Simple Talk: DevOps

Last updated by | shakibesaib | Sep 29, 2023 at 3:16 PM GMT+6

DevOps is a culture, development or practice that accentuates the joint effort and correspondence of both developers (Dev) and other Operations professionals (Ops), along with Quality Assurance (QA) representatives, while automating the procedure of software development and framework changes. DevOps targets setting up a culture and condition where building, testing, and releasing software can happen quickly, much of the time, and more dependably.

Furthermore, the nine pillars of DevOps are leadership, collaborative culture, design for DevOps, continuous integration, continuous testing, elastic infrastructure, continuous monitoring, continuous security and continuous delivery.

What Is DevOps? Definition, Goals, Methodology, and Best Practices | Spiceworks It Security

In simple words, **DevOps** is a process between Devs(codes, developers) and It(operators, deploy, manages VMs etc.). Previously, operators had to handle lots of stuff(i.e. letting know the devs where to deploy etc.), at the same time the both party usually have a large communication gap. To improve this type of situation, DevOps being introduced or it's become big help as it helps automate the process(i.e. deploying process, where to deploy) which can resolves any conflicts, confusions etc. which results in a smother flow. Also it helps in testing phases. For example: it would've been hard to know that it worked in a particular VM or not easily. With the ability of creating these automation processes, devOps can easily test it. As a result, can conclude with the **benefits** written below:

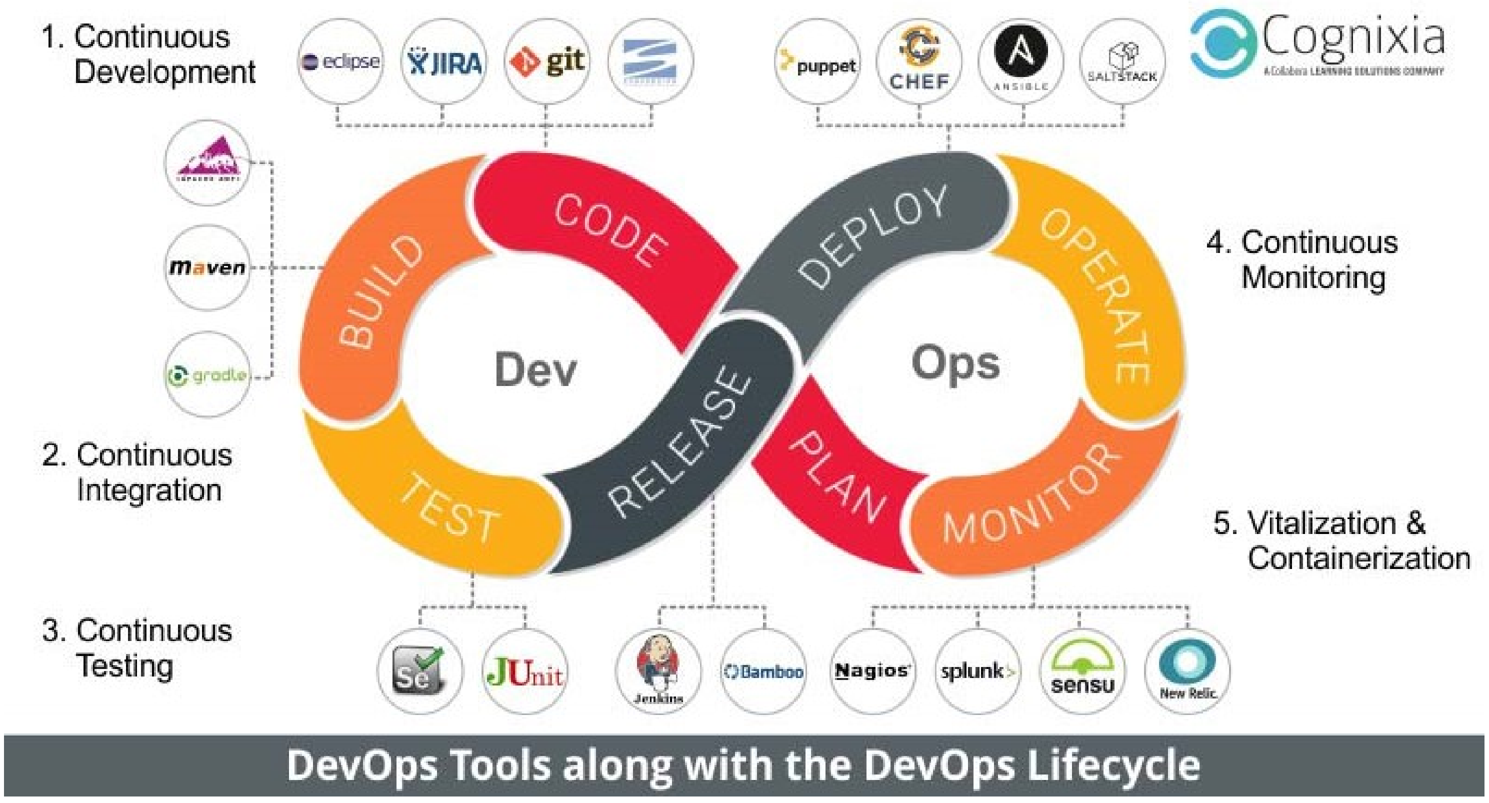
- Faster, efficient product delivery.
- Faster issue resolution and reduced complexity.
- Better scalability and availability.
- Stable operating environments.
- Better resource utilization.
- Automate extra works.
- Greater visibility into system results.

