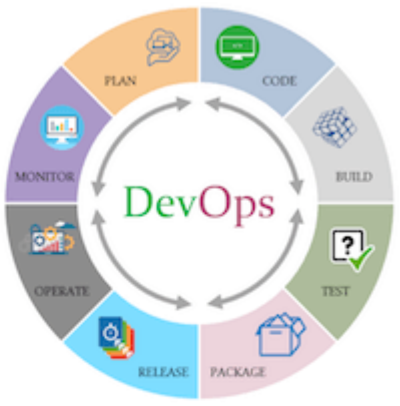
DevOps Architecture



- The DevOps is the combination of two words, one is **Development** and other is **Operations**.

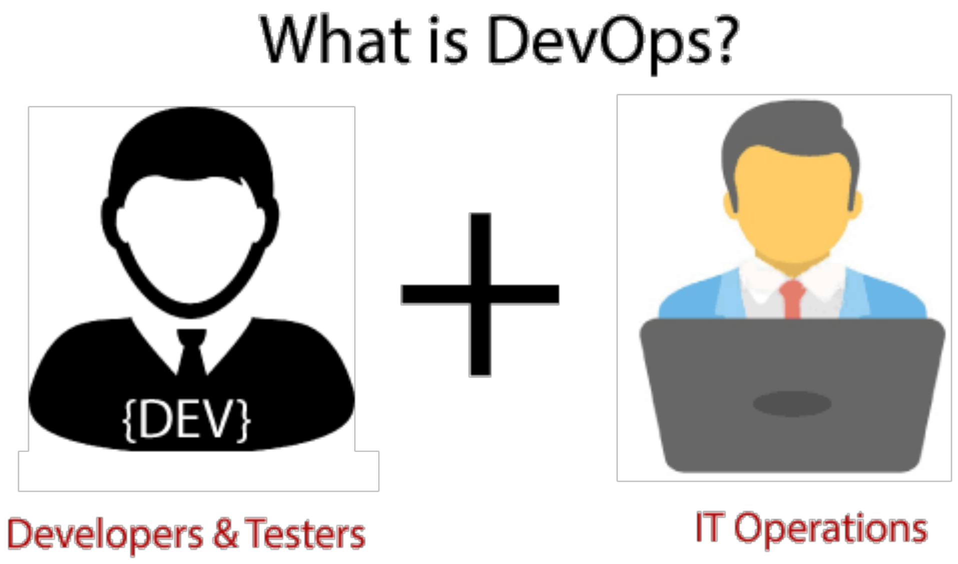
- It is a culture to promote the development and operation process collectively.

What is DevOps?

The DevOps is a combination of two words, one is software Development, and second is Operations.

This allows a single team to handle the entire application lifecycle, from development to **testing, deployment**, and **operations**.

DevOps helps you to reduce the disconnection between software developers, quality assurance (QA) engineers, and system administrators.



DevOps promotes collaboration between Development and Operations team to deploy code to production faster in an automated & repeatable way.

DevOps helps to increase organization speed to deliver applications and services.

It also allows organizations to serve their customers better and compete more strongly in the market.

DevOps can also be defined as a sequence of development and IT operations with better communication and collaboration.

DevOps has become one of the most valuable business disciplines for enterprises or organizations.

With the help of DevOps, **quality**, and **speed** of the application delivery has improved to a great extent.

DevOps is nothing but a practice or methodology of making "**Developers**" and "**Operations**" folks work together.

DevOps represents a change in the IT culture with a complete focus on rapid IT service delivery through the adoption of agile practices in the context of a system-oriented approach.

DevOps is all about the integration of the operations and development process.

Organizations that have adopted DevOps noticed a 22% improvement in software quality and a 17% improvement in application deployment frequency and achieve a 22% hike in customer satisfaction. 19% of revenue hikes as a result of the successful DevOps implementation.

Usage of DevOps?

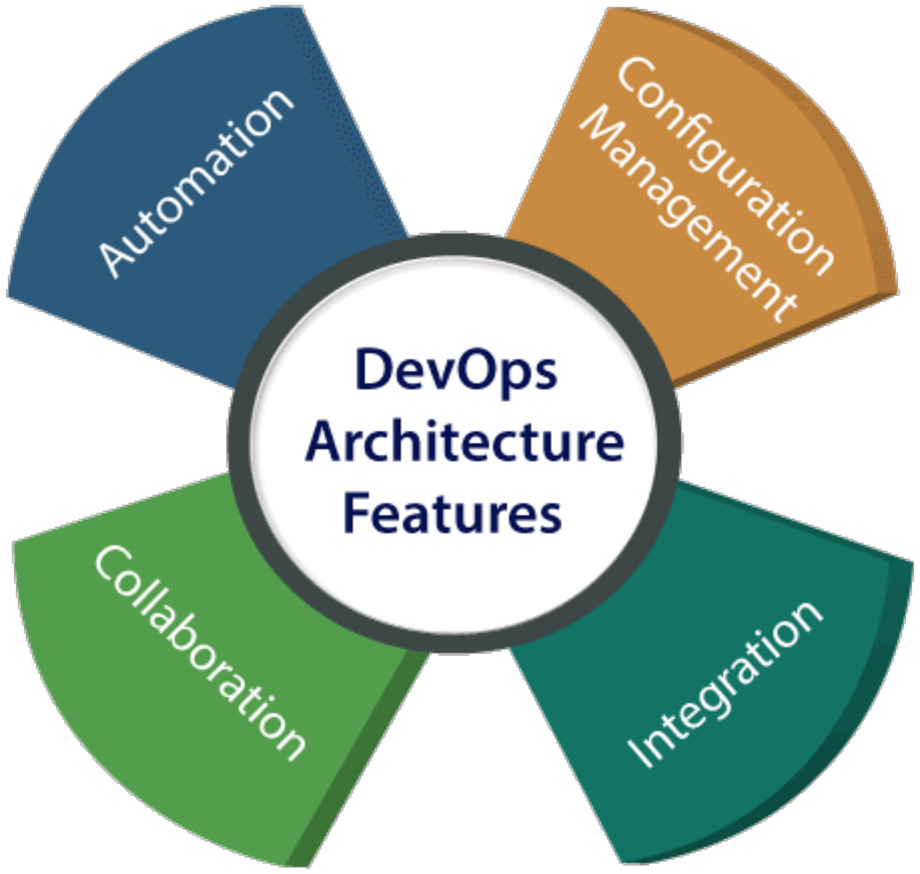
- The operation and development team worked in complete isolation.
- After the design-build, the testing and deployment are performed respectively. That's why they consumed more time than actual build cycles.
- Without the use of DevOps, the team members are spending a large amount of time on designing, testing, and deploying instead of building the project.
- Manual code deployment leads to human errors in production.
- Coding and operation teams have their separate timelines and are not in synch, causing further delays.

