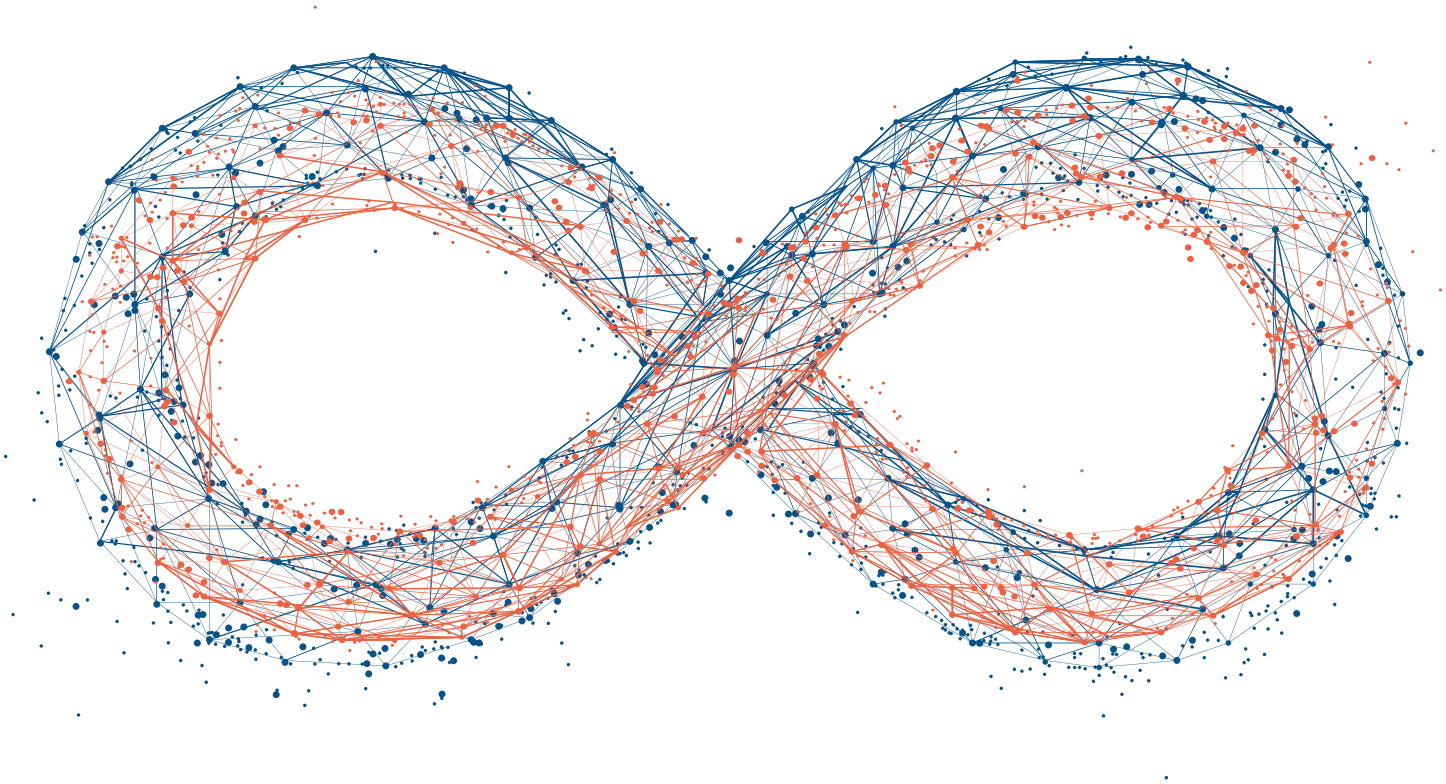


GoDgtl
by PruTech

DEVOPS EXPLAINED & BEST PRACTICES

Table Of Contents

Introduction

Software engineering teams are always on the lookout for ways that can help them improve their development process. Not just that, but also enhance efficiency. DevOps is one of the best solutions that come into the role here. It helps bring together a company's software developmen and IT operations teams, promoting collaboration and enhancing relationships.

But what is DevOps? How well do you know it? What advantages can it bring to your organization? Well, these are some questions that this white paper will address.

Additionally, this white paper also sheds insights into DevOps best practices and how they can unlock your business' potential.

What is DevOps?

DevOps is more than just an IT tool. It's about developing a mindset and having a cultural shift where teams focus on new processes to improve their work and working methods.

A DevOps culture means developers get closer to the user by understanding user requirements and needs. Operations teams get involved in the development process, add maintenance requirements, and meet customer needs.

Development and operations teams are no longer "siloed" under a DevOps model. Sometimes, these teams work as a single team. This means the engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function. Moreover, quality assurance and security teams integrate with development and operations throughout the application lifecycle in some DevOps models.

