GitLab

A guide to getting started in DevOps

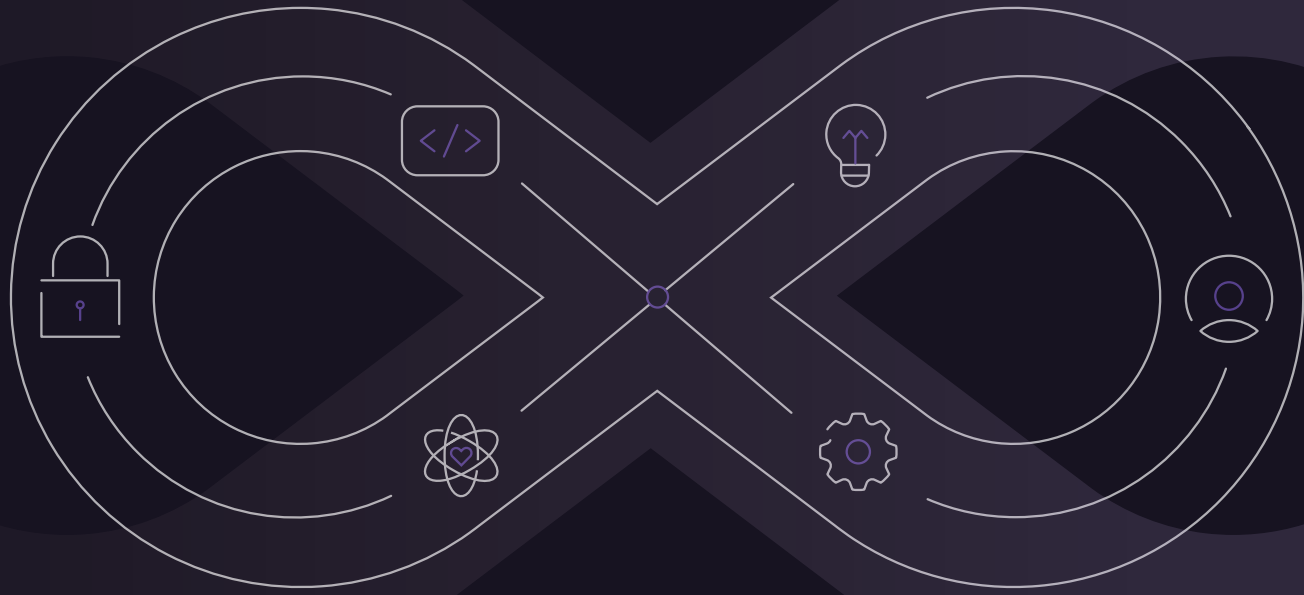


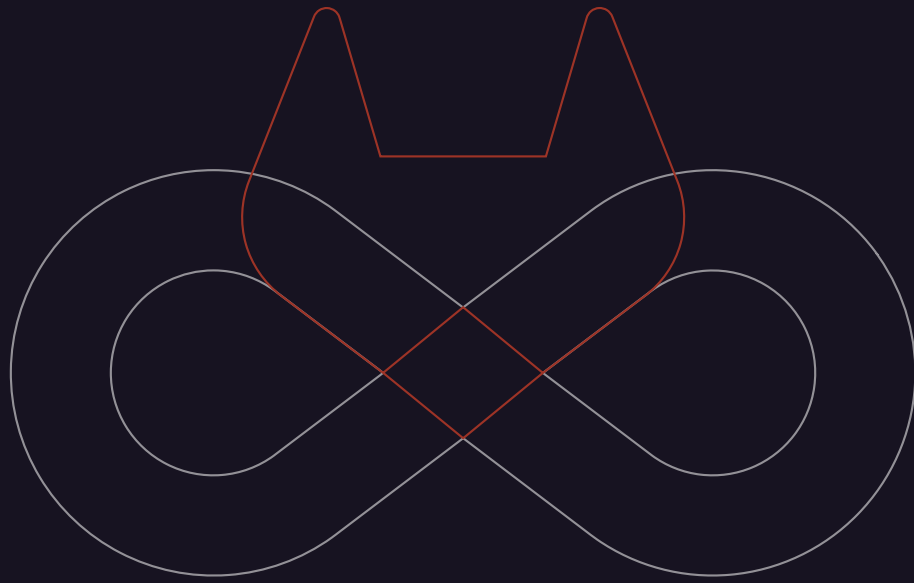
Table of contents

Introduction

With the number of organizations using DevOps growing dramatically, it only makes sense that a lot of people on those DevOps teams are new — some even really new — to the technology and practice. According to GitLab's **2024 Global DevSecOps Survey**, 40% of respondents reported using DevOps methodologies, up from 35% in 2023.

