DevOps

NAME- ASHWINI R. JAWALE

ROLL NO- 226237

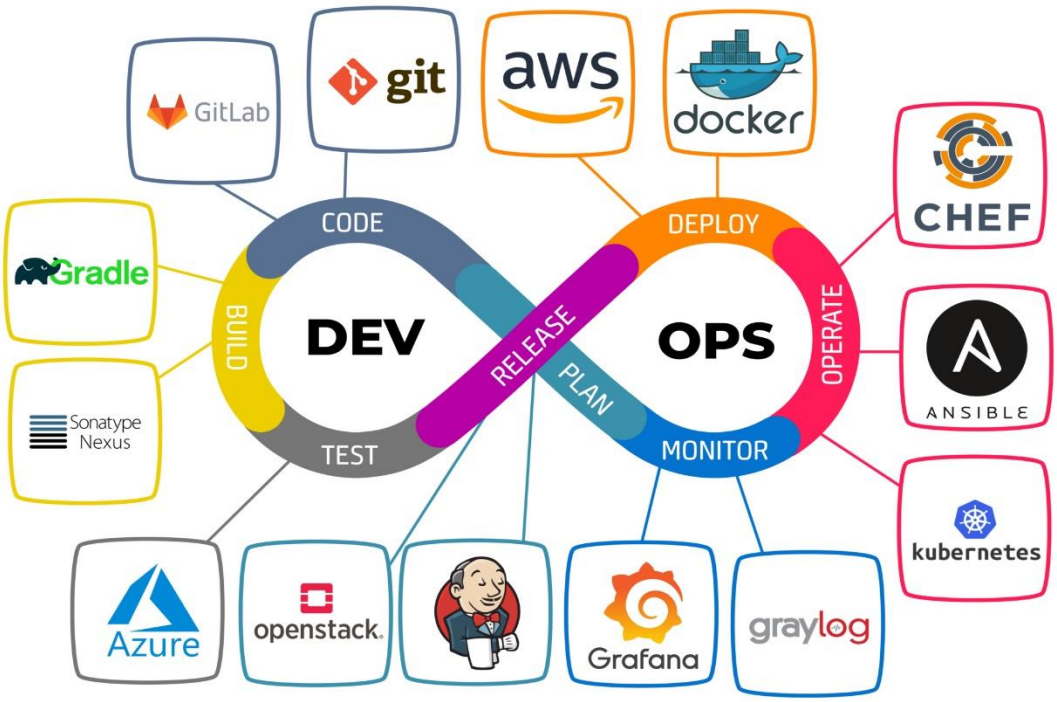
MSC-CS (SEM-III)

What is DevOps?

The DevOps is the combination of two words, one is Development and other is Operations**.** Used as a set of practices and tools, DevOps integrates and automates the work of software development (*Dev*) and IT operations (*Ops*) as a means for improving and shortening the systems development life cycle.

By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster.

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. Creating backlogs, tracking bugs, managing agile software development with Scrum, using Kanban boards, and visualizing progress with dashboards are some of the ways DevOps teams plan with agility and visibility.

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. In this phase, teams define a release management process with clear manual approval stages. They also set automated gates that move applications between stages until they're made available to customers. Automating these processes makes them scalable, repeatable, controlled. This way, teams who practice DevOps can deliver frequently with ease, confidence, and peace of mind.

