données externes via des connecteurs.
- Étroitement intégré à ADF
- Instructions CREATE TABLE AS ou INSERT … SELECT
- Peut définir une table intermédiaire de type HEAP pour un chargement rapide.
- Il prend en charge des lignes d'une longueur maximale de 1 Mo.

BCP

BCP permet d'importer et d'exporter des données à partir de n'importe quel environnement SQL Server, y compris Azure Synapse Analytics. Il offre les fonctions suivantes :

- Il prend en charge des lignes d'une longueur supérieure à 1 Mo.
- Il a été développé à l'origine pour des versions antérieures de Microsoft SQL Server.

Reportez-vous à https://docs.microsoft.com/sql/tools/bcp-utility pour en savoir plus sur l'utilitaire BCP.

API SqlBulkCopy

SqlBulkCopy API est l'équivalent d'API de la fonctionnalité BCP. Il offre les fonctions suivantes :

- Elle autorise l'implémentation de processus de chargement à l'aide d'un programme.
- Possibilité de charger en bloc des tables SQL Server avec des données provenant de sources sélectionnées

Reportez-vous à https://docs.microsoft.com/dotnet/api/system.data.sqlclient.sqlbulkcopy pour en savoir plus sur cette API.

Support SQL standard

Azure Synapse Analytics prend en charge la norme SQL, y compris la possibilité de :

- Chargez des lignes individuelles ou des résultats d'instructions **`SELECT`** dans des tables de l'entrepôt de données.
- Il est possible d'utiliser l'instruction **`INSERT … SELECT`** dans PolyBase pour insérer des données en bloc à partir de données extraites via des sources externes dans des tables de l'entrepôt de données.

Cette section présente des considérations architecturales et la méthodologie globale à suivre pour planifier, préparer et effectuer une migration réussie d'un système de stockage de données hérité vers Azure Synapse Analytics. Elle contient une multitude d'informations que vous pourrez consulter ultérieurement lorsque vous commencerez votre projet de migration vers Azure Synapse Analytics.

Résumé

Service d'analyse illimité, Azure Synapse Analytics offre des délais d'extraction des informations incomparables. Cela a pour effet d'accélérer le développement des applications BI, IA et intelligentes destinées aux entreprises. Vous bénéficierez de nombreux avantages en migrant votre entrepôt de données hérité vers Azure Synapse Analytics, notamment en matière de performances, de rapidité, de sécurité et de conformité, d'élasticité, d'infrastructure gérée, d'évolutivité et de coûts.

Avec Azure Synapse, les professionnels des données disposant de différentes compétences peuvent collaborer, gérer et analyser facilement leurs données les plus importantes, le tout au sein du même service. De l'intégration d'Apache Spark avec le moteur SQL puissant et fiable, à l'intégration et à la gestion des données sans code, Azure Synapse est conçu pour tous les professionnels des données.

Ce chapitre présente les considérations architecturales et une méthodologie générale afin de préparer et d'exécuter la migration d'un système d'entrepôt de données hérité existant vers Azure Synapse Analytics.

Pour mener à bien un projet de migration de données, il faut commencer par rédiger un plan précis. Un plan efficace répertorie les nombreux points à prendre en compte, en prêtant une attention particulière à l'architecture et à la préparation des données.

Après avoir migré vers Azure Synapse, vous pouvez explorer d'autres technologies Microsoft dans le riche écosystème analytique Azure pour moderniser davantage votre architecture de stockage des données.

Voici quelques idées à creuser :

- Déléguez les zones intermédiaires et le traitement ELT à Azure Data Lake et ADF.
- Concevez des produits de données fiables dans le format de modèle de données commun, que vous pourrez consommer partout (pas seulement dans votre entrepôt de données).
- Encouragez le développement collaboratif de pipelines de préparation des données par le service commercial et l'équipe informatique à l'aide du mappage ADF et du flux de wrangling data.
- Concevez des pipelines analytiques dans ADF pour analyser les données par lots et en temps réel.
- Concevez et déployez des modèles de Machine Learning pour ajouter des informations supplémentaires aux connaissances déjà acquises.
- Intégrez votre entrepôt de données à des données de streaming en direct.
- Simplifiez l'accès aux données et aux informations stockées dans plusieurs magasins de données analytiques Azure en créant un entrepôt de données logique à l'aide de PolyBase.

Dans le chapitre suivant, vous apprendrez en détail sur Azure cognitive services, en nous concentrant sur les solutions d'architecture qui incluent l'intelligence comme moteur de base.

19

Création de solutions intelligentes

La technologie Cloud a changé beaucoup de choses, y compris la création d'applications intelligentes d'une manière agile, évolutive et avec paiement basé sur la consommation. Les applications antérieures à l'avènement de la technologie Cloud n'intégraient généralement pas l'intelligence en elles-mêmes, principalement parce que :

- Cela prenait beaucoup de temps et était sujet aux erreurs.
- Il était difficile d'écrire, de tester et d'expérimenter des algorithmes de manière continue.
- Les données n'étaient pas en nombre suffisant.
- C'était extrêmement coûteux.

Au cours de la dernière décennie, deux choses ont changé et ont conduit à la création d'applications beaucoup plus intelligentes que par le passé. Ces deux choses sont l'évolutivité illimitée à la demande et rentable du Cloud, ainsi que la disponibilité des données en termes de volume, de variété et de vitesse.

Dans ce chapitre, nous allons examiner les architectures qui peuvent aider à créer des applications intelligentes avec Azure. Certains des sujets abordés dans ce chapitre sont les suivants :

- L'évolution de l'IA
- Processus d'IA Azure
- Azure Cognitive Services
- Création d'un service de reconnaissance optique de caractères
- Création d'un service de fonctions visuelles à l'aide du SDK .NET Cognitive Search

L'évolution de l'IA

L'IA n'est pas un nouveau domaine de connaissances. En fait, la technologie est le résultat de décennies d'innovation et de recherche. Toutefois, sa mise en œuvre au cours des décennies précédentes était un défi pour les raisons suivantes :

1. **Coût** : les expériences d'IA étaient coûteuses par nature, et il n'y avait pas de technologie Cloud. Toute l'infrastructure avait été achetée ou embauchée auprès d'un tiers. La configuration des expériences prenait également du temps et d'immenses compétences étaient nécessaires pour démarrer. Une grande quantité de stockage et de puissance de calcul était également nécessaire, mais elle n'était généralement pas en quantité suffisante dans la communauté dans son ensemble, et était détenue entre les mains de quelques-uns seulement.

2. **Manque de données** : il n'y avait pratiquement pas d'appareils ni de capteurs intelligents à disposition, pour générer des données. Les données étaient de nature limitée et devaient être achetées, ce qui rendait les applications d'IA onéreuses. Les données étaient également moins fiables et il y avait un manque général de confiance dans les données elles-mêmes.

3. **Difficulté** : les algorithmes d'IA n'étaient pas suffisamment documentés et se trouvaient principalement dans les domaines des mathématiciens et des statisticiens. Ils étaient difficiles à créer et à utiliser au sein des applications. Imaginez si l'on avait voulu créer un système de **reconnaissance optique de caractères** (**OCR**) il y a 15 ans. Il n'y avait guère de bibliothèques, de données, de puissance de traitement ou de compétences nécessaires pour développer des applications à l'aide de l'OCR.