DevOps Culture

require major changes in normal working culture. But it creates the environment for high-performing teams to develop.

- Collaboration, visibility, and alignment
- Shifts in scope and accountability
- Shorter release cycles
- Continuous learning

Life cycle



Each phase relies on the others, and the phases are not role-specific. In a true DevOps culture, each role is involved in each phase to some extent.

How DevOps improves the 4 phases of an application lifecycle?

Plan (Agility and Visibility): teams define the application or system they are building along with capabilities and features. Teams can benefit from agility and visibility that in this phase, by creating **backlogs**, tracking **bugs**, using **boards**, and visualizing progress with **dashboards**. Azure boards also have reporting feature.

Develop (Automation): Development consists of writing, testing, reviewing, and integrating coding, and the process of deploying artifacts into environments. People who work with Azure DevOps can use VS Code for faster coding. **Azure Pipelines** automates testing and enables continuous integration in the cloud and environments can be quickly provisioned with **Azure DevTest Labs**.

Deliver (Easily Repeatable): When deploying applications into a production environment, teams need to do this in a consistent and reliable way. Automating processes in the deliver phase allows them to be **scalable**, **repeatable**, and **controlled**.

Operate (Monitor & Security): Dev teams create system reliability, high availability, and aim for zero downtime while reinforcing security and governance. When there are issues, teams want the opportunity to identify issues before they affect the user experience and solve them quickly when they do occur. Can implement full stack-monitoring, get actionable alerts and gain insights from logs (i.e. Azure Monitor). Furthermore, DevOps can help managing cloud environment, helping the teams to stay compliant (i.e. Azure Automation, Azure Blueprints). Then, with these technologies, DevOps Engineers can help limit threat exposure and fix vulnerabilities (i.e. Azure Security Center).

DevOps Practices

CI/CD: Continuous integration is the practice of integrating all your code changes into the main branch of a shared source code repository early and often, automatically testing each change when we merge them, and automatically kicking off a build. With continuous integration, errors and security issues can be identified and

fixed more easily, and much earlier in the software development lifecycle.
Continuous delivery is a software development practice that works in conjunction with continuous integration to automate the infrastructure provisioning and 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 can be deployed as packaged with everything it needs to deploy to any environment at any time. Continuous delivery can cover everything from provisioning the infrastructure to deploying the application to the testing or production environment.

Version Control: Version control is the practice of managing code in versions—tracking revisions and change history to make code easy to review and recover.
The use of version control is a fundamental DevOps practice, helping development teams work together, divide coding tasks between team members, and store all code for easy recovery if needed.
Version control is also a necessary element in other practices such as continuous integration and infrastructure as code.

Agile Software Development: Agile is a software development approach that emphasizes team collaboration, customer and user feedback, and high adaptability to change through short release cycles. Teams that practice Agile provide continual changes and improvements to customers, collect their feedback, then learn and adjust based on customer wants and needs.

Infrastructure as Code: Infrastructure as code defines system resources and topologies in a descriptive manner that allows teams to manage those resources as they would code. Those definitions can also be stored and versioned in version control systems, where they can be reviewed and reverted—again like code.
An approach to managing data center server, storage, and networking infrastructure. It is meant to significantly simplify large-scale configuration and management. For example: for Azure - ARM, for cross-platform -terraform

Configuration management: Configuration management refers to managing the state of resources in a system including servers, virtual machines, and databases. Using configuration management tools, teams can roll out changes in a controlled, systematic way, reducing the risks of modifying system configuration.

Continuous Monitoring: full, real-time visibility into the performance and health of the entire application stack, from the underlying infrastructure running the application to higher-level software components.