7. OPERATE

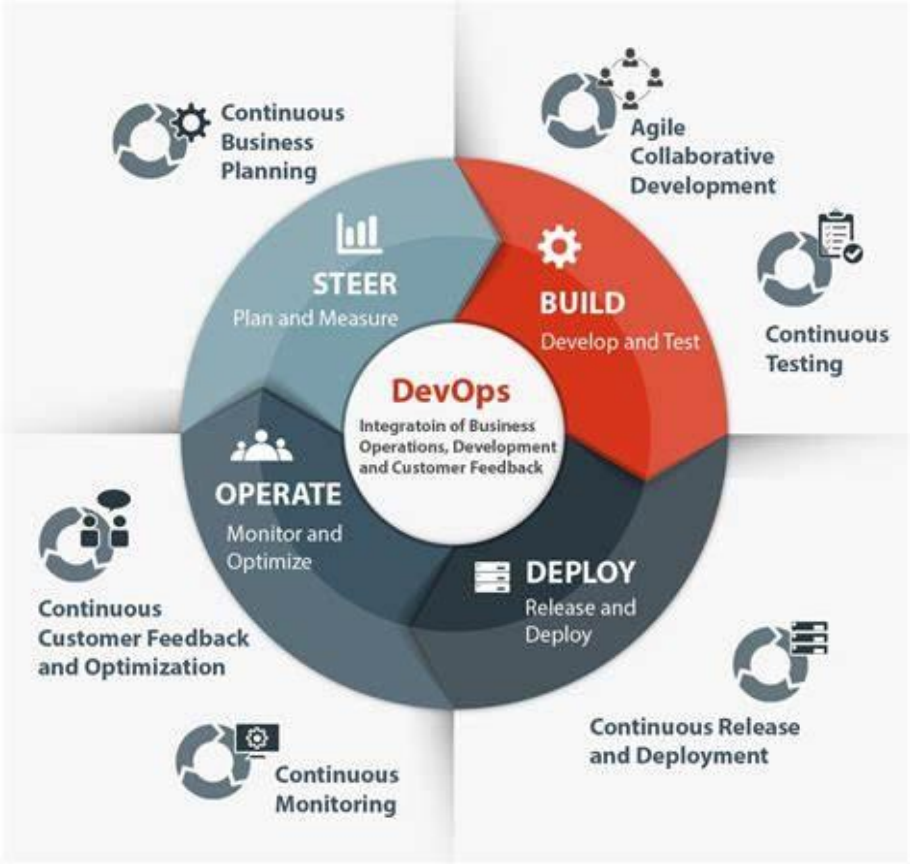
The operate phase involves maintaining, monitoring, and troubleshooting applications in production environments. In adopting DevOps practices, teams work to ensure system reliability, high availability, and aim for zero downtime while reinforcing security and governance. DevOps teams seek to identify issues before they affect the customer experience and mitigate issues quickly when they do occur. Maintaining this vigilance requires rich telemetry, actionable alerting, and full visibility into applications and the underlying system.

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 Model:

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes.



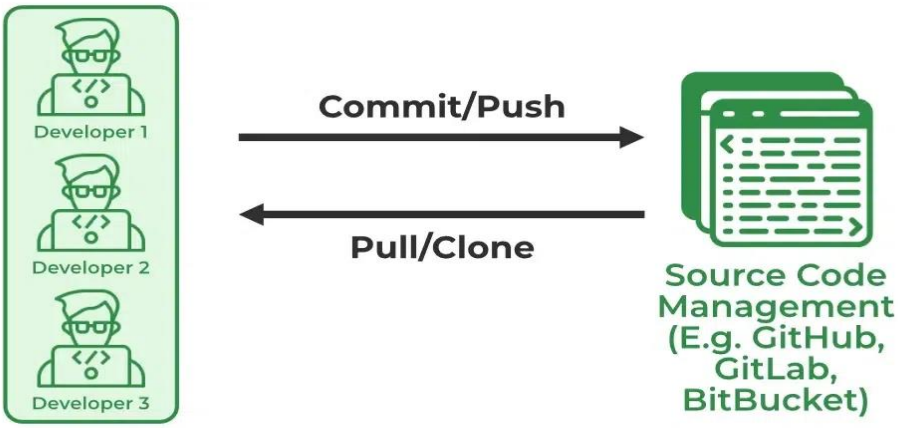
How DevOps works?

DevOps is a software development methodology that improves the collaboration between developers and operations teams using various automation tools. These automation tools are implemented using various stages which are a part of the DevOps Lifecycle.