If your organization has recently adopted DevOps — or is getting ready to make the switch — you may not be entirely familiar with the tools and practices behind it. This guide will help you get up to speed. We'll explain what DevOps is (and isn't), and key technologies and terms you'll need to understand, as well as why collaboration is so critical. We'll showcase an example of DevOps working in the real world, and then explain why DevOps can help your career and your paycheck. Finally, we've got an extensive list of resources.

Let's dive in.



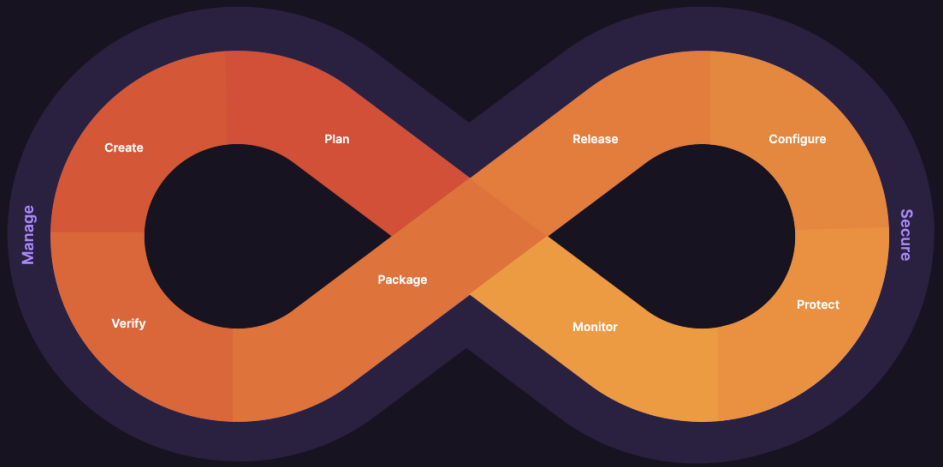
What DevOps is and how it helps your company

The first thing you should **know about DevOps** is that it's all about empowering teams – enabling organizations to work collaboratively to develop and deliver secure software faster and more efficiently. For many years, software development was anything but streamlined and efficient. Processes were siloed, leading to bottlenecks and costly delays, while security was, at best, an afterthought. DevOps sprung from deep-seated frustrations with the old way of doing things, and brought with it the promise of simplicity and speed.

We think DevOps is done best on a single end-to-end platform. A DevOps platform allows teams to move from, or avoid, an often complex and confusing multitude of tools by using a single, complete software development ecosystem, eliminating the need for team members to jump from one tool to another, saving both time and money. That ecosystem can be used to conceive, build, and, ultimately, deliver better, more compliant and secure software more efficiently, continuously, and at top speed. And that helps DevOps teams be **more agile**, but it also provides more agility to the overall business – enabling companies to more quickly meet customer needs, remain compliant, stay ahead of competitors, and turn on a dime to take advantage of changing business climates. So DevOps is the engine that drives agility in both software development and deployment, as well as in business.

By using automation, shifting security to the left, and making processes repeatable and measurable, a DevOps platform leads to better software and reduces the time between designing new, higher-quality features and rolling them out into production. And that maximizes the overall return on software development.

And to talk about how DevOps works, you really need to talk about **the culture**, or mindset, behind it. It's not development as usual. The bedrock of DevOps culture is **collaboration and joint responsibility**, along with a focus on a constant cycle of rapid iteration, measurement, assessment, and reevaluation. Again, it's all about agility, and being able to learn and deploy fast. All of that leads to continuous, iterative improvements and feature deployment.



Fundamental technologies and processes

Stages of the DevOps process

A DevOps education isn't complete unless you understand the lifecycle stages, which take the process from planning all the way through to launching new features, analysis, and gathering feedback.

As you work in DevOps, you'll need to learn these phases since each one is an integral part of the process. So, from a 40,000-foot view, there are three overarching stages executed in a logical order – build, test, and deploy. It's the natural workflow. Build the code, then test it, and, if all is in working order, deploy it.

