A beginner's guide to everything DevOps

Take a fresh look at why DevOps is important, what it means for IT professionals, and its methods, frameworks, and tools.

By Sameer S Paradkar

February 27, 2020 | 3 Comments | 11 min read

Image by: *Opensource.com*

A great deal has happened since DevOps became a common term in the IT world. With so much of the ecosystem being open source, it's important to review why it started and what it means to IT careers.

What is DevOps?

While there is no single definition, I consider DevOps to be a process framework that ensures collaboration between development and operations teams to deploy code to production environments faster in a repeatable and automated way. We will spend the rest of this article unpacking that statement.

The word "DevOps" is an amalgamation of the words "development" and "operations." DevOps helps increase the speed of delivering applications and services. It allows organizations to serve their customers efficiently and become more competitive in the market. In simple terms, DevOps is an alignment between development and IT operations with better communication and collaboration.

DevOps assumes a culture where collaboration among the development, operations, and business teams is considered a critical aspect of the journey. It's not solely about the tools, as DevOps in an organization creates continuous value for customers. Tools are one of its pillars, alongside people and processes. DevOps increases organizations' capability to deliver high-quality solutions at a swift pace. It automates all processes, from build to deployment, of an application or a product.

The DevOps discussion centers around the relationship between developers, the individuals who write software for a living, and operators, those responsible for maintaining that software.

Challenges for the development team

Developers are typically enthusiastic and willing to adopt new approaches and technologies to solve organizations' problems. However, they do face challenges, such as:

- The competitive marketplace creates a lot of pressure for on-time delivery.
- They must cater to production-ready code management and new capability implementation.
- The release cycle is long, so the development team has to make several assumptions before application deployment. In such a scenario, it takes more time to resolve issues that occur during deployment in the production or staging environment.

Challenges for the operations team

Operators are historically focused on the stability and reliability of IT services. The operations team is, therefore, concerned about making changes to resources, technologies, or approaches as they look for stability. Their challenges include:

- Managing resource contention as demands for resources increase
- Handling redesigns or tweaks required for application execution in a production environment
- Diagnosing and resolving production-related issues after application deployment in isolation

How DevOps addresses the dev and ops challenges

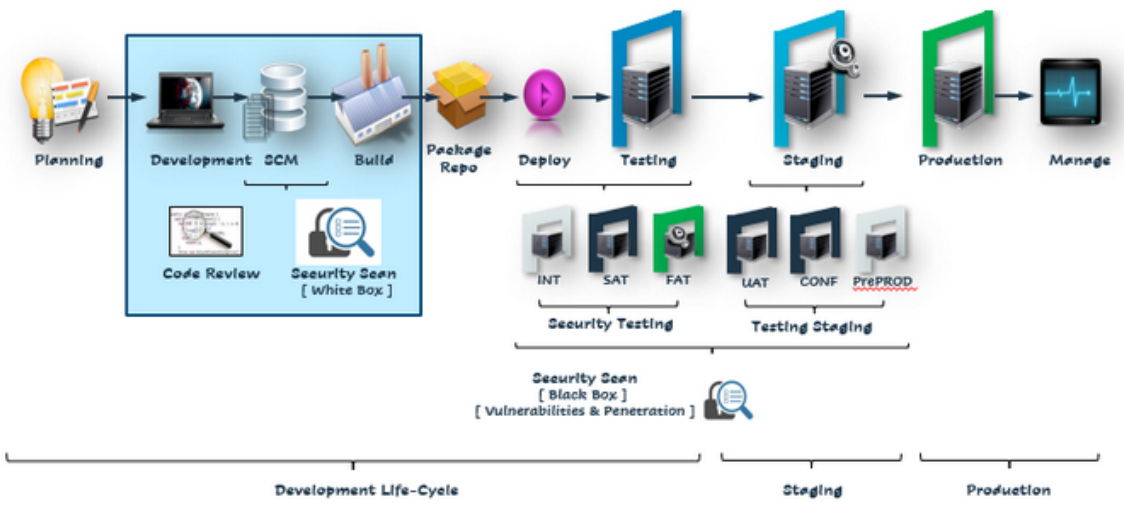
Instead of releasing a large number of application features at once, companies are trying to see if they can roll out a small number of features to their customers through a series of release iterations. This has several advantages, like better software quality, quicker feedback from customers, etc. These, in turn, ensure high customer satisfaction. To achieve these objectives, companies are required to:

- Lower the failure rate for new releases
- Increase deployment frequency
- Achieve quicker mean time to recovery in the event a new release crashes the application
- Shorten the lead time between fixes

DevOps fulfills all these objectives and helps achieve seamless delivery. Organizations adopt DevOps to achieve levels of performance that were unthinkable even a few years ago. They are doing tens, hundreds, or even thousands of deployments per day while delivering world-class reliability, stability, and security. (Read more about batch sizes and the impact on software delivery.)