The **Goal** of DevOps is to increase an organization speed when it comes to delivering applications and services. Many companies have successfully implemented DevOps to enhance their user experience like Amazon, Netflix etc.

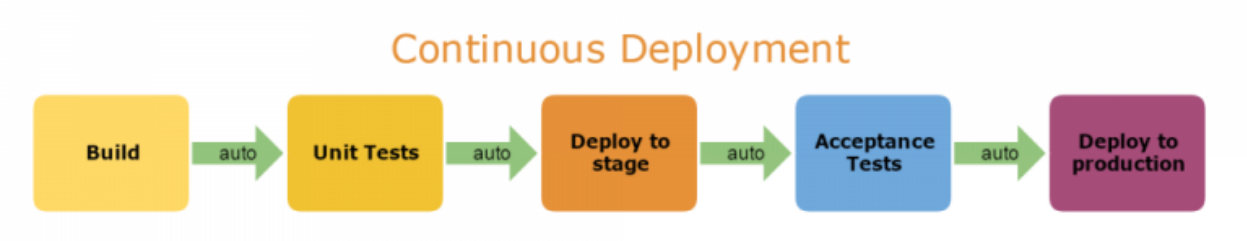
1. **Continuous Development**

This stage involves committing code to version control tools such as Git or SVN for maintaining the different versions of the code, and tools like Ant, Maven, Gradle for building/packaging the code into an executable file that can be forwarded to the QAs for testing.



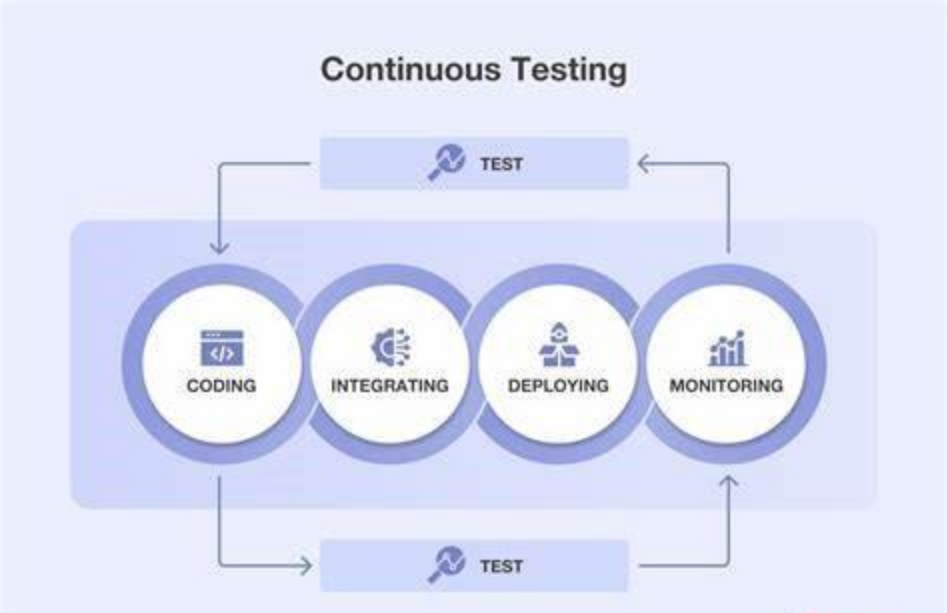
2. **Continuous Deployment**

In this stage the code is built, the environment or the application is containerized and is pushed onto the desired server. The key processes in this stage are Configuration Management, Virtualization etc.



3. **Continuous Testing**

The stage deals with automated testing of the application pushed by the developer. If there is an error, the message is sent back to the integration tool, this tool, in turn, notifies the developer of the error, If the test was a success, the message is sent to Integration-tool which pushes the build on the production server.



4. **Continuous Monitoring**

The stage continuously monitors the deployed application for bugs or crashes. It can also be set up to collect user feedback. The collected data is then sent to the developers to improve the application.