However, we need to dig deeper to uncover more complex layers of these stages. Each one is a key driver to producing software and business value. Understanding and using this flow will create efficiency, reliability, speed, and agility. Here is a closer look at nine key stages:

- **Plan** is the stage of DevOps that encompasses everything that happens before the first line of code is written. It's about creating a product roadmap that guides upcoming development, helping the team organize resources and priorities, align, and track projects.
- **Create** is the first stage of the CI/CD pipeline. This is where code is designed and developed using version control to coordinate changes made by multiple developers to the same code base. This is one of the keys to improving velocity.
- **Verify** is a process focused on confirming the quality of code. To get feedback quickly to developers and testers, and instant insights into every commit, this process relies on security testing, code quality analysis, parallel execution, and automation. Enabling developers to find and fix flaws while they are developing has proven to be **more cost-effective and efficient**.
- The **Package** stage comes after code has been created and tested. Packaging applications and dependencies, managing containers, and building artifacts maintains a consistent software supply chain.
- **Release** or deployment is about pushing code updates into the production environment. With DevOps, releases can be deployed as iterations are created, tested, and ready – and not as on a preplanned, static, bulk release date.
- **Configure** is about setting up, managing, and maintaining application environments. Automated configuration management is designed to handle these complex environments across servers, networks, and storage systems.
- **Monitor** is a proactive, automated part of the process, focused on tracking software, infrastructure, and networks to trace status and raise alerts to problems. This increases security, reliability, and agility.
- **Protect** is about securing your applications and their runtime environment, from intrusions, and new vulnerabilities.
- **Manage** is about visibility and control across your end-to end software development lifecycle by managing permissions, standardizing DevOps build and deployment processes, and automating guardrails to ensure security and compliance policies are met.

What about security? Good question. That's the beauty of DevOps – security isn't an afterthought. It's part of EVERY stage of the process, from documenting requirements to automated testing to validating those requirements. It ensures that new code and features actually work exactly the way they are designed, and that bugs, security threats, and compliance issues haven't been created.

These stages are all part of an ongoing cycle. All of the information created in these stages is instantly available through the platform to all participants, across stages, and provides a single source of truth for improved visibility and collaboration. Another key benefit of a united platform is the ability to manage and control the entire software development lifecycle from one place.

What powers these DevOps stages

Source Code Management (SCM)

This is how a repository of code is shared among many developers without one person's changes negating another's. The code is divided and managed into projects and groups of projects. An individual developer checks out existing code or adds code to what's there and the SCM tool identifies conflicting edits to the same code and flags it for resolution. This process allows multiple developers to work on one project at once, a key to increasing the velocity of software updates. DevOps thrives on Git repositories, **an important distinction from old-school version control systems**, because of the powerful capabilities their modern architecture enables.

Continuous Integration(CI)

This is the step that enables iteration by committing changes to a shared source code repository early and often — many times a day — and automatically testing each change and kicking off a build.

Continuous integration is all about efficiency. By automating manual work and testing code more frequently, teams can iterate faster and deploy new features with fewer bugs more often. Other **benefits of CI** include identifying and fixing problems more easily, less context-switching for your team, and happier users and customers.

To get the most out of continuous integration, make sure your setup includes these core elements:

- A source code repository with all the necessary files and scripts to create builds.
- Automated builds with scripts that include everything you need to build from a single command.
- Self-testing builds that automate your policies (for instance, fail if any test fails).
- Frequent commits and iterations so there are fewer places for conflicts to hide.
- Stable testing environments that accurately reflect the production environment.
- Visibility so every developer can access the latest executables and see any changes made to the repository.

Continuous Delivery (CD)

