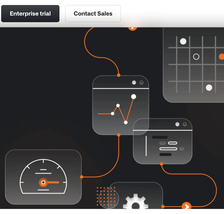
What is DevOps?

DevOps is a holistic business practice that combines people, technologies, cultural practices, and processes to bring previously silved teams together to deliver better software faster. Successful DevOps implementations are viewed as an organizational change when it comes to software development. Learn about DevOps and how to optimize software delivery with DevOps throughout the software development lifecycle (SDLC).



Overview DevOps benefits DevOps and CI/CD DevSecOps Examples Request a demo

Defining DevOps

By bringing people, processes, and products together, DevOps enables development teams to continuously deliver value to their customers.

What do we mean when we say DevOps?

A culture shift? A way of working? Specific tooling?

When it comes to the meaning of DevOps, it's more than just automation. It's really about organizations following a process enabled by software to help them deliver value. DevOps is a business practice where development, IT operations, and security teams partner to build, test, and provide regular feedback throughout the entire SDLC.

To successfully adopt the DevOps methodology, people shift the way they work and collaborate. From culture change to automation with continuous integration and continuous deployment (CI/CD), let's explore how DevOps teams develop shared context, expertise, tooling, and roles to ship better products.

How DevOps was created: Silos to shared workflows

DevOps sets a new standard for how we build software. Before the mid-2000s, developers, IT operations, and security teams typically worked in silos and followed a linear process:

- Developers wrote code, packaged the application with documentation, and shipped it to QA.
- The QA team tested the application and handed it off to the production operations team.
- The operations team deployed the software to production and managed the software's uptime and stability—with little or no direct interaction with the development team.
- The security team only checked code for vulnerabilities after deployment.
- If someone reported an issue, the entire process started over again.

This linear approach to software development was slow and frustrating for everyone involved. Then in 2008, system administrator Patrick Debois and software developer Andrew Shafer created a working group to help bridge the gap between operations and development teams. Their best practices started the journey toward the DevOps processes we know today.

Learn more about the history of DevOps

Bridging the gap between operations and development

In a DevOps culture, teams work closely together in a shared workflow to continuously improve their software delivery and management processes. The DevOps Engineer plays an important role by continuously managing team communications and designing and implementing strategies. In addition to code-related tasks, the DevOps Engineer is responsible for solving issues related to testing, monitoring, release management, and operational planning.

What is a DevOps model?

Since every DevOps implementation is unique to the company's business needs, there really is no one-size-fits-all DevOps model. However, the core philosophies of DevOps include the breaking down of silos, and the ideas that everyone should take responsibility for a product and work together as a team across all parts of the SDLC.

Benefits of DevOps

Traditional deployments are difficult, manual, and often break down, leading to employee burnout and unsustainable release cycles. In comparison, recent research shows that using DevOps methodologies and principles like automation drive productivity and improve the developer experience.

DevOps makes software development faster, friendlier, and more collaborative. Engineers who work on DevOps teams focus on their goals, prioritize work-life balance, and get more time for the projects they care about. This type of satisfaction builds on what we know from organizational psychology: Teams do their best work when they're personally invested in the outcome.

Improved productivity
Reduce manual tasks and gain consistency, reliability, and efficiency.

Cloud-ready code
Easily provision environments in the cloud to help teams work faster at scale and meet demand.

Security
Protect customers and code while managing access across locations and environments.

Collaboration
Build and foster a collaborative culture with improved communication.

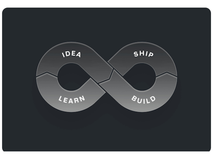
> "We've gone from over 30 siloed software engineering groups to a largely cohesive team. By using standards and automation, the DevOps journey is now three steps.

Denis Canty
McKesson Labs // Vice President of Developer Services and Technology Labs

How DevOps works

The DevOps lifecycle

The DevOps lifecycle spans all aspects of software development. DevOps teams collaborate continuously and share tools and responsibilities across all lifecycle phases to efficiently build and ship software.



Four phases of the DevOps lifecycle

Idea

Teams gather requirements and feedback and start sketching out the resources they'll need. DevOps teams focus on continuous planning—plans that can be updated frequently, even daily. To build out roadmaps and track progress towards their goals, teams leverage project management tools everyone can access and use.