5. **Continuous Customer Feedback**

Once the application is released into the market the end users will use the application and they will give us feedback about the performance of the application and any glitches affecting the user experience after getting multiple feedback from the end users' the DevOps team will analyze the feedbacks given by end users and they will reach out to the developer team tries to rectify the mistakes they are performed in that piece of code by this we can reduce the errors or bugs.

DevOps Best Practices:

Beyond establishing a DevOps culture, teams bring DevOps to life by implementing certain practices throughout the application lifecycle. Some of these practices help accelerate, automate, and improve a specific phase.

- **Have a Centralized Unit for DevOps**

  In large organizations, a centralized unit is often used for DevOps. This centralized unit is responsible for creating DevOps tools like Jenkins, Docker, Ansible, Puppet, etc. This unit has ownership of all the DevOps tools that are developed and uses agile for each of the teams involved.

  Among all the different tools available, the centralized DevOps team is responsible for choosing the ones that are most relevant, beneficial, and important for the organization.

- **Continuous Integration and Continuous Delivery (CI/CD)**

  The most important process of DevOps is Continuous Integration (CI) and Continuous Delivery (CD). In short, CI / CD processes enable software companies to develop and deliver software in very short cycles.

  Using configuration management tools, teams can roll out changes in a controlled, systematic way, reducing the risks of modifying system configuration. Teams use configuration management tools to track system state and help avoid configuration drift, which is how a system resource's configuration deviates over time from the desired state defined for it.

- **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. This practice is usually implemented using version control systems such as Git which allow multiple developers to collaborate in authoring code. These systems provide a clear process to merge code changes that happen in the same files, handle conflicts, and roll back changes to earlier states. We can divide coding tasks between team members, and store all code for easy recovery if needed.

- **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**

  Practicing infrastructure as code helps teams deploy system resources in a reliable, repeatable, and controlled way. Infrastructure as code also helps automate deployment and reduces the risk of human error, especially for complex large environments. This repeatable, reliable solution for environment deployment lets teams maintain development and testing environments that are identical to production. Duplicating environments to different data centers and cloud platforms likewise becomes simpler and more efficient.

- **Use the Right Tools**

  For a DevOps approach to be successful, the processes have to be automated. There are so many DevOps tools available for different purposes, such as measuring different metrics, detecting security issues, etc. In order to save time and shorten the life cycle of software development, it is mandatory to use the right kind of tool.

- **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.

- **Monitor the Right Metrics**

  In order to have DevOps be integrated into the organization effectively, it is important to monitor the right kind of metrics. These metrics are the ones that give information on the effectiveness of DevOps. The metric can range in different categories such as lead time, the severity of the issue, the average time it takes to detect an issue, and so on.

- **Decide which Processes and Tests to automate first**

  Some important steps in software development that you can automate first include compiling codes, User Interface (UI) testing, functional testing, etc., based on the goal that you are trying to achieve. Automate processes that are most important and most frequently used before moving on to processes that are less important or less frequently used.

