A.V.C COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER APPLICATIONS

(A Recognized as a Research Centre

Approved by Anna University , Chennai)

"CAS Newsletter"

Volume:14 **Month: May'22** **Issue:5**

Dear Students,

Warm welcome to the academic session 2022-23, Our actions are born from our thoughts, our thoughts are the product of our values, and values come from our own belief system. If we believe life is miserable, we attract misery; if we believe life is beautiful, we attract happiness. Wish you all Happy and Successful Session 2022-23.

I wish them all success.

Dr. S.SELVAMUTHUKUMARAN.

LEARNER → WRITER

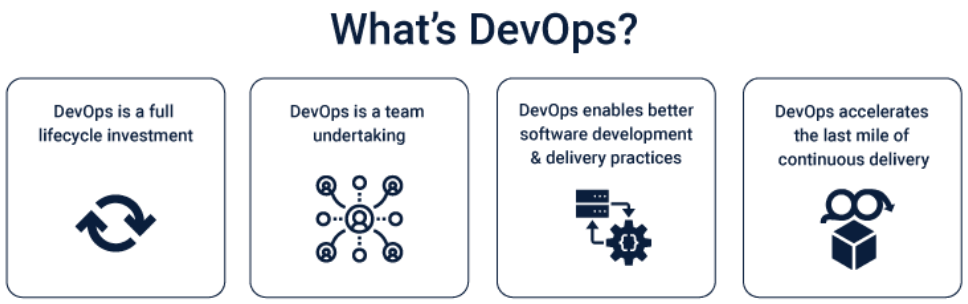
DevOps

M. Selvakumar - I MCA

Introduction

What is DevOps?

The word *DevOps* is a combination of the terms development and *operations*, meant to represent a collaborative or shared approach to the tasks performed by a company's application development and IT operations teams.



DevOps is a philosophy that promotes better communication and collaboration between these teams -- and others -- in an organization. In its most narrow interpretation, DevOps describes the adoption of iterative software development, automation, and programmable infrastructure deployment and maintenance. The term also covers culture changes, such as building trust and cohesion between developers and systems administrators and aligning technological projects to business requirements. DevOps can change the software delivery chain, services, job roles, IT tools and best practices.

These include the following:

- continuous integration and continuous delivery or continuous deployment (CI/CD) tools, with an emphasis on task automation;
- systems and tools that support DevOps adoption, including real-time monitoring, incident management, configuration management and collaboration platforms; and
- cloud computing, microservices and containers implemented concurrently with DevOps methodologies.

A DevOps approach is one of many techniques IT staff use to execute IT projects that meet business needs. DevOps can coexist with

- Agile software development;
- IT service management frameworks, such as ITIL;
- project management directives, such as Lean and Six Sigma;

How does DevOps work?

DevOps is a methodology meant to improve work throughout the software development lifecycle. You can visualize a DevOps process as an infinite loop, comprising these steps: plan, code, build, test, release, deploy, operate, monitor and -- through feedback -- plan, which resets the loop. Ideally, DevOps means that an IT team writes software that perfectly meets user requirements, deploys without any wasted time and runs optimally on the first try. Organizations use a combination of culture and technology to pursue this goal. To align software to expectations, developers and stakeholders communicate about the project, and developers work on small updates that go live independently of each other. To avoid wait times, IT teams use CI/CD pipelines and other automation to move code from one step of development and deployment to another. Teams review changes immediately and can enforce policies to ensure releases meet standards. It's easy to write software quickly; writing software that works is another story. To deploy good code to production, DevOps adherents use containers or other methods to make the software behave the same way from development through testing and into production. They deploy changes individually so that problems are traceable. Teams rely on configuration management for consistent deployment and hosting environments. Problems they discover in live operations lead to code improvements, often through a blameless post-mortem investigation and continuous feedback channels.

Developers might support the live software, which puts the onus on them to address runtime considerations. IT operations administrators might be involved in the software design meetings, offering guidance on how to use resources efficiently and securely. Anyone can contribute to blameless post-mortems. The more these specialists collaborate and share skills, the more they can foster a DevOps culture.

What problems does DevOps solve?

DevOps solves communication and priority problems between IT specializations. To build viable software, development teams must understand the production environment and test their code in realistic conditions. A traditional structure puts development and operations teams in silos. This means developers are satisfied when their code delivers functionality -- and if the release breaks in production, it's up to the operations team to fix the problems. With a DevOps culture, developers don't resort to the "It worked on my machine" response when a problem arises. Changes rolled out to production are small and reversible. Plus, the whole team understands the changes, which greatly simplifies incident management. With a faster process from idea to live software, companies can capitalize on market opportunities. In this way, DevOps provides a competitive advantage for businesses.

The evolution of DevOps

Patrick Debois, a software development consultant, is credited with creating the term *DevOps* in 2009 by naming a conference DevOps Days. DevOps addressed a shortcoming of the Agile software development methodology: Iterative, rapid code development did not necessarily lead to iterative, rapid code deployment.