Même si l'afflux de données augmentait avec le temps, il y avait encore un manque d'outils pour donner un sens aux données d'une manière ajoutant de la valeur commerciale. En outre, les bons modèles d'IA sont basés sur des données suffisamment précises et sont formés à l'aide d'algorithmes capables de résoudre des problèmes de la vie réelle. La technologie Cloud et le grand nombre de capteurs et d'appareils portatifs ont redéfini ce paysage.

La technologie Cloud permet de configurer le stockage à la demande et les ressources de calcul pour les applications basées sur l'IA. L'infrastructure Cloud fournit de nombreuses ressources pour la migration de données, le stockage, le traitement et le calcul, ainsi que pour générer des informations et éventuellement fournir des rapports et des tableaux de bord. Elle y parvient à un coût minime et de manière plus rapide, car aucun aspect physique n'est impliqué. Examinons les coulisses de la conception d'une application basée sur l'IA.

Processus d'IA Azure

Chaque projet basé sur l'IA doit traverser un certain nombre d'étapes avant d'être opérationnel. Examinons ces sept phases :

Ingestion des données

Au cours de cette phase, les données sont capturées à partir de différentes sources et stockées de manière à pouvoir être consommées au cours de la phase suivante. Les données sont nettoyées avant d'être stockées et les écarts par rapport à la norme sont ignorés. Cela fait partie de la préparation des données. Les données peuvent avoir une vitesse, une variété et un volume différents. Elles peuvent être structurées de la même manière que les bases de données relationnelles, semi-structurées comme les documents JSON, ou non structurées comme les images, les documents Word, etc.

Transformation des données

Les données ingérées sont transformées en un autre format, car elles peuvent ne pas être consommables dans leur format actuel. La transformation des données inclut généralement le nettoyage et le filtrage des données, l'élimination des biais des données, l'augmentation des données en les associant à d'autres jeux de données, la création de données supplémentaires à partir des données existantes, et bien plus encore. Cela fait également partie de la préparation des données.

Analyse

Les données de la dernière phase sont réutilisées à des fins d'analyse. La phase d'analyse contient des activités liées à la recherche de modèles dans les données, à la réalisation d'analyses exploratoires de données et à la génération d'informations supplémentaires. Ces informations sont ensuite stockées en même temps que les données existantes pour la consommation au cours de la phase suivante. Cela fait partie du processus de conditionnement du modèle.

Modélisation des données

Une fois les données augmentées et nettoyées, les données appropriées et nécessaires sont mises à la disposition des algorithmes d'IA pour générer un modèle propice à la réalisation de l'objectif global. Il s'agit d'un processus itératif appelé expérimentation, à l'aide de différentes combinaisons de données (ingénierie des fonctionnalités) pour s'assurer que le modèle de données est solide. Cela fait également partie du processus de conditionnement du modèle.

Les données sont transmises à des algorithmes d'apprentissage afin d'identifier les modèles. Ce processus est généralement désigné sous le terme de formation du modèle. Plus tard, les données de test sont utilisées sur le modèle pour en vérifier l'efficacité.

Validation du modèle

Une fois le modèle créé, un ensemble de données de test est utilisé pour déterminer son efficacité. Si l'analyse obtenue à partir des données de test reflète la réalité, le modèle est sain et utilisable. Les tests sont un aspect important du processus d'IA.

Déploiement

Le modèle est déployé en production afin que les données en temps réel puissent y être introduites pour obtenir le résultat prédit. Ce résultat peut ensuite être utilisé dans les applications.

Pilotage

Le modèle déployé en production est surveillé de façon continue pour l'analyse future de toutes les données entrantes et pour reformer et améliorer les modèles d'efficacité.

Les étapes et les processus d'IA, par nature, prennent beaucoup de temps et sont itératifs. Ainsi, les applications basées sur ceux-ci présentent un risque inhérent d'être longues, expérimentales et gourmandes en ressources, tout en étant retardées par les dépassements de coûts et en ayant de faibles chances de réussite.

Si l'on garde ces éléments à l'esprit, il devrait y avoir des solutions basées sur l'IA prêtes à l'emploi que les développeurs peuvent utiliser dans leurs applications pour les rendre intelligentes. Ces solutions d'IA doivent être facilement consommables à partir des applications et doivent avoir les fonctionnalités suivantes :

- **Multi-plateforme** : les développeurs qui utilisent n'importe quelle plateforme doivent être en mesure de consommer ces services. Ils doivent être déployés et consommés sous Linux, Windows ou Mac sans aucun problème de compatibilité.
- **Inter-langages** : les développeurs doivent être en mesure d'utiliser n'importe quel langage pour consommer ces solutions. Non seulement les développeurs bénéficieront d'une courbe d'apprentissage plus courte, mais en plus, ils n'auront pas besoin de modifier leur choix de langage préféré pour utiliser ces solutions.

Ces solutions doivent être déployées en tant que services à l'aide de normes et de protocoles du secteur. En règle générale, ces services sont disponibles en tant que points de terminaison HTTP qui peuvent être appelés à l'aide de n'importe quel langage et plateforme de programmation.

Il existe de nombreux types de services qui peuvent être modélisés et déployés pour la consommation des développeurs. Voici quelques exemples :

- **Traduction de langue** : dans ces services, l'utilisateur fournit du texte dans une langue et obtient le texte correspondant dans une autre langue en sortie.
- **Reconnaissance de caractères** : ces services lisent des images et restituent le texte qui y est présent.
- **Reconnaissance vocale** : ces services peuvent convertir la parole en texte.

Maintenant que nous avons passé en revue les détails de la création d'un projet basé sur l'IA/ML, nous allons explorer les applications de divers services cognitifs offerts par Azure.

Azure Cognitive Services

Azure fournit un service global connu sous le nom d'Azure Cognitive Services. Azure Cognitive Services est un ensemble de services que les développeurs peuvent utiliser au sein de leurs applications pour les rendre intelligentes.

Vision	Recherche Web	Langue	Parole	Décision
• Vision par ordinateur • Visage • Indexeur vidéo • Custom Vision • Form Recognizer (version préliminaire) • Ink Recognizer (version préliminaire)	• Suggestions automatiques Bing • Recherche personnalisée Bing • Recherche d'entité Bing • Moteur de recherche Bing Image • Moteur de recherche Bing News • Vérification orthographique Bing • Moteur de recherche Bing Video • Recherche Web Bing	• Lecteur immersif (version préliminaire) • Language Understanding • QnA Maker • Analyse de texte • Traducteur	• Reconnais-sance vocale • Conversion de texte en parole • Traduction de parole • Reconnaissance des interlocuteurs (version préliminaire)	• Détection d'anomalies (version préliminaire) • Content Moderator • Personalizer

Tableau 19.1 : Azure Cognitive Services

Les services ont été répartis en cinq grandes catégories selon leur nature. Ces cinq catégories sont les suivantes :

Vision

Cette API fournit des algorithmes pour la classification des images et contribue au traitement d'images en fournissant des informations utiles. La vision par ordinateur peut fournir une variété d'informations à partir d'images sur différents objets, personnes, personnages, émotions, etc.

Recherche

Ces API sont utiles dans les applications associées à la recherche. Elles facilitent la recherche en fonction du texte, des images, de la vidéo et de la fourniture d'options de recherche personnalisées.

Langage

Ces API sont basées sur le traitement du langage naturel et permettent d'extraire des informations sur l'intention du texte soumis par l'utilisateur, ainsi que sur la détection des entités. Elles sont également utiles dans l'analyse de données et la traduction dans différentes langues.