DevOps History

- In 2009, the first conference named **DevOpsdays** was held in Ghent Belgium. Belgian consultant and Patrick Debois founded the conference.
- In 2012, the state of DevOps report was launched and conceived by Alanna Brown at Puppet.
- In 2014, the annual State of DevOps report was published by Nicole Forsgren, Jez Humble, Gene Kim, and others. They found DevOps adoption was accelerating in 2014 also.
- In 2015, Nicole Forsgren, Gene Kim, and Jez Humble founded DORA (DevOps Research and Assignment).
- In 2017, Nicole Forsgren, Gene Kim, and Jez Humble published "Accelerate: Building and Scaling High Performing Technology Organizations".

DevOps Architecture Features

Here are some key features of DevOps architecture, such as:



1) Automation

Automation can reduce time consumption, especially during the testing and deployment phase.

The productivity increases, and releases are made quicker by automation.

This will lead in catching bugs quickly so that it can be fixed easily.

For contiguous delivery, each code is defined through automated tests, cloud-based services, and builds. This promotes production using automated deploys.

2) Collaboration

The Development and Operations team collaborates as a DevOps team, which improves the cultural model as the teams become more productive with their productivity, which strengthens accountability and ownership.

The teams share their responsibilities and work closely in sync, which in turn makes the deployment to production faster.

3) Integration

Applications need to be integrated with other components in the environment.

The integration phase is whore the existing code is combined with new functionality and then tested.

Continuous integration and testing enable continuous development.

The frequency in the releases and micro-services leads to significant operational challenges.

To overcome such problems, continuous integration and delivery are implemented to deliver in a **quicker, safer**, and **reliable manner**.

4) Configuration management

It ensures the application to interact with only those resources that are concerned with the environment in which it runs.

The configuration files are not created where the external configuration to the application is separated from the source code.

The configuration file can be written during deployment, or they can be loaded at the run time, depending on the environment in which it is running.

DevOps Advantages and Disadvantages

Here are some advantages and disadvantages that DevOps can have for business, such as:

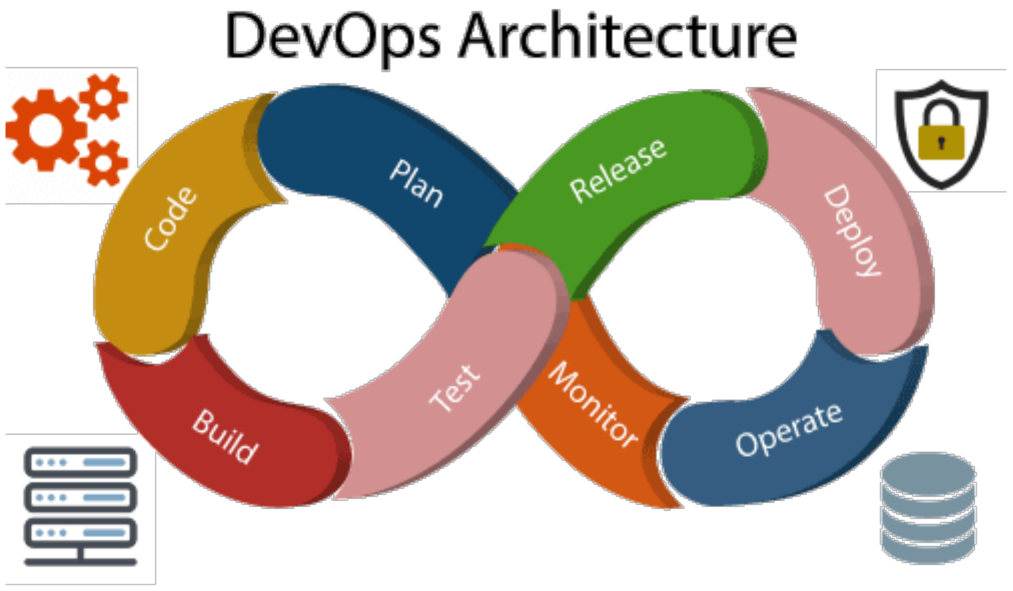
Advantages

- DevOps is an excellent approach for quick development and deployment of applications.
- It responds faster to the market changes to improve business growth.
- DevOps escalate business profit by decreasing software delivery time and transportation costs.
- DevOps clears the descriptive process, which gives clarity on product development and delivery.
- It improves customer experience and satisfaction.
- DevOps simplifies collaboration and places all tools in the cloud for customers to access.
- DevOps means collective responsibility, which leads to better team engagement and productivity.