Ideation stack

GitHub Issues and Project Boards

Learn more about planning and tracking

Build

As project managers assign tasks, developers get to work. Here, DevOps comes to life. With version control and cloud-based development environments, developers can make ongoing changes and review code together in real time. Tools like GitHub Codespaces make it easy for developers to quickly, safely, and easily make changes without having to create or persist a desktop development environment. Continuous integration (CI) also kicks in, where automated tools turn code changes into builds, run tests against required checks, and merge and prepare code for deployment.

Continuous integration stack

GitHub Codespaces, GitHub Actions

Learn more about continuous integration

Ship

After passing initial checks, developers use continuous delivery (CD) tools to automatically push code changes to a non-production testing or staging environment. Operations teams can immediately deploy the changes to production, without any surprises.

Continuous deployment stack

CD pipelines with GitHub Actions, GitHub Packages, Microsoft Azure

Learn more about deployment

Learn

Software may be out in the world, but the work isn't over. Operations teams keep an eye on releases with monitoring tools that measure performance and monitor the impact of code changes. They ensure stability and uptime, gather customer feedback, and stay in close contact with developers to push required fixes and address incidents faster.

Observability stack

Netdata, Lightstep, New Relic, App Dynamics, Sentry, Raygun, Honeycomb, Splunk

Automation at every step

DevOps emphasizes automation alongside collaboration for quicker releases. Automating tasks via CI/CD in a DevOps pipeline accelerates software development and deployment and helps ensure high-quality output with testing, reporting, and monitoring.

Learn more about automation >

Adopting a DevOps model and culture

Successful DevOps depends on cultural change—adopting a collaboration-first approach to software originally developed by open source teams. In open source, developers encourage transparency, shared workloads, continuous feedback and reviews, and robust documentation. Industry reports like DORA's State of DevOps from researchers Nicole Forsgren, Jez Humble and Gene Kim, and popular DevOps frameworks like CALMS offer use these best practices as benchmarks for a successful DevOps culture.

Collaboration

Collaboration is at the heart of DevOps. Developers and operations teams regularly communicate about timelines and goals, and everyone is responsible for the project's success.

Automation

As teams work together and share tools, automation supports consistency, reliability, and efficiency across the organization so it's easier to discover and troubleshoot problems.

Transparency

Ongoing feedback and open access to repositories encourages collaboration within and outside individual teams. It also promotes InnerSource, the process of using open source best practices to build internal code.

Evolved DevOps operations

With DevOps, operations teams are now embedded in development and have a stake in empowering developers. This often means working more closely with developers to provide on-demand access to compliant, secure environments and tools. It also means turning to automation for more of their own repeatable tasks, such as updating systems and resolving incidents.

Effective DevOps management

DevOps managers are responsible for the implementation of the DevOps framework throughout the development and deployment process. This includes taking responsibility for successful software delivery, monitoring workflows, and overseeing software release management. Effective DevOps managers are leaders with excellent interpersonal and problem-solving skills, and a strong technical background.

DevOps best practices for implementation

DevOps is implemented by following DevOps practices throughout the application lifecycle. Some practices help speed, improve, and automate a specific DevOps phase, and some practices help improve productivity across several phases.

> “We’ve seen significant savings for our organization and now our engineers have even more time to spend in what matters the most, the business logic.

Stuart Davidson
`Skyscanner // Engineering Manager`

Continuous integration

Developers use CI to automate, merge, and test code. CI helps catch bugs early in the development cycle when they're less expensive to fix. CI also executes automated tests to help ensure quality. To drive frequent deployments, teams use CI to produce deployable artifacts, including infrastructure and apps, that feed release processes.

Continuous deployment

In this process, code is built, tested, and deployed to test in production environments. To increase quality, teams deploy and test in multiple environments. CD systems are also used for automated release processes and fixes to existing software.

Version control

Using version control is a fundamental DevOps practice that empowers several developers to collaborate on authoring code. A version control system provides a clear process to merge code changes, resolve conflicts, and roll back changes to earlier versions when needed.

Agile software development

Through short release cycles, the agile software development approach enables high adaptability to change. Agile teams provide continual improvements to their customers, collect feedback, and adjust based on customer needs and wants.