Voix

Ces API permettent de convertir la parole en texte, le texte en parole et offrent une traduction vocale. Elles peuvent être utilisées pour ingérer des fichiers audio et prendre des mesures en fonction du contenu pour le compte des utilisateurs. Cortana est un exemple qui utilise des services similaires pour prendre des mesures pour les utilisateurs en fonction de la parole.

Décision

Ces API aident à détecter les anomalies et à modérer le contenu. Elles peuvent vérifier le contenu dans les images, les vidéos et le texte et trouver des modèles qui doivent être mis en évidence. Un avertissement sur le contenu pour adultes est un exemple d'une telle application.

Maintenant que vous comprenez le cœur de Cognitives Services, nous allons discuter de son fonctionnement de manière détaillée.

Comprendre Cognitive Services

Azure Cognitive Services se compose de points de terminaison HTTP qui acceptent les demandes et renvoient les réponses à l'appelant. Presque toutes les requêtes sont des requêtes HTTP POST et se composent à la fois d'un en-tête et d'un corps.

Le provisionnement de Cognitive Services génère deux artefacts importants qui permettent à un appelant d'appeler un point de terminaison avec succès. Il génère une URL de point de terminaison et une clé unique.

Le format de l'URL est **`https://{azure location}.api.cognitive.microsoft.com/{cognitive type}/{version}/{sub type of service}?{query parameters}`**. Voici un exemple d'URL :

```
https://eastus.api.cognitive.microsoft.com/vision/v2.0/
ocr?language=en&detectOrientation=true
```

Cognitive Service est configuré dans la région Azure de l'Est des États-Unis. Le type de service est la vision par ordinateur à l'aide de la version 2 et le sous-type est OCR. Il existe généralement quelques sous-types pour chaque catégorie de niveau supérieur. Enfin, il existe quelques paramètres de chaîne de requête, tels que **`language`** et **`detectOrientation`**. Ces paramètres de requête sont différents pour chaque catégorie et sous-catégorie de service.

L'en-tête ou les paramètres de requête doivent fournir la valeur de clé pour que l'invocation du point de terminaison réussisse.

La valeur de clé doit être affectée à la clé d'en-tête **`Ocp-Apim-Subscription-Key`** avec la requête.

Le contenu du corps de la requête peut être une chaîne simple, binaire ou une combinaison des deux. Selon la valeur, l'en-tête content-type approprié doit être défini dans la requête.

Les valeurs d'en-tête possibles sont les suivantes :

- **`Application/octet-stream`**
- **`multipart/form-data`**
- **`application/json`**

Utilisez **`octet-stream`** lors de l'envoi de données binaires et **`json`** pour l'envoi de valeurs de chaîne. **`form-data`** peut être utilisé pour l'envoi de plusieurs valeurs de combinaison de binaire et de texte.

La clé est une chaîne unique utilisée pour valider si l'appelant a reçu l'autorisation d'invoquer l'URL. Cette clé doit être protégée de telle sorte que d'autres personnes qui ne doivent pas être en mesure d'appeler les points de terminaison n'y aient pas accès. Plus loin dans ce chapitre, vous verrez comment protéger ces clés.

Utilisation de Cognitives Services

Il existe deux façons d'utiliser des Cognitives Services :

- **À l'aide d'un point de terminaison HTTP directement** : dans ce cas, le point de terminaison est appelé directement en élaborant à la fois l'en-tête et le corps avec des valeurs appropriées. La valeur de retour est ensuite analysée et les données sont extraites de celle-ci. Tous les services IA dans Cognitive Services sont des API REST. Ils acceptent les requêtes HTTP dans JSON, ainsi que d'autres formats, et des réponses au format JSON.
- **À l'aide d'un SDK** : Azure fournit plusieurs **kits de développement logiciel (SDK)**. Des SDK sont disponibles pour les langages .NET, Python, Node.js, Java et Go.

Dans la section suivante, nous allons étudier l'utilisation de l'un des Cognitives Services à l'aide des deux méthodes. Examinons cela en créant des services d'IA à l'aide de points de terminaison HTTP.

Création d'un service OCR

Dans cette section, nous allons utiliser certains des services d'IA à l'aide de C# et de PowerShell pour montrer leur utilisation à l'aide du point de terminaison HTTP directement. La section suivante se concentrera sur la même chose à l'aide d'un SDK .NET.

Avant d'entrer dans la création d'un projet à l'aide de Cognitives Services, la première étape consiste à configurer l'API elle-même.

La reconnaissance optique de caractères est disponible en tant qu'API Vision et peut être mise en service à l'aide du portail Azure, comme illustré ci-après. Créez une API Vision en accédant à **Cognitive Services > Compute Vision > Create** , comme illustré à la *Figure 19.1* :

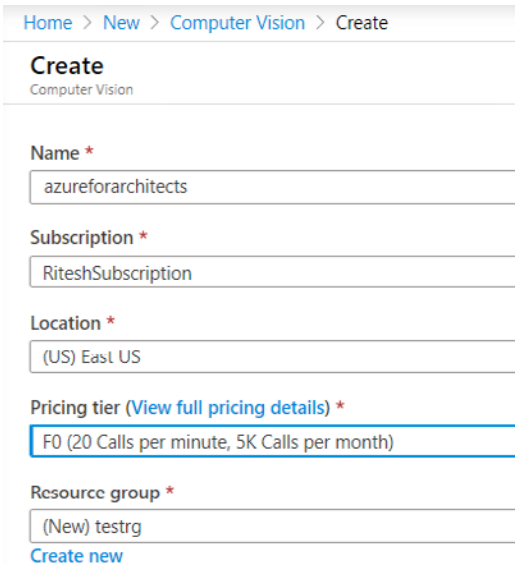


Figure 19.1 : Créer une API Vision

Une fois l'API configurée, la page de présentation fournit tous les détails pour consommer l'API. Elle fournit l'URL de base et les informations de clé. Notez la clé, car elle sera utilisée ultérieurement :

