# What is DevOps?

DevOps is a modern software development approach that combines development (Dev) and operations (Ops) to streamline collaboration, accelerate delivery, and improve software quality.

Development (Dev) focuses on creating and building new software and features, while Operations (Ops) is responsible for maintaining, deploying, and running the software in a stable and reliable production environment

## DevOps defined

DevOps is a software development methodology that unites development and operations teams to deliver software faster, more securely, and more efficiently.

DevOps combines development (Dev) and operations (Ops) to increase the efficiency, speed, and security of software development and delivery compared to traditional processes. A more nimble software development lifecycle results in a competitive advantage for businesses and their customers.

# DevOps explained

DevOps practices enable software development (dev) and operations (ops) teams to accelerate delivery through automation, collaboration, fast feedback, and iterative improvement.

Stemming from an Agile approach to software development, a DevOps process expands on the cross-functional approach of building and shipping applications in a faster and more iterative manner.

In adopting a DevOps development process, you are making a decision to improve the flow and value delivery of your application by encouraging a more collaborative environment at all stages of the development cycle.DevOps represents a change in mindset for IT culture.

In building on top of Agile, lean practices, and systems theory, DevOps focuses on incremental development and rapid delivery of software. Success relies on the ability to create a culture of accountability, improved collaboration, empathy, and joint responsibility for business outcomes.

> DevOps is a combination of software development (dev) and operations (ops). It is defined as a software engineering methodology which aims to integrate the work of development teams and operations teams by facilitating a culture of collaboration and shared responsibility.

# DevOps methodology

The DevOps methodology aims to shorten the systems development lifecycle and provide continuous delivery with high software quality.

It emphasizes collaboration, automation, integration and rapid feedback cycles. These characteristics help ensure a culture of building, testing, and releasing software that is more reliable and at a high velocity.

This methodology comprises four key principles that guide the effectiveness and efficiency of application development and deployment. These principles, listed below, center on the best aspects of modern software development.

## Core DevOps principles

1. Automation of the software development lifecycle

This includes automating testing, builds, releases, the provisioning of development environments, and other manual tasks that can slow down or introduce human error into the software delivery process.

2. Collaboration and communication

A good DevOps team has automation, but a great DevOps team also has effective collaboration and communication.

3. Continuous improvement and minimization of waste

From automating repetitive tasks to watching performance metrics for ways to reduce release times or mean-time-to-recovery, high performing DevOps teams are regularly looking for areas that could be improved.

4. Hyperfocus on user needs with short feedback loops.

Through automation, improved communication and collaboration, and continuous improvement, DevOps teams can take a moment and focus on what real users really want, and how to give it to them.

By adopting these principles, organizations can improve code quality, achieve a faster time to market, and engage in better application planning.

## The four phases of DevOps

The evolution of DevOps has unfolded across four distinct phases, each marked by shifts in technology and organizational practices.

This progression reflects the growing complexity within DevOps, driven primarily by two key trends:

- **Transition to Microservices:** As organizations shift from monolithic architectures to more flexible microservices architectures, the demand for specialized DevOps tools has surged. This shift aims to accommodate the increased granularity and agility offered by microservices.
- **Increase in Tool Integration:** The proliferation of projects and the corresponding need for more DevOps tools have led to a significant rise in the number of integrations between projects and tools. This complexity has

prompted organizations to rethink their approach to adopting and integrating DevOps tools.

The evolution of DevOps has unfolded through four distinct phases, each addressing the growing demands and complexities of software development and delivery.

This four phases are as follows:

## Phase 1: Bring Your Own DevOps (BYOD)

In the Bring Your Own DevOps phase, each team selected its own tools. This approach caused problems when teams attempted to work together because they were not familiar with the tools of other teams. This phase highlighted the need for a more unified toolset to facilitate smoother team integration and project management.

## Phase 2: Best-in-class DevOps

To address the challenges of using disparate tools, organizations moved to the second phase, Best-in-class DevOps. In this phase, organizations standardized on the same set of tools, with one preferred tool for each stage of the DevOps lifecycle. It helped teams collaborate with one another, but the problem then became moving software changes through the tools for each stage.

## Phase 3: Do-it-yourself (DIY) DevOps

To remedy this problem, organizations adopted do-it-yourself (DIY) DevOps, building on top of and between their tools. They performed a lot of custom work to integrate their DevOps point solutions together.

However, since these tools were developed independently without integration in mind, they never fit quite right. For many organizations, maintaining DIY DevOps was a significant effort and resulted in higher costs, with engineers maintaining tooling integration rather than working on their core software product.

## Phase 4: DevOps Platform

A single-application platform approach improves the team experience and business efficiency. A DevOps platform replaces DIY DevOps, allowing visibility throughout and control over all stages of the DevOps lifecycle.

By empowering all teams – Development, Operations, IT, Security, and Business – to collaboratively plan, build, secure, and deploy software across an end-to-end unified system, a DevOps platform represents a fundamental step-change in realizing the full potential of DevOps.