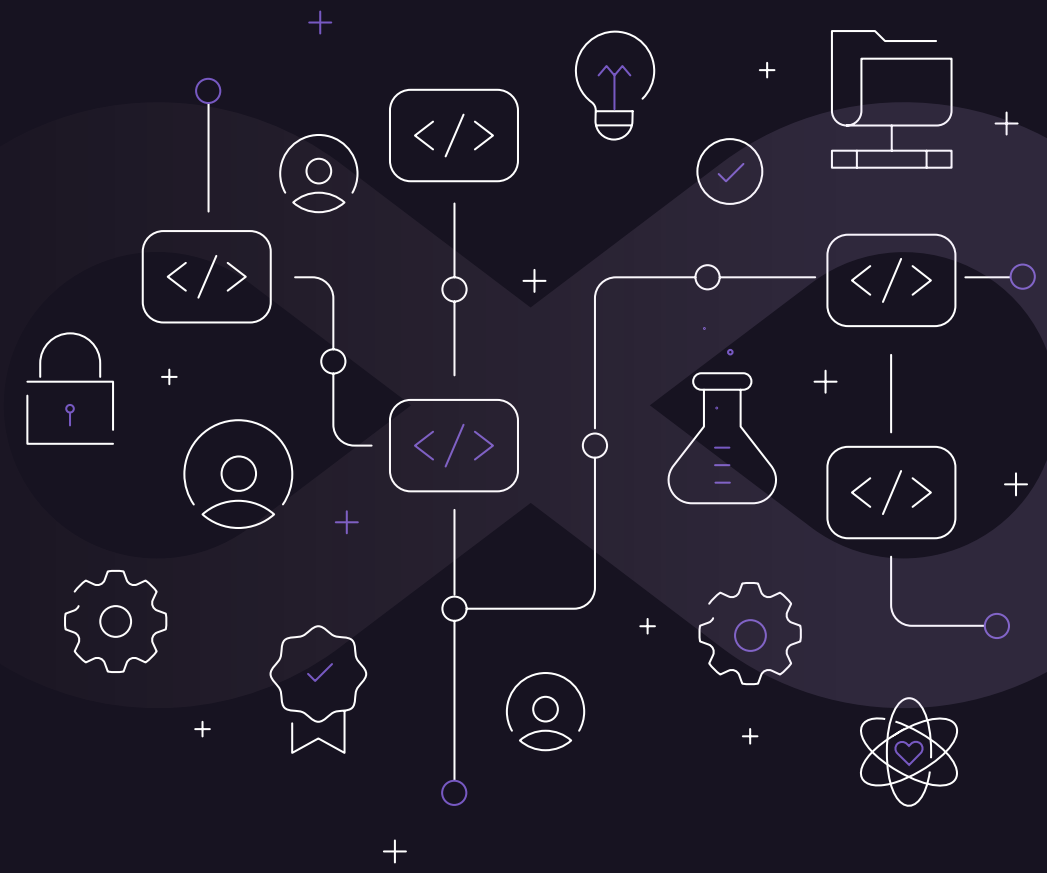
Continuous delivery is a software development process that works in conjunction with continuous integration to automate the application release process. Once code has been tested and built as part of the CI process, continuous delivery takes over during the final stages to ensure it's packaged with everything it needs to deploy to any environment at any time. Continuous delivery can cover everything from provisioning the infrastructure environment to deploying the tested application to test/staging or production.

With continuous delivery, software is built so it can be deployed to production at any time. Then you can trigger the deployments manually or automate the process.

When continuous delivery is done well, your software release **processes become boring** — that is, they are low-risk, consistent, and repeatable. Then you can confidently plan release processes and schedules, automate infrastructure and deployments, and manage your cloud resources more effectively.

Automated testing

This is key to fully adopting DevOps and continuous integration — and releasing higher quality code more frequently. With testing built into your CI pipeline, every committed code change triggers a build, and then the build runs tests to ensure the changes pass all tests, policies, and code compliance standards you established for your application. With this in place, bugs are identified earlier and with greater context to simplify their resolution, your teams can deploy more frequently and confidently, and you minimize manual testing and reworking late in the process.



Shifting security left

A fundamental process for successful DevOps is incorporating security into the end-to-end automation. This is often referred to as DevSecOps. By integrating testing and the security review process earlier in the software development lifecycle, there is more opportunity to adequately address any security issues. If security testing is treated as an afterthought or doesn't happen until code is ready for production, it can be difficult to go back and correct problems, and it's often too late to fix them quickly and efficiently. This can lead to delayed deployments, vulnerabilities making it into production, greater technical debt, and inefficient silos between security and the rest of the DevOps teams.

To shift security left, you'll want to integrate security testing into your CI pipelines so code is continually tested, not only against other commits in the shared repository, but for overall security, as well. Some types of security testing you may want to include early on in your development lifecycle include:

- Static Application Security Testing (SAST)
- Dynamic Application Security Testing (DAST)
- Container and cluster image scanning
- Dependency scanning
- Secret detection
- Infrastructure-as-code (IAC) scanning
- API testing

Documentation

Although sometimes overlooked, documentation is invaluable to successfully implementing DevOps practices. Creating and maintaining internal documentation for the services and applications that your team works on and what your DevOps process looks like can go a long way to improving your team's performance and the software you put out. The **2024 Accelerate State of DevOps Report** noted that teams that focus on the user by increasing the quality of product documentation see an increase in product performance.