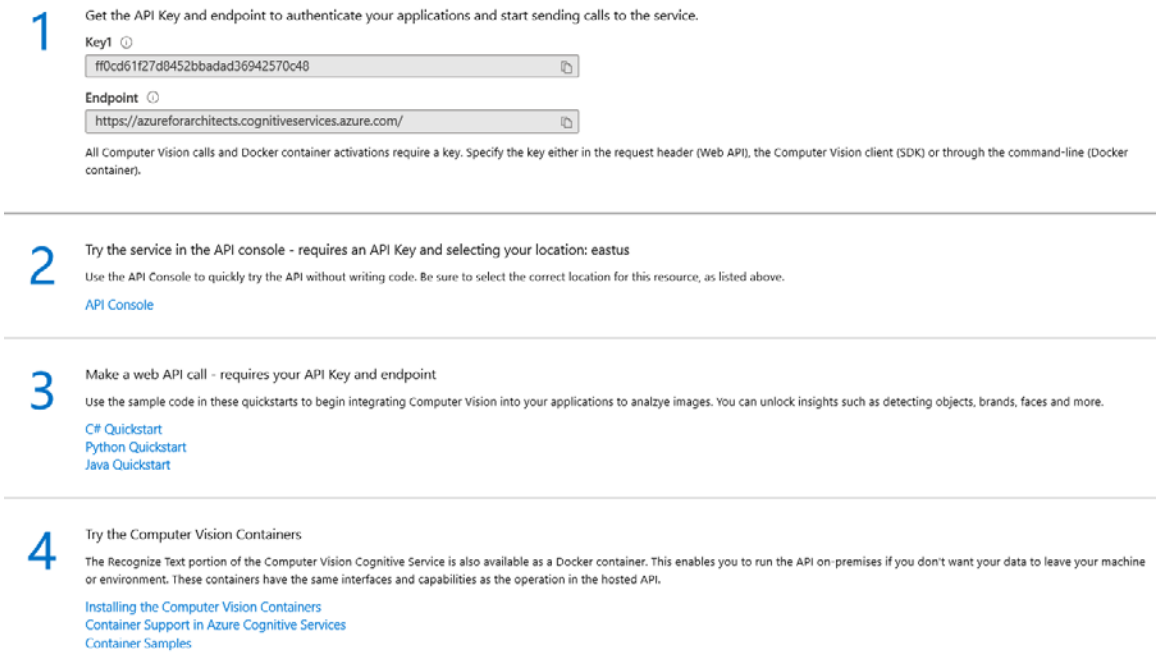


Figure 19.2 : page de présentation

Elle fournit également une console API pour tester rapidement l'API. En cliquant dessus, vous ouvrez une nouvelle fenêtre qui contient tous les points de terminaison relatifs à ce service. Un clic sur **OCR** affiche un formulaire qui peut être rempli avec les données appropriées et exécuter les points de terminaison de service. Une réponse complète est également fournie. Cela est illustré à *Figure* 19.3. L'URL est disponible en tant qu'URL de requête, et la requête est une requête HTTP typique avec une méthode **`POST`**. L'URL pointe vers le point de terminaison dans la région Azure de l'Est des États-Unis. Elle est également associée au groupe Vision des API, version 2, et au point de terminaison OCR.

La clé d'abonnement est transmise dans l'en-tête avec le nom **`ocp-apim-subscription-key`**. L'en-tête contient également la clé content-type avec **`application/json`** en tant que valeur. En effet, le corps de la requête contient une chaîne JSON. Le corps est sous la forme de JSON avec l'URL de l'image à partir de laquelle le texte doit être extrait :

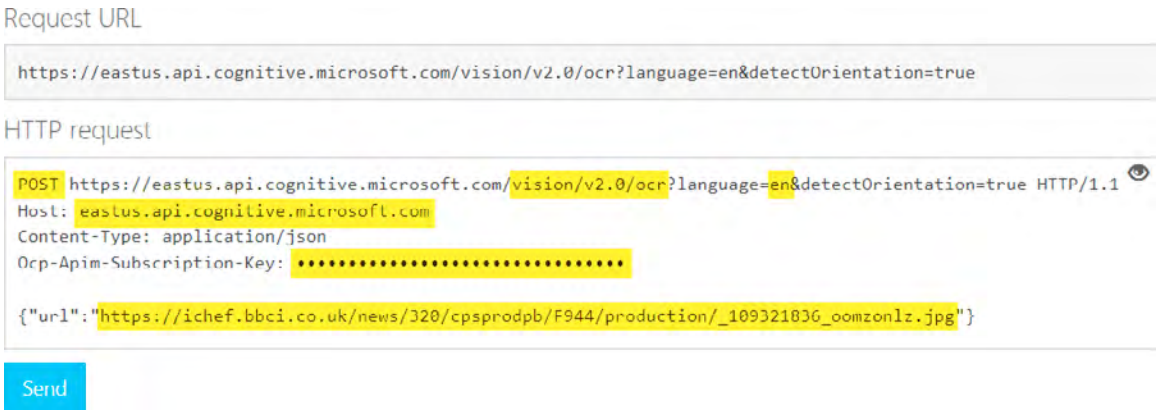


Figure 19.3 : URL de requête

La requête peut être envoyée au point de terminaison en cliquant sur le bouton **Envoyer**. Cela se traduira par une réponse HTTP 200 OK, comme illustré ci-après, si tout se passe bien. S'il y a une erreur dans les valeurs de la requête, la réponse sera un code HTTP d'erreur :

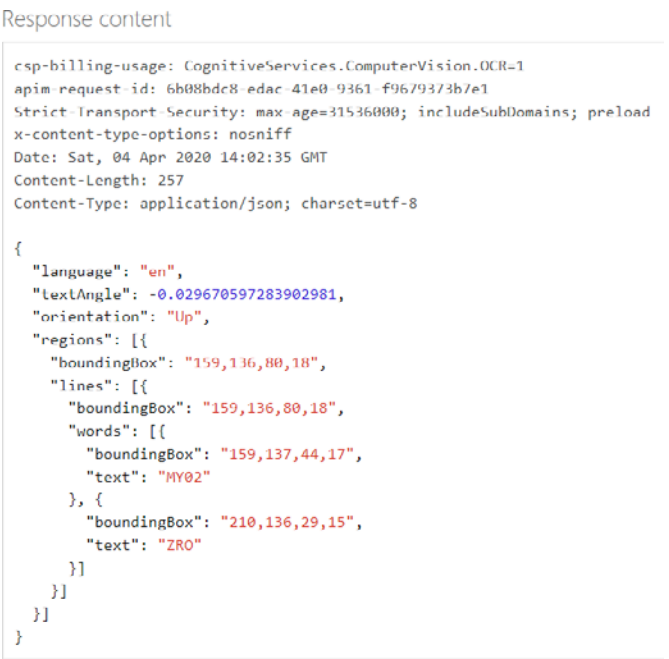


Figure 19.4 : réponse HTTP 200 OK

La réponse se compose de détails liés à l'utilisation de la facturation, d'un ID de requête interne généré par le point de terminaison, de la longueur du contenu, du type de contenu de réponse (JSON), ainsi que des données et de l'heure de la réponse. Le contenu de la réponse se compose d'une charge utile JSON avec les coordonnées du texte et le texte réel lui-même.

Utilisation de PowerShell

La même requête peut être créée à l'aide de PowerShell. Le code PowerShell suivant peut être exécuté à l'aide de PowerShell ISE.

Le code utilise la cmdlet **`Invoke-WebRequest`** pour appeler le point de terminaison Cognitive Services en transmettant l'URL au paramètre **`Uri`** à l'aide de la méthode **`POST`** et en ajoutant les deux en-têtes appropriés comme indiqué dans la dernière section, et enfin, le corps constitué de données au format JSON. Les données sont converties en JSON à l'aide de la cmdlet **`ConvertTo-Json`** :

```
$ret = Invoke-WebRequest -Uri "https://eastus.api.cognitive.microsoft.
com/vision/v2.0/ocr?language=en&detectOrientation=true"  -Method Post
-Headers @{"Ocp-Apim-Subscription-Key"="ff0cd61f27d8452bbadad36942
570c48"; "Content-type"="application/json"} -Body $(ConvertTo-Json
-InputObject  @{"url"="https://ichef.bbci.co.uk/news/320/cpsprodpb/F944/
production/_109321836_oomzonlz.jpg"})


$val = Convertfrom-Json $ret.content


foreach ($region in $val.regions) {
    foreach($line in $region.lines) {
        foreach($word in $line.words) {
            $word.text
        }
    }
}
```

La réponse de la cmdlet est enregistrée dans une variable qui se compose également de données au format JSON. Les données sont converties en objet PowerShell à l'aide de la cmdlet **`ConvertFrom-Json`** et sont regroupées pour trouver les mots dans le texte.