Disadvantages

- DevOps professional or expert's developers are less available.
- Developing with DevOps is so expensive.
- Adopting new DevOps technology into the industries is hard to manage in short time.
- Lack of DevOps knowledge can be a problem in the continuous integration of automation projects.

DevOps Architecture



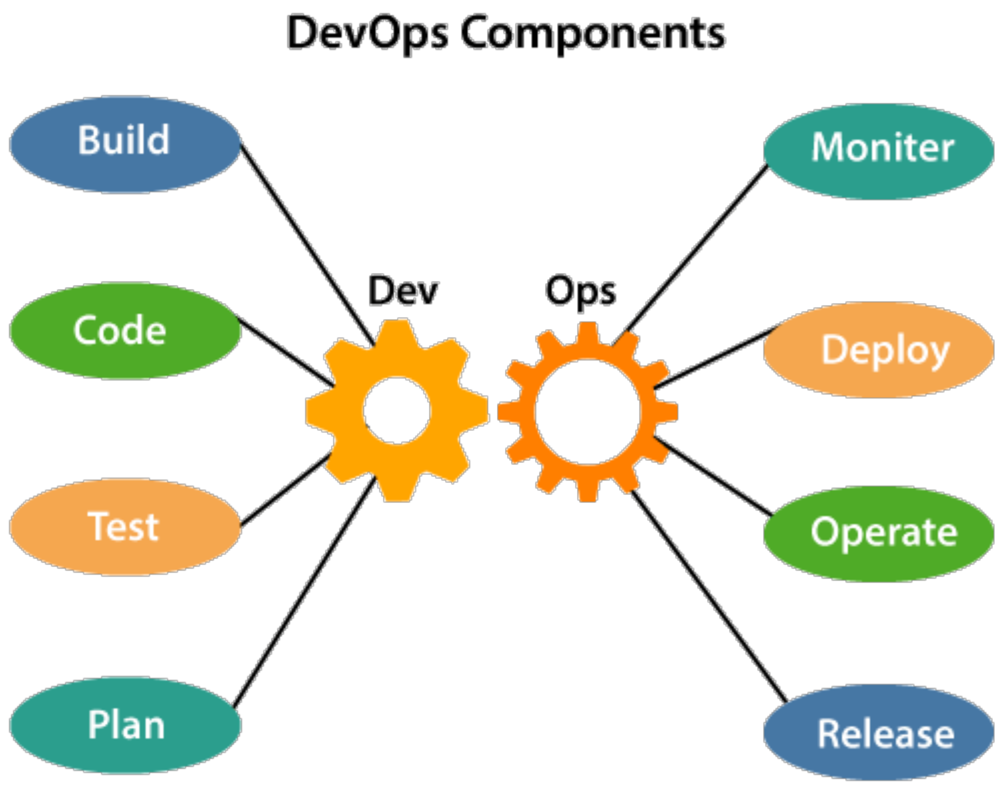
- Development and operations both play essential roles in order to deliver applications.

- The deployment comprises analyzing the **requirements, designing, developing**, and **testing** of the software components or frameworks.

- The operation consists of the administrative processes, services, and support for the software.

- When both the development and operations are combined with collaborating, then the DevOps architecture is the solution to fix the gap between deployment and operation terms; therefore, delivery can be faster.
- DevOps architecture is used for the applications hosted on the cloud platform and large distributed applications. Agile Development is used in the DevOps architecture so that integration and delivery can be contiguous. When the development and operations team works separately from each other, then it is time-consuming to **design, test**, and **deploy**. And if the terms are not in sync with each other, then it may cause a delay in the delivery. So DevOps enables the teams to change their shortcomings and increases productivity.

Below are the various components that are used in the DevOps architecture:



1) Build

- Without DevOps, the cost of the consumption of the resources was evaluated based on the pre-defined individual usage with fixed hardware allocation.
- And with DevOps, the usage of cloud, sharing of resources comes into the picture, and the build is dependent upon the user's need, which is a mechanism to control the usage of resources or capacity.

2) Code

Many good practices such as Git enables the code to be used, which ensures writing the code for business, helps to track changes, getting notified about the reason behind the difference in the actual and the expected output, and if necessary reverting to the original code developed. The code can be appropriately arranged in **files, folders**, etc. And they can be reused.

3) Test

- The application will be ready for production after testing. In the case of manual testing, it consumes more time in testing and moving the code to the output.

  The testing can be automated, which decreases the time for testing so that the time to deploy the code to production can be reduced as automating the running of the scripts will remove many manual steps.

4) Plan

- DevOps use Agile methodology to plan the development. With the operations and development team in sync, it helps in organizing the work to plan accordingly to increase productivity.

5) Monitor

- Continuous monitoring is used to identify any risk of failure. Also, it helps in tracking the system accurately so that the health of the application can be checked. The monitoring becomes more comfortable with services where the log data may get monitored through many third-party tools such as **Splunk**.

6) Deploy

- Many systems can support the scheduler for automated deployment. The cloud management platform enables users to capture accurate insights and view the optimization scenario, analytics on trends by the deployment of dashboards.