Feedback

This is an essential piece of the puzzle as organizations should always be looking for ways to improve the user experience and the overall DevOps process. In traditional software development, the feedback loop can be a complex path to navigate. With DevOps, closer collaboration and rapid iterations means teams have constant access to manageable data, so they can incorporate feedback, tune their efforts, and deliver improvements efficiently and quickly. Automating the process is a key step since it will ensure the information is collected and distributed to the right parts of the team, creating rapid adjustment to new code updates.

Other key DevOps technologies you should understand

When you're getting up to speed with DevOps, there are several areas where you should be sure to educate yourself and make sure you stay current. Here are several areas you should understand and stay current on:

The cloud

If you're working in DevOps, understanding **the cloud** is a good idea. The majority of modern software depends on the cloud and cloud-native infrastructure, including containers and orchestrators, which help automate the process of software development and delivery. Applications developed this way allow DevOps professionals to deploy anywhere and to get the most out of **multiple cloud platforms**, or even private clouds within their own data center.

A cloud-native approach is more scalable because it distances the code from the hardware it uses. To be able to help their companies and their own careers, DevOps professionals need to understand cloud providers, services, and platforms. And it's a focus for a lot of DevOps professionals. According to **GitLab's 2024 Global DevSecOps Survey**, 55% of respondents said they are running at least half of their apps in the cloud, up from 32% of respondents in 2023.

A culture of collaboration

You need to embrace collaboration. It's a tenet of DevOps practice and philosophy. By inviting discussion, input, and assistance from both experienced and new team members, a culture can be built around learning from and relying on others' expertise. Being part of a true DevOps culture also means being able to listen, stay calm under pressure, create trust among team members, and be able to take ownership of a situation or problem.

Collaboration, though, is about more than DevOps teammates working together. It's also about collaboration between DevOps and other parts of the business – members of the security team, marketing, finance, customer service, and the C-suite. Collaboration between DevOps and security, for example, is a way to integrate security into the entire development process. Don't make the mistake of dismissing this as a soft skill that's less important than technical skills. **Develop key skills** like communication, knowing how to talk about business needs, and working together to solve problems.

Key programming languages

DevOps engineers need to be able to code, but even more importantly, they must consider the processes, tools, and methodologies used across the end-to-end DevOps lifecycle stages identified above. Some languages are more conducive to this end-to-end process than others. There are a lot of programming languages out there so it can be a big job to figure out where to start. First, you need to understand what your DevOps teams need. What projects are being worked on, and what languages are needed now, as well as what languages will be needed for future projects?

Some of the most popular programming languages are Python, Golang, Ruby, JavaScript, Perl, Java, Bash, and PHP. According to the **2024 Stack Overflow Developer Survey**, JavaScript was the most popular programming language overall, while Python was the most popular language for people learning to code.

Need some practice? Why not **volunteer your coding time** or contribute to open source projects to fill out your resume?