DevOps attempts to address a variety of problems that result from predecessor methodologies, including:

- The development and operation teams working in isolation
- Testing and deployment as isolated phases done after design and build and requiring more time than the build cycles
- Team members spending excessive time in testing, deploying, and designing instead of focusing on the core—creating business services
- Manual code deployment leading to errors in production
- Development and operations teams on separate, asynchronous timelines, which causes additional delays



DevOps vs. agile vs. traditional IT

DevOps is often discussed in relation to other IT practices, notably agile and waterfall IT.

Agile is a set of principles, values, and methods for producing software. For example, if you have an idea that you want to convert into software, you can leverage agile principles and values. But that software might only work in the development or test environment. You need a way to quickly and repeatably transfer the software into the production environment in a simple and safe way, and the way is through DevOps tools and techniques. Agile software development methodology focuses on development processes, but DevOps is responsible for development and deployment—in the safest and most reliable way.

Comparing the traditional software waterfall model with DevOps is a good way to understand the benefits DevOps provides. The following example assumes the application is scheduled to go live in four weeks, coding is 85% complete, the application is a fresh launch, and the process of procuring the servers to ship the code has just been initiated.

More DevOps resources

- What is DevOps?
- The ultimate DevOps hiring guide
- DevOps monitoring tools guide
- Getting started with DevSecOps
- Download the DevOps glossary
- eBook: Ansible for DevOps
- Latest DevOps articles

Traditional process	DevOps process
After placing an order for new servers, the development team works on testing. The operations team works on extensive paperwork, as required in enterprises, to deploy the infrastructure.	After placing an order for new servers, the development and operations teams work together on the processes and paperwork to set up the new servers. This results in better visibility into the infrastructure requirements.
Details about failover, redundancy, data-center locations, and storage requirements are skewed, as no inputs are available from the development team that has deep knowledge of the application.	Details about failover, redundancy, disaster recovery, data-center locations, and storage requirements are known and correct due to the inputs from the development team.
The operations team has no idea of the development team's progress. The operations team develops a monitoring plan based on their understanding.	The operations team is completely aware of the progress the development team is making. The operations team interact with the development team, and they jointly develop a monitoring plan that caters to the IT and business needs. They also leverage application performance monitoring (APM) tools.
Before go-live, the load test crashes the application, which delays the release.	Before the go-live, the load test makes the application slow. The development team swiftly fixes the bottlenecks, and the application is released on time.

DevOps lifecycle

DevOps includes the adoption of certain common practices.

Continuous planning

Continuous planning leverages lean principles in order to start smaller by identifying the resources and outcomes needed to test the business value or vision, adapt continually, measure progress, learn from the customers' needs, shift direction as needed with agility, and update the business plan.

Collaborative development

The collaborative development process enables collaboration between business, development, and test teams that are spread across different time zones to deliver quality software continuously. This includes multiplatform development, support for polyglot programming, creation of user stories, elaboration of ideas, and lifecycle management. Collaborative development includes the process and practice of continuous integration, which promotes frequent code integrations and automatic builds. By integrating application code frequently, integration issues are identified earlier in the lifecycle (when they are easier to fix), and the overall

integration effort is reduced via continuous feedback because the project shows continuous and demonstrable progress.

Continuous testing

Continuous testing reduces the testing cost while helping the development teams balance speed and quality. It also eliminates testing bottlenecks through virtualized services and simplifies the creation of virtualized test environments that can be easily shared, deployed, and updated as systems change. These capabilities reduce the cost of provisioning and maintaining test environments and shorten test-cycle times by allowing integration testing early in the lifecycle.

Continuous release and deployment

This adoption path entails one major practice: continuous release and deployment. Continuous release and deployment provide a continuous delivery pipeline that automates key processes. It reduces the number of manual processes, wait times for resources, and amount of rework by enabling push-button deployment that ensures higher numbers of releases, reduced errors, and end-to-end transparency.

