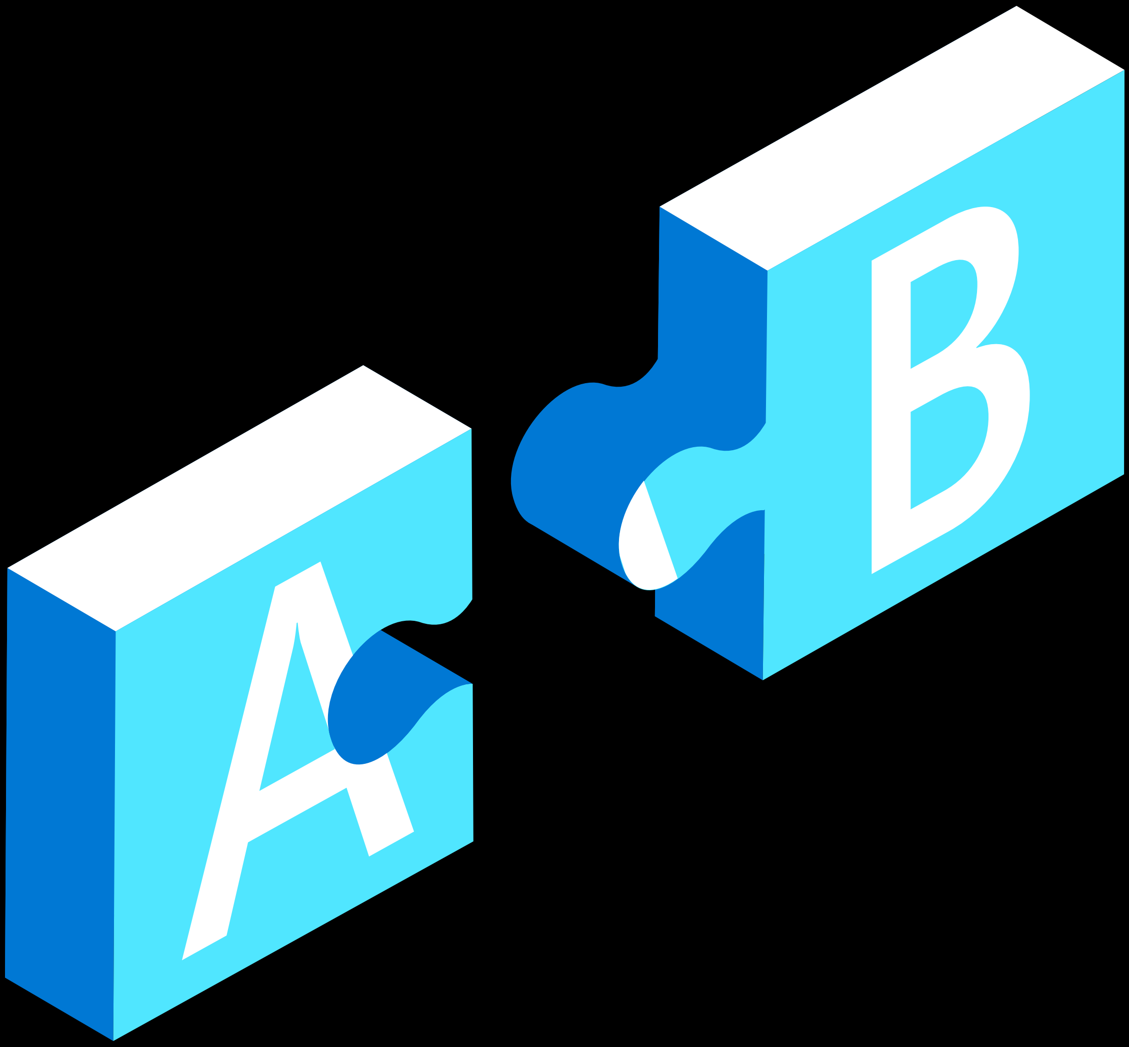
E-book series

Microsoft Azure

How to Choose the Right Azure Services for Your Applications —It's Not A or B



How to Choose the Right Azure Services for Your Applications —It's Not A or B

Preface

Who should read this book?

Technical and business decision makers, C-suite through architects and developers who are considering migrating their applications to Azure or building cloud-native applications in Azure.

Why did we write this book?

Having worked with several customers, a consistent theme that we observed across our customers has been a struggle to pick the right service in Azure to host as many of their applications as possible, if not all. For example, they ask questions like, "should we use Azure Kubernetes Service or Azure App Service to host all our web applications?" This kind of A or B thinking stems from traditional on-premises practices that were based on the constraints of the on-premises world, such as packaged software delivery model, big upfront investments, and long lead times required to build and deploy any application platform. They bring the same mindset to cloud and spend a significant amount of time identifying and preparing that one Azure service that can host all their applications, and then trying to force-fit all their applications into this single platform introducing delays and creating unnecessary roadblocks. But there is a better way—an A+B approach that fosters modular thinking and helps your company achieve better results in the cloud.

This book is about changing the way you think about the right platform for your applications. We wrote this book to equip you with the principles of an A+B mindset that you can use to choose the right Azure Services for your applications, thereby achieving greater efficiencies and faster time to market.

Navigating this book

The book is written in a logical order as follows:

- An introduction to the A+B mindset for choosing the right Azure services for your applications.
- Traditional reasons why organizations used A or B mindset for choosing Azure services to host their applications.
- How cloud enables A+B mindset and the benefits of using A+B mindset.
- Principles of A+B mindset and application of these principles to seven real-world use cases using Azure Kubernetes Service and Azure Spring Apps as example destinations. These principles can be put to practice right away by any organization by applying the A+B mindset.
- Key takeaways.
- Finally, as an addendum, we have provided a detailed comparison of Azure Kubernetes Service and Azure Spring Apps for hosting Spring Boot applications for those who are interested in learning more about running Spring Boot apps in Azure.

We recommend reading this book in the order it is written.

Introduction

If you have been working with Azure for any period, you might have grappled with the question, which Azure service is best to run my apps on? This is an important decision because the services you choose will dictate your resource planning, budget, timelines, and ultimately the time to market for your business. It impacts the cost for not only the initial delivery but also the ongoing maintenance of your applications.

Traditionally, organizations have thought that they must choose between two platforms, technologies, or competing solutions to build and run their software applications. For example, they ask questions like—"Do we use Web Logic or WebSphere for hosting our Java Enterprise applications?," "Should Docker Swarm be the enterprise-wide container platform or Kubernetes?," or "Do we adopt containers or just stick with virtual machines (VMs)?" They try to fit all their applications on platform A or B. This A or B mindset stems from on-premises practices that were based on the constraints of the on-premises world, such as packaged software delivery models, significant upfront investments in infrastructure and software licensing, and long lead times required to build and deploy any application platform. They bring the same mindset to Azure and spend a lot of time building a single platform based on a single Azure service that can host as many of their applications as possible, if not all. Then they try to force-fit all their applications into this single platform, introducing delays and roadblocks that could have been avoided. But there is a better approach that is possible in Azure that will produce better returns on investment (ROI).

An A+B mindset simply means instead of limiting yourself to a predetermined service, you choose the service(s) that best meet your application needs; you choose the right tool for the right job.

A+B mindset applies to any application written in any language.

As you transition to Azure, where you provision and deprovision resources on an as-needed basis, you don't have to choose between A or B. Azure makes it easy and cost effective to take a different approach, the A+B approach. Azure enables you to shift your thinking from an A or B to an A+B mindset, which has many benefits, as explained later in this book. An A+B mindset simply means instead of limiting yourself to a predetermined service, you choose the service(s) that best meet your application needs; you choose the right tool for the right job.

As organizations expand their decision-making process and technical strategy from an A or B mindset to encompass the possibilities and new opportunities offered with an A+B mindset, there are many new considerations. In this book, we introduce the principles of the A+B mindset that you can use to choose the right Azure services for your applications. We have illustrated the A+B approach using Azure Spring Apps (formerly known as Azure Spring Cloud) and Azure Kubernetes Service (AKS) as examples; however, you can apply these principles to evaluate *any number of Azure Services* for hosting your applications (Azure App Service, AKS, Azure Container Apps, Azure Spring Apps, and Virtual Machines are commonly used Azure Services for application hosting). A+B mindset applies to *any application, written in any language* though we used Java Spring Boot Apps as examples for the purpose of this book.

Why organizations think A or B

The following are some reasons why organizations think A or B:

- **Resource optimization:** Before the cloud, a typical organization handled all layers of its application stacks, which translated to a significant upfront investment in hardware, software licensing, skilling up teams, hiring subject matter experts (SMEs), configuration, deployment, and operations. Conventional wisdom says it is more economical to invest in a single platform and use it for as many applications as possible (if not all) than to invest in multiple platforms.
- **External influence:** Partners, system integrators (SIs), and internal teams pushing for solutions that drive value for themselves in the near- and long-term rather than making decisions that are best for their customers.
- **Compliance:** Many software applications are required to meet compliance and regulatory requirements based on industry and geography. In addition to this, organizations have their own internal governance and security standards that the applications must comply with. This leads to choosing what they perceive as the most compliant platform.
- **Culture:** Companies that do not have an agile and growth-mindset culture typically have long planning cycles and move slowly. They tend to pick one platform per technology for the entire company and spend more of their energy enforcing those frameworks across all the applications with little respect for the use cases that the applications are aiming at solving.
- **Knowledge gap:** Lack of deeper understanding of technologies and their application, incorrect assumptions, and a bias toward the familiar. For example, some are confused between Spring Boot, Spring Cloud, and Azure Spring Apps (more details in the Feature comparison—Azure Spring Apps and Azure Kubernetes Service for Spring Boot applications section). It is important to have a good understanding of what each technology is meant for and its application before making your platform choice.

Conventional wisdom says it is more economical to invest in a single platform and use it for as many applications as possible (if not all) than to invest in multiple platforms.

- **Generic is better than opinionated:** Some companies prefer generic platforms over opinionated ones as a technology strategy and therefore have a bias for generic platforms. As a result, they go with that one generic platform that they think will satisfy most, if not all, use cases.
- **Politics:** Politics among different organizations of a company and the desire to control also contribute to an organization enforcing a platform on their application teams as opposed to doing an objective analysis based on the application needs.
- **Job preservation, complacency:** Those who have developed a certain level of knowledge and experience in a particular platform have a sense of security and are hesitant to move away from the familiar and give room to new. Complacency and a lack of innovative thinking also contribute to companies holding onto the platform they are comfortable with for as long as possible.
- **We have always done it that way—tyranny of the familiar:** Comfortable with the familiar, fear of change, and unwillingness to take risks also contribute to the bias toward as few platforms as possible.

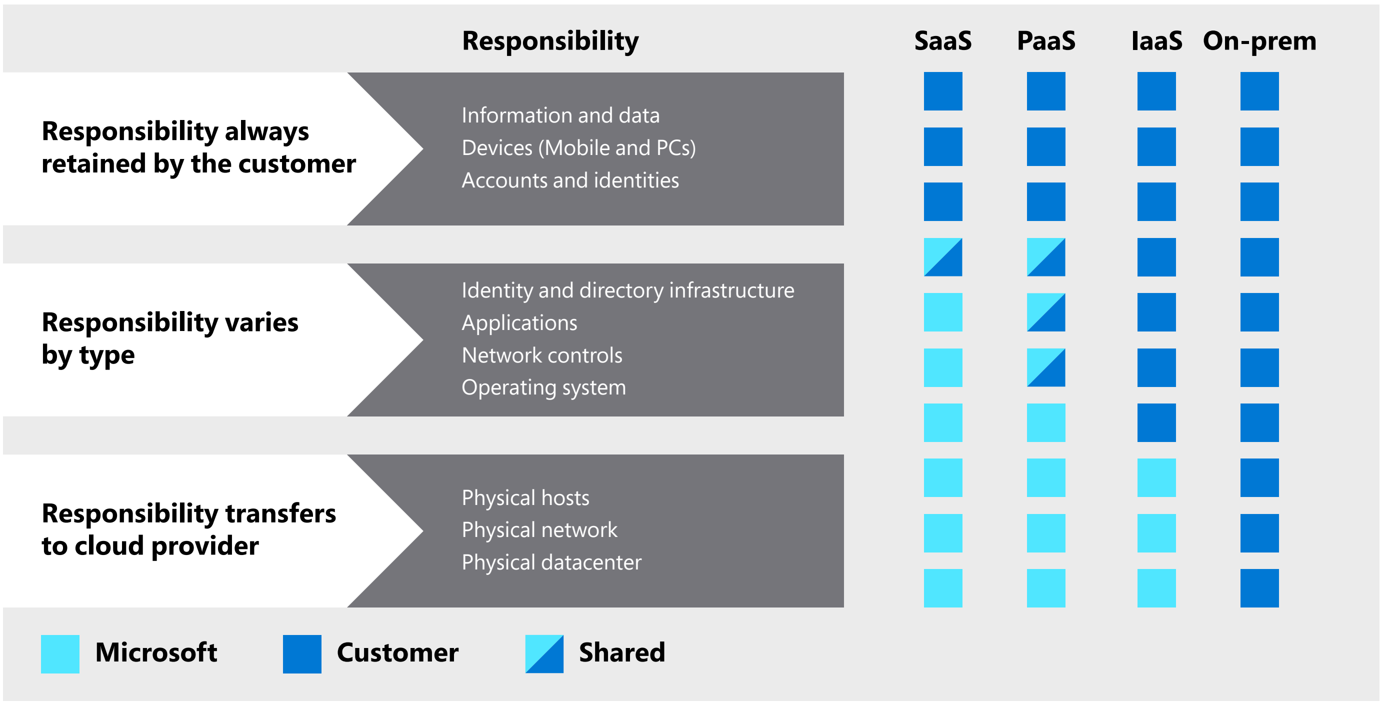
Organizations tend to bring some of the previously mentioned mindsets to the cloud and therefore try to pick a single platform for all applications rather than choosing what is right for the application.