Early proponents of DevOps included these key players:

- Debois and collaborator Andrew Clay Shafer;
- *The Phoenix Project* authors Gene Kim, Kevin Behr and George Spafford;
- Paul Hammond and John Allspaw, influential early adopters at Flickr;
- continuous delivery pioneers Jez Humble and Dave Farley;
- containerization advocate John Willis; and
- State of DevOps Report organizers, including Alanna Brown and Nicole Forsgren.

As DevOps became popular, organizations formalized DevOps approaches. Retailer Target originated the DevOps Dojo training method, for example. Vendors touted DevOps-enabling capabilities in tools, from collaboration chatbots to CI/CD suites built into cloud services.

DevOps continues to evolve, as artificial intelligence surfaces to aid in everything from code creation to incident management. AI for DevOps (or *AIOps*) means smarter automation and even shorter wait times, even seamless translations from business need to technological offering -- but plenty of barriers remain before this becomes reality.

Methodologies, principles and strategies

DevOps is associated with Agile software development because Agile practitioners promoted DevOps as a way to extend the methodology into production. The approach has even been labeled a counterculture to the IT service management practices championed in ITIL. DevOps does not have an official framework. To hone their strategies, organizations should understand the related contexts of DevOps, Agile and Waterfall development, site reliability engineering (SRE) and SysOps, and even the variations within DevOps.

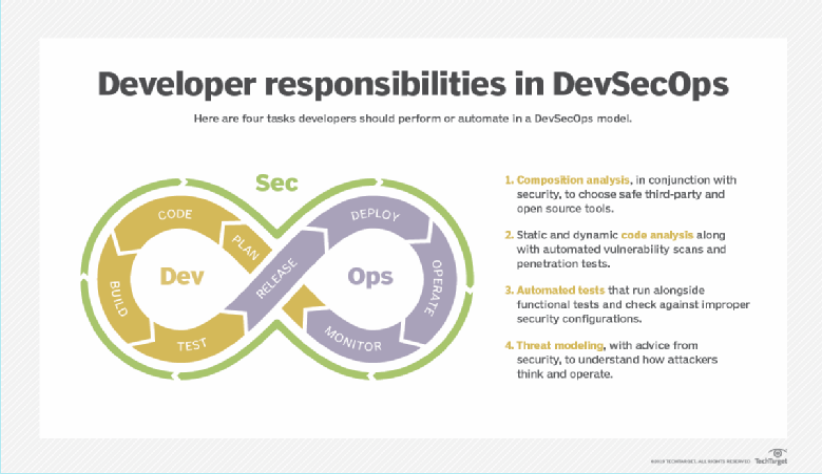
DevOps vs. Agile development.

Agile is a software development approach defined in the Agile Manifesto. Agile teams focus on incremental and rapid cycles of code creation and delivery, referred to as *sprints*. Each sprint iterates upon the last, which makes the software highly flexible and adaptable to changing requirements. It is possible for the original vision of a project to be lost through this cycle. DevOps arose from Agile's success at improving development speed, and the realization that disconnects between development and operations teams -- as well as between IT and the business side of the organization -- significantly

hindered the Agile software's delivery to users. In an Agile-only workflow, development and operations teams have separate objectives and leadership. When an organization uses DevOps and Agile together, both development and operations teams manage code throughout the software development lifecycle. While Agile work is often formalized with a framework, such as Scrum, DevOps does not have a framework.

DevOps vs. SRE.

Site reliability engineering arose concurrently with Agile and DevOps. Started in the early 2000s at Google, it is essentially a programming- and automation-focused approach to the software development lifecycle. Problems should be solved in a way that prevents them from occurring again. Rote tasks should be minimized. The SRE toolbox closely matches that for DevOps. Both disciplines aim for continuous improvement. SRE and DevOps engineers seek to abolish silos between development and operations. While DevOps also can extend to business stakeholders, SRE typically stays within the confines of IT processes.



DevOps benefits and challenges

DevOps benefits include the following:

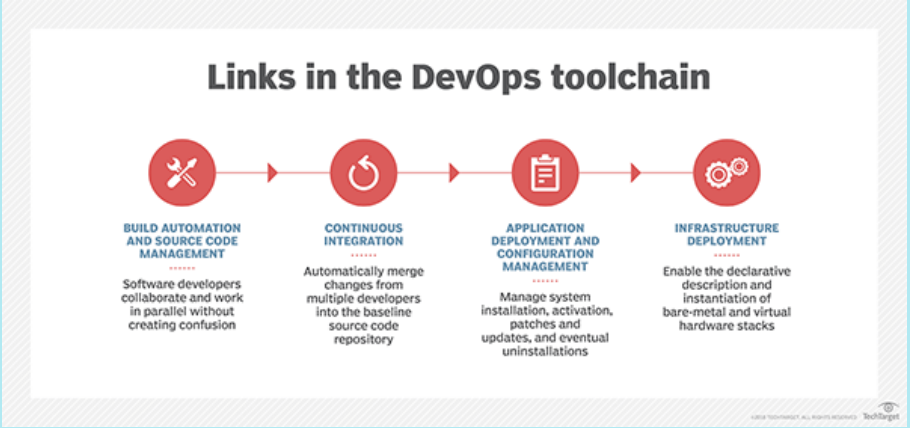
- Fewer silos and increased communications between IT groups;
- Faster time to market for software;
- Rapid improvement based on feedback;
- Less downtime;
- Improvement to the entire software delivery pipeline through builds, validations and deployment;
- Less menial work, thanks to automation;