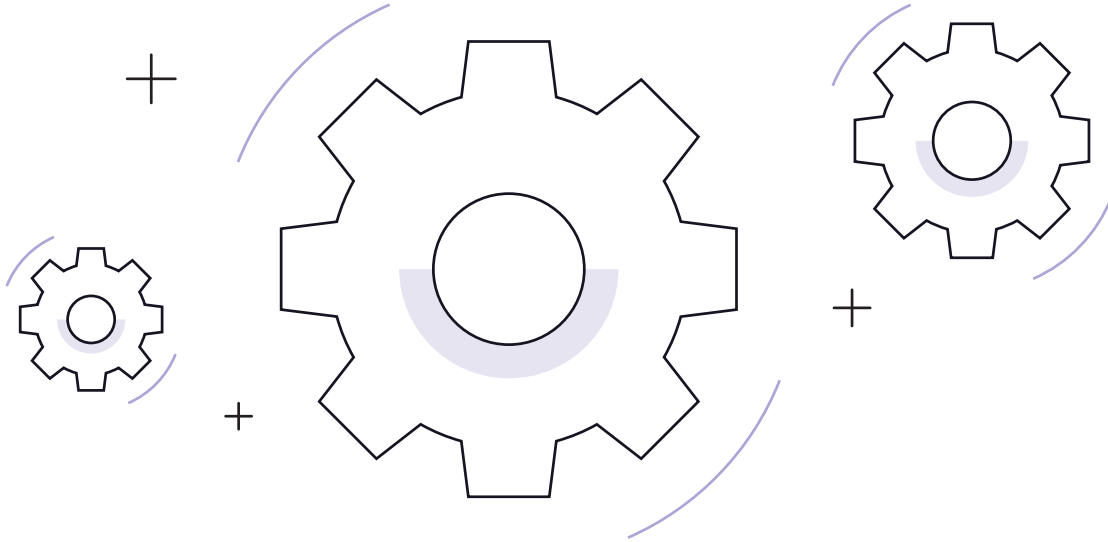
Automation

DevOps teams are increasingly looking to automate processes throughout the development and deployment lifecycle, so it only makes sense that you understand how automation works and how to use it. Automation, which cuts time and money spent on repetitive tasks and eliminates human errors, streamlines the whole DevOps process. With automation, each task is performed identically and with consistency, reliability, and accuracy. This promotes speed and increases deliveries, and ultimately deployments. While it doesn't remove humans from the picture, automation minimizes dependency on humans for managing recurring tasks, such as monitoring for availability, performance, or security problems; consistently configuring software environments; testing new application versions against predefined quality standards; integrating code; speeding deployments; aiding CI/CD testing software throughout the development process; and managing logs and documentation. Yes, there's a lot to this.

According to GitLab's 2024 Global DevSecOps Survey, 67% of respondents reported that their software development lifecycle is "mostly" or "completely" automated. Another 29% said it is "somewhat" automated. That means if you're in DevOps, you should be a student of automation.

Monitoring

As your organization's application stack and the number of DevOps teams working on it grow, the number of moving pieces will only multiply. Keeping track of all of those pieces can be overwhelming. That makes continuous monitoring a requirement for being able to maintain



an end-to-end, real-time situational awareness of your ecosystem. With DevOps, complex applications could be updated and deployed every day – even multiple times a day. Sophisticated and automated monitoring is a proactive way to cut down on bugs, improve deployment speed and efficiency, detect security threats and compliance issues, eliminate breaking changes, and maintain documentation. It's used from planning through development, integration, testing, deployment, and even operations.

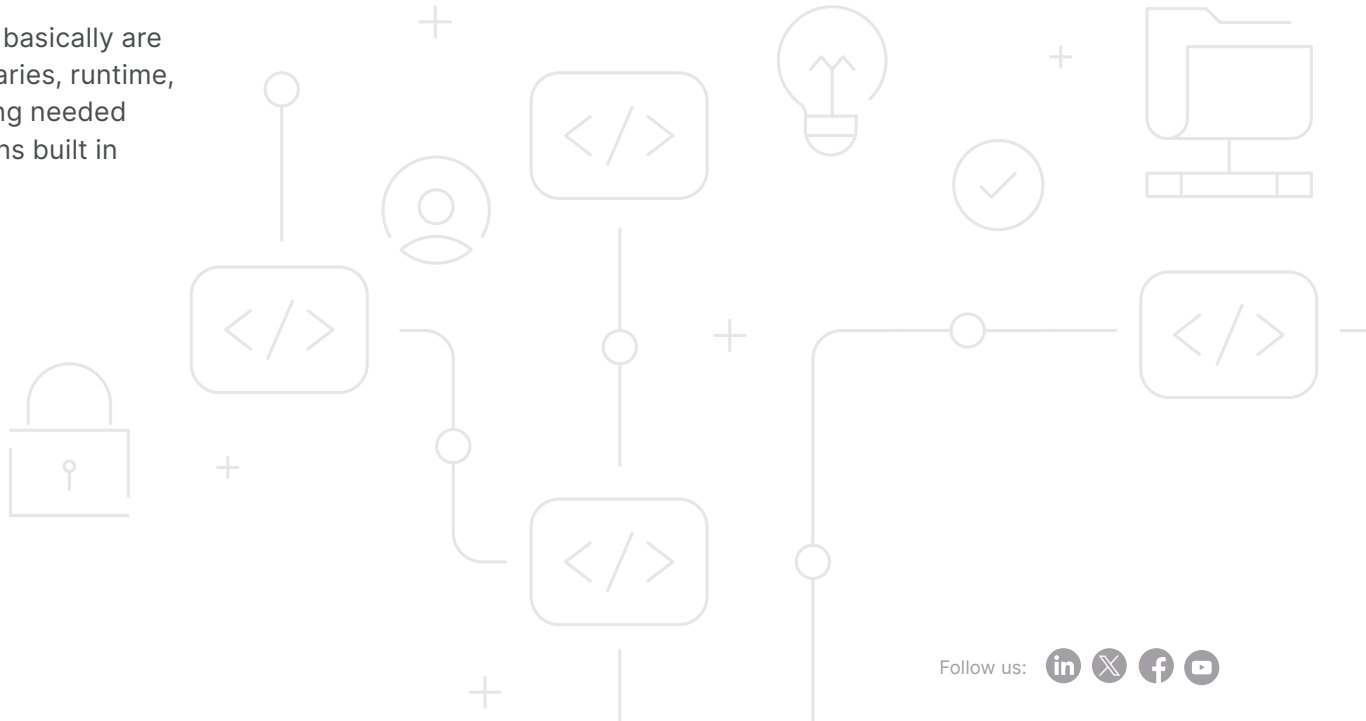
That means monitoring is not just a process needed by developers, but also by project leaders and security teams. Monitoring doesn't just track processes. It's set up to raise alarms about performance and threats throughout the pipeline.