Why it is not necessary to think A or B

While some of the reasons are behavioral—not technical, and should be addressed by a culture and mindset change, the key constraints to embracing the A+B mindset on-premises are the long lead times to build and deploy software platforms (like application servers and databases), the huge upfront investments in hardware, and potentially, the higher costs involved with purchasing, deploying, and maintaining multiple application-hosting solutions. A good architecture will not ignore resource and cost optimization, so naturally, A or B thinking made sense for on-premises, but the cloud changes everything.

Cloud is on-demand and elastic, with no upfront capital expenditures—you only pay for what you use. You can scale your apps up or down based on your requirements. You can provision and deprovision resources in the cloud very easily compared to on-premises.

As you move to Azure, a significant portion of the infrastructure and platform responsibility is transferred to Microsoft, as seen in Figure 1. The effort involved in provisioning and operating any two Azure services is typically comparable with each other.



	Responsibility	SaaS	PaaS	IaaS	On-prem
Responsibility always retained by the customer	Information and data	■	■	■	■
	Devices (Mobile and PCs)	■	■	■	■
	Accounts and identities	■	■	■	■
Responsibility varies by type	Identity and directory infrastructure	◩	◩	■	■
	Applications	□	◩	■	■
	Network controls	□	◩	■	■
	Operating system	□	□	■	■
Responsibility transfers to cloud provider	Physical hosts	□	□	□	■
	Physical network	□	□	□	■
	Physical datacenter	□	□	□	■

□ Microsoft ■ Customer ◩ Shared

Figure 1: Shared responsibility model between customer and cloud provider

Benefits of the A+B mindset

The benefits of the A+B mindset are quite intuitive. Customers can use the right tool for the right job, avoid unnecessary work involved in force-fitting the use case to a predetermined solution, and achieve greater agility, higher cost efficiency, faster time to market, and operational excellence.

Since you do not need to worry about upfront investments and additional costs associated with owning and managing multiple platforms, you now have the luxury to use the right tool for the right job. Now you can choose the platform that best fits your application needs, rather than trying to force-fit every application into the same platform.

For example, with regard to AKS versus Azure Spring Apps for Spring Boot applications, you do not have to choose between one or the other but use the technology that is best suited to solve the use case at hand. Moreover, we have noticed an increasing adoption of microservices in the recent past. Microservices naturally promote technology diversity because a key tenant of microservices is to use the best tool for the job; consequently, A+B is a natural outcome of adopting a microservices mindset.

Benefits of the A+B mindset

The benefits of the A+B mindset are quite intuitive. Customers can use the right tool for the right job, avoid unnecessary work involved in force-fitting the use case to a predetermined solution, and achieve greater agility, higher cost efficiency, faster time to market, and operational excellence.

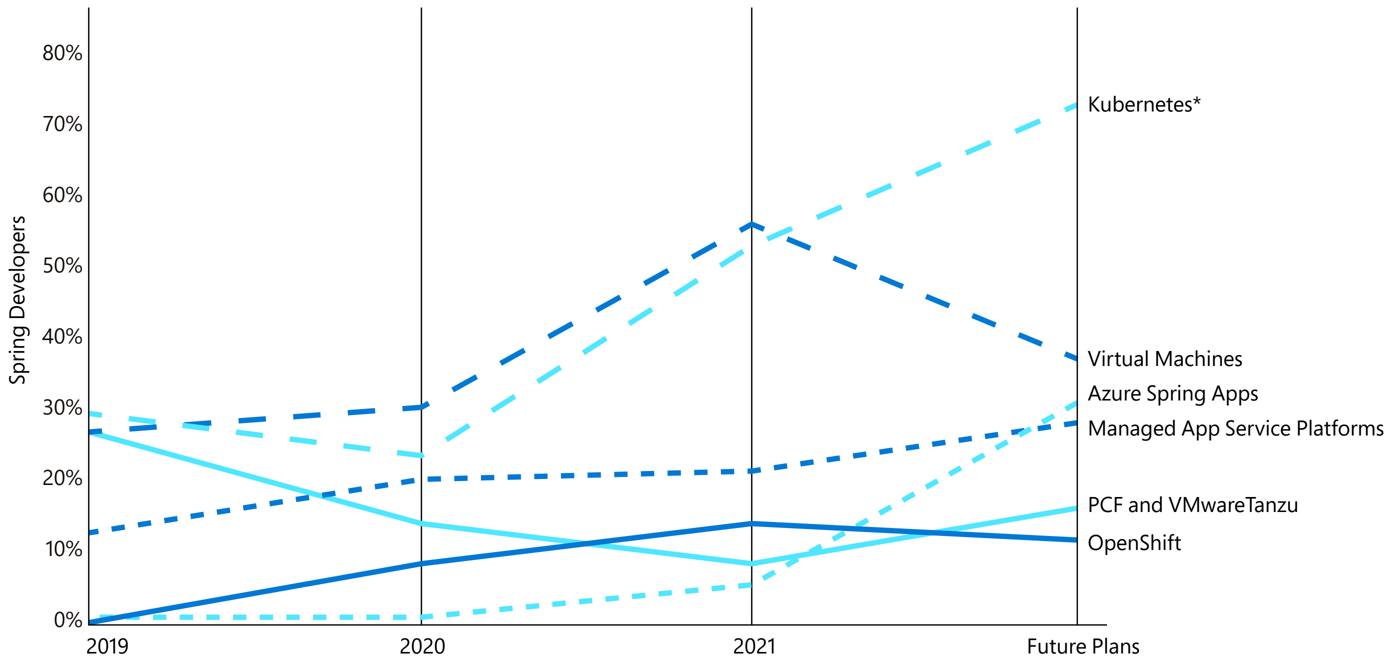
Current industry trends

Many organizations are gradually transitioning from an A or B mindset to an A+B mindset when it comes to choosing Azure services for hosting their applications.

One out of five organizations said they are willing to use more than one platform

In a survey conducted during the 2021 SpringOne conference, 21% of participants said they plan to use more than one type of platform for their Spring Boot apps. Many indicated plans to move away from VMs to Kubernetes or Platform-as-a-Service (PaaS).

Platforms Targeted for Spring Boots Apps



● One out of five organizations said they would use more than one platform

● *50% plan to use a managed Kubernetes service

● *36% plan to use Kubernetes on virtual machines

Figure 2: Target platforms for Spring Boot apps

J.B. Hunt migrated their on-premises apps to multiple Azure services

J.B. Hunt is an innovator in North American trucking and a provider of transportation, end-to-end shipping, and logistics services. J.B. Hunt 360°® is its cloud-based digital service that seamlessly connects shippers and carriers, matching the right loads with the right trucks. Originally, J.B. Hunt 360° ran on an on-premises distributed architecture backed by mainframe databases, supported by more than 80 internally built application services, most written in Java. To scale the platform for up to 100,000 third-party carriers, J.B. Hunt wanted to migrate those services to Azure.

First, they launched a hybrid-cloud version of J.B. Hunt 360° by migrating 20 core services to VM scale sets and other services to the Web Apps feature of Azure App Service. Then, to create a full cloud offering, the team began moving the remaining services to Azure Kubernetes Service (AKS) with Kubernetes for orchestration.

By first evaluating the specific needs of each unique service as well as the business requirements, budgets, and timelines, J.B. Hunt used a variety of Azure services, including VM scale sets, Azure App Service, and AKS, to migrate and modernize their applications. Find more on J.B. Hunt in the Examples of applying the A+B mindset to AKS and Azure Spring Apps section.

To scale the platform for up to 100,000 third-party carriers, J.B. Hunt wanted to migrate those services to Azure.

Barracuda's strategy is to use the Azure service that best meets their need

Barracuda provides email and data protection and application, cloud, and network security to more than 220,000 customers worldwide, in addition to its on-premises and cloud backup solutions. The company's customers wanted to be able to find sensitive, personally identifiable information stored in Microsoft 365. So, in collaboration with Microsoft and building on top of Microsoft Azure services, Barracuda developed Data Inspector, which uses AI to discover sensitive categories of information to make compliance easier.

If an Azure service can perform or augment a function that we need, we use it.

Andy Blyler, vice president of engineering for data inspector at Barracuda.

"Our primary customers and partners are Microsoft users, so when we first started designing the solution, we chose to use Azure as much as we could," says Andy Blyler, vice president of engineering for data inspector at Barracuda. "If an Azure service can perform or augment a function that we need, we use it."

The solution relies on a core set of Azure services, including Azure Cosmos DB, Azure Blob Storage, Azure Functions, Azure App Service, AKS, Azure Pipelines, Azure DevOps, Azure Front Door, and Azure Content Delivery Network. To extract text from images, Data Inspector uses Azure Cognitive Services and its optical character recognition (OCR) capabilities. Using all these Azure services, Barracuda was able to get to market faster and invest engineering resources in building customer value into the solution.

This is another example of forward-thinking companies embracing the A+B mindset for faster innovation, greater speed to market, and better customer value.

Practical guidance for transitioning to the A+B mindset

This section enumerates some key principles that can be used as a guideline for transitioning to the A+B mindset and continuing with it:

- **Go from use case to solution, not the other way around:** Often, many software teams decide on technology first and then try to force-fit the use cases and design, thus in many cases incurring a significant overhead in terms of cost, development time, resources, and operational expenses. Get clarity on your use cases, and functional and non-functional requirements before jumping into the solution.
- **Understand your business goals, nature of business, your competition, and how often you need to roll out new features to production:** Your solution should always be designed to meet your business goals and objectives.

In the age of the cloud, where everything is accessed over the internet, security is crucial and non-negotiable.

- **Understand security and compliance requirements:** In the age of the cloud, where everything is accessed over the internet, security is crucial and non-negotiable. In addition to this, depending on the industry you serve, your application may need to meet certain compliance requirements. You must design your solution to weather advanced security attacks and to meet your compliance requirements.
- **Understand your budget and timelines:** Have a clear understanding of your budget for the initial development, ongoing operations, and future releases. Additionally, understand your timelines—the cost of delayed projects, both in terms of additional expenses and negative business impact is often underestimated. Design your solution to meet both your budget and timeline.

- **Think cloud-native where applicable:** Cloud-native architecture and technologies are an approach to designing, constructing, and operating workloads that are built in the cloud and take full advantage of the cloud computing model. With cloud-native, you will be able to build and deploy applications to production at a much faster rate. The cloud also provides capabilities that might not be possible on-premises, for example, elasticity, global scale, advanced analytics, AI, and ML capabilities. Design your solution based on cloud-native technologies as much as possible.
- **Think DevOps culture:** DevOps is not just tools or processes; it is a software development practice that promotes collaboration between development and operations, resulting in faster and more reliable software delivery. Commonly referred to as a culture, DevOps connects people, processes, and technology to deliver continuous value.
- **Choose the solution that meets your business and nonfunctional requirements**, one that is:
  a. Fastest to implement.
  b. Cost-effective in terms of costs involved for skilling up, building, deployment, and operations.
  c. Easy to operate.
  d. Fully compatible with automation.
  e. Supportive of DevOps by design.

These principles will help you keep your focus where it should be—on building a solution that meets your business goals rather than force-fitting the solution to a predetermined platform.

Examples of applying the A+B mindset to AKS and Azure Spring Apps

In this section, we will consider a few real-world applications and apply the principles of the A+B mindset and see whether they land in Azure Spring Apps or Azure Kubernetes Service (AKS).

Please note that Azure Spring Apps and AKS are used only as examples; the same approach can be used to evaluate any set of Azure services.