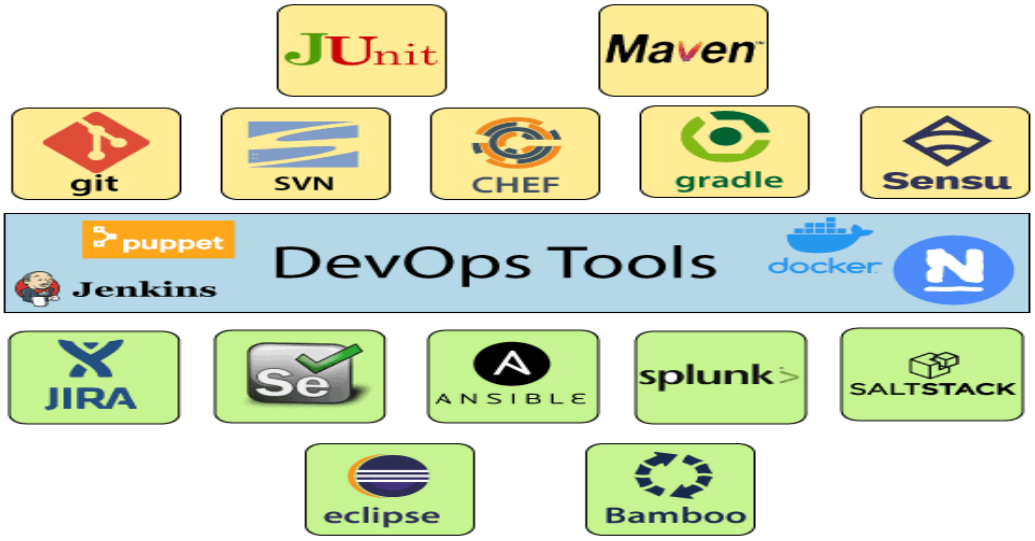
- **Switch to Microservices**

  Any work that is split into its smaller counterparts is much easier to finish than a single large chunk of work. The traditional method of working involves combining all the smaller counterparts into a single program which is harder to work with. The microservice architecture involves deploying all the smaller applications and services independently.

- **Communication and Collaboration**

  The use of DevOps tooling and automation of the software delivery process establishes collaboration by physically bringing together the workflows and responsibilities of development and operations. This helps speed up communication across developers, operations, and even other teams like marketing or sales, allowing all parts of the organization to align more closely on goals and projects.

DevOps Tools:



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.

2. **Ansible**

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. It makes it easier for DevOps teams to scale automation and speed up productivity.

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.

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.

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.

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.

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.

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.

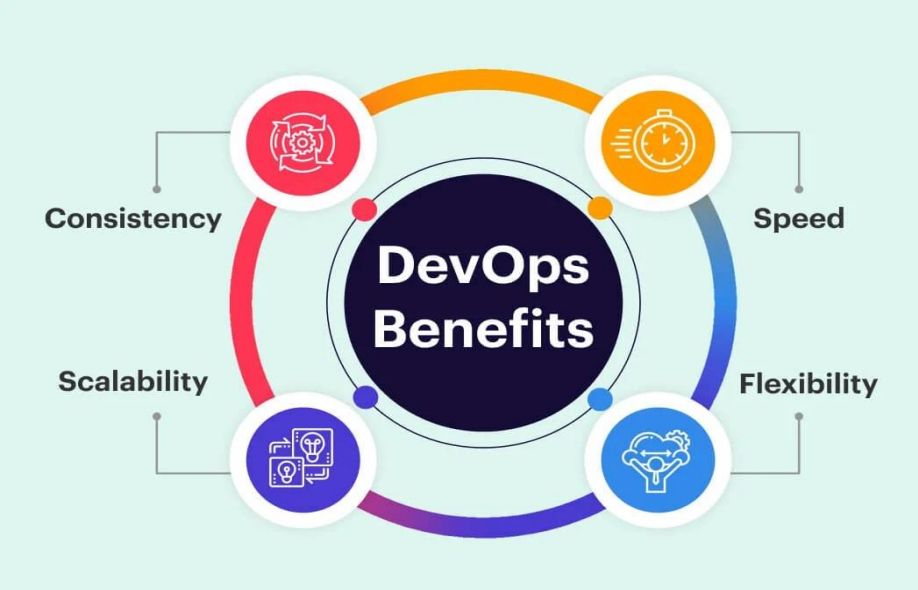
9. Splunk

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.

10. Selenium

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

Benefits of DevOps:



- Maintaining systems stability.

- Speed- Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. For example, microservices and continuous delivery let teams take ownership of services and then release updates to them quicker.

- Improving the mean time to recovery.

- Reliability- Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive

experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe.

- Rapid Delivery- Increase the frequency and pace of releases so you can innovate and improve your product faster. The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs.

- Adapting to the market and competition.

- Scale- Operate and manage your infrastructure and development processes at scale. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

- Collaboration- Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time.

- Accelerating time to market.

- Security- You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code.

Thank you