7) Operate

- DevOps changes the way traditional approach of developing and testing separately. The teams operate in a collaborative way where both the teams actively participate throughout the service lifecycle. The operation team interacts with developers, and they come up with a monitoring plan which serves the IT and business requirements.

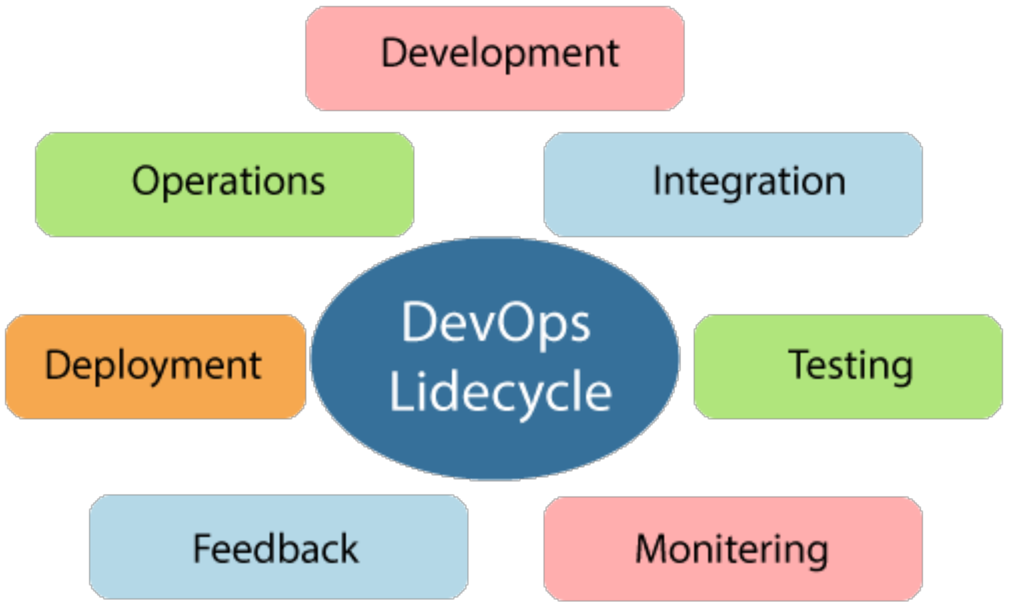
8) Release

- Deployment to an environment can be done by automation. But when the deployment is made to the production environment, it is done by manual triggering. Many processes involved in release management commonly used to do the deployment in the production environment manually to lessen the impact on the customers.

DevOps Lifecycle

DevOps defines an agile relationship between operations and Development.

It is a process that is practiced by the development team and operational engineers together from beginning to the final stage of the product.



DevOps lifecycle includes seven phases as given below:

1) Continuous Development

This phase involves the planning and coding of the software.

The vision of the project is decided during the planning phase.

And the developers begin developing the code for the application.

There are no DevOps tools that are required for planning, but there are several tools for maintaining the code.

2) Continuous Integration

This stage is the heart of the entire DevOps lifecycle.

It is a software development practice in which the developers require to commit changes to the source code more frequently.

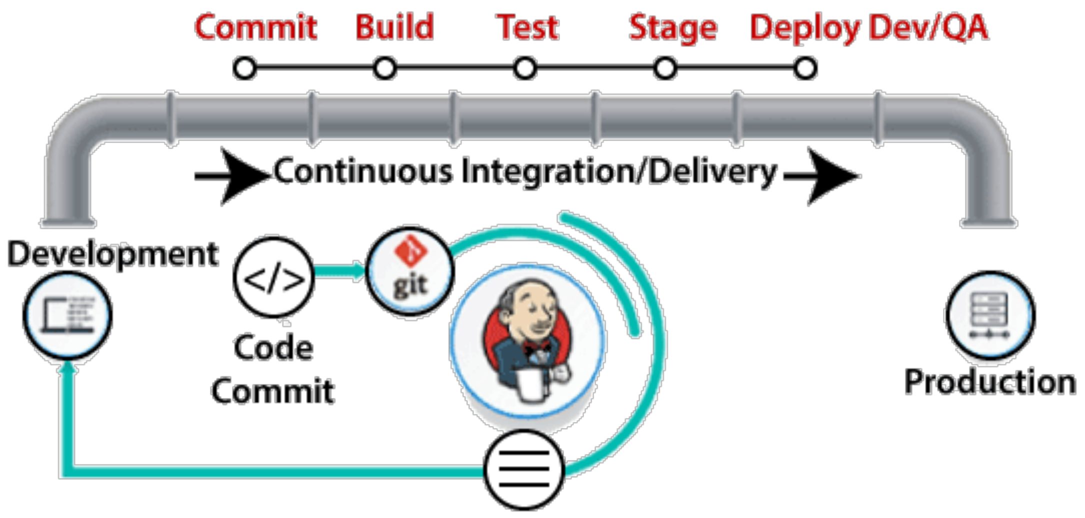
This may be on a daily or weekly basis.

Then every commit is built, and this allows early detection of problems if they are present.

Building code is not only involved compilation, but it also includes **unit testing, integration testing, code review**, and **packaging**.

The code supporting new functionality is continuously integrated with the existing code.

The updated code needs to be integrated continuously and smoothly with the systems to reflect changes to the end-users.



Jenkins is a popular tool used in this phase.

Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a build of that code, which is an executable file in the form of war or jar.

Then this build is forwarded to the test server or the production server.