When security is the focus for a DevOps team, it is referred to as DevSecOps. These teams use practices to automate processes that historically have been manual and slow. They use a technology stack and tools, which help them operate and evolve applications quickly and reliably. These tools also help engineers deploy code or provisioning infrastructure that normally would have required assistance from other teams, increasing a team's velocity.

Benefits of DevOps

Speed: The DevOps model enables your developers to innovate for customers faster, adapt to changing markets better, and grow more efficiently. For instance, microservices and continuous delivery let teams take ownership of services and release updates quicker.

Rapid Delivery: The quicker you can release new features and fix bugs, the faster you respond to your customers' needs and build a competitive advantage. DevOps help you increase the frequency and pace of releases to innovate and improve your product faster. Its continuous integration and rapid delivery automate the software release process, from build to deployment.

Reliability: DevOps ensure the quality of application updates and infrastructure changes so you can reliably deliver at a rapid pace while maintaining a positive end-user experience. You can use continuous integration and continuous delivery practices for testing to ensure each change is functional and safe. Furthermore, monitoring and logging practices help you stay informed of performance in real-time.

Scale: DevOps help you operate and manage your infrastructure and development processes at scale. With its automation capabilities, you can manage complex or changing systems efficiently and reduce risk. For example, its infrastructure as code helps you manage development, testing, and production environments repetitively and efficiently.

Improved Collaboration: You can build more effective teams under a DevOps cultural model, emphasizing values such as ownership and accountability. As developers and operations teams collaborate closely, it reduces inefficiencies and saves time. For example, it decreases handover time between developers and operations.

Security: You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For instance, you can define and then track compliance at scale using infrastructure as code and policy as code.

DevOps: Best Practices Explained

DevOps' best practices help organizations innovate faster. These include automation, streamlining software development, and infrastructure management processes. And the use of the right tools helps accomplish these practices.

Organizations using a DevOps model deploy updates more often than those using traditional software development practices. This is because they perform very frequent but small updates, usually more incremental than the occasional updates performed under traditional release practices. This helps teams address bugs faster since teams can identify the last deployment causing the error.

Besides, companies might also use a microservices architecture to make their applications flexible and enable quicker innovation. Such microservices architecture decouples large, complex systems into simple, independent projects, breaking applications into multiple individual components. Each service is scoped to a single purpose or function. Moreover, it operates independently of its peer services and the application as a whole. This architecture reduces the coordination overhead of updating applications.

Additionally, organizations can proceed quickly when each service is paired with small, agile teams. However, the combination of microservices and increased release frequency leads to more deployments. This can present operational challenges.

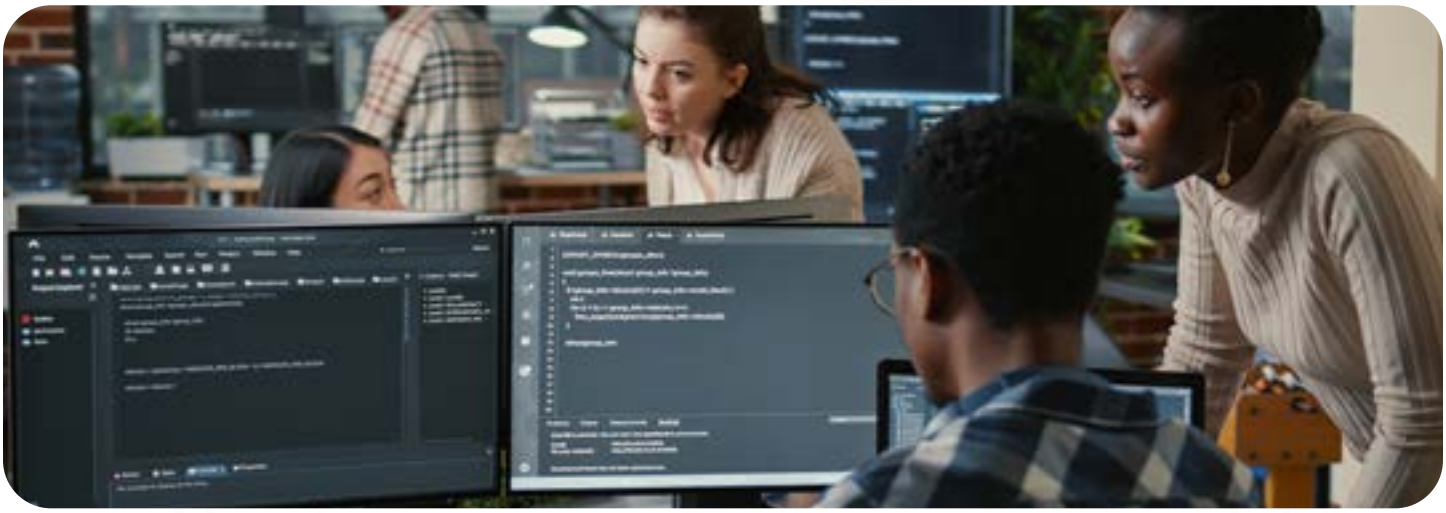
DevOps practices like continuous integration and continuous delivery help organizations deliver rapidly, safely, and reliably. Moreover, infrastructure automation practices like infrastructure as code and configuration management help keep computing resources elastic and responsive to frequent changes. Monitoring and logging help engineers track the performance of applications and infrastructure. Hence, they can react quickly to problems.