Utilisation de C#

Dans cette section, nous allons créer un service qui doit accepter les requêtes des utilisateurs, extraire l'URL de l'image, construire la requête HTTP et l'envoyer au point de terminaison Cognitive Services. Le point de terminaison Cognitive Services renvoie une réponse JSON. Le contenu textuel approprié est extrait de la réponse et renvoyé à l'utilisateur.

Architecture et design

Une application intelligente est une application ASP.NET Core MVC. Une application MVC est créée par un développeur sur une machine de développeur, passe par le pipeline d'intégration et de livraison continus, génère une image Docker et télécharge l'image Docker vers le registre de conteneurs Azure. Les principaux composants de l'application sont expliqués ici, ainsi que leur utilisation :

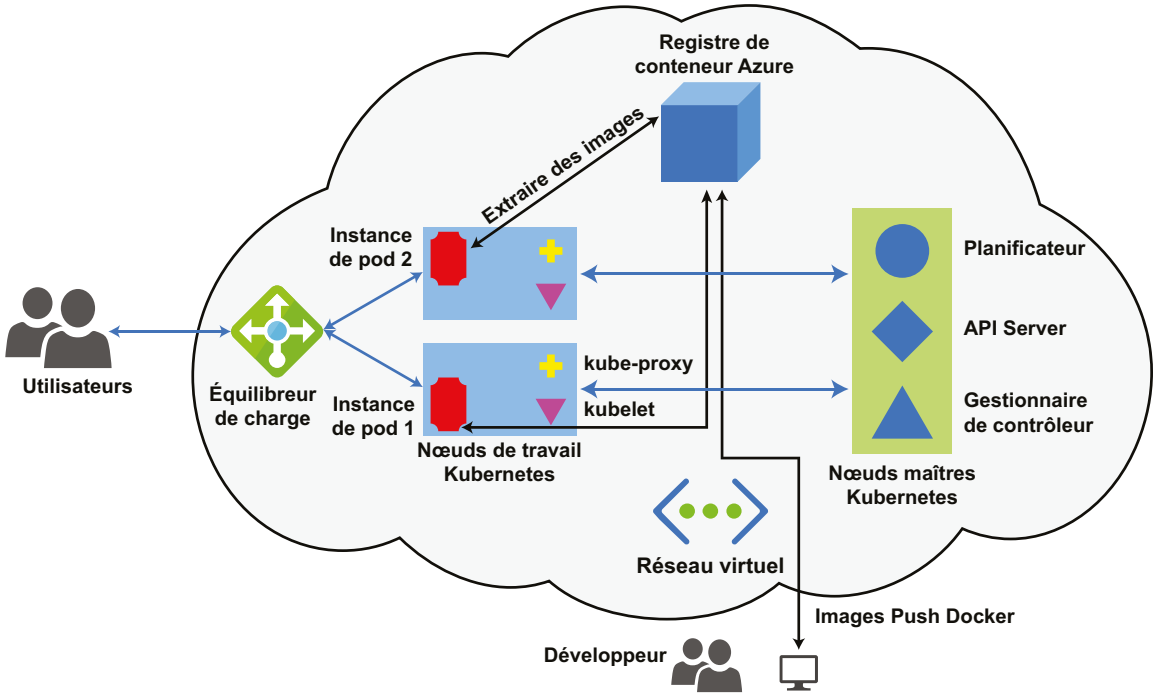


Figure 19.5 : Charge de travail d'une application intelligente

Docker

Docker est l'un des principaux acteurs des technologies de conteneurs et est disponible sur plusieurs plateformes, y compris Linux, Windows et Mac. Le développement d'applications et de services avec la conteneurisation à l'esprit offre la souplesse nécessaire pour les déployer dans les clouds et les emplacements, ainsi que sur site. Il supprime également toutes les dépendances sur la plateforme hôte, ce qui permet une fois de plus de moins recourir à la plateforme en tant que service. Docker facilite la création d'images personnalisées, et les conteneurs peuvent être créés à partir de ces images. Les images contiennent toutes les dépendances, les binaires et les infrastructures nécessaires pour que l'application ou le service fonctionne, et ils sont complètement autonomes. Il s'agit donc d'un excellent objectif de déploiement pour les services tels que les microservices.

Registre de conteneur Azure

Azure Container Registry est un registre similaire à Docker Hub pour le stockage des images de conteneur dans un référentiel. Il est possible de créer plusieurs référentiels et de télécharger plusieurs images dans ceux-ci. Une image possède un nom et un numéro de version, formant ensemble un nom qualifié complet utilisé pour y faire référence dans une définition de Kubernetes Pod. Ces images sont accessibles et téléchargées par n'importe quel écosystème Kubernetes. Une condition préalable est que les secrets appropriés pour extraire l'image doivent déjà être créés à l'avance. Elle ne doit pas forcément être sur le même réseau que les nœuds Kubernetes. Il n'est d'ailleurs pas nécessaire qu'un réseau crée et utilise Azure Container Registry.

Azure Kubernetes Service

L'application intelligente qui accepte l'URL d'une image pour récupérer le texte peut être hébergée sur des machines virtuelles standard ou même dans Azure App Service. Toutefois, le déploiement dans Azure Kubernetes Service offre de nombreux avantages, qui ont été abordés dans le *Chapitre 8*, *Créer des solutions Azure Kubernetes*. Pour l'instant, il est important de savoir que ces applications se réparent spontanément et qu'un nombre minimal d'instances est automatiquement maintenu par le maître Kubernetes, tout en offrant la souplesse nécessaire pour les mettre à jour de plusieurs façons, y compris les déploiements bleu-vert et les mises à jour canari.

Pods, jeux de réplicas et déploiements

Le développeur crée également un fichier YAML associé au déploiement de Kubernetes qui référence les images au sein de la spécification Pod et fournit également une spécification pour le jeu de réplicas. Il fournit ses propres spécifications relatives à la stratégie de mise à jour.

Conception d'exécution

L'architecture et la conception restent les mêmes que dans la section précédente. Toutefois, lorsque l'application ou le service est déjà actif et opérationnel, il a déjà téléchargé les images à partir de Azure Container Registry et a créé des Pods qui exécutent les conteneurs qu'ils possèdent. Lorsqu'un utilisateur fournit une URL d'image pour le décodage du texte qu'il contient, l'application dans le Pod appelle l'API de vision par ordinateur Azure Cognitive Services et lui transmet l'URL, puis attend une réponse du service :