AKS simplifies deploying a managed Kubernetes cluster in Azure by offloading the operational overhead to Azure. As a hosted Kubernetes service, Azure handles critical tasks, like health monitoring and maintenance. Since Kubernetes masters are managed by Azure, you only manage and maintain the agent nodes. Thus, AKS is free; you only pay for the agent nodes within your clusters, not for the masters. See the Feature comparison—Azure Spring Apps and Azure Kubernetes Service for Spring Boot applications section for more information.

Azure Spring Apps makes it easy to deploy Spring Boot applications to Azure without any code changes. The service manages the infrastructure of Spring Boot applications so developers can focus on their code. Azure Spring Apps provides life cycle management using comprehensive monitoring and diagnostics, configuration management, service discovery, continuous integration and continous deployment (CI/CD) integration, blue-green deployments, and more.

Bosch

For our first case study, we will consider two business-critical applications, Track and Trace and RIoT, developed by Bosch, a world-renowned manufacturer and a leader in IoT solutions. These applications are currently in production, running on Azure. We will see how the A+B mindset can be applied to these applications to determine the right destinations in Azure for these applications.

a. **Track and Trace** is an IoT solution that provides real-time information on the location and load status of shipping assets, such as load carriers, truck trailers/semitrailers, locomotives, railroad cars, and ocean freight containers.

Requirements:

- The application consists of several microservices that process data received from wireless tags attached to the assets that are tracked.
- The team is focused on development, and they want to use a major share of their capacity for product development. They do not have the interest or capacity to get involved in managing Kubernetes or building container images.
- The team wanted as little build and deployment overhead as possible. They previously worked with Cloud Foundry and its build packs and wanted a similar experience.
- Scalability is a huge factor because Track and Trace is growing and replacing many more tracking devices in the field.
- Security and monitoring are undisputable requirements enforced by corporate IT.

Now let's apply the *principles of the A+B mindset* that are relevant to this use case, to find out which service is the best fit for this use case:

- **Go from use case to solution:** The key requirements to consider for this use case are:
  1) The application requires a microservices platform.
  2) The team wants to focus on development and they do not have the desire or capacity to manage Kubernetes or a containerization process.
  3) They want minimum deployment overhead.
  4) Security and monitoring are minimum requirements.

  Requirement 2 makes Azure Spring Apps a good choice for this application, but we still need to make sure Azure Spring Apps satisfactorily meets all the application requirements. As stated previously, Azure Spring Apps is secure, scalable, and a fully managed platform as a service (PaaS) for running Spring Boot applications and microservices. It provides full life cycle management—customers only need to bring their Java ARchive (JAR) file and they are good to go with the deployment—and it supports comprehensive monitoring and diagnostics. This makes Azure Spring Apps a good option for running the Track and Trace application; however, we need to run through the rest of the A+B mindset criteria to make sure it checks those boxes as well.
- **Understanding business goals and nature of business:** Track and Trace expects to grow rapidly, so it requires a platform that can scale as the business grows. Azure Spring Apps is a fully managed solution with the ability to scale dynamically. Thus Azure Spring Apps supports this requirement.

- **Understand security and compliance requirements:** Security and compliance requirements can be fully met by using Azure security features such as Azure Virtual Network and subnets for network segregation, managed identities, key vaults, data encryption, and enforcing security policies during provisioning using infrastructure as code. Azure Spring Apps also provides best practices for securely deploying Spring Boot apps on Azure Spring Apps.
- **Cost and timelines:** Based on the approach to cost calculation in the Feature comparison — Azure Spring Apps and Azure Kubernetes Service for Spring Boot applications section, Azure Spring Apps is very cost-effective; however, cost is not the primary deciding factor for this use case.

Azure Spring Apps is a fully managed solution with the ability to scale dynamically.

Walking through the decision criteria, we find that Azure Spring Apps meets the business requirements, supports rapid implementation and deployment, is cost-effective and easy to operate, enables automation, and supports DevOps.

Based on the A+B analysis, Azure Spring Apps is the right destination for this application.

b. **RIoT** (**Residential IoT Services GmBH**), a start-up created by Bosch, built a smart home platform called the RIoT platform that provides back-end services for the Home Connect Plus app that the home users use to connect, automate, and centrally control various smart home devices from a single user interface. The RIoT team already chose Azure as their cloud and wanted to find the right destination in Azure for their microservices-based RIoT platform.

Requirements:

- The development team went with a distributed and highly scalable microservices architecture for this application and decided to base their implementation on the Distributed Application Runtime (Dapr) framework, which makes building microservices much easier.
- The RIoT platform needed to efficiently support a large number of concurrent users within the first year of launch and grow from there.
- They wanted the ability to scale out subsystems that require more resources without scaling out the entire application.

- The services need to be reliable and highly available.
- The development team wanted an abstraction layer so they don't have to learn about individual Azure services like Service Bus that run under the hood to facilitate service to service communication. They also wanted the ability to switch underlying components between environments; for example, use Cosmos DB in production but use Redis in a local development environment to store the state.

Let's apply the *principles of the A+B mindset* that are relevant to this use case:

- **Go from use case to solution:** Since the development team based their microservices implementation on Dapr, the key consideration for this would be which Azure service is best suited for hosting Dapr-based microservices. AKS is the clear answer; however, we still need to validate AKS against the rest of the principles of the A+B mindset.
- **Understanding business goals and nature of business:** The RIoT platform must support many concurrent users and scale as their customer base grows. AKS supports running multiple instances of a microservice concurrently, thus supporting many concurrent users. AKS can also scale up or scale down the number of microservice instances based on demand.
- **Understand security and compliance requirements:** Security and compliance requirements are not explicitly mentioned, but it is assumed that they are table stakes. AKS and the Azure platform supports best-in-class security.
- **Cost and timelines:** AKS is very cost-effective; however, cost is not a primary deciding factor for this use case.

“ AKS and the Azure platform supports best-in-class security.

Walking through the decision criteria, we find that AKS meets the application requirements, supports rapid implementation and deployment, is cost-effective and easy to operate, enables automation, and supports DevOps.

Based on the preceding analysis, AKS is the right Azure service for this application.

Note: In the rest of the case studies, we have analyzed only one application per company to decide whether it lands in AKS or Azure Spring Apps. This does not mean that we are recommending a single Azure service for all applications. Every application must undergo A+B analysis to find its destination in Azure.

Digital Realty

Digital Realty supports the data center colocation, and interconnection strategies of customers across the world. It operates over 290 data centers in 24 countries and six continents. Digital Realty wanted to build a new customer-facing platform that allowed its customers to build integrations and consume data about Digital Realty's services.

Requirements:

- Customer-facing application with approximately 90 microservices in scope. Has global customers, requires global access with low latency.
- Team did not have the desire or skills to manage infrastructure (Kubernetes clusters).
- Wanted to start development almost immediately without long runways for setting up infrastructure, networking, and Kubernetes clusters.
- Wanted a globally scalable solution that scales quickly and easily.
- Wanted automated global deployments.
- Wanted real-time end-to-end monitoring and the ability to use external monitoring tools for advanced monitoring capabilities.
- Best-in-class security with zero trust, plus the use of existing investments in F5 and Palo Alto.
- Support heavy loads from across the globe.
- Rapid business growth, ability to roll out new features rapidly

Now let's apply the *principles of the A+B mindset* that are relevant to this use case, to find out which service is the best fit for this use case:

- **Go from use case to solution:** The key requirements to consider in deciding which platform is better suited for this application are:
  1) The team did not have the desire or skills to manage AKS clusters.
  2) Rapid business growth and the ability to roll out new features rapidly.

  Clearly, the only option left is Azure Spring Apps, but we need to make sure that Azure Spring Apps meets all the application requirements. Azure Spring Apps is a secure, scalable, and fully managed multiregional service that comes with a rich set of monitoring capabilities and the ability to integrate with external monitoring tools, thus meeting the key requirements.
- **Understanding business goals and nature of business:** We see from the requirements that Digital Realty anticipates rapid business growth, and they require the ability to scale the solution as they add new customers. They also want the ability to add new features rapidly. Azure Spring Apps can scale dynamically and comes with CI/CD integration, which makes rolling out new releases effortless.
- **Understand security and compliance requirements:** Security and compliance requirements can be fully met by using Azure security features such as Azure Virtual Networks and subnets for network segmentation, support for third-party firewalls like F5 and Palo Alto, which can be used to front the applications running on Azure Spring Apps, managed identities, access policies, RBAC, encryption and a key vault for storing keys and secrets, and code scanning in release pipelines for credentials and vulnerabilities.
- **Cost and timelines:** Based on the approach to cost calculation in the Feature comparison—Azure Spring Apps and Azure Kubernetes Service for Spring Boot applications section, Azure Spring Apps is very cost-effective for the 90 microservices that are in scope. Timelines were already addressed in the second point previously.

Walking through the decision criteria, we find that Azure Spring Apps checks all the boxes—supports rapid implementation and deployment, is cost-effective and easy to operate, enables automation, and supports DevOps.

Based on the preceding analysis, Azure Spring Apps is the best destination for this application.

Kroger

The Kroger In-Stock application was Kroger's #1 application in terms of business impact in 2019, significantly impacting the revenue in terms of recovered lost sales. The In-Stock application that was running on-premises on Pivotal Cloud Foundry (hosted on VMs) had scale challenges. They wanted to scale their application to support several more stores than the on-premises solution could support.

Requirements:

- The application has 25+ services and 20+ databases.
- Scalable solution to increase the active store capacity from 150 to 2,500+ stores.
- Production-to-production migration. The application was in production and cannot have any outages during normal operations.
- Securely connect to on-premises data sources. Secure access from in-store mobile apps.
- Kroger has a cloud-first corporate strategy.
- Monitoring dashboard post-deployment in the cloud to make sure the application is up and running.

Let's apply the *principles of the A+B mindset* to this use case:

- **Go from use case to solution:** The key requirements to consider in terms of deciding which Azure service to use are:
  1) Scalability: Kroger wanted to scale this application over 4x from 625 to 2,500+ stores.
  2) The existing solution is based on Pivotal Cloud Foundry.
  3) Requires a monitoring dashboard post-production.

  While both AKS and Azure Spring Apps could meet the requirements, since the application is already built on Pivotal Cloud Foundry, Azure Spring Apps is a better natural fit than AKS. Azure Spring Apps also meets their other requirements—it can scale dynamically and provides extensive monitoring and dashboarding through integration with Azure Monitor—and as with the other two use cases, Azure Spring Apps also comes with the additional advantage of not having to manage the infrastructure.

- **Understanding the goals and nature of business:** This is a retail application used by the store employees to quickly identify the in-stock status and location of a product in-store. The key requirement is the ability to scale from a few stores to all stores. Azure Spring Apps's dynamic scaling ability will meet this requirement.
- **Understand security and compliance requirements:** Azure provides many security controls such as network isolated deployment of Azure Spring Apps inside an Azure Virtual Network, Azure Firewall for managing egress, Azure Front Door for ingress, Azure Key Vault for storing secrets, and ExpressRoute for private connectivity to on-premises data sources, using which the security requirements can be met.
- **Cost and timelines:** Based on the approach to cost calculation in the Feature Comparison — Azure Spring Apps and Azure Kubernetes Service for Spring Boot Applications section, Azure Spring Apps is very cost-effective for this solution as well. There are no specific requirements on the timeline, so we will assume that it is not one of the deciding factors.

Walking through the decision criteria, we find that Azure Spring Apps meets the primary requirements. Its ability to scale dynamically meets the scalability requirement. Azure Monitor supports extensive monitoring and dashboarding across all Azure services. You can prebuild the dashboard prior to go-live and continuously monitor pre, during, and post-go-live to make sure the application is fully operational.

Based on the preceding analysis, Azure Spring Apps is the best destination for this application.

Azure Monitor supports extensive monitoring and dashboarding across all Azure services.

AIA Singapore

AIA Singapore is a subsidiary of AIA Group Limited, a leader in life insurance and financial services with branches and subsidiaries across 18 Asia-Pacific markets. AIA has several Java as well as some .NET applications that are running on servers on-premises. To account for unexpected spikes in traffic, they had to over-provision their on-premises hardware, incurring significant cost overhead. They wanted to migrate to the cloud to achieve more cost efficiency, performance enhancement, and the ability to accelerate the delivery of new and innovative solutions to their customers.

Requirements:

- AIA has several enterprise Java applications, including their flagship iPoS application, that are hosted on application servers like JBoss EAP. They also have a few ASP.NET applications. AIA is looking to quickly modernize these applications and host them in the cloud to gain more cost efficiency and performance enhancements.
- AIA wants a cloud platform that not only supports modernizing existing apps but also that helps them rapidly build and deploy new applications, accelerating the delivery of new and innovative solutions to their customers.
- Nine to 12 months to first production workload.
- AIA does not have any constraints regarding deploying and managing AKS clusters.