Continuous Integration

Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository. After which, automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process. It entails both an automation component (i.e., a CI or build service) and a cultural component (i.e., learning to integrate frequently). The key goals of continuous integration are finding and addressing bugs quicker, improving software quality, and reducing the time it takes to validate and release new software updates.

Earlier, developers worked in isolation for an extended period and only merged their changes to the master branch. This made merging code changes difficult and time-consuming, resulting in bugs accumulating for a long time without correction. These factors made it harder to deliver updates to customers quickly.

With continuous integration, developers frequently commit to a shared repository using a version control system such as Git. Before each commit, they may run local unit tests on their code as an extra verification layer prior to integrating. Besides, a continuous integration service automatically builds and runs unit tests on the new code changes to immediately surface errors.



Continuous Delivery

Continuous delivery is a DevOps software development practice where code changes are automatically prepared for production. Continuous delivery, known as the pillar of modern application development, expands upon continuous integration by deploying all code changes to testing or production environments after the build stage. When appropriately implemented, developers will always have a deployment-ready build artifact passed through a standardized test process.

Continuous delivery lets developers automate testing beyond just unit tests so they can verify application updates across multiple dimensions before deploying. These tests may include UI testing, load testing, integration testing, API reliability testing, etc. This helps developers more thoroughly validate updates and preemptively discover issues. It is easy and cost-effective to automate the creation and replication of multiple environments for testing, which was previously difficult to do on-premises with the cloud.

Microservices

The microservices architecture is a design approach to build a single application as a set of small services. Each service runs in its own process and communicates with other services through a well-defined interface using a lightweight mechanism, typically an HTTP-based application programming interface (API). Microservices are built around business capabilities; each service is scoped to a single purpose. You can use different frameworks or programming languages to write microservices and deploy them independently, as a single service, or as a group of services.

Netflix, eBay, Amazon, Twitter, PayPal, and other tech stars evolved from monolithic to microservices architecture. Unlike microservices, a monolith application is built as a single, autonomous unit. This makes changes to the application slow as it affects the entire system. A modification made to a small code section might require building and deploying an entirely new version of the software. Scaling specific functions of an application also means you have to scale the entire application.

Microservices solve these challenges of monolithic systems by being as modular as possible. In the simplest form, they help build an application as a suite of small services, each running in its own process and are independently deployable. These services may be written in different programming languages and may use different data storage techniques. While this results in developing scalable and flexible systems, it needs a dynamic makeover. Microservices are often connected via APIs and can leverage many tools and solutions grown in the RESTful and web service ecosystem. Testing these APIs can help validate the data flow and information throughout your microservice deployment.

Infrastructure as Code

Infrastructure as code is a practice that provisions and manages infrastructure using code and software development techniques, like version control and continuous integration. The cloud's API-driven model enables developers and system administrators to interact with infrastructure programmatically and at scale instead of manually setting up and configuring resources. Thus, engineers can interface with infrastructure using code-based tools and treat infrastructure like application code. Because they are defined by code, infrastructure and servers can quickly be deployed using standardized patterns, updated with the latest patches and versions, or duplicated in repeatable ways.

Configuration Management

Developers and system administrators use code to automate the operating system and host configuration, operational tasks, and more. The use of code makes configuration changes repeatable and standardized. It frees developers and system administrators from manually configuring operating systems, system applications, or server software.

Policy as Code

With infrastructure and its configuration codified with the cloud, organizations can monitor and enforce compliance dynamically and at scale. Infrastructure described by code can thus be tracked, validated, and reconfigured automatedly. This makes it easier for organizations to govern changes over resources and ensure that security measures are properly enforced in a distributed manner (For example, information security or compliance with PCI-DSS or HIPAA). This allows teams within an organization to move at a higher velocity since non-compliant resources can be automatically flagged for further investigation or even brought back into compliance.

Monitoring and Logging

Organizations monitor metrics and logs to see how application and infrastructure performance impacts their product's end-user experience. By capturing, categorizing, and analyzing data and logs generated by applications and infrastructure, they understand how changes or updates impact users, shedding insights into the root causes of problems or unexpected changes. Active monitoring becomes increasingly important as services must be available 24/7 and as application and infrastructure update frequency increases. Creating alerts or performing real-time analysis of this data also helps organizations proactively monitor their services. Developers and system administrators use code to automate the operating system and host configuration, operational tasks, and more. The use of code makes configuration changes repeatable and standardized. It frees developers and system administrators from manually configuring operating systems, system applications, or server software.