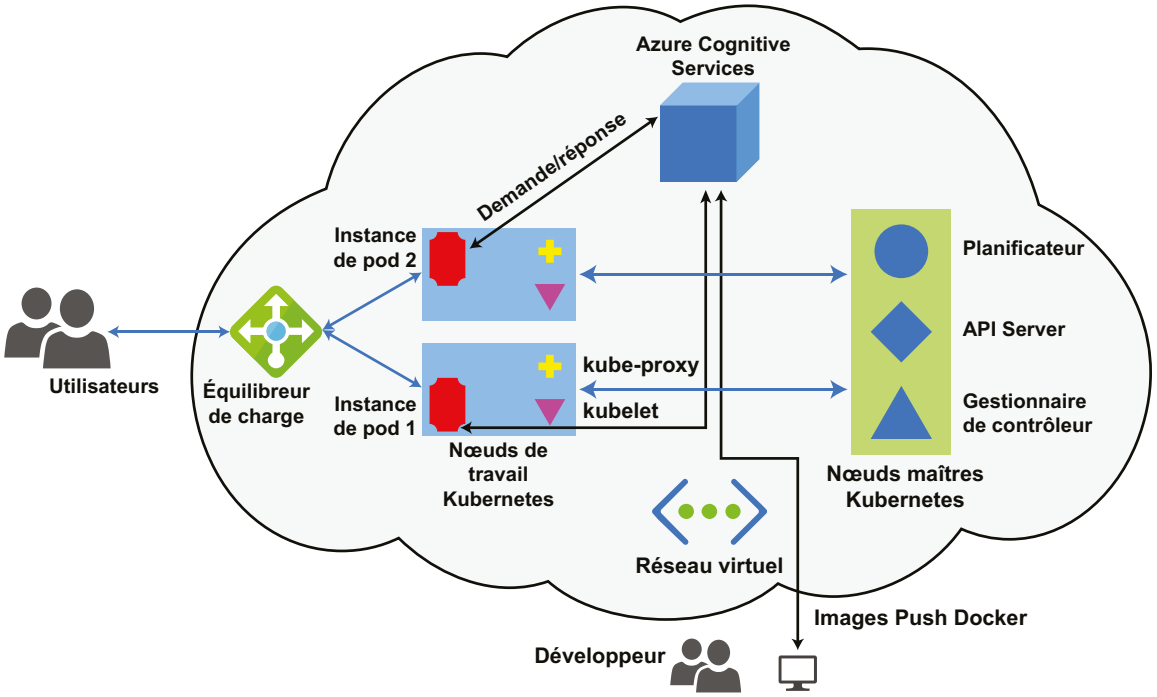


Figure 19.6 : Charge de travail d'une application intelligente

Une fois qu'elle reçoit la réponse JSON des services, elle peut récupérer les informations et les renvoyer à l'utilisateur.

Le processus de développement

L'environnement de développement peut être Windows ou Linux. Il fonctionnera avec Windows 10 et Windows 2016/19 Server. Lors de l'utilisation de Windows, il est utile de déployer Docker pour Windows afin qu'il crée à la fois un environnement Linux et Windows Docker.

Lors de la création d'un projet d'application Web ASP.NET Core à l'aide de Visual Studio 2019, l'option **Docker support** doit être sélectionnée avec **Windows** ou **Linux** en tant que valeurs. Selon la valeur choisie, le contenu approprié sera généré dans **`Dockerfile`**. La principale différence dans **`Dockerfile`** est le nom de l'image de base. Il utilise des images différentes pour Linux par rapport à Windows.

Lors de l'installation de docker pour Windows, il installe également une machine virtuelle Linux. Il est donc important d'activer l'hyperviseur Hyper-V.

Dans cet exemple, au lieu d'envoyer les données sous forme de chaîne JSON, l'image est téléchargée et les données binaires sont envoyées au point de terminaison Cognitive Services.

Il possède une fonction qui accepte une entrée de chaîne pour les valeurs d'URL. Il appelle ensuite Cognitive Services avec les valeurs d'en-tête appropriées et un corps contenant l'URL. Les valeurs d'en-tête doivent contenir la clé fournie par Cognitive Services lors du provisionnement du service. La valeur dans le corps peut contenir des valeurs de chaîne standard sous la forme de JSON ou des données d'image binaires elle-même. La propriété d'en-tête content-type doit être définie en conséquence.

Le code déclare l'URL et la clé associées à Cognitive Services. Ce point est montré à des fins de démonstration uniquement. L'URL et la clé doivent être placées dans des fichiers de configuration.

À l'aide de l'objet **`HttpClient`**, l'image correspondant à l'URL fournie par l'utilisateur est téléchargée et stockée dans la variable **`responseMessage`**. Un autre objet **`HttpClient`** est instancié et ses en-têtes sont remplis avec **`Ocp-Apim-Subscription-Key`** et **`content-type keys`**. La valeur de l'en-tête content-type est **`application/octet-stream`**, car les données binaires sont transmises au point de terminaison.

Une requête post est effectuée après l'extraction du contenu de la variable **`responseMessage`** et sa transmission en tant que corps d'une requête au point de terminaison du service cognitif.

Le code pour l'action du contrôleur s'affiche ensuite :

```
        [HttpPost]
        public async Task<string> Post([FromBody] string value)
        {
            string myurl = " https://eastus.api.cognitive.microsoft.com/
vision/v2.0/ocr?language=en&detectOrientation=true
            string token = "……";
            using (HttpClient httpClient = new HttpClient())
            {
                var responseMessage = await httpClient.GetAsync(value);
```

```
                using (var httpClient1 = new HttpClient())
               {
                    httpClient1.BaseAddress = new Uri(myurl);
                  httpClient1.DefaultRequestHeaders.Add("Ocp-Apim-
Subscription-Key", token);

                   HttpContent content = responseMessage.Content;
                   content.Headers.ContentType = new
MediaTypeWithQualityHeaderValue("application/octet-stream");
                   var response = await httpClient1.PostAsync(myurl,
content);

                   var responseContent = await response.Content.
ReadAsByteArrayAsync();
                   string ret = Encoding.ASCII.GetString(responseContent,
0, responseContent.Length);
                   dynamic image = JsonConvert.
DeserializeObject<object>(ret);
                   string temp = "";
                   foreach (var regs in image.regions)
                   {
                       foreach (var lns in regs.lines)
                       {
                           foreach (var wds in lns.words)
                           {
                               temp += wds.text + " ";
                           }
                       }
                   }
                   return temp;
               }
           }
       }
```

Une fois que le point de terminaison a terminé son traitement, il renvoie la réponse à l'aide d'une charge utile JSON. Le contexte est extrait et désérialisé en objets .NET. Plusieurs boucles sont codées pour extraire le texte de la réponse.

Dans cette section, nous avons créé une application simple qui utilise Cognitive Services pour fournir des extractions de mots à partir de fonctionnalités qui utilisent l'API OCR et déployées dans des Pods Kubernetes. Ce processus et cette architecture peuvent être utilisés dans n'importe quelle application qui souhaite consommer des API Cognitive Services. Ensuite, nous allons jeter un œil à une autre API Cognitive Services, connue sous le nom de fonctions visuelles.

Création d'un service de fonctions visuelles à l'aide du SDK .NET Cognitive Search

La dernière section portait sur la création d'un service qui utilise un point de terminaison cognitif d'OCR pour renvoyer le texte dans les images. Dans cette section, un nouveau service sera créé et renverra des fonctions visuelles au sein d'une image, telles que des descriptions, des balises et des objets.

