GitHub Datasheet

6 DevOps tips to help engineering leaders deliver software at scale

DevOps is about speed: faster software development, faster updates, and faster shipments, all of which lead to shortened systems development lifecycles. According to Microsoft's Enterprise DevOps Report, organizations that successfully transition to a DevOps model ship code faster and outperform other companies by 4-5x.

It's no small wonder organizations of just about all sizes and across vertical markets and global geographies have embraced DevOps. Another survey found that more than 80% of IT decision makers say their organizations have invested in DevOps.

But there are real-world challenges for any organization seeking success in DevOps. Among the top challenges? Simplifying a complex technology stack, confronting organizational resistance to change, dealing with legacy architectures, mitigating any security risks, and applying automation throughout the software development lifecycle (SDLC), among other things.

Below, we'll explore six actionable tips that high-performing organizations follow to maximize their investment in and benefit from DevOps.

1. Building a strong DevOps culture matters.

According to the annual State of DevOps Report, companies often "hyper-focus on the automation aspects of DevOps to the detriment of team interactions, fast flow, collaboration, and optimization of the whole system." That focus on automation often leaves out a more foundational element in any successful DevOps practice: building a DevOps culture that favors collaboration and knowledge sharing.

This sharing is critical to building a strong culture of cross-functional collaboration, which helps create a positive developer experience. At GitHub, we know the value of investing in the developer experience throughout the SDLC—because when developers are invested in their work, the outcome and software delivery is far better.

"High-evolution organizations have done both the top-down and bottom-up work to build momentum behind their DevOps practices, created knowledge and pattern sharing practices that enable fast flow optimization, and established productive change approval processes," write the authors of the State of DevOps Report. Developers need

CONTINUED NEXT PAGE →

a lightweight platform to securely share early successes and proven practices across the organization. This sharing is critical to building a strong culture of cross-functional collaboration and creating decision-making logs and documentation.

The bottom line: Don't try overdoing it at the beginning of your DevOps journey. Instead, focus on building a culture where collaboration, automation, and cross-functional communication are prioritized from the outset.

2. Prioritize security right from the start.

DevSecOps is a growing practice that builds on DevOps to make security a core part of every software development stage. This means automating security configurations to the greatest extent possible. It also means adopting DevOps and DevSecOps security tools such as static application security testing (SAST) and dynamic application security testing (DAST) tools, among other things.

GitHub Enterprise provides a fully integrated DevSecOps solution that is GDPR compliant, encrypts all data in transit, and supports industry-leading control considerations with the Cloud Security Alliance CSA-CAIQ Assessment. Organizations also have the option to use a self-hosted solution, which supports strict security, auditing, and compliance requirements. With a cohesive developer experience designed to minimize context switching, GitHub Enterprise helps teams shift security left without creating additional friction for developer teams and DevOps organizations.

GitHub also offers the add-on security suite GitHub Advanced Security, which offers a number of industry-leading security solutions to automate the detection of issues, vulnerabilities, and coding flaws throughout the SDLC. That makes it easier to focus on keeping organizational and customer data secure at every step. And most importantly, teams can build more secure code faster with comprehensive tools that offer pre-commit intelligence on vulnerable code. That empowers developers to fix the vulnerabilities that matter the most for your organization in minutes.

The bottom line: DevOps organizations need to shift their focus from the details of audits and compliance to developing and delivering secure code across every stage of the SDLC.

CONTINUED NEXT PAGE →

3. Apply automation strategically as much as you can.

Automation and DevOps aren't synonymous—there are organizations practicing DevOps that apply some or no automation to their core processes. But automation does enable DevOps organizations to achieve higher degrees of performance by removing repetitive tasks and limiting the chance for human error, among other things.

And when it's properly implemented, automation can also improve the developer experience leading to faster software deployments. Why? Because when organizations strategically use automation within the SDLC, they can actively help their developers solve bigger problems without getting bogged down in repetitive tasks that aren't directly related to building new code.

Some strategic areas to apply automation in a DevOps practice include systems configurations, workflows, and systems provisioning. This infrastructure automation tackles the common challenge of keeping developers' work on pace and not running ahead of operational capabilities—and that's directly related to the ability to deploy software.

But automation also serves up another key benefit. It enables organizations to achieve greater efficiency by bringing a greater degree of repeatability and velocity to the SDLC. GitHub offers an array of CI/CD and automation templates that allow users to automatically configure workflows and keep the status of project board cards in sync with associated issues and pull cards. Users can automate actions based on triggering events to eliminate time-consuming manual tasks involved in managing a project board—or the broader SDLC.