GitLab's DevOps platform is a single application powered by a cohesive user interface, agnostic of self-managed or SaaS deployment. It is built on a single codebase with a unified data store, that allows organizations to resolve the inefficiencies and vulnerabilities of an unreliable DIY toolchain.

## How DevOps can benefit from AI and ML?

AI and ML enhance DevOps by automating testing, detecting anomalies, and improving security and performance monitoring.

Artificial intelligence (AI) and machine learning (ML) are still maturing in their applications for DevOps, but there is plenty for organizations to take advantage of today. They assist in analyzing test data, identifying coding anomalies that could lead to bugs, as well as automating security and performance monitoring to detect and proactively mitigate potential issues.

- AI and ML can find patterns, figure out the coding problems that cause bugs, and alert DevOps teams so they can dig deeper.
- Similarly, DevOps teams can use AI and ML to sift through security data from logs and other tools to detect breaches, attacks, and more. Once these issues are found, AI and ML can respond with automated mitigation techniques and alerting.
- AI and ML can save developers and operations professionals time by learning how they work best, making suggestions within workflows, and automatically provisioning preferred infrastructure configurations.

AI and ML excel in parsing vast amounts of test and security data, identifying patterns and coding anomalies that could lead to potential bugs or breaches. This capability enables DevOps teams to proactively address vulnerabilities and streamline alerting processes.

Read more about the benefits of AI and ML for DevOps

## What is a DevOps platform?

A DevOps platform is a unified system that integrates all development and operations tools into a single application for greater collaboration and visibility.

DevOps brings the human silos together and a DevOps platform does the same thing for tools. Many teams start their DevOps journey with a disparate collection of tools, all of which have to be maintained and many of which don't or can't integrate. A DevOps platform brings tools together in a single application for unparalleled collaboration, visibility, and development velocity.

A DevOps platform is how modern software should be created, secured, released, and monitored in a repeatable fashion. A true DevOps platform means teams can iterate faster and innovate together because everyone can contribute.

This integrated approach is pivotal for organizations looking to navigate the complexities of modern software development and realize the full potential of DevOps.

## Benefits of a DevOps culture

The benefits of DevOps and of a DevOps culture lies in the ability to improve the production environment in order to deliver software faster with continuous improvement.

You need the ability to anticipate and respond to industry disruptors without delay. This becomes possible within an Agile software development process where teams are empowered to be autonomous and deliver faster, reducing work in progress. Once this occurs, teams are able to respond to demands at the speed of the market.

There are some fundamental concepts that need to be put into action in order for DevOps to function as designed, including the need to:

- Remove institutionalized silos and handoffs that lead to roadblocks and constraints, particularly in instances where the measurements of success for

one team is in direct odds with another team's key performance indicators (KPIs).
- Implement a unified tool chain using a single application that allows multiple teams to share and collaborate. This will enable teams to accelerate delivery and provide fast feedback to one another.

### Key benefits:

Adopting a DevOps culture brings numerous benefits to an organization, notably in operational efficiency, faster delivery of features, and improved product quality. Key advantages include:

- **Enhanced Collaboration:** Breaking down silos between development and operations teams fosters a more cohesive working environment, leading to better communication and collaboration.
- **Increased Efficiency:** Automation of the software development lifecycle reduces manual tasks, minimizes errors, and accelerates delivery times. Continuous Improvement: DevOps encourages a culture of continuous feedback, allowing teams to quickly adapt and make improvements, ensuring that the software meets user needs effectively.
- **Higher Quality and Security:** With practices like continuous integration and delivery (CI/CD) and proactive security measures, DevOps ensures that the software is not only developed faster but also maintains high quality and security standards.
- **Faster Time to Market:** By streamlining development processes and improving team collaboration, organizations can reduce the overall time from conception to deployment, offering a competitive edge in rapidly evolving markets.

## What is the goal of DevOps?

The goal of DevOps is to increase software delivery speed and quality by fostering collaboration, accountability, and continuous improvement.

DevOps represents a change in mindset for IT culture. In building on top of Agile practices, DevOps focuses on incremental development and rapid delivery of software. Success relies on the ability to create a culture of accountability, improved collaboration, empathy, and joint responsibility for business outcomes.

Adopting a DevOps strategy enables businesses to increase operational efficiencies, deliver better products faster, and reduce security and compliance risk.

## The DevOps lifecycle and how DevOps works

The DevOps lifecyle stretches from the beginning of software development through to delivery, maintenance, and security. The stages of the DevOps lifecycle are:

- **Plan:** Organize the work that needs to be done, prioritize it, and track its completion.
- **Create:** Write, design, develop and securely manage code and project data with your team.
- **Verify:** Ensure that your code works correctly and adheres to your quality standards — ideally with automated testing.
- **Package:** Package your applications and dependencies, manage containers, and build artifacts.
- **Secure:** Check for vulnerabilities through static and dynamic tests, fuzz testing, and dependency scanning.
- **Release:** Deploy the software to end users.
- **Configure:** Manage and configure the infrastructure required to support your applications.
- **Monitor:** Track performance metrics and errors to help reduce the severity and frequency of incidents.
- **Govern**: Manage security vulnerabilities, policies, and compliance across your organization.