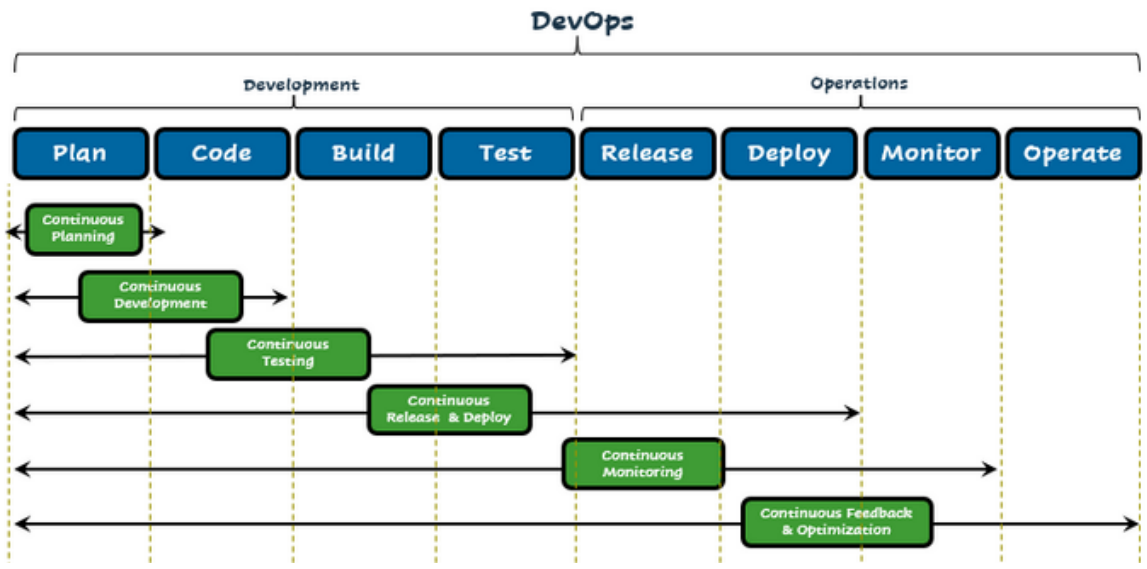
Automation plays a key role in ensuring the software is released repeatably and reliably. One critical objective is to take manual processes like build, regression, deployment, and infrastructure provisioning and automate them. This requires version control for source code; test and deployment scripts; infrastructure and application configuration data; and the libraries and packages the application depends on. The ability to query the state of all environments is another important factor.

Continuous monitoring

Continuous monitoring ensures enterprise-grade reporting capabilities that help development teams understand the availability and performance of applications in the production landscape, even before they are deployed to production. The early feedback provided by continuous monitoring is critical for lowering the cost of errors and for steering projects in the right direction. This practice often includes observability tools, which tend to expose metrics related to application performance.

Continuous feedback and optimization

Continuous feedback and optimization provide visual evidence for analyzing customer journeys and pinpointing pain areas. Feedback can be enabled for both pre- and post-production phases to maximize value and ensure that even more transactions are successfully completed. This provides immediate visibility into the root cause of customer struggles that affect behavior and impact business.



Benefits of DevOps

DevOps can facilitate a collaborative environment where developers and operators work as a team toward common ends. A major milestone in this process is the implementation of continuous integration and continuous delivery (CI/CD). This enables teams to launch software to the market faster with fewer errors.

Important benefits of DevOps are:

Predictability: DevOps offers a significantly lower failure rate for new releases.

Maintainability: It enables effortless recovery in the event of a new release crashing or disabling the application.

Reproducibility: Versioning the builds or the code enables earlier versions to be restored as needed.

Higher quality: Incorporating infrastructure issues improves application development quality.

Time to market: Streamlined software delivery reduces time to market by 50%.

Reduced risk: Incorporating security into the software lifecycle reduces defects across its lifecycle.

Cost-efficiency: Driving cost-efficiency in software development pleases senior management.

Resiliency: The software system is more stable, secure, and changes are auditable.

Breaks larger codebase into manageable pieces: DevOps is based on the agile programming method, which supports breaking down larger codebases into smaller, more manageable chunks.

DevOps principles

DevOps adoption has produced several principles that have evolved (and are still evolving). Most solution providers have developed their own variants. All these principles take a holistic approach to DevOps, and organizations of all sizes can adopt them.

Develop and test against a production-like environment

The objective is to allow the development and quality assurance (QA) teams to develop and test against systems that behave like production systems, so they can see how the application behaves and performs well before it is ready for deployment.