Let's apply the *principles of the A+B* mindset to this use case:

- **Go from use case to solution:** The key requirements to consider in terms of deciding which Azure service to use are:
  1) Achieving cost efficiency, scalability, and performance enhancements for existing applications.
  2) Ability to rapidly build and deploy new and innovative solutions.
  3) Existing Java apps are primarily enterprise Java apps, with most of them being hosted on JBoss EAP.
  4) They also have non-Java applications.

Containerization is a proven and the quickest way to achieve higher application density, thus more cost efficiency for existing applications. AKS is a container orchestrator platform that makes it extremely easy to deploy, run, manage, and scale container-based apps. Moreover, AIA has Java apps that are not necessarily Spring/Spring Boot apps, and in addition to that, they have other non-Java applications. So, in this instance, AKS is a better choice than Azure Spring Apps. Customers can quickly containerize their existing apps and run them on AKS.

- **Understanding the goals and nature of business:** Innovation and delivering customer value rapidly is at the forefront of AIA's business strategy. In addition to cost and performance efficiency, AIA also wanted to rapidly deliver new and innovative solutions to their customers. With infrastructure as code, CI/CD, and built-in Kubernetes echo system tools like kubectl and Helm charts, AIA can automate both the build and deployment of their applications, thus accelerating the time to market.
- **Understand security and compliance requirements:** There are no specific security requirements mentioned; however, best-in-class security requirements are assumed, which AKS meets.
- **Cost and timelines:** Based on the approach to cost calculation in the Feature comparison—Azure Spring Apps and Azure Kubernetes Service for Spring Boot applications section, customers can compute the cost for AKS clusters depending on the number of cluster instances that are required across the regions of their choice. AKS is very cost-effective, and the high application density should provide additional cost savings compared to an on-premises solution. AKS is a managed solution and can be spun up on demand. Customers can containerize the existing apps without having to rewrite code and deploy them on AKS within nine to 12 months.

Walking through the decision criteria, we find that AKS meets their primary requirements; it is cost-effective compared to on-premises and easy to operate, and it supports automation and DevOps.

Based on the preceding analysis, AKS is the best destination for this application.

J.B. Hunt

J.B. Hunt, an innovator in North American trucking for more than 50 years, has expanded from being a transportation provider to being an end-to-end shipping and logistics provider. J.B. Hunt 360° is the company's technology platform, which hosts multiple application services that provide diverse services to shippers and carriers. J.B. Hunt 360° runs on an on-premises distributed architecture backed by mainframe databases, supported by more than 80 internally built application services, mostly written in Java with Jenkins automation. To scale the platform for up to 100,000 third-party carriers, J.B. Hunt wanted to migrate those services to Azure.

Requirements:

- J.B. Hunt wanted to scale their existing Java-based services to 100,000 carriers.
- J.B. Hunt wanted to continue to use Jenkins and Azure DevOps for CI/CD automation.
- They wanted to migrate these services to Azure quickly and efficiently.

Let's apply the *principles of the A+B mindset* to this use case:

- **Go from use case to solution:** The key requirements to consider in terms of deciding which Azure service to use are:
  1) Scaling existing application services.
  2) Minimal to no code refactoring.
  3) These are Java services but not necessarily Spring or Spring Boot applications.
  4) Need to retain existing CI/CD tools.

  As discussed before, containerization is a proven and the quickest way to achieve higher application density as well as scalability. As we know, AKS is a container orchestrator platform that makes it extremely easy to deploy, run, manage, and scale container-based apps. So, in this instance, AKS is a better choice than Azure Spring Apps.
- **Understanding the goals and nature of business:** The key business objective is to scale multiple carriers. AKS is a scalable container platform, and it provides multiple levers to scale based on the load at either the pod level or the node level. So, AKS is a good fit for this use case.

- **Understand security and compliance requirements:** There are no specific security requirements mentioned; however, best-in-class security requirements are assumed, which AKS meets.
- **Cost and timelines:** There are no specific requirements on the timeline, so we will assume that they are not the deciding factors.

Walking through the decision criteria, we find that AKS meets their primary requirements; it is cost-effective compared to on-premises and easy to scale, and it supports automation and DevOps.

Based on the preceding analysis, AKS is the best destination for this application.

Exceptions

Like anything else, there are exceptions to A+B. This is not an exhaustive list but will provide you with directional guidance on some exceptions that you might encounter:

- **Enterprise strategy:** For example, enterprise-wide adoption of containers to build and deploy applications because they may have multiple programming languages at play, and they want to build and deploy all applications in a unified manner.
- **Too far down the line with execution:** You may have chosen a solution before going through the A+B analysis. If you are already deep into execution of your solution, continue with it but for the next application use the principles of A+B mindset to choose the right solution for your use case.
- **Large scale data center migrations:** To accelerate their journey to the cloud, enterprises commonly use a strategy called "lift and shift" that involves migrating servers (hosting their applications) in bulk to Azure using tools like Azure Migrate. Some use this approach to migrate data centers to Azure and shut them down in an efficient and cost effective manner. In this scenario, we recommend using A+B mindset to modernize applications after migrating to Azure.

If you are already deep into the execution of your solution, continue with it, but for the next application, use the principles of the A+B mindset to choose the right solution for your use case.

Conclusion

In this book, we provided you with the framework for thinking and the principles that you can use to choose the right destinations in Azure for your applications.

**It's not one size fits all;
it's not A or B, but A+B.**

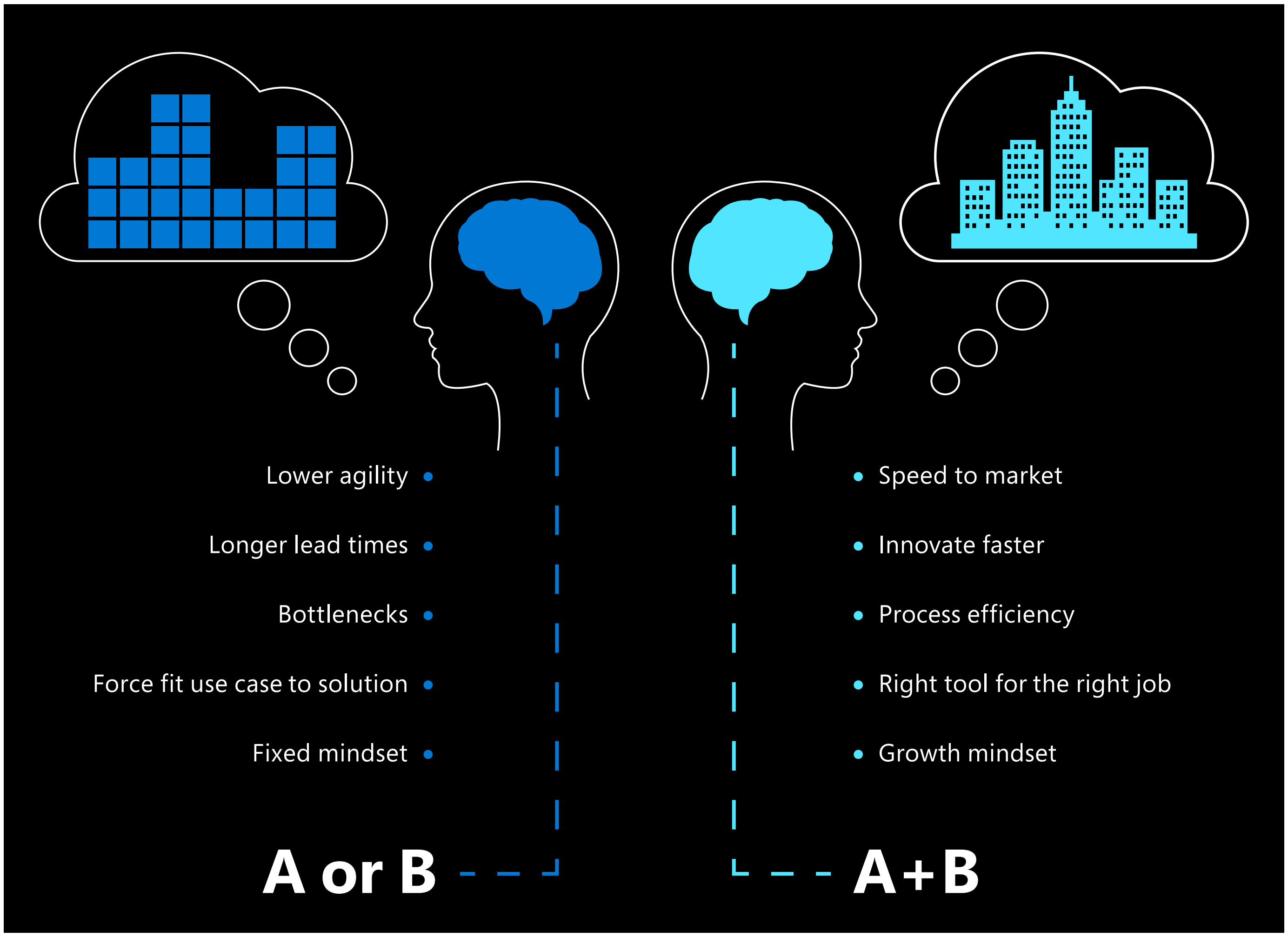


Figure 3: A or B versus A+B mindset

Key takeaways

Don't put the cart before the horse
—understand the requirements before considering the solution.

It's not A or B
—choose the Azure service that best meets the requirements of your application.

Look for a win-win
—focus and rely on your strength, which is building business applications, and let Microsoft take care of running the platform and infrastructure for you so that you can be agile and go to market faster than ever.

Note: As you run your applications through the principles of an A+B mindset, you might find that the same Azure service meets the A+B criteria for several of your applications. That is the best scenario. We are not advising against using a single platform; we are just asking you to make sure you are picking the Azure service that best meets your business goals.

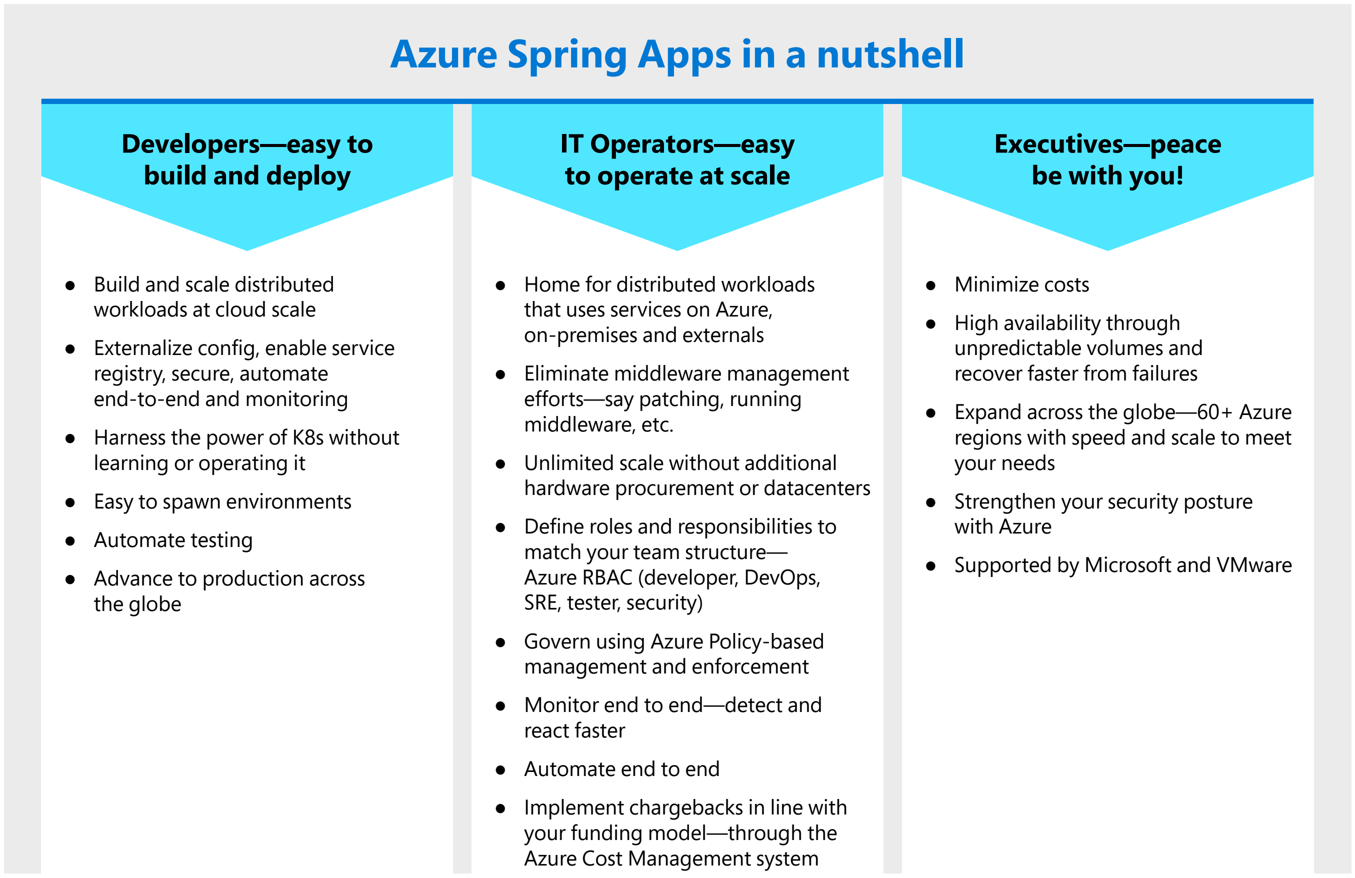
Note: The principles outlined in this book should not be misconstrued as an argument against reusability. We are not against "build once, reuse many times"; in fact, you will reuse many Azure platform features like Azure Monitor and Microsoft Defender for Cloud (formerly Azure Security Center) across all your applications. We strongly recommend using Azure landing zones to build your foundation in Azure once and reuse or extend it across multiple applications. The reusability at the platform level is what enables you and sets you free to use the A+B mindset to choose the right Azure service for hosting your application.