The bottom line: Look for areas across your SDLC that can be automated such as in the build, test, and deployment stages. For some organizations, this may start with explorations into building a CI/CD pipeline—or improving an existing pipeline. Whatever path you take, it's important to identify repetitive manual tasks as candidates for automation to improve the developer experience and help your teams ship software faster.

CONTINUED NEXT PAGE →

4. Simplify your technology stack.

Some of the top challenges in finding success with DevOps are people-oriented and often focused on eliminating process bottlenecks and streamlining developer collaboration.

Butanother common challenge is selecting, implementing, and managing an ecosystem of specific DevOps tools for key functions such as CI/CD, application monitoring, infrastructure management, and automation. One solution to this challenge is choosing an extensible development platform that integrates with the widest array of tools possible. However, many organizations are now seeking to sidestep this issue altogether by leveraging a development platform that offers multiple capabilities ranging from CI/CD to project management and security.

Their goal? Simplifying their technology stack to prioritize one platform that accomplishes multiple business goals.

At GitHub, we have built GitHub Enterprise as a singular platform with built-in code management, project management, CI/CD, security, IDEs, AI-powered coding tools, and more. We did this to help simplify the technology stacks software teams need to navigate and improve the overall developer experience by reducing the amount of context switching teams need to do while working.

The result: You can manage all of your software development and DevOps practices in one place.

The bottom line: Prioritize development platforms that reduce the number of additional tools you'll need in your DevOps practice and also offer an ecosystem of integrations to meet you where you are with the technology stack you have.

---

5. Maximize open source efficiency with innersource.

SA fast-growing number of organizations are adopting and using innersource, a development methodology that seeks to combine the best practices from large-scale open source projects with security. These kinds of highly complex, multi-development team efforts require coordination across hundreds—if not thousands—of developers and teams.

CONTINUED NEXT PAGE →

Innersource distills lessons about how to build large software projects in a distributed and asynchronous way—and it's proving to be a formidable weapon to combat the many challenges posed by large-scale DevOps projects. It has also proved to be a great way to boost efficiency through code reuse. Companies can access and leverage open source best practices behind the innersource firewall.

GitHub is home to the world's largest open source community, and the platform itself has been built, designed, and optimized to facilitate large-scale development efforts across distributed teams. This makes it simpler for DevOps organizations to adopt innersource methodologies. Developers and teams on the GitHub platform can view, modify and distribute a project for any purpose they choose, as enforced by open source licenses. And GitHub offers a broad series of self-help guides for finding users for your project, building welcoming communities, investing in maintenance and governance, and analyzing open source metrics, to name a few things.

The bottom line: Innersource enables companies to incorporate open source best practices into their own development efforts. Invest in adopting and implementing innersource methodologies among your software teams to enable greater efficiency, more transparency, and clearer communication.

6. Promote continuous feedback, continuous integration, change management, and delivery.

Discovering common practices and industry standards in DevOps is one thing. But it's another thing to integrate them and embed them deeply into a process flow that crosses data and information silos from development through deployment and operations.

To this end, continuous integration builds on test automation to ensure that new code is added to an application. As a practice, it focuses on quality at the application level as new or modified code is integrated. Change management brings operations into the picture to allow input on what other systems could be impacted and what consequences or opportunities a change might expose at a broader level. Incremental deployment strategies often fall victim to operational constraints that can limit or stop continuous deployment in its tracks. This is a people issue, not a technology problem. And the remedy is the inclusion of all stakeholders in application development, operations, and support.

CONTINUED NEXT PAGE →

This kind of continuous intradepartmental input will significantly boost the likelihood that an application will also meet user needs and expectations.

The bottom line: Strategically build automated checkpoints throughout the SDLC to ensure continuous testing, change management, and software delivery—all with real-time feedback to make sure the right people know when something goes wrong. This level of automation removes the risk of human error and enables teams to work faster at scale to build quality software.

CONTINUED NEXT PAGE →

Ship software faster on the complete developer platform.

GitHub offers the complete developer platform to deliver secure software at scale. Trusted by more than 83 million developers and 90% of the Fortune 100, GitHub helps DevOps teams of every size collaborate securely and deliver better customer experiences, faster.

Find out why the world's most innovative companies use GitHub.

Start a free trial →

Questions about DevOps? We're here to help.
Contact us at sales@github.com. Learn more at github.com/enterprise