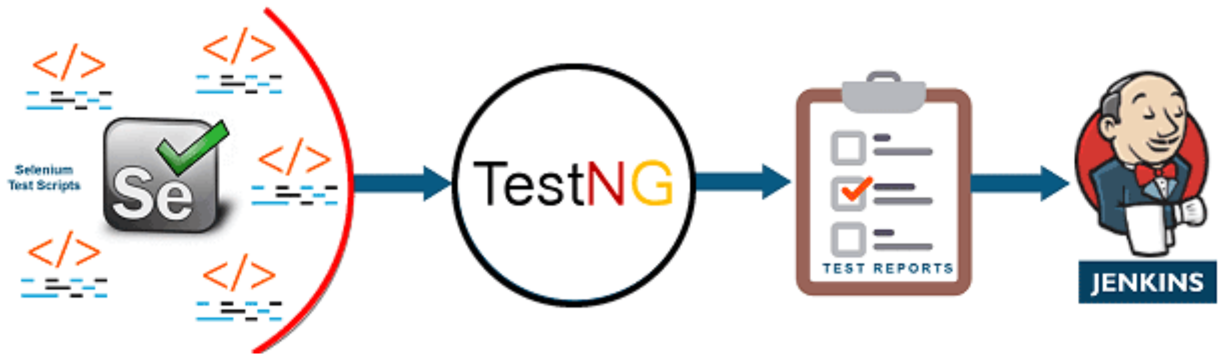
3) Continuous Testing

This phase, where the developed software is continuously testing for bugs.

For constant testing, automation testing tools such as **TestNG, JUnit, Selenium**, etc are used.

These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality.

In this phase, **Docker** Containers can be used for simulating the test environment.



Selenium does the automation testing, and TestNG generates the reports.

This entire testing phase can automate with the help of a Continuous Integration tool called **Jenkins**.

Automation testing saves a lot of time and effort for executing the tests instead of doing this manually.

Apart from that, report generation is a big plus.

The task of evaluating the test cases that failed in a test suite gets simpler.

we can schedule the execution of the test cases at predefined times.

After testing, the code is continuously integrated with the existing code.

4) Continuous Monitoring

Monitoring is a phase that involves all the operational factors of the entire DevOps process, where important information about the use of the software is recorded and carefully processed to find out trends and identify problem areas.

Usually, the monitoring is integrated within the operational capabilities of the software application.

It may occur in the form of documentation files or maybe produce large-scale data about the application parameters when it is in a continuous use position.

The system errors such as server not reachable, low memory, etc are resolved in this phase.

It maintains the security and availability of the service.

5) Continuous Feedback

The application development is consistently improved by analyzing the results from the operations of the software.

This is carried out by placing the critical phase of constant feedback between the operations and the development of the next version of the current software application.

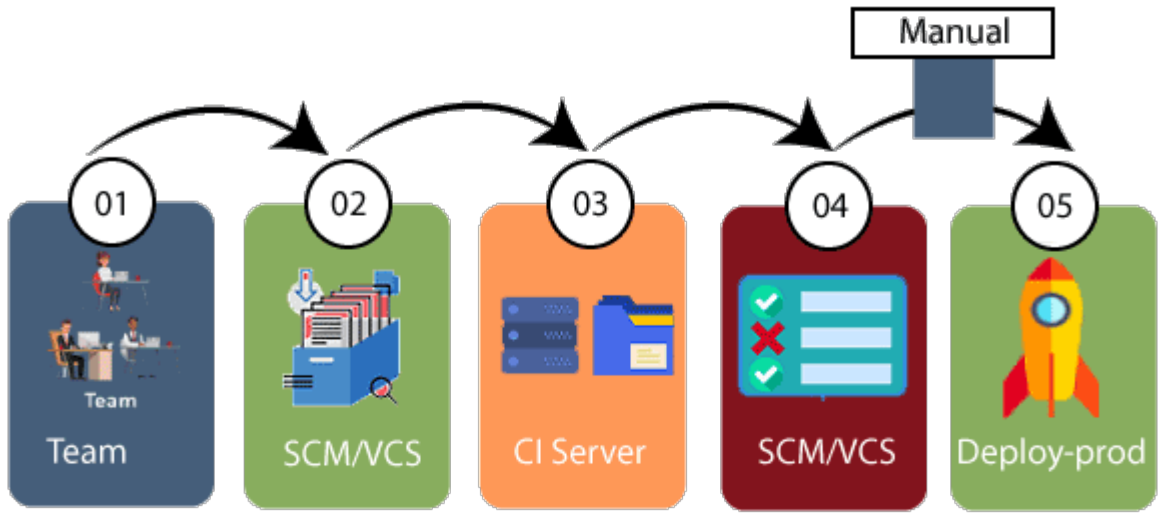
The continuity is the essential factor in the DevOps as it removes the unnecessary steps which are required to take a software application from development, using it to find out its issues and then producing a better version.

It kills the efficiency that may be possible with the app and reduce the number of interested customers.

6) Continuous Deployment

In this phase, the code is deployed to the production servers.

Also, it is essential to ensure that the code is correctly used on all the servers.



The new code is deployed continuously, and configuration management tools play an essential role in executing tasks frequently and quickly.

Here are some popular tools which are used in this phase, such as **Chef, Puppet, Ansible**, and **SaltStack**.

Containerization tools are also playing an essential role in the deployment phase.

Vagrant and **Docker** are popular tools that are used for this purpose.

These tools help to produce consistency across development, staging, testing, and production environment.

They also help in scaling up and scaling down instances softly.