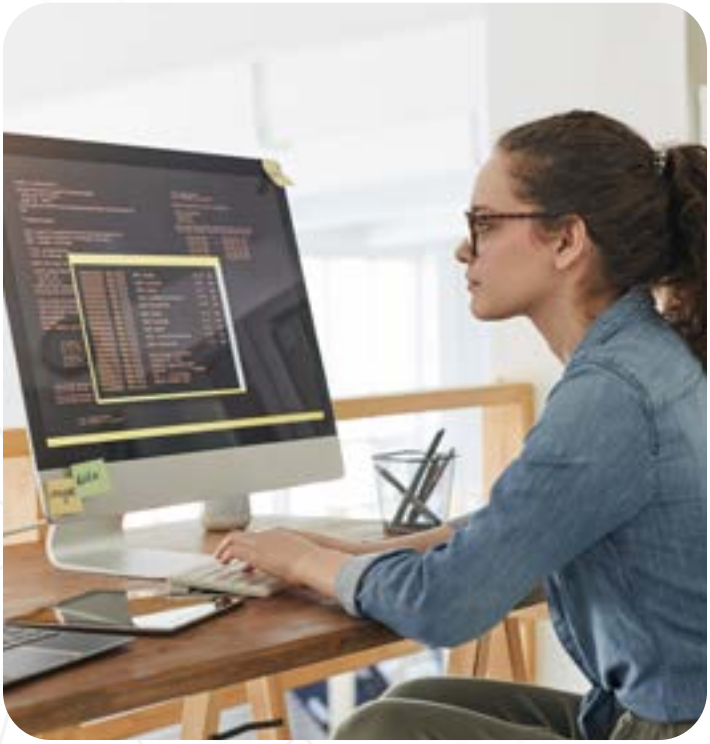
Communication and Collaboration

The use of DevOps tooling and automation of the software delivery process establishes collaboration by bringing together the workflows and responsibilities of development and operations teams. Moreover, these teams set strong cultural norms around information sharing and facilitating communication through chat applications, issue or project tracking systems, and wikis. This helps speed up communication across developers, operations, and other teams like marketing or sales, allowing the organization to align more closely on goals and projects.

Using AWS for DevOps

AWS is used for implementing DevOps culture with its tools and services. AWS provides many services that help you practice DevOps. These tools automate manual tasks and help teams manage complex environments and scale.

- **Get Started Fast:** You are not required to download or install any software. You just need to create an AWS account and use it for every AWS service. These services help you take advantage of AWS resources and operating infrastructure on your own.
- **Scalability**: With each service, you make the most out of AWS resources too quickly by simplifying, provisioning, configuring, and scaling. You can even scale up thousands of instances using AWS services.
- **Programmable**: You have the option to use each service via the AWS Command Line Interface or through APIs and SDKs. You can also model and provision AWS resources and your entire AWS infrastructure using declarative AWS CloudFormation templates.



- **Automation**: AWS helps you use automation to build faster and more efficiently. Using AWS services, you can automate manual tasks or processes such as deployments, development & test workflows, container management, and configuration management.
- **Secure**: Use AWS Identity and Access Management (IAM) to set user permissions and policies. This gives you granular control over who can access your resources and how they access those resources.
- **Pay-As-You-Go**: AWS gives you the freedom to pay only for what you use. It has no upfront fees, termination penalties, or long-term contracts. You can purchase services as you need them.

How Does GoDgtl Collaborate With AWS?

GoDgtl brings a team of experienced cloud experts who work directly with AWS to bring value and real solutions for your cloud projects. With direct access to AWS resources and in-house cloud consulting talent, GoDgtl guides you through your cloud journey, regardless of where you are on the path. Whether it is more knowledge-based information on cloud topics such as security, governance, compliance, or basic cloud migration aspects, or even if an assessment is needed, GoDgtl can provide a roadmap for your path to project completion and success.

GoDgtl
by PruTech

Our mission is to **help client organizations like yours access the latest resources and make their DX goals a reality.** Connect with our teams at Go-Dgtl to embrace new ideas and key enablers. **We promise to make your digital acceleration journey a success.**

go-dgtl.com/contact-us

ENABLE | TRANSFORM | ACHIEVE | ANALYZE | ADAPT

OUR LOCATIONS // Charlotte | Bangalore | Hyderabad | Mexico City | New Jersey (Iselin) | New York | Washington DC

CONTACT US // info@go-dgtl.com | (646) 536-7777 | go-dgtl.com