Feature comparison—Azure Spring Apps and Azure Kubernetes Service for Spring Boot applications

If you are interested in learning more about Spring Boot apps on Azure, this section provides a detailed comparison of Azure Kubernetes Service and Azure Spring Apps for hosting Spring Boot applications.

Azure Spring Apps in a nutshell

Azure Spring Apps is a fully-managed PaaS service for hosting any Spring Boot app—from monolith to microservices. Backed by a fully managed AKS cluster, it provides a truly serverless experience in a cost-effective manner, for building, deploying and managing your Spring Boot applications in the cloud.



Azure Spring Apps in a nutshell

Developers—easy to build and deploy

- Build and scale distributed workloads at cloud scale
- Externalize config, enable service registry, secure, automate end-to-end and monitoring
- Harness the power of K8s without learning or operating it
- Easy to spawn environments
- Automate testing
- Advance to production across the globe

IT Operators—easy to operate at scale

- Home for distributed workloads that uses services on Azure, on-premises and externals
- Eliminate middleware management efforts—say patching, running middleware, etc.
- Unlimited scale without additional hardware procurement or datacenters
- Define roles and responsibilities to match your team structure—Azure RBAC (developer, DevOps, SRE, tester, security)
- Govern using Azure Policy-based management and enforcement
- Monitor end to end—detect and react faster
- Automate end to end
- Implement chargebacks in line with your funding model—through the Azure Cost Management system

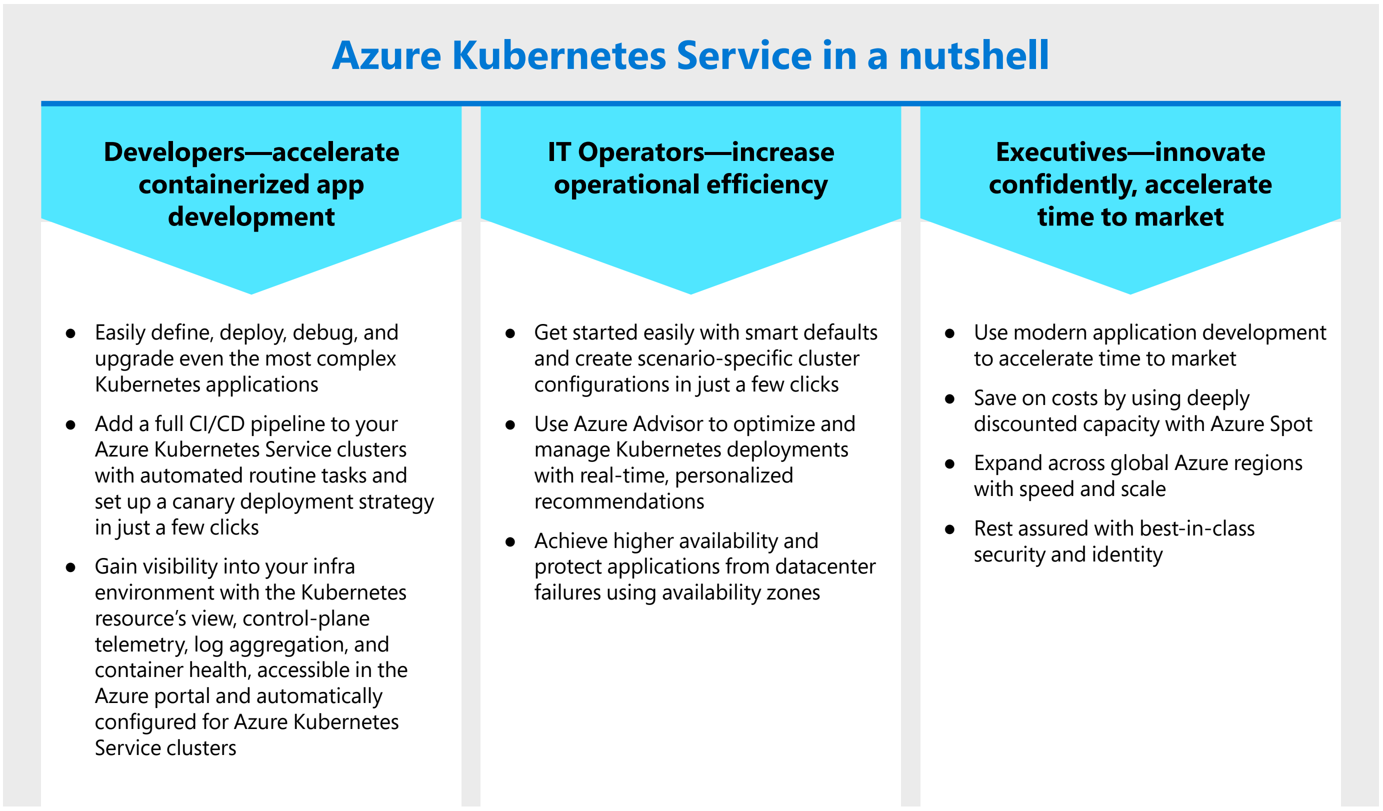
Executives—peace be with you!

- Minimize costs
- High availability through unpredictable volumes and recover faster from failures
- Expand across the globe—60+ Azure regions with speed and scale to meet your needs
- Strengthen your security posture with Azure
- Supported by Microsoft and VMware

Figure 4: Azure Spring Apps in a nutshell

AKS in a nutshell

Azure Kubernetes Service is a managed Kubernetes service for hosting containerized applications. With AKS you can rapidly build,deploy, and scale applications cost effectively, with confidence.



Azure Kubernetes Service in a nutshell

Developers—accelerate containerized app development

- Easily define, deploy, debug, and upgrade even the most complex Kubernetes applications
- Add a full CI/CD pipeline to your Azure Kubernetes Service clusters with automated routine tasks and set up a canary deployment strategy in just a few clicks
- Gain visibility into your infra environment with the Kubernetes resource's view, control-plane telemetry, log aggregation, and container health, accessible in the Azure portal and automatically configured for Azure Kubernetes Service clusters

IT Operators—increase operational efficiency

- Get started easily with smart defaults and create scenario-specific cluster configurations in just a few clicks
- Use Azure Advisor to optimize and manage Kubernetes deployments with real-time, personalized recommendations
- Achieve higher availability and protect applications from datacenter failures using availability zones

Executives—innovate confidently, accelerate time to market

- Use modern application development to accelerate time to market
- Save on costs by using deeply discounted capacity with Azure Spot
- Expand across global Azure regions with speed and scale
- Rest assured with best-in-class security and identity

Figure 5: AKS in a nutshell

Cost calculations

This section provides directional guidance on how to calculate the cost of hosting Spring Boot applications on AKS and Azure Spring Apps. In this example, we will be calculating costs for a single region. For production, typically, you will have at least one more region for disaster recovery. You can apply the same logic for each region and compute the total cost for your application.

You can use the Azure Pricing Calculator to estimate the cost of your services. Go to Pricing Calculator, search for Azure Spring Apps, and choose the appropriate options for your service. Repeat the same process for AKS.

In calculating the cost, you want to consider the following base application parameters:

- **Number of applications you want to run:** Indicates the number of Spring Boot applications that you plan to run.
- **Average CPU per app:** Indicates the number of virtual CPU (vCPU) cores required for your app.
- **Average memory per app:** Indicates the amount of memory required for your app.
- **Monthly usage hours:** Indicates the number of hours per month that you will run your apps.

Azure Spring Apps: As shown in the example below, for Azure Spring Apps you only need to input the total number of vCPUs and total memory you require across all your applications.

Azure Kubernetes Service: For Azure Kubernetes Service there are some additional considerations for computing the cost for the AKS cluster that hosts your Spring Boot applications.

- **Right-sizing the cluster:** You are responsible for right-sizing the cluster by mapping the total vCPUs and memory required by your apps to the number of VMs in the AKS cluster. For example, if you have 10 apps each requiring 2 vCPUs and 2 GB of memory, you will need at least 3 D8s v4 VMs (each VM has 8 vCPUs and 32 GB RAM). In addition to this, you will need to allocate one or more VMs for resiliency depending on your desired number of failures to tolerate (FTT).
- **VM resource overhead:** An AKS cluster uses some of the cluster node's compute capacity for cluster operations. In addition to that, you will also need some compute capacity to host Spring middleware, and other components like ingress control. In summary, you will need more compute to accommodate Spring middleware, ingress control, resiliency, and VM fragmentation, and you will need to estimate the VM overhead accordingly.
- **Cost of labor:** You will also need to include the additional labor cost for the initial setup and configuration of the cluster and ongoing operations.

Example cost calculation

In this section, we'll work through an example cost calculation:

Application parameters:

- Number of app Instances: 100
- Monthly usage in hours: 730
- Average vCPU per app: 1
- Average memory (GiB) per app: 1
- Region: US East 2

Azure Spring Apps parameters:

- Azure Spring Apps pricing tier: Standard

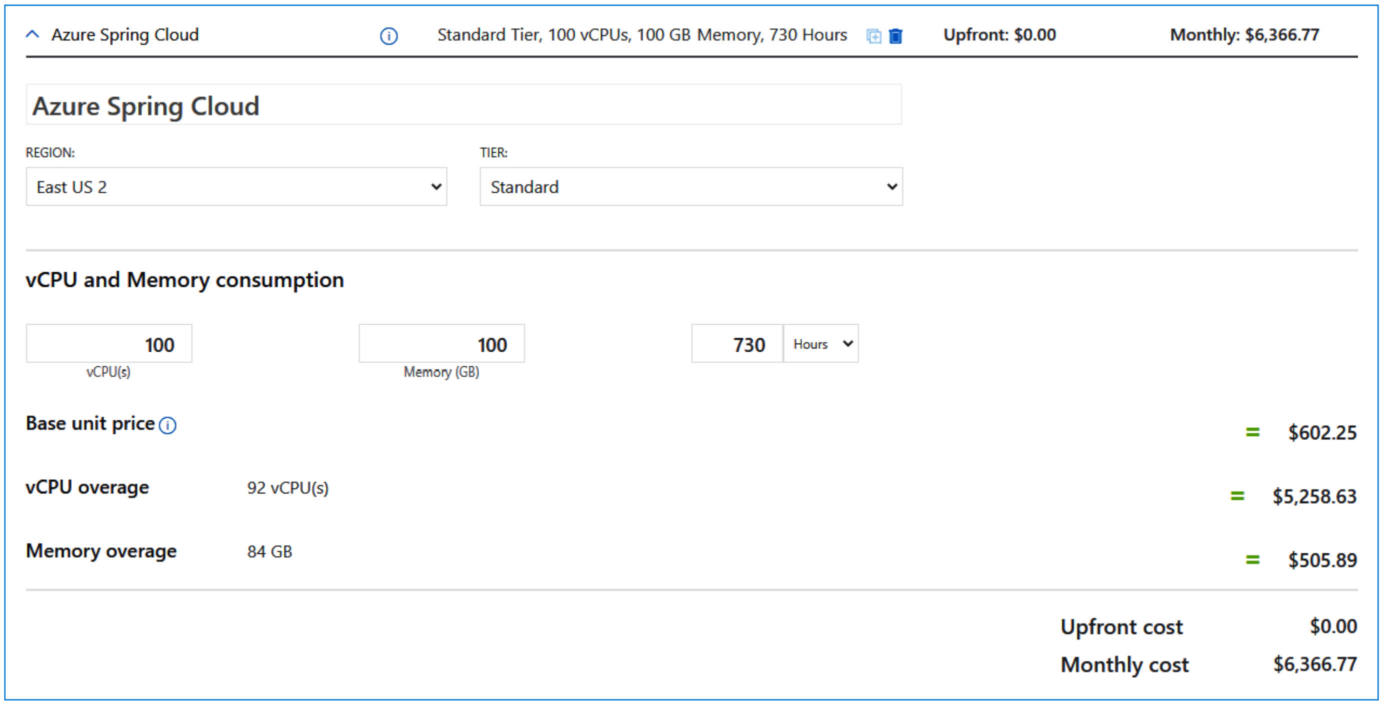
AKS parameters:

- Number of AKS clusters :1
- VM SKU for AKS Cluster: D8 v3
- Approximate VM overhead:
  Node Reservation: 20%
  Config server: 2%
  Registry: 2%
  Ingress: 2%
  VM Fragmentation: 20%
  Resiliency: 2 FTT
  Labor cost for operation: This example uses the standard salary for a Kubernetes Admin in the USA.