Containerization tools help to maintain consistency across the environments where the application is tested, developed, and deployed.

There is no chance of errors or failure in the production environment as they package and replicate the same dependencies and packages used in the testing, development, and staging environment.

It makes the application easy to run on different computers.

7) Continuous Operations

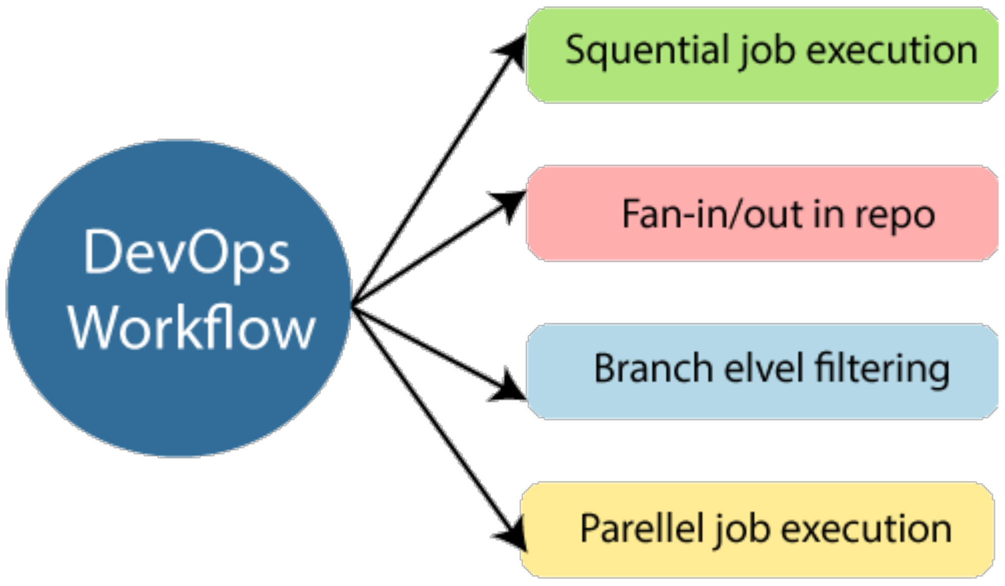
All DevOps operations are based on the continuity with complete automation of the release process and allow the organization to accelerate the overall time to market continuingly.

It is clear from the discussion that continuity is the critical factor in the DevOps in removing steps that often distract the development, take it longer to detect issues and produce a better version of the product after several months.

With DevOps, we can make any software product more efficient and increase the overall count of interested customers in your product.

DevOps Workflow

DevOps workflow provides a visual overview of the sequence in which input is provided. Also, it tells about which one action is performed, and output is generated for an operations process.



DevOps workflow allows the ability to separate and arrange the jobs which are top requested by the users. Also, it gives the ability to mirror their ideal process in the configuration jobs.

DevOps Principles

The main principles of DevOps are Continuous delivery, automation, and fast reaction to the feedback.

End to End Responsibility: DevOps team need to provide performance support until they become the end of life. It enhances the responsibility and the quality of the products engineered.

Continuous Improvement: DevOps culture focuses on continuous improvement to minimize waste. It continuously speeds up the growth of products or services offered.

1. **Automate Everything:** Automation is an essential principle of the DevOps process. This is for software development and also for the entire infrastructure landscape.
2. **Custom Centric Action:** DevOps team must take customer-centric for that they should continuously invest in products and services.
3. **Monitor and test everything:** The DevOps team needs to have robust monitoring and testing procedures.
4. **Work as one team:** In the DevOps culture role of the designers, developers, and testers are already defined. All they needed to do is work as one team with complete collaboration.

These principles are achieved through several DevOps practices, which include frequent deployments, QA automation, continuous delivery, validating ideas as early as possible, and in-team collaboration.