DevOps teams that want to be on top of their game will increasingly be using monitoring, so you should understand it.

Containers

The DevOps world has gone all-in with containers, which basically are packages of software code, its configuration, system libraries, runtime, and the rest of its environment. Containers hold everything needed for the application to run, and they ensure that applications built in one environment run consistently and seamlessly in others, solving the problem of how to get software to run reliably when moved between environments. These modular units, or building blocks, are set up to enable DevOps teams to build, test, deploy, and maintain applications efficiently, with fewer resources, at top speed, and securely.

Since they easily can be shared between teams, they not only speed development and deployment, but they feed into a culture of collaboration, which is key in DevOps. And knowledge of containers means understanding Docker, a popular application container technology, as well as Kubernetes, an open-source container-orchestration system that controls how and where containers run.



How DevOps solves real-world problems

If you're trying to figure out exactly how important and transformational adopting DevOps can be for your company, check out what a DevOps platform did for HackerOne. As the world's most trusted, hacker-powered security platform, HackerOne gives organizations access to the largest community of hackers on the planet. With a presence in more than 70 global locations, the company found cross-functional collaboration challenging. For instance, when developers on different continents had to pick up where others left off on a code project, lengthy pipeline times interrupted handoffs. With an engineering team that had tripled, HackerOne needed to speed development and deployment, reduce toolchain complexity, and enable teams to efficiently manage multiple projects. They found a solution in a DevOps platform. With a single, unified DevOps platform, HackerOne's teams were able to find code issues earlier in the pipeline, work iteratively to resolve security flaws, and simplify audits. They also increased deployments from once or twice a day to up to five times daily, and saved four hours of development time for each developer every week.

Resources

Here's a look at just some of the resources available to you out there:

Podcasts to help you dive in to DevOps

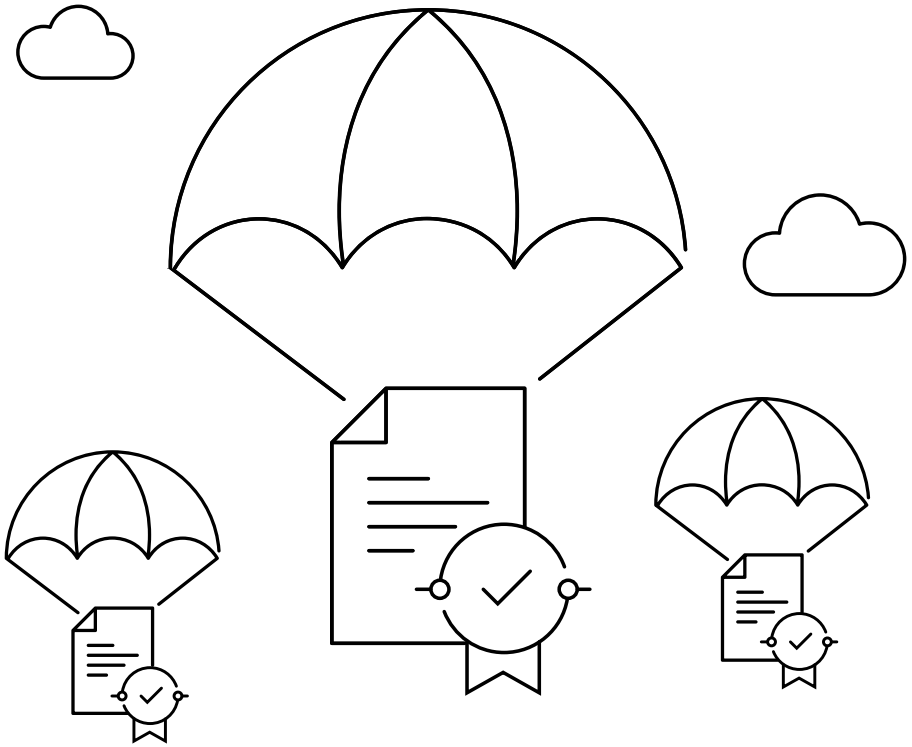
- **The Humans of DevOps Podcast Series** offers insights on things like upskilling, the art of DevOps, and women working in Devops.
- **Arrested DevOps** talks with top techies about the state of DevOps.
- **Real World DevOps** talks with people who are organizing DevOps conferences, writing related books, or building tech.
- **The Cloudcast** is exactly what you'd expect – focused on all things cloud-related.
- **Greater Than Code** focuses on both human and tech issues in DevOps, and the tech field in general.
- **Code Newbie Podcast** is aimed at people just getting started with software development.
- **DevOps Paradox** has industry luminaries walk you through what DevOps is all about.

Helpful books and eBooks

- **Seven Tips to Get the Most out of Your DevOps Platform** this is an eBook from GitLab that focuses on making sure your team is poised to get the most out of a DevOps platform.
- **Continuous Delivery** this is a book that one reviewer called "required reading" for anyone working to tie the whole development and delivery process together.
- **Practical DevOps** talks about how DevOps works, and then moves into code storage, code testing, and deployment.
- **The DevOps Handbook** is considered a go-to for anyone in the field. It not only talks about the advantages of DevOps, but what it can do to give companies a competitive edge.
- **GitLab Quick Start Guide** is an eBook and a great guide for how to migrate to the GitLab platform.
- **Big Little Book on Git** is an eBook that talks to both the experienced DevOps professional and the beginner.

Certifications

- **DevOps Institute offers certifications** in areas such as development, DevOps engineering, DevOps testing, and security engineering.
- **GitLab has its own certifications** in areas like CI/CD, project management, and DevOps security.
- Always go to the source. For training on using Google Cloud, for instance, look to **the company's site for certifications**.



Bootcamps and courses

- For a monthly fee, the online training platform **A Cloud Guru** offers cloud certifications. It's set up to give users video content, hands-on labs, learning tools, quizzes, and exams.
- **LinkedIn Learning's DevOps Foundations** offers a solid knowledge base for anyone new, or fairly new, to DevOps. Free to anyone with a LinkedIn subscription, videos give users a rundown of the industry, along with key principles and technologies, like automation, collaboration, monitoring, and culture.
- **The DevOps Implementation Boot Camp**, offered by consulting firm Cprime, is a three-day course that starts at $1,695. Training is offered in-person or live online. Private team training also is available.
- **DevOps Culture and Mindset** is run by the University of California, Davis via Coursera, an online course provider. Fully online, the approximately 15-hour course focuses on foundational principles of DevOps. It's free with a Coursera subscription.
- **Continuous Delivery & DevOps** is an online, beginner-level, 8-hour course offered by the University of Virginia, through Coursera. It focuses on continuous delivery, testing, and infrastructure as code. It's free with a Coursera subscription.

About GitLab

GitLab is the most comprehensive, AI-powered DevSecOps Platform for software innovation. GitLab provides one interface, one data store, one permissions model, one value stream, one set of reports, one spot to secure your code, one location to deploy to any cloud, and one place for everyone to contribute. The platform is the only true cloud-agnostic end-to-end DevSecOps platform that brings together all DevSecOps capabilities in one place.

With GitLab, organizations can create, deliver, and manage code quickly and continuously to translate business vision into reality. GitLab empowers customers and users to innovate faster, scale more easily, and serve and retain customers more effectively. Built on open source, GitLab works alongside its growing community, which is composed of thousands of developers and millions of users, to continuously deliver new innovations.


GitLab