Now that we have our parameters, we can begin calculating the cost.

Azure Spring Apps: We need a total of 100 vCPUs and 100 GiB for the applications. Plugging these numbers into Azure Pricing Calculator, we see that the Azure Spring Apps monthly cost is $6,337.

100 app instances in Azure Spring Apps


Azure Spring Cloud — Standard Tier, 100 vCPUs, 100 GB Memory, 730 Hours — Upfront: $0.00 — Monthly: $6,366.77

Azure Spring Cloud

REGION: East US 2

TIER: Standard

vCPU and Memory consumption

100 vCPU(s)

100 Memory (GB)

730 Hours

Base unit price		= $602.25
vCPU overage	92 vCPU(s)	= $5,258.63
Memory overage	84 GB	= $505.89

Upfront cost	$0.00
Monthly cost	$6,366.77

Figure 6: Cost for 100 application instances in Azure Spring Apps

AKS:

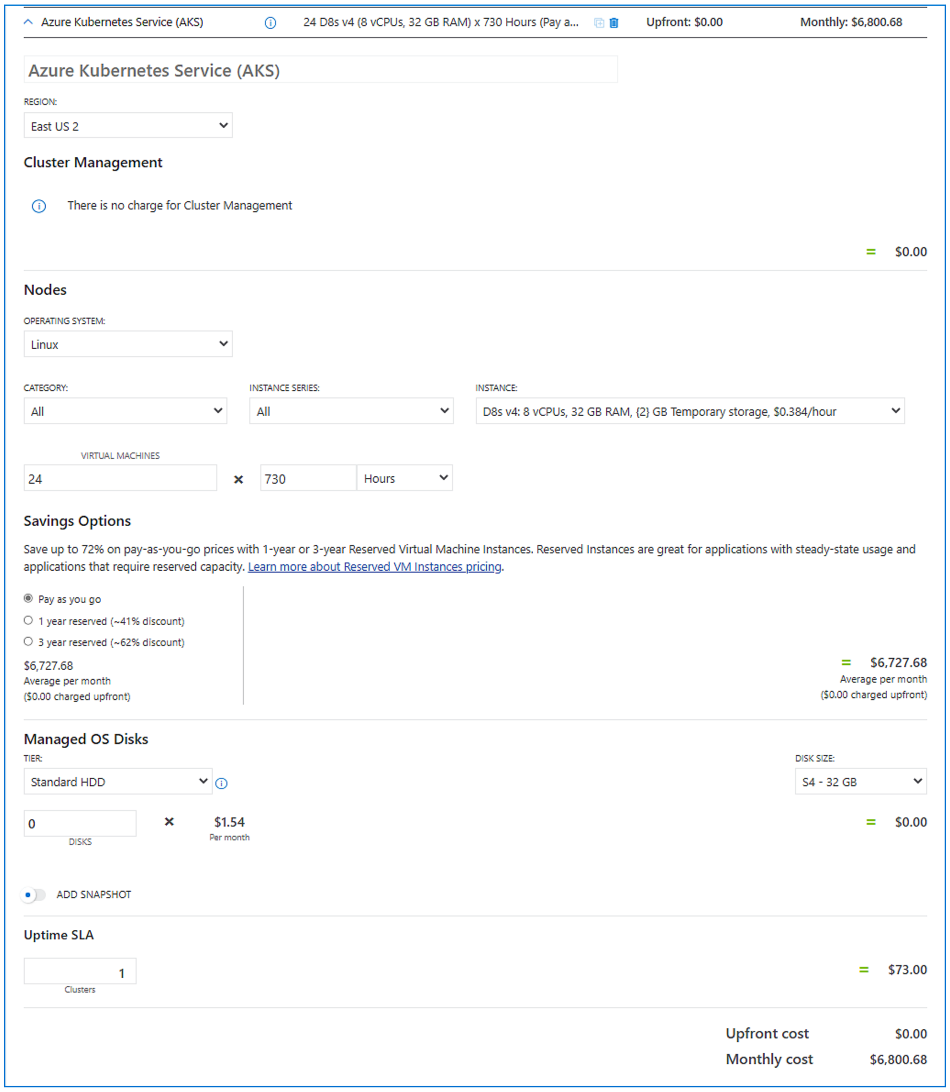
- A D8 v3 comes with 8 vCPUs and 32 GB RAM.
- For a total of 100 vCPUs, we will need at least 12.5 nodes (VMs).
- For a total of 100 GB memory, we will need at least 3.125 nodes.
- So, the total number of nodes required is max (12.5,3.125) = 12.5.
- If 20% of the total is consumed for defragmentation, the total number of VMs before defragmentation will be 12.5/(1-0.2) = 15.625; rounding up, it will be 16 nodes.

Let's add the rest of the VM overhead:

- Reservation + Ingress + Config + Registry = (0.2 + 0.1 + 0.02 + 0.02)*16 = 5.44.
- Adding FTT to this, we have 5.44 + 2 = 7.44. Let's round it up to 8.
- So, the total number of VMs required would be 16 + 8 = 24.

Plugging this into Azure Pricing Calculator, we have an estimated cost of $6,801.

100 app instances in Azure Kubernetes Service



Azure Kubernetes Service (AKS) | 24 D8s v4 (8 vCPUs, 32 GB RAM) x 730 Hours (Pay a... | Upfront: $0.00 | Monthly: $6,800.68

Azure Kubernetes Service (AKS)

REGION: East US 2

Cluster Management

There is no charge for Cluster Management

= $0.00

Nodes

OPERATING SYSTEM: Linux

CATEGORY: All
INSTANCE SERIES: All
INSTANCE: D8s v4: 8 vCPUs, 32 GB RAM, (2) GB Temporary storage, $0.384/hour

VIRTUAL MACHINES: 24 × 730 Hours

Savings Options

Save up to 72% on pay-as-you-go prices with 1-year or 3-year Reserved Virtual Machine Instances. Reserved Instances are great for applications with steady-state usage and applications that require reserved capacity. Learn more about Reserved VM Instances pricing.

[✓ Pay as you go]
1 year reserved (~41% discount)
3 year reserved (~62% discount)

$6,727.68
Average per month
($0.00 charged upfront)

= $6,727.68
Average per month
($0.00 charged upfront)

Managed OS Disks

TIER: Standard HDD
DISK SIZE: S4 - 32 GB

0 DISKS × $1.54 Per month = $0.00

ADD SNAPSHOT

Uptime SLA

1 Clusters = $73.00

Upfront cost $0.00
Monthly cost $6,800.68

Figure 7: Cost for 100 app instances in AKS

The final cost we need to add is the labor cost for AKS operations. According to ZipRecruiter, the annual salary of a Kubernetes Admin is $105K/year. The total cost for a salaried employee is typically 68% salary and 32% benefits, bringing the total salary to $147K/year. Assuming 1 unit of labor can operate 800 app instances, the monthly labor cost would be ((annual salary/12)*no of app instances)/800 = ((147,000/12)*100))/800 = $1,531. Thus, the total cost of AKS would be 6,801 + 1,531 = $8,332/month.

In summary, for 100 app instances that require 1 vCPU and 1GB memory:

Azure Spring Apps cost = $6,367 per month

Azure Kubernetes Service cost = $8,332 per month

These are directional costs assuming a simple configuration. The actual cost may vary depending on the exact configuration. The reason Azure Spring Apps came out less expensive is because of economies of scale. Azure Spring Apps has dozens of dedicated engineers managing it as well as the AKS clusters that Azure Spring Apps runs on. The cost of our engineers managing the clusters is distributed across thousands of clusters, which results in a much lower cost per engineer per cluster compared to our customers managing the clusters themselves.

The following chart shows AKS versus Azure Spring Apps costs for 50, 100, 500, 1,000, and 2,000 app instances.

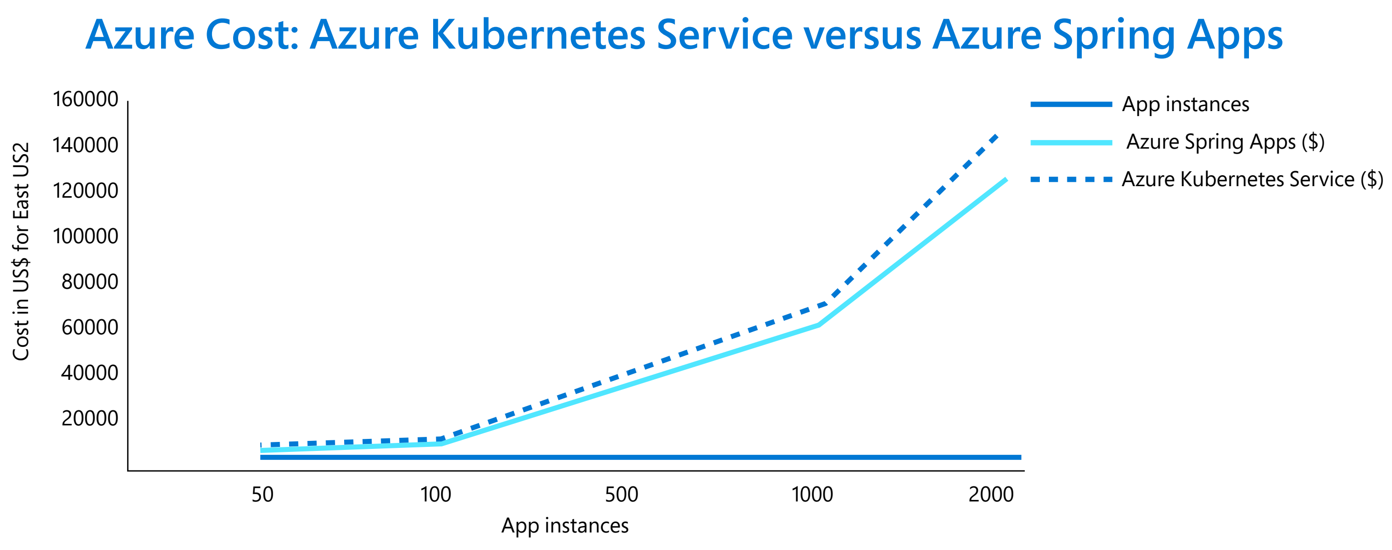


Figure 8: Azure Spring Apps and AKS—cost variance by number of instances

Comparing the user experiences for Spring Boot applications —AKS and Azure Spring Apps

This section provides additional information on what your experience will look like in **AKS versus Azure Spring Apps for Spring Boot applications** and how you can choose from one of the two options.

When to use what

	Azure Spring Apps	Azure Kubernetes Service
What workloads?	For Spring Boot applications	For containers—general purpose
When should the customer use it?	Prefers to focus on business	Requires full control
Pricing estimate	$600/month—pay as you go	Pay for infra Pay for uptime SLA

Table 1: AKS versus Azure Spring Apps for Java Spring applications

In Figure 8, you can see a simple decision tree to guide you in picking the right Azure Service for Spring Boot applications. Does your business require infrastructure control? If not, move down to the managed services path, Azure Spring Apps.

If you need infrastructure control, the application destination hinges on your skillsets or your desire to acquire skillsets to manage AKS clusters and container images or outsource them to SIs or partners.