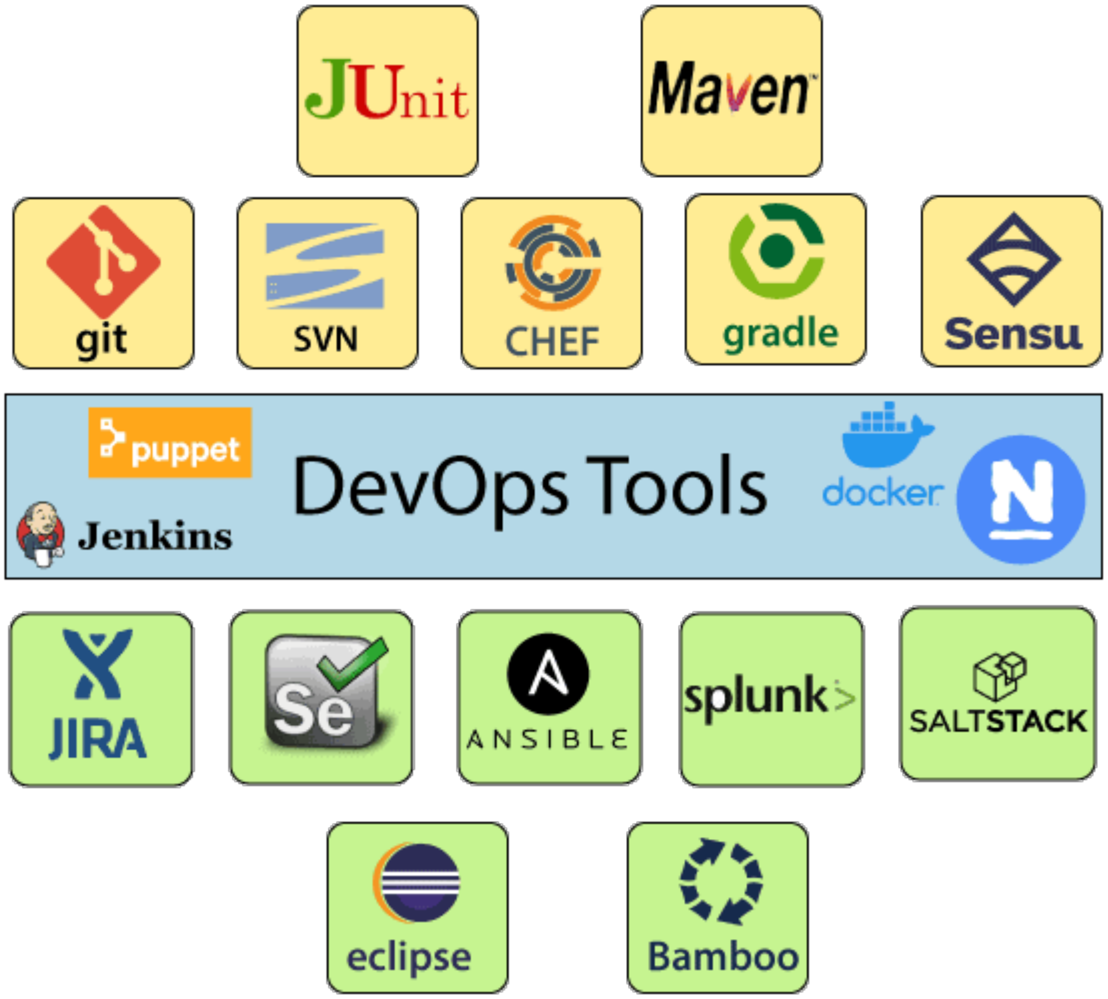
DevOps Practices

Some identified DevOps practices are:

- Self-service configuration
- Continuous build
- Continuous integration
- Continuous delivery
- Incremental testing
- Automated provisioning
- Automated release management

DevOps Tools

Here are some most popular DevOps tools with brief explanation shown in the below image, such as:



1) Puppet

Puppet is the most widely used DevOps tool. It allows the delivery and release of the technology changes quickly and frequently. It has features of versioning, automated testing, and continuous delivery. It enables to manage entire infrastructure as code without expanding the size of the team.

Features

- Real-time context-aware reporting.
- Model and manage the entire environment.
- Defined and continually enforce infrastructure.
- Desired state conflict detection and remediation.
- It inspects and reports on packages running across the infrastructure.
- It eliminates manual work for the software delivery process.
- It helps the developer to deliver great software quickly.

2) Ansible

Ansible is a leading DevOps tool. Ansible is an open-source IT engine that automates application deployment, cloud provisioning, intra service orchestration, and other IT tools. It makes it easier for DevOps teams to scale automation and speed up productivity.

Ansible is easy to deploy because it does not use any **agents** or **custom security** infrastructure on the client-side, and by pushing modules to the clients. These modules are executed locally on the client-side, and the output is pushed back to the Ansible server.

Features

- It is easy to use to open source deploy applications.
- It helps in avoiding complexity in the software development process.
- It eliminates repetitive tasks.
- It manages complex deployments and speeds up the development process.

3) Docker

Docker is a high-end DevOps tool that allows building, ship, and run distributed applications on multiple systems. It also helps to assemble the apps quickly from the components, and it is typically suitable for container management.

Features

- It configures the system more comfortable and faster.
- It increases productivity.
- It provides containers that are used to run the application in an isolated environment.
- It routes the incoming request for published ports on available nodes to an active container. This feature enables the connection even if there is no task running on the node.
- It allows saving secrets into the swarm itself.

4) Nagios

Nagios is one of the more useful tools for DevOps. It can determine the errors and rectify them with the help of network, infrastructure, server, and log monitoring systems.

Features

- It provides complete monitoring of desktop and server operating systems.
- The network analyzer helps to identify bottlenecks and optimize bandwidth utilization.
- It helps to monitor components such as services, application, OS, and network protocol.
- It also provides to complete monitoring of Java Management Extensions.

5) CHEF

A chef is a useful tool for achieving scale, speed, and consistency. The chef is a cloud-based system and open source technology. This technology uses Ruby encoding to develop essential building blocks such as recipes and cookbooks. The chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.

Chef has got its convention for different building blocks, which are required to manage and automate infrastructure.

Features

- It maintains high availability.
- It can manage multiple cloud environments.
- It uses popular Ruby language to create a domain-specific language.
- The chef does not make any assumptions about the current status of the node. It uses its mechanism to get the current state of the machine.

6) Jenkins

Jenkins is a DevOps tool for monitoring the execution of repeated tasks. Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. It helps to integrate project changes more efficiently by finding the issues quickly.

Features

- Jenkins increases the scale of automation.
- It can easily set up and configure via a web interface.
- It can distribute the tasks across multiple machines, thereby increasing concurrency.
- It supports continuous integration and continuous delivery.
- It offers 400 plugins to support the building and testing any project virtually.
- It requires little maintenance and has a built-in GUI tool for easy updates.

7) Git

Git is an open-source distributed version control system that is freely available for everyone. It is designed to handle minor to major projects with speed and efficiency. It is developed to co-ordinate the work among programmers. The version control allows you to track and work together with your team members at the same workspace. It is used as a critical distributed version-control for the DevOps tool.