Utilisation de PowerShell

Le code dans PowerShell est similaire à l'exemple OCR précédent. il n'est donc pas répété ici. L'URL est différente de l'exemple de code précédent :

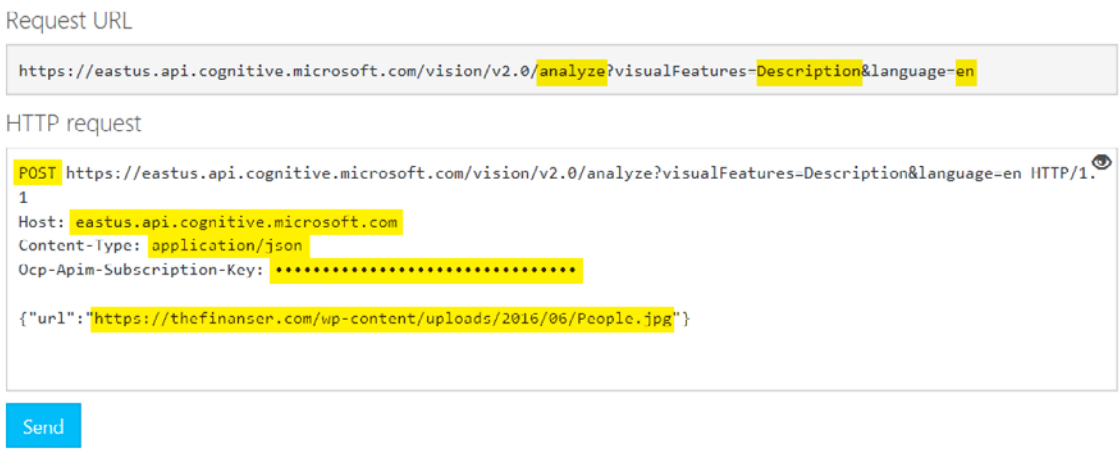


Figure 19.7 : URL de requête

La requête est effectuée à l'aide d'une méthode **`POST`**, et l'URL pointe vers le point de terminaison dans la région Azure de l'Est des États-Unis. Elle utilise également la version 2 et l'API Vision.

La clé d'accès Cognitive Services fait partie de l'en-tête HTTP nommé **`ocp-apim-subscription-key`**. L'en-tête contient également l'en-tête content-type avec **`application/json`** en tant que valeur. En effet, le corps de la requête contient une valeur JSON. Le corps contient l'URL de l'image à partir de laquelle le texte doit être extrait.

La réponse sera au format JSON, présentant le contenu de l'image et une description.

Utilisation de .NET

Cet exemple est à nouveau une application ASP.NET Core MVC et comporte le package NuGet **`Microsoft.Azure.CognitiveServices.Vision.ComputerVision`** :

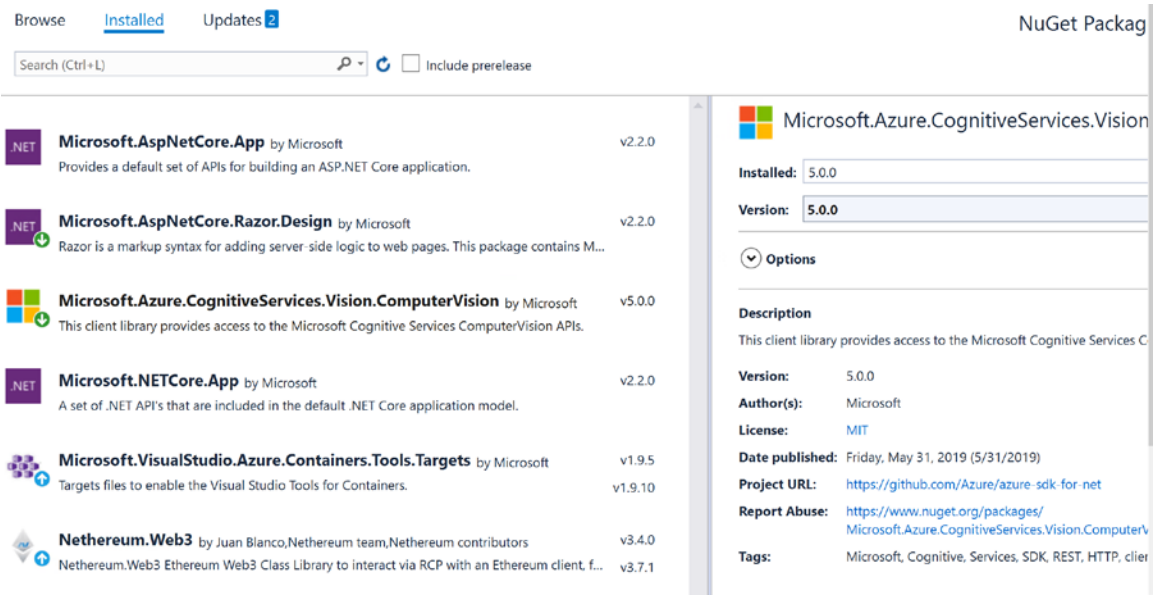


Figure 19.8 : application ASP.NET Core MVC avec le package NuGet Microsoft.Azure.CognitiveServices.Vision.ComputerVision

Le code pour l'action du contrôleur s'affiche ensuite. Dans ce code, le service cognitif et la clé sont déclarés. Il déclare également des variables pour les objets **`ComputerVisionClient`** et **`VisionType`**. Il crée une instance du type **`ComputerVisionClient`**, en lui fournissant l'URL et la clé.

La liste **`VisionTypes`** se compose de plusieurs types de données recherchées à partir de l'image : les balises, les descriptions et les objets sont ajoutés. Seuls ces paramètres seront extraits de l'image.

Un objet **`HttpClient`** est instancié pour télécharger l'image à l'aide de l'URL fournie par l'utilisateur et envoie ces données binaires au point de terminaison Cognitive Services à l'aide de la fonction **`AnalyzeImageInStreamAsync`** de type **`ComputerVisionClient`** :

```
[HttpPost]
        public string Post([FromBody] string value)
        {
        private string visionapiurl = " https://
eastus.api.cognitive.microsoft.com/vision/v2.0/
analyze?visualFeaure=tags,description,objects&language=en";
        private string apikey = "e55d36ac228f4d718d365f1fcddc0851";
        private ComputerVisionClient client;
        private List<VisualFeatureTypes> visionType =  new
List<VisualFeatureTypes>();


client = new ComputerVisionClient(new ApiKeyServiceClientCredentials(apikey))
{
                Endpoint = visionapiurl
            };
            visionType.Add(VisualFeatureTypes.Description);
            visionType.Add(VisualFeatureTypes.Tags);
            visionType.Add(VisualFeatureTypes.Objects);


            string tags = "";
            string descrip   = "";
            string objprop = "";
            using (HttpClient hc = new HttpClient()) {
                var responseMessage =  hc.GetAsync(value).GetAwaiter().
GetResult();
                Stream streamData = responseMessage.Content.
ReadAsStreamAsync().GetAwaiter().GetResult();
                var result = client.AnalyzeImageInStreamAsync(streamData,
visionType).GetAwaiter().GetResult();
                foreach (var tag in result.Tags) {
                    tags += tag.Name + " ";
                }


                foreach (var caption in result.Description.Captions)
```

```
            {
                descrip += caption.Text + " ";
            }

            foreach (var obj in result.Objects)
            {
                objprop += obj.ObjectProperty + " ";
            }

        }
    return tags;
    // return descrip or objprop




    }
```

Les résultats sont en boucle et les balises sont renvoyées à l'utilisateur. De même, les descriptions et les propriétés des objets peuvent également être renvoyées à l'utilisateur. Maintenant, nous allons voir comment nous pouvons protéger l'exposition des clés de service.

Protection de la clé Cognitive Services

Il existe plusieurs façons de protéger l'exposition des clés à d'autres acteurs. Vous pouvez utiliser la ressource API Management dans Azure. Vous pouvez également utiliser Azure Functions Proxies.