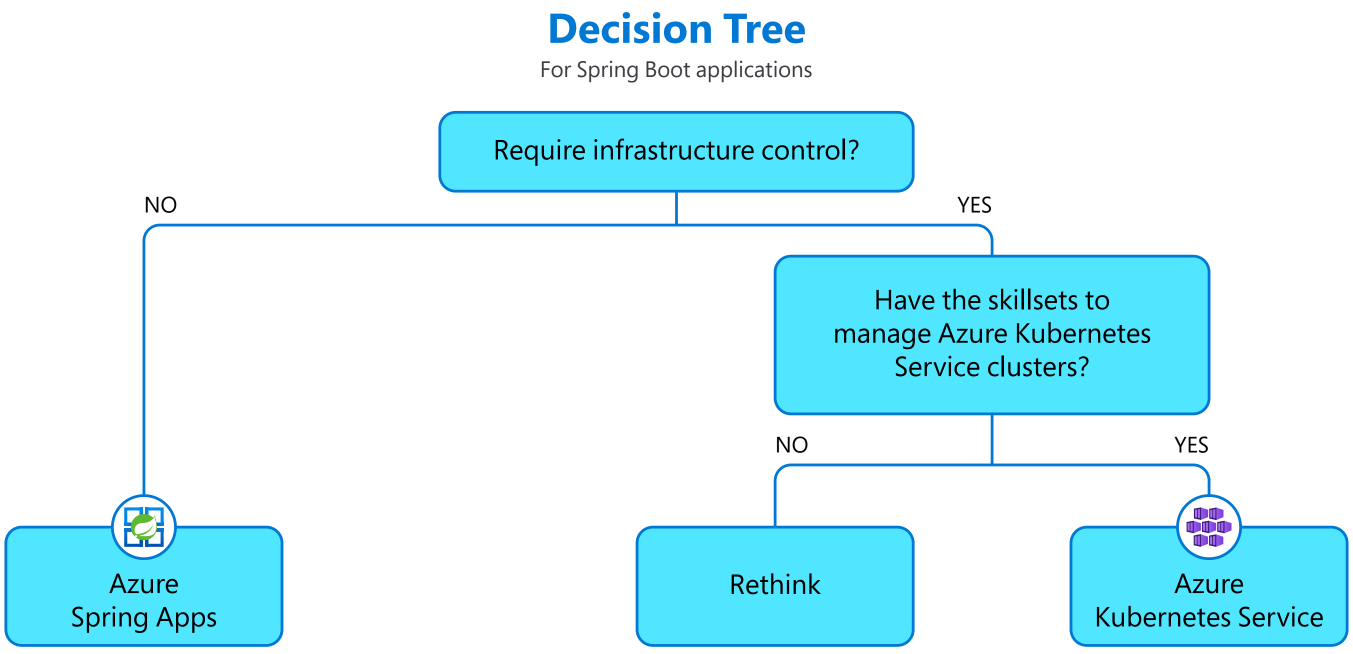

Decision Tree
For Spring Boot applications
Require infrastructure control?
NO
YES
Have the skillsets to manage Azure Kubernetes Service clusters?
NO
YES
Azure Spring Apps
Rethink
Azure Kubernetes Service

Figure 9: AKS and Azure Spring Apps—decision tree

Examples of requiring infrastructure control: Having business requirements to manage everything from top to bottom, including ingress to Kubernetes to underlying VMs and requirements to manage and patch those, or having software that depends on the Kubernetes API or needs interactions using kubectl.

Spring Boot applications on Azure Spring Apps versus AKS —infrastructure management

With AKS, the customer is responsible for managing some infrastructure components as well as Spring Cloud middleware components. Azure Spring Apps manages these infrastructure components transparent to the customer.

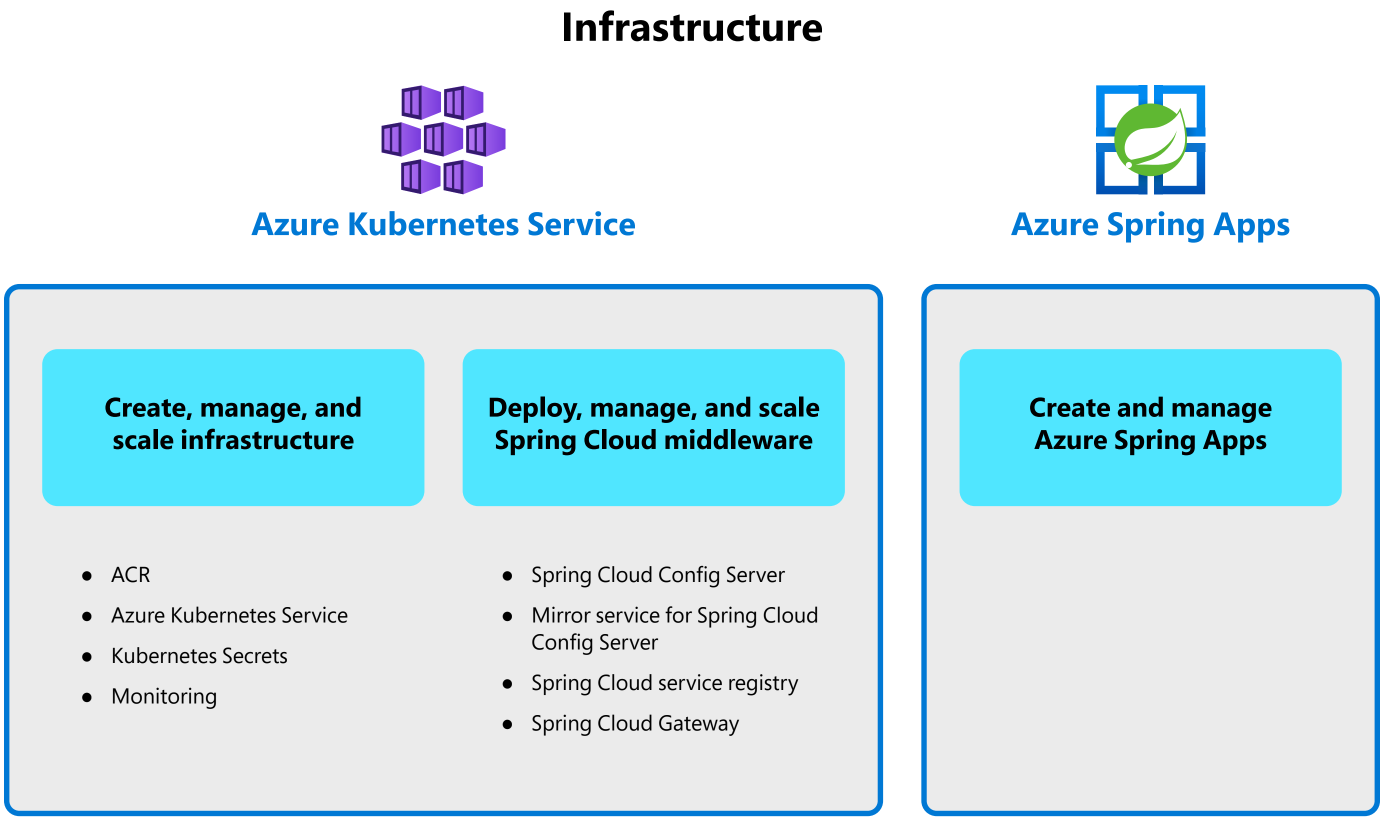
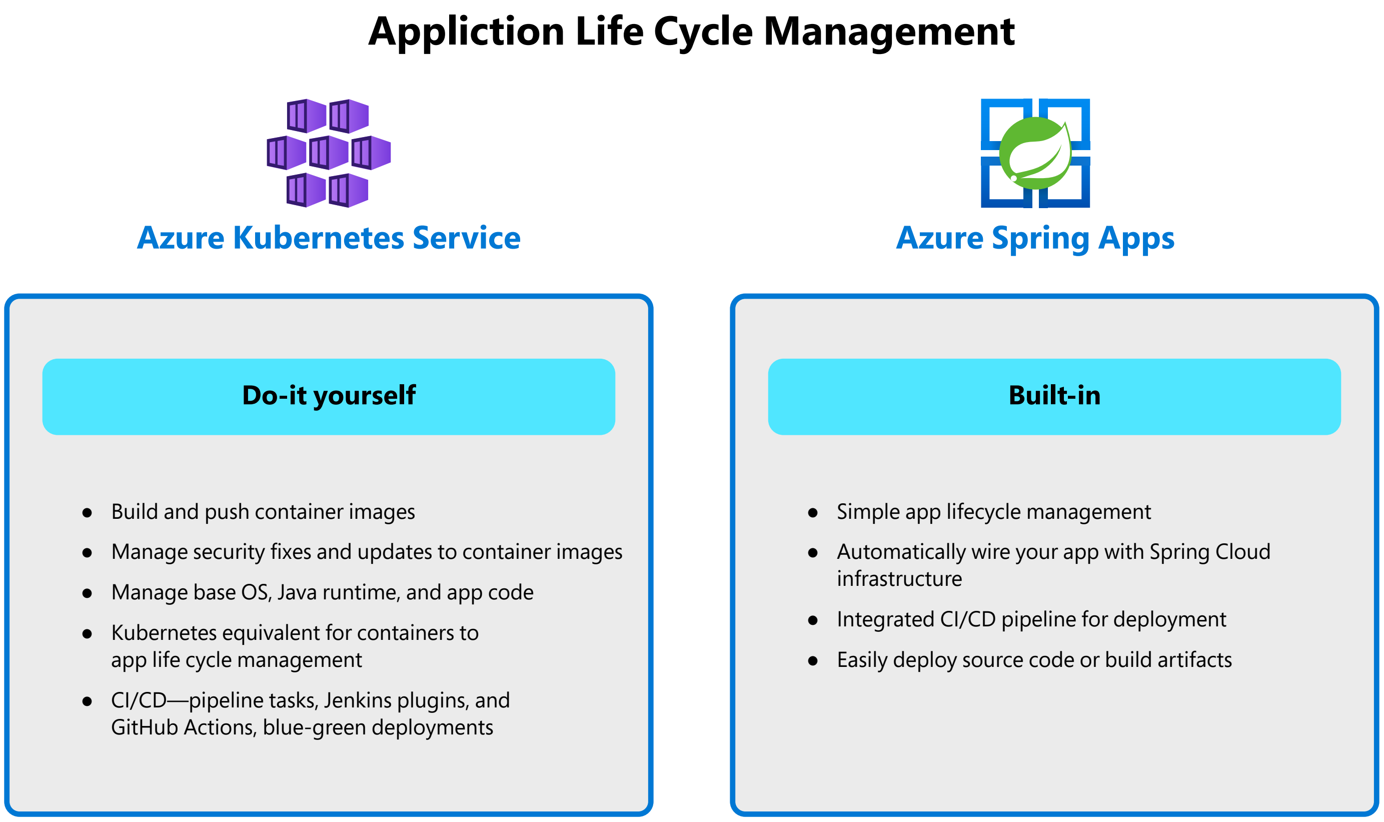

Infrastructure
Azure Kubernetes Service
Azure Spring Apps
Create, manage, and scale infrastructure
Deploy, manage, and scale Spring Cloud middleware
Create and manage Azure Spring Apps
ACR
Azure Kubernetes Service
Kubernetes Secrets
Monitoring
Spring Cloud Config Server
Mirror service for Spring Cloud Config Server
Spring Cloud service registry
Spring Cloud Gateway

Figure 10: AKS and Azure Spring Apps—infrastructure management responsibilities

Spring Boot applications on Azure Spring Apps versus AKS—application life cycle management

As shown in Figure 10, Azure Spring Apps simplifies application life cycle management to a great extent.



Appliction Life Cycle Management

Azure Kubernetes Service

Do-it yourself

- Build and push container images
- Manage security fixes and updates to container images
- Manage base OS, Java runtime, and app code
- Kubernetes equivalent for containers to app life cycle management
- CI/CD—pipeline tasks, Jenkins plugins, and GitHub Actions, blue-green deployments

Azure Spring Apps

Built-in

- Simple app lifecycle management
- Automatically wire your app with Spring Cloud infrastructure
- Integrated CI/CD pipeline for deployment
- Easily deploy source code or build artifacts

Figure 11: AKS and Azure Spring Apps—application life cycle management responsibilities

Spring Boot applications on Azure Spring Apps versus AKS—monitoring

Monitoring is a built-in capability in Azure Spring Apps, whereas some additional installations, integrations, and configurations for monitoring are required for AKS.