Features

- It is a free open source tool.
- It allows distributed development.
- It supports the pull request.
- It enables a faster release cycle.
- Git is very scalable.
- It is very secure and completes the tasks very fast.

8) SALTSTACK

Stackify is a lightweight DevOps tool. It shows real-time error queries, logs, and more directly into the workstation. SALTSTACK is an ideal solution for intelligent orchestration for the software-defined data center.

Features

- It eliminates messy configuration or data changes.
- It can trace detail of all the types of the web request.
- It allows us to find and fix the bugs before production.
- It provides secure access and configures image caches.
- It secures multi-tenancy with granular role-based access control.
- Flexible image management with a private registry to store and manage images.

9) Splunk

Splunk is a tool to make machine data usable, accessible, and valuable to everyone. It delivers operational intelligence to DevOps teams. It helps companies to be more secure, productive, and competitive.

Features

- It has the next-generation monitoring and analytics solution.
- It delivers a single, unified view of different IT services.
- Extend the Splunk platform with purpose-built solutions for security.
- Data drive analytics with actionable insight.

10) Selenium

Selenium is a portable software testing framework for web applications. It provides an easy interface for developing automated tests.

Features

- It is a free open source tool.
- It supports multiplatform for testing, such as Android and ios.
- It is easy to build a keyword-driven framework for a WebDriver.
- It creates robust browser-based regression automation suites and tests.

DevOps Automation

Automation is the crucial need for DevOps practices, and automate everything is the fundamental principle of DevOps.

Automation kick starts from the code generation on the developers machine, until the code is pushed to the code and after that to monitor the application and system in the production.

Automating infrastructure set up and configurations, and software deployment is the key highlight of DevOps practice.

DevOps practice id is dependent on automation to make deliveries over a few hours and make frequent deliveries across platforms.

Automation in DevOps boosts speed, consistency, higher accuracy, reliability, and increases the number of deliveries.

Automation in DevOps encapsulates everything right from the building, deploying, and monitoring.

DevOps Automation Tools

In large DevOps team that maintain extensive massive IT infrastructure can be classified into six categories, such as:

- Infrastructure Automation
- Configuration Management
- Deployment Automation
- Performance Management
- Log management
- Monitoring

Below are few tools in each of these categories let see in brief, such as:

Infrastructure Automation

Amazon Web Services (AWS): Being a cloud service, you don't need to be physically present in the data center, they are easy to scale on-demand, and there are no up-front hardware costs.

It can be configured to provide more servers based on traffic automatically.

Configuration Management

Chef: Chef is a handy DevOps tool for achieving speed, scale, and consistency.

It can be used to ease out of complex tasks and perform configuration management.

With the help of this tool, the DevOps team can avoid making changes across ten thousand servers.

Rather, they need to make changes in one place, which is automatically reflected in other servers.

Deployment Automation

Jenkins: It facilitates continuous integration and testing.

It helps to integrate project changes more efficiently by quickly finding issues as soon as built is deployed.

Performance Management

App Dynamic: It offers real-time performance monitoring. The data collected by this tool help developers to debug when issues occur.

Log Management

Splunk: This DevOps tool solves issues such as storing, aggregating, and analyzing all logs in one place.

Monitoring

Nagios: It notified people when infrastructure and related service go down. Nagios is a tool for this purpose, which helps the DevOps team to find and correct problems.

DevOps Pipeline

A pipeline in software engineering team is a set of automated processes which allows DevOps professionals and developer to reliably and efficiently compile, build, and deploy their code to their production compute platforms.

The most common components of a pipeline in DevOps are build automation or continuous integration, test automation, and deployment automation.

A pipeline consists of a set of tools which are classified into the following categories such as:

- Source control
- Build tools
- Containerization
- Configuration management
- Monitoring

Continuous Integration Pipeline

Continuous integration (CI) is a practice in which developers can check their code into a version-controlled repository several times per day.

Automated build pipelines are triggered by these checks which allows fast and easy to locate error detection.

Some significant benefits of CI are:

- Small changes are easy to integrate into large codebases.
- More comfortable for other team members to see what you have been working.
- Fewer integration issues allowing rapid code delivery.
- Bugs are identified early, making them easier to fix, resulting in less debugging work.

Continuous Delivery Pipeline

Continuous delivery (CD) is the process that allows operation engineers and developers to deliver bug fixes, features, and configuration change into production reliably, quickly, and sustainably.

Continuous delivery offers the benefits of code delivery pipelines, which are carried out that can be performed on demand.

Some significant benefits of the CD are:

- Faster bug fixes and features delivery.
- CD allows the team to work on features and bug fixes in small batches, which means user feedback received much quicker. It reduces the overall time and cost of the project.

DevOps Methodology

We have a demonstrated methodology that takes an approach to cloud adoption.

It accounts for all the factors required for successful approval such as people, process, and technology, resulting in a focus on the following critical consideration:

- **The Teams:** Mission or project and cloud management.
- **Connectivity:** Public, on-premise, and hybrid cloud network access.
- **Automation:** Infrastructure as code, scripting the orchestration and deployment of resources.
- **On-boarding Process:** How the project gets started in the cloud.
- **Project Environment:** TEST, DEV, PROD (identical deployment, testing, and production).
- **Shared Services:** Common capabilities provided by the enterprise.
- **Naming Conventions:** Vital aspect to track resource utilization and billing.
- **Defining Standards Role across the Teams:** Permissions to access resources by job function.