## DevOps tools, concepts and fundamentals

DevOps tools and practices include version control, Agile, CI/CD, and shift-left testing to accelerate delivery and improve quality.

DevOps covers a wide range of practices across the application lifecycle. Teams often start with one or more of these practices in their journey to DevOps success.

| Topic | Description |
|---|---|
| **Version control** | The fundamental practice of tracking and managing every change made to source code and other files. Version control is closely related to source code management. |
| **Agile** | Agile development means taking iterative, incremental, and lean approaches to streamline and accelerate the delivery of projects. |
| **Continuous Integration (CI)** | The practice of regularly integrating all code changes into the main branch, automatically testing each change, and automatically kicking off a build. |
| **Continuous Delivery (CD)** | Continuous delivery works in conjunction with continuous integration to automate the infrastructure provisioning and application release process. They are commonly referred to together as CI/CD. |
| **Shift left** | A term for shifting security and testing much earlier in the development process. Doing this can help speed up development while simultaneously improving code quality. |

## How does DevSecOps relate to DevOps?

DevSecOps extends DevOps by embedding security checks and compliance into every stage of the development lifecycle.

Security has become an integral part of the software development lifecycle, with much of the security shifting left in the development process. DevSecOps ensures that DevOps teams understand the security and compliance requirements from the very beginning of application creation and can properly protect the integrity of the software.

By integrating security seamlessly into DevOps workflows, organizations gain the visibility and control necessary to meet complex security demands, including vulnerability reporting and auditing. Security teams can ensure that policies are being enforced throughout development and deployment, including critical testing phases.

DevSecOps can be implemented across an array of environments such as on-premises, cloud-native, and hybrid, ensuring maximum control over the entire software development lifecycle.

## How are DevOps and CI/CD related?

CI/CD is a core DevOps practice that automates building, testing, and deploying software for faster and more reliable releases.

CI/CD — the combination of continuous integration and continuous delivery — is an essential part of DevOps and any modern software development practice. A purpose-built CI/CD platform can maximize development time by improving an organization's productivity, increasing efficiency, and streamlining workflows through built-in automation, continuous testing, and collaboration.

As applications grow larger, the features of CI/CD can help decrease development complexity. Adopting other DevOps practices — like shifting left on security and creating tighter feedback loops — helps break down development silos, scale safely, and get the most out of CI/CD.

## How does DevOps support the cloud-native approach?

DevOps supports cloud-native development by enabling scalable, collaborative, and faster application delivery in the cloud.

Moving software development to the cloud has so many advantages that more and more companies are adopting cloud-native computing. Building, testing, and deploying applications from the cloud saves money because organizations can scale resources more easily, support faster software shipping, align with business goals, and free up DevOps teams to innovate rather than maintain infrastructure.

Cloud-native application development enables developers and operations teams to work more collaboratively, which results in better software delivered faster.

Read more about the benefits of cloud-native DevOps environments

## What is a DevOps engineer?

A DevOps engineer is responsible for all aspects of the software development lifecycle, including communicating critical information to the business and customers.

Adhering to DevOps methodologies and principles, they efficiently integrate development processes into workflows, introduce automation where possible, and test and analyze code. They build, evaluate, deploy, and update tools and platforms (including IT infrastructure if necessary). DevOps engineers manage releases, as well as identify and help resolve technical issues for software users.

DevOps engineers require knowledge of a range of programming languages and a strong set of communication skills to be able to collaborate among engineering and business groups.

## Benefits of DevOps

The main benefits of DevOps include shorter release cycles, stronger collaboration, better responsiveness, and higher-quality software.

Adopting DevOps breaks down barriers so that development and operations teams are no longer siloed and have a more efficient way to work across the entire development and application lifecycle. Without DevOps, organizations often experience handoff friction, which delays the delivery of software releases and negatively impacts business results.

The DevOps model is an organization’s answer to increasing operational efficiency, accelerating delivery, and innovating products. Organizations that have implemented a DevOps culture experience the benefits of increased collaboration, fluid responsiveness, and shorter cycle times.

## Collaboration

Adopting a DevOps model creates alignment between development and operations teams; handoff friction is reduced and everyone is all in on the same goals and objectives.

## Fluid responsiveness

More collaboration leads to real-time feedback and greater efficiency; changes and improvements can be implemented quicker and guesswork is removed.

## Shorter cycle time

Improved efficiency and frequent communication between teams shortens cycle time; new code can be released more rapidly while maintaining quality and security.