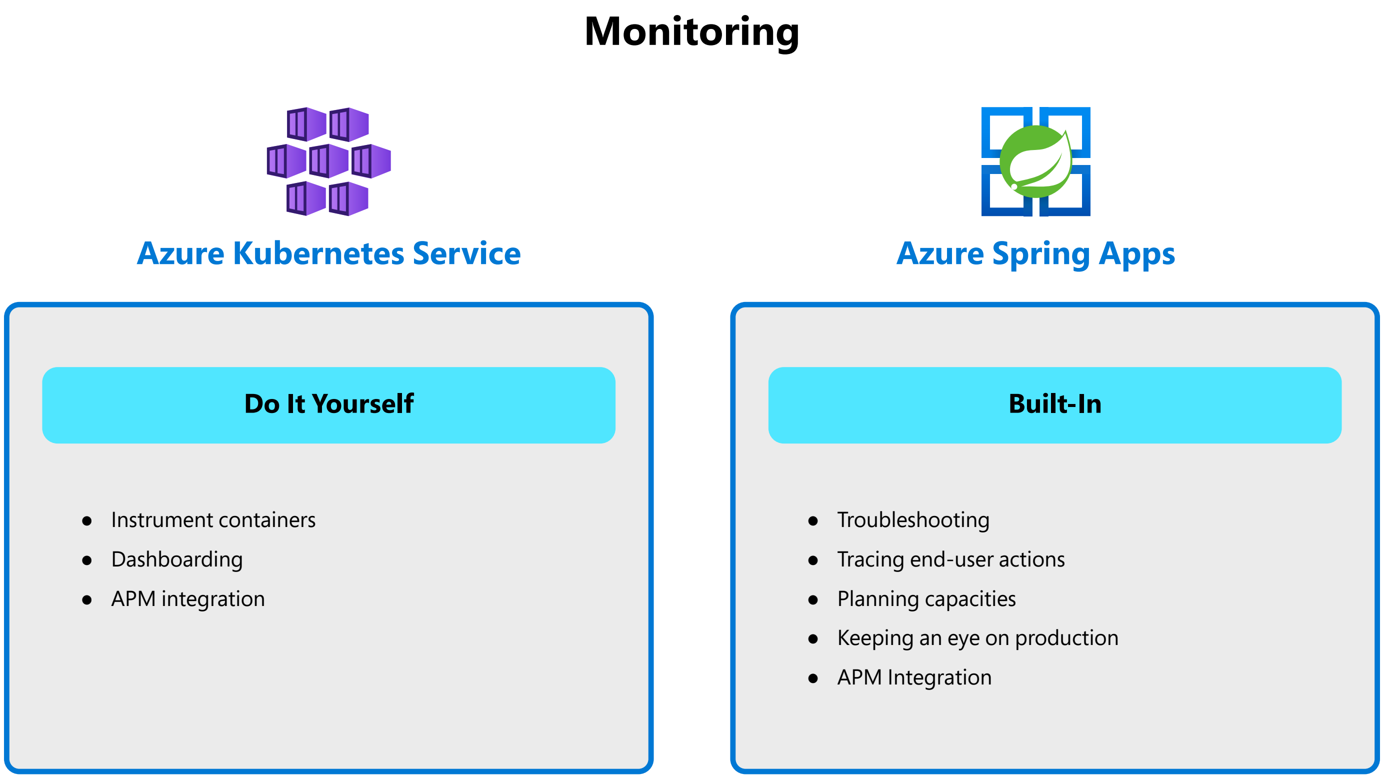


Figure 12: AKS and Azure Spring Apps—monitoring responsibilities

Feature grid—deploy + secure + scale + monitor + SLA

As discussed earlier, enabling Spring Runtime on AKS, securing, and operationalizing it involves configuring several additional capabilities. For example, in AKS, you are responsible for deploying and managing Spring Cloud Config Server, high availability for microservices, and custom domain names, whereas Azure Spring Apps provides all the capabilities out of the box.

Feature	Azure Spring Apps	Azure Kubernetes Service
Spring Cloud Runtime		
Managed Spring Cloud Config Server	Azure	☞
Mirror Service for Spring Cloud Config Server	Azure Enterprise Tier	☞
Managed Spring Cloud Service Registry	Azure	☞
Managed Spring Cloud Gateway	Azure Enterprise Tier	☞
Service essentials		
High availability for apps and Spring Cloud Runtime—99.9	Azure	☞
Service connectors for Azure data, cache, messaging, and directory services	Azure	☞
Autoscale in or out apps	Azure	☞
Autopatching— service, Spring Cloud Runtime, and Java Runtime Engine	Azure	☞
Azure Container Registry	Azure	☞
.NET apps—Steeltoe	Azure	☞

Key

☞ customer responsibility

Table 2: AKS and Azure Spring Apps—feature grid

Feature	Azure Spring Apps	Azure Kubernetes Service
Monitoring		
Monitoring—logs, metrics, and alerts	Azure	☞
Integration with Application Insights—performance, failures, live metrics stream, application map, and distributed tracing	Azure	☞
Self-diagnostics as a service	Azure	☞
Networking		
Custom domain	Azure	☞
Integration with Traffic Manager, Application Gateway, and Azure Front Door	Azure	☞
VNet for isolating apps from the internet	Azure	Azure
VNet for accessing on-premises resources	Azure	Azure
VNet for placing apps on corporate networks	Azure	Azure
SLA and support		
SLA for apps on Spring Cloud	Azure	☞
Support for Spring Cloud (and Spring family)	Azure	☞

Key

☞ customer responsibility

Table 2: AKS and Azure Spring Apps—feature grid, continued

Feature	Azure Spring Apps	Azure Kubernetes Service
Security		
Managed identity	Azure	☞
RBAC for granting access to Azure Spring Apps resources	Azure	☞
Integration with Key Vault	Azure	☞
TLS for consumer to app communications (BYOC—bring your own certificates)	Azure	☞
TLS for app to Spring Cloud Runtime	Azure	☞
TLS for app-to-app communications (BYOC)	Azure	☞
TLS for app-to-external resource communications (BYOC)	Azure	☞
End-user AuthN and AuthZ using AAD and AAD B2C (via "gateway")	Azure Enterprise Tier	☞
Redacting secrets and personally identifiable information from logs	Azure	☞
Encryption at REST—Microsoft key	Azure	Azure
Encryption at REST—BYOK	Azure	☞
Azure Security Center and Policy management integration	Azure for apps and services	Azure for containers

Key

☞ customer responsibility

Table 2: AKS and Azure Spring Apps—feature grid, continued

Feature grid—developer experiences and automation

You will have a difference in developer experience and automation between Azure Spring Apps and AKS. As you can see in the following table, most of the activities, like JAR to containers and enabling application performance monitoring (APM), are handled for you by Azure Spring Apps.

Feature	Azure Spring Apps	Azure Kubernetes Service
Tooling—Maven, IntelliJ, Eclipse, and VS Code	Azure	Third parties
Log stream for dev and troubleshooting	Azure	K8s
Source | JAR to containers	Azure	☞
Configuring certificates for TLS (apps)	Azure	☞
Enabling APMs—New Relic, AppDynamics, Dynatrace, and others	Azure	☞
Scanning apps for vulnerabilities	Azure	☞
Building out Tomcat for WAR apps	Azure	☞
Inspect app dependencies, metadata, and audit	Azure Enterprise Tier	☞
DevOps automation using pipelines	Azure	☞
DevOps automation using GitHub Actions workflows	Azure	☞
Deploy apps without disruption—blue-green	Azure	☞
Azure Security Center and Policy management integration	Azure for apps and services	Azure for containers

Key

☞ customer responsibility

Table 3: AKS and Azure Spring Apps—developer experience feature grid

Ongoing responsibilities

Let's assume that you have a production system running Spring-based microservices. The following table shows the difference between Azure Spring Apps and AKS in terms of what it takes to keep the lights on.

Ongoing Responsibilities	Azure Spring Apps	Azure Kubernetes Service
Updating libraries*	☞	☞
Updating the Spring Cloud middleware components—Spring Cloud Config Server, Spring Cloud service registry, Spring Cloud Gateway, and so on*	Azure	☞
Updating the Java Runtime Engine*	Azure	☞
Triggering Kubernetes updates**	Azure	☞
Reconciling non-backward-compatible Kubernetes API changes	Azure	☞
Updating the container base image*	Azure	☞
Updating the operating system*	Azure	Azure
Detecting and restarting failed instances	Azure	Azure
Implementing draining and rolling restart for updates	Azure	Azure
Infrastructure management	Azure	☞
Monitoring and alerts	☞	☞

Key

☞ customer responsibility

*Includes vulnerability remediation
**Performed by Azure with a manual trigger but you are on-point for the downstream impact of Kubernetes updates

Table 4: AKS and Azure Spring Apps—ongoing responsibilities feature grid

Azure Spring Apps is a managed service for hosting any Spring Boot application—from a monolith to a microservice.

Spring Boot versus Spring Cloud versus Azure Spring Apps

Many are confused about the difference between Spring Boot, Spring Cloud, and Azure Spring Apps. Spring Boot is an open-source, Java-based framework for building apps cleanly and quickly. Spring Boot apps may have dependencies on the back end, such as data, cache, messaging, and eventing. Those dependencies are satisfied using modules like Spring Data, Spring Messaging, Spring Cloud Stream Binder, scalable Azure data, cache, messaging, and eventing services.

Optionally, Spring Boot apps can use Spring Cloud components to coordinate anything between applications. Spring Cloud provides mechanisms to facilitate scalable, survivable configuration, communication, and coordination among mission-critical systems of Spring Boot applications—using Spring Cloud components such as Spring Cloud Config, Spring Cloud Registry, and Spring Cloud Gateway.

Azure Spring Apps is a managed service for hosting any Spring Boot application—from a monolith to a microservice.

References

- **Bosch case study 1**—Accelerate Spring Apps to Cloud at Scale—Discussion with Azure Spring Apps Customers – YouTube
- **Bosch case study 2**—Bosch delivers supply chain efficiencies using Java on Azure
- **Bosch case study 3**—Microsoft Customer Story—Bosch builds smart homes using Dapr and Azure
- **Digital Realty case study 1**—Accelerate Spring Apps to Cloud at Scale—Discussion with Azure Spring Apps Customers – YouTube
- **Digital Realty case study 2**—Digital Realty powers global portal and REST APIs using Azure Spring Apps
- **Kroger case study**—Kroger shares how they Migrated their stock management Java application to Azure at Microsoft Build – YouTube
- **AIA Singapore case study**—Microsoft Customer Story—AIA Singapore Private Limited drives performance enhancements and achieves more cost efficiencies after moving critical Java applications to Azure
- **AIA Singapore webinar**—AIA Singapore Drives Digital Transformation with Java on Azure
- **J.B. Hunt case study**—Microsoft Customer Story—The fast lane to digital disruption: J.B. Hunt builds load-matching cloud service for shippers and carriers
- **Barracuda case study**—Microsoft Customer Story—Barracuda uses Azure to develop a cloud-native service for discovering sensitive data
- **Salary and benefits**—Employer Costs for Employee Compensation—June 2021 (bls.gov)
- **Pricing Calculator**—Pricing Calculator | Microsoft Azure
- **ZipRecruiter**—Kubernetes Administrator Annual Salary ($104,682 Avg | Nov 2021)—ZipRecruiter
- **Azure Landing Zones**—What is an Azure landing zone?—Cloud Adoption Framework | Microsoft Docs

Acknowledgments

We are extremely thankful to the following contributors for carefully reviewing and providing their valuable input that greatly helped in enhancing the structure and content of this book:

- Nate Ceres, product marketing manager, Java on Azure at Microsoft
- Karim Vaes, specialist at Microsoft
- Adam Hurwitz, director specialist for FSI Intelligence Cloud at Microsoft
- Michael McKechney, principal cloud solution architect at Microsoft
- Frank Campise, director cloud solution architecture at Microsoft
- Jeremiah Gutherie, senior cloud solution architect at Microsoft
- Courtney Reyers, senior content strategist at Microsoft

We cannot thank Brian Diffin enough (Executive Vice President and CTO, Global Technology at Wolters Kluwer Tax and Accounting) for reviewing this book from an Azure customer's perspective. We are very encouraged by his positive feedback, support and resonance with the A+B mindset.

About the Authors

Ajai Peddapanga is a principal cloud solution architect at Microsoft and a Technical Leader in the App Innovation space. Since 2014, Ajai has been helping large enterprise customers build their cloud strategies, adopt Azure, and migrate their data centers to Azure. A Top Contributor Award winner, he works closely with customer technical and business decision makers, and C-suite executives on strategies for application modernization and increasing developer velocity. Prior to Microsoft, Ajai spent 10 years as a product owner, building multi-tenant SaaS solutions for the supply chain industry. He loves to hear from his readers.

LinkedIn

Asir Selvasingh is a principal architect, Java on Microsoft Azure, on-point for everything developers and customers need to build, migrate, and scale Java applications on Azure. Asir started his software engineering career in the early days of Java, in 1995, and built enterprise products, applications, and open-source projects for many years. He works closely with customer technical and business decision-makers, and C-suite executives on strategies for application modernization and increasing developer velocity. And, he works with the community, delivering sessions at Java conferences and fostering strategic relations that enrich the Java ecosystem. He loves to hear from you.

LinkedIn