However, DevOps challenges abound:

- Organizational and IT departmental changes, including new skills and job roles;
- Expensive tools and platforms, including training and support to use them effectively;
- Development and IT tool proliferation;
- Unnecessary, fragile or unsafe automation;
- Scaling devops across multiple projects and teams;
- Riskier deployment due to a fail-fast mentality and job generalization vs. Specialization;
- Regulatory compliance, especially when role separation is required.

DevOps Tools

DevOps is a mindset, not a tool set. But it's hard to do anything in an IT team without the right tools. In general, DevOps practitioners rely on a CI/CD pipeline, containers and cloud hosting. Tools can be open source, proprietary or supported distributions of open source technology.



Code repositories.

Version-controlled source code repositories enable multiple developers to work on code. Developers check code out and in, and they can revert to a previous version of code if needed. These tools keep a record of modifications made to the source code. Without tracking, developers may struggle to follow which changes are recent and which versions of the code are available to end users.

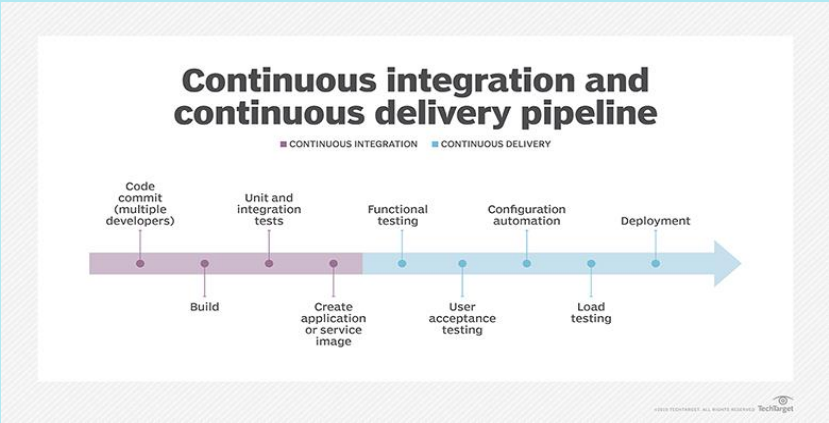
In a CI/CD pipeline, a code change committed in the version-control repository automatically triggers next steps, such as a static code analysis or build and unit tests. Tools for source code management include Git and GitHub.

Artifact repositories.

Source code is compiled into an artifact for testing. Artifact repositories enable version-controlled, object-based outputs. Artifact management is a good practice for the same reasons as version-controlled source code management. Examples of artifact repositories include JFrog Artifactory and Nexus Repository.

CI/CD pipeline engines.

CI/CD enables DevOps teams to frequently validate and deliver applications to the end user through automation during the development lifecycle. The continuous integration tool initializes processes so that developers can create, test and validate code in a shared repository as often as needed without manual work. Continuous delivery extends these automatic steps through production-level tests and configuration setups for release management. Continuous deployment goes a step further, invoking tests, configuration and provisioning, as well as monitoring and potential rollback capabilities. Common tools for CI, CD or both include Jenkins, GitLab and CircleCI.



Configuration management.

Configuration management systems enable IT to provision and configure software, middleware and infrastructure based on a script or template. The DevOps team can set up deployment environments for software code releases and enforce policies on servers, containers and VMs through a configuration management tool. Changes to the deployment environment can be version controlled and tested, so DevOps teams can manage infrastructure as code. Configuration management tools include Puppet and Chef.

Cloud environments.

DevOps organizations often concurrently adopt cloud infrastructure because they can automate its deployment, scaling and other management tasks. AWS and Microsoft Azure are among the most used cloud providers. Many cloud vendors also offer CI/CD services.

Monitoring.

Additionally, monitoring tools enable DevOps professionals to observe the performance and security of code releases on systems, networks and infrastructure. They can combine monitoring with analytics tools that provide operational intelligence. DevOps teams use these tools together to analyze how changes to code affect the overall environment. Choices are wide-ranging, but include New Relic One, Dynatrace, Prometheus, Datadog and Splunk.

As-a-service models.

DevOps as a service is a delivery model for a set of tools that facilitates collaboration between an organization's software development team and the IT operations team. In this delivery model, the provider assembles a suite of tools and handles the integrations to seamlessly cover the overall process of code creation, delivery and maintenance.