Infrastructure as code

By defining system topologies and resources in a descriptive manner, infrastructure as code empowers teams to manage resources like they manage code. This helps them deploy system resources in a repeatable, reliable, and controlled way, and maintain testing and development environments that are identical to production.

Configuration management

Configuration management means managing the state of resources in a system, including databases, virtual machines, and servers. Configuration management tools help teams reduce risk by rolling out changes in a systematic, controlled way. Teams also use these tools to track system state and operate complex environments at scale.

Continuous monitoring

Through continuous monitoring, teams gain full, real-time visibility into the health and performance of the entire application stack. High-performing DevOps teams make sure they set meaningful, actionable alerts and collect rich telemetry event and log data. Insights derived from this data help the team mitigate issues in real time and improve the application in future releases.

What can you do with DevOps?

See how high-performing teams put collaboration and automation to work.

Mckesson Labs

Building a cohesive DevOps technology stack.

Explore Mckesson >

Dell Technologies

Standardizing to support innersource and scalability.

Explore Dell Technologies >

The Home Depot

Modernizing infrastructure and developer tooling.

Explore The Home Depot >

Dow Jones

Bringing automated security to CI/CD pipelines.

Explore Dow Jones >

Autodesk

Improving collaboration and reliability in the cloud.

Explore Autodesk >

American Airlines

Fostering innovation by using GitHub as an incubator for ideas.

Explore American Airlines >

> “The access, sharing, and collaboration that happens when we’re all one organization is fantastic.

Cynthia Payne
`Nationwide // Vice President of IT App Services`

DevOps and DevSecOps

What is DevSecOps?

Like development and operations, DevSecOps integrates automated security testing into every part of DevOps culture, tooling, and processes.

Learn more about DevSecOps >

For many practitioners, security was always meant to be part of DevOps. Instead of happening at the end of the SDLC, DevOps security starts at the source: Code. This approach is commonly known as DevSecOps or "shifting left." GitHub has always considered good DevOps to mean DevSecOps, and GitHub tools make it easier for DevOps teams to include security at every step of the SDLC.

Using automated security tools, developers find and address security vulnerabilities as they code instead of waiting for security teams to address them after deployment. By empowering developers to take responsibility for security throughout the DevOps lifecycle, DevSecOps allows development, operations, and security teams to find and remediate security issues faster.

> “You fix it and you move forward, because guess what? It takes you a minute to solve that security issue in development when it could potentially take you days in production.

Shomal Siwan
`California Department of Technology // Lead DevOps Engineer/Solutions Architect`

DevOps products

Each phase of the DevOps lifecycle has unique considerations that one or more products can help solve. Organizations might invest in building out a collection of tools to use in its DevOps practice, called a "DevOps Toolchain" to address each phase of the lifecycle. The most successful DevOps organizations will have these types of tools to help them deliver higher-quality software faster.

See how GitHub compares to other DevOps platforms >

CI/CD

Throughout this article, we've discussed how CI/CD is fundamental to any DevOps practice. GitHub Actions, a complete CI/CD DevOps software solution, empowers DevOps teams to build more secure code from the start and deploy software with confidence —without sacrificing speed.

Workflow automation

Automation is another fundamental component of DevOps and GitHub Actions makes it easy to automate DevOps software workflows, with world-class CI/CD. This includes code reviews, branch management, and issue triaging.

Version and source control

The best version and source control platforms, like GitHub, integrate with the organization's DevOps toolchain and empower product teams to collaborate throughout the DevOps lifecycle. They include productivity features like pull requests, which help developers get reviews on proposed code changes before they're integrated into the main code branch.

Security

Companies that invest in DevOps often invest in several tools to model threats and apply automated security testing. Integrated solutions like GitHub Advanced Security help ensure software security through every phase of the SDLC. For example, GitHub Advanced Security scans code as it's created to help developers secure their code in minutes, watches their repositories, and notifies them of secrets issued by more than 45 leading secret providers, and helps turn security best practices into better development processes.

Start building your DevOps workflow

Whether you're ready to dive in or still have questions, we've got you covered.

Explore best practices > | Get a GitHub demo

Frequently asked questions

- What is DevOps automation? +
- What is the difference between DevOps and SRE? +
- What is the DevOps model? +
- What is the difference between DevOps and DevSecOps? +
- What is considered good DevOps? +
- What does a DevOps Engineer do? +

GitHub Inc. © 2024