The application should be exposed to production-like systems as early in the lifecycle as possible to address three major potential issues. First, this permits the application to be tested in an environment that is close to the actual environment. Second, it allows application-delivery processes to be tested and validated upfront. Third, it enables the operations team to verify early in the lifecycle how their environment will behave when applications are deployed, thereby allowing them to create a finely tweaked, application-aware environment.

Deploying with repeatable, reliable processes

This principle allows the development and operations teams to support an agile software development process throughout the lifecycle. Automation is critical to creating processes that are iterative, reliable, and repeatable. Hence, the organization must create a delivery pipeline that enables continuous, automated deployment and testing. Frequent deployments also allow teams to test the deployment processes, thereby lowering the risk of deployment failures during actual releases.

Monitor and validate operational quality

Organizations are good at monitoring applications in production because they have tools that capture metrics and key performance indicators (KPIs) in real time. This principle moves monitoring to early in the lifecycle, ensuring that automated testing monitors functional and nonfunctional attributes of the application early in the process. Whenever an application is tested and deployed, quality metrics should be captured and analyzed. Monitoring tools provide early warning about operational and quality issues in the landscape that may occur in production. These metrics should be gathered in a format that all business stakeholders can understand and articulate.

Amplify feedback loops

One objective of DevOps processes is to enable organizations to react and make changes more rapidly. In software delivery, this goal requires an organization to get early feedback and then learn rapidly from every action it takes. This principle calls for organizations to create communication channels that allow stakeholders to access and act on feedback. Development may act by adjusting its project plans or priorities. Production may act by enhancing production environments.

DevOps tools cheat sheet

#	Domain	Lifecycle	Popular Open Source DevOps Tools
1	Dev	Plan	Kanboard, Wekan, and other alternatives to Trello; GitLab, Tuleap, Redmine, and other alternatives to JIRA; Mattermost, Roit.im, IRC, and other alternatives to Slack
2		Code	Git, Gerrit, Bugzilla; Jenkins and other open source CI/CD tools
3		Build	Apache Maven, Gradle, Apache Ant, Packer

#	Domain	Lifecycle	Popular Open Source DevOps Tools
4		Test	JUnit, Cucumber, Selenium, Apache JMeter
5	Ops	Release	Kubernetes, Nomad, Jenkins, Zuul, Spinnaker, Ansible, Apache ZooKeeper, etcd, Netflix Archaius, Terraform
6		Deploy	
7		Operate*	
8		Monitor	Grafana, Prometheus, Nagios, InfluxDB, Fluentd, and others covered in this guide

* Operating is best numbered as the final step for Operations teams, but its tooling overlaps with release and deploy lifecycle stages. It was moved for ease of readability.

In conclusion

DevOps is an increasingly popular methodology that aims to connect developers and operators into a cohesive unit. It's uniquely differentiated from traditional IT operations and complements (but is not the same as) agile.

Would you consider your IT department to be DevOps adopters? Please tell your story in the comments.

This article is an adaptation of DevOps Tools & Frameworks: Everything You Need To Know. It has been republished with permission.

What to read next

DevOps vs Agile: What's the difference?

The difference between the two is what happens after development.

Taz Brown (Alumni)

7 CI/CD tools for sysadmins

An easy guide to the top open source continuous integration, continuous delivery, and continuous deployment tools.

Dan Barker (Alumni)

A beginner's guide to building DevOps pipelines with open source tools

If you're new to DevOps, check out this five-step process for building your first pipeline.

Bryant Son (Alumni)

Tags: DEVOPS

Sameer S Paradkar

Sameer Paradkar is an Enterprise Architect with 15+ years of extensive experience in the ICT industry, which spans across Consulting, Product Development, and

Systems Integration. He is an Open Group TOGAF, Oracle Master Java EA, TMForum NGOSS, IBM SOA Solutions, IBM Cloud Solutions, IBM MobileFirst, ITIL Foundation V3, COBIT 5, and AWS certified enterprise architect.

More about me

3 Comments

These comments are closed.

Marco Bravo | February 27, 2020 No readers like this yet.

Nice article. I like it. Thanks for sharing.

Dandenong | February 28, 2020 No readers like this yet.

nice

os | March 16, 2020 No readers like this yet.

Nice Article, really its very helpful to the beginner's.

Related Content

Delegate common tasks with an open source automation tool

4 questions open source engineers should ask to mitigate risk at scale

How the Gherkin language bridges the gap between customers and developers

A note on advertising: Opensource.com does not sell advertising on the site or in any of its newsletters.