Utilisation d'Azure Functions Proxies

Azure Functions Proxies peut faire référence à n'importe quelle URL, qu'elle soit interne ou externe. Lorsqu'une requête atteint Azure Functions Proxies, elle utilise l'URL du service cognitif ainsi que la clé pour appeler le point de terminaison cognitif. Elle substitue également les paramètres de requête et ajoute l'URL de l'image entrante, puis les ajoute à l'URL de point de terminaison cognitive en tant que données POST. Lorsqu'une réponse revient du service, elle remplace la réponse, supprime les en-têtes et transmet les données JSON à l'utilisateur.

Utilisation de Cognitives Services

L'utilisation de Cognitives Services suit un modèle cohérent. Chaque service cognitif est disponible en tant qu'API REST, et chaque API attend différents ensembles de paramètres sur lesquels travailler. Les clients qui appellent ces URL doivent consulter la documentation relative aux paramètres associés et leur fournir des valeurs. La consommation d'URL est une méthode relativement brute d'utilisation de Cognitives Services. Azure fournit des SDK pour chaque service et pour plusieurs langages. Les clients peuvent utiliser ces SDK pour travailler avec Cognitive Services.

L'API de création **LUIS (Language Understanding Intelligence Service)** est disponible à l'adresse **`https://{luis resource name}-authoring.cognitiveservices.azure.com/`** et l'API de production est disponible à l'adresse

`https://{azure region}.api.cognitive.microsoft.com/luis/prediction/v3.0/apps/{application id}/slots/production/predict?subscription-key={cognitive key}&verbose=true&show-all-intents=true&log=true&query=YOUR_QUERY_HERE`.

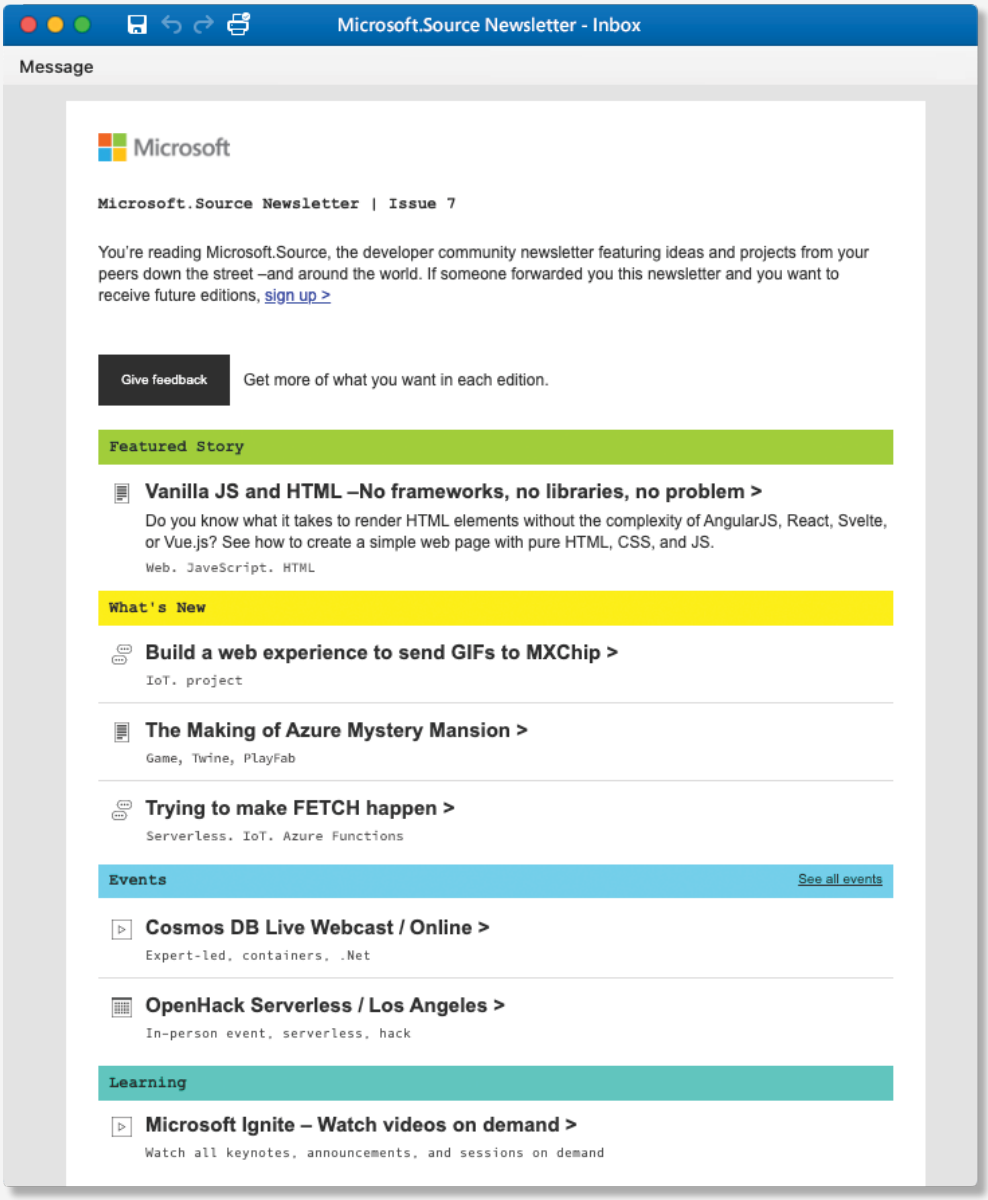
De même, l'API Face est disponible à l'adresse **`https://{endpoint}/face/v1.0/detect[?returnFaceId][&returnFaceLandmarks][&returnFaceAttributes][&recognitionModel][&returnRecognitionModel][&detectionModel]`**.

Il existe de nombreuses API Cognitive Services, chacune ayant plusieurs options en termes d'URL. La meilleure façon de connaître ces URL est d'utiliser la documentation Azure.

Résumé

Dans ce chapitre, vous avez acquis une compréhension de l'architecture de déploiement et d'application pour la création d'applications intelligentes dans Azure. Azure fournit Cognitives Services avec de nombreux points de terminaison. Chacun est responsable de l'exécution d'un algorithme associé à l'IA et de la fourniture de sorties. Presque tous les points de terminaison Cognitives Services fonctionnent de la même manière en ce qui concerne les demandes et les réponses HTTP. Ces points de terminaison peuvent également être invoqués à l'aide de SDK fournis par Azure pour différents langages. Vous avez vu un exemple d'obtention de fonctionnalités visuelles à l'aide de ces SDK. Il existe plus de 50 points de terminaison différents. Il est conseillé de comprendre la nature des points de terminaison à l'aide de la fonctionnalité de console API fournie par Azure.

Microsoft



Par les développeurs, pour les développeurs

Newsletter Microsoft.Source

Obtenez des articles techniques, des exemples de code et des informations sur les événements à venir dans Microsoft.Source, la newsletter mensuelle de la communauté des développeurs.

- Tenez-vous au courant des dernières technologies
- Connectez-vous avec vos pairs lors d'événements communautaires
- Apprenez avec des ressources pratiques

Créer un compte

Index

À propos de

Tous les principaux mots-clés utilisés dans ce livre sont classés par ordre alphabétique dans cette section. Chacun est accompagné du numéro de page de l'endroit où il apparaît.

A

B

C

D

E

F

G

H

I

J

K

R

S

T

U

V

W