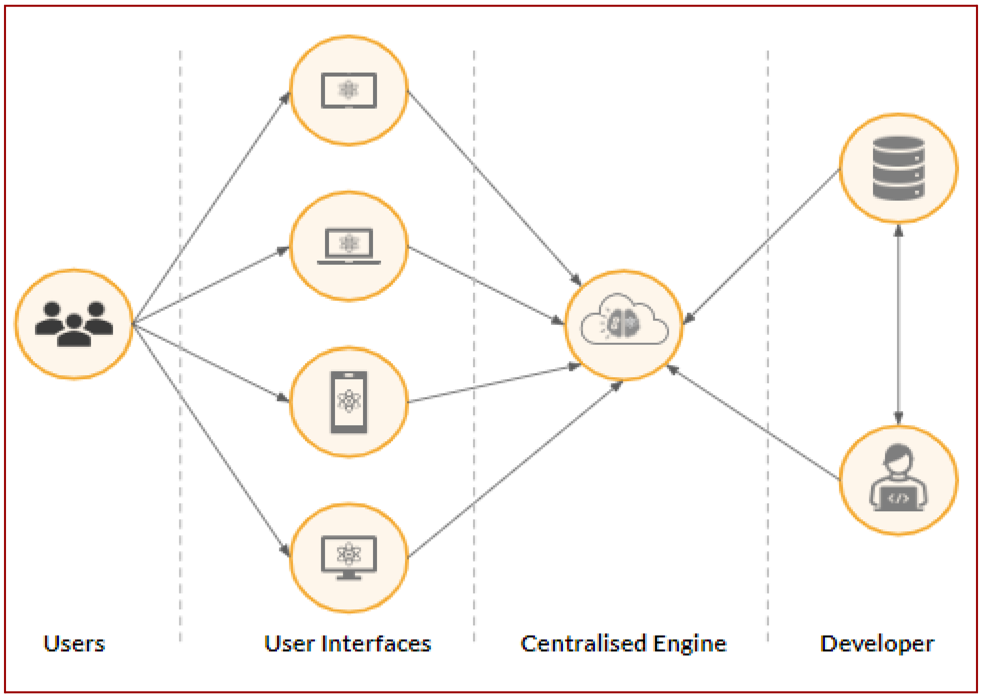
Session - 1: Introduction to DevOps

1.1 Challenges in Adding Features

Today, companies are becoming more agile and adaptive to incorporating customer needs as it helps in increasing revenue. Let's understand how the development of any product or new feature happens. A development team creates the code, which is tested by the testing team, deployed and maintained by the operations team. As you can see, there are multiple steps and different teams involved. With such complexity, various things can go wrong. For example, often, there are situations in apps or websites when something is not loading or there is some error or downtime. But, these issues are rarely witnessed with Google or Amazon products. What do these companies do differently? Well, among many other things, they follow the DevOps principle to its core.

Now, adding a new feature(code) to an existing centralised server is not an e. The below diagram shows what the backend architecture of any application looks like.



Once the code runs locally without any errors, it is integrated into the centralised engine. Now, an application can be accessed from different devices such as laptops, smartphones and tablets. Therefore, it needs to be compatible with all such devices.

There are also some bottlenecks that can arise when adding new features.

- High probability of errors
- Difficult and expensive changes
- Rollback to previous versions

1.2 Understanding DevOps

- DevOps can be defined as a culture or a philosophy that aims to enhance the collaboration between the development team (Dev) and the operations team (Ops), resulting in the creation of better-quality software.

- The Dev and Ops teams worked in silos earlier. Now in DevOps, they work together as a single engineering team.

- This increased collaboration and single unified vision across different teams play a vital role in the growth of many organisations.

- For example, Netflix had only 8.4 million subscribers in 2008, but when the company moved to the cloud and scaled its systems following DevOps practices, it saw tremendous growth. By 2015, the number of subscribers increased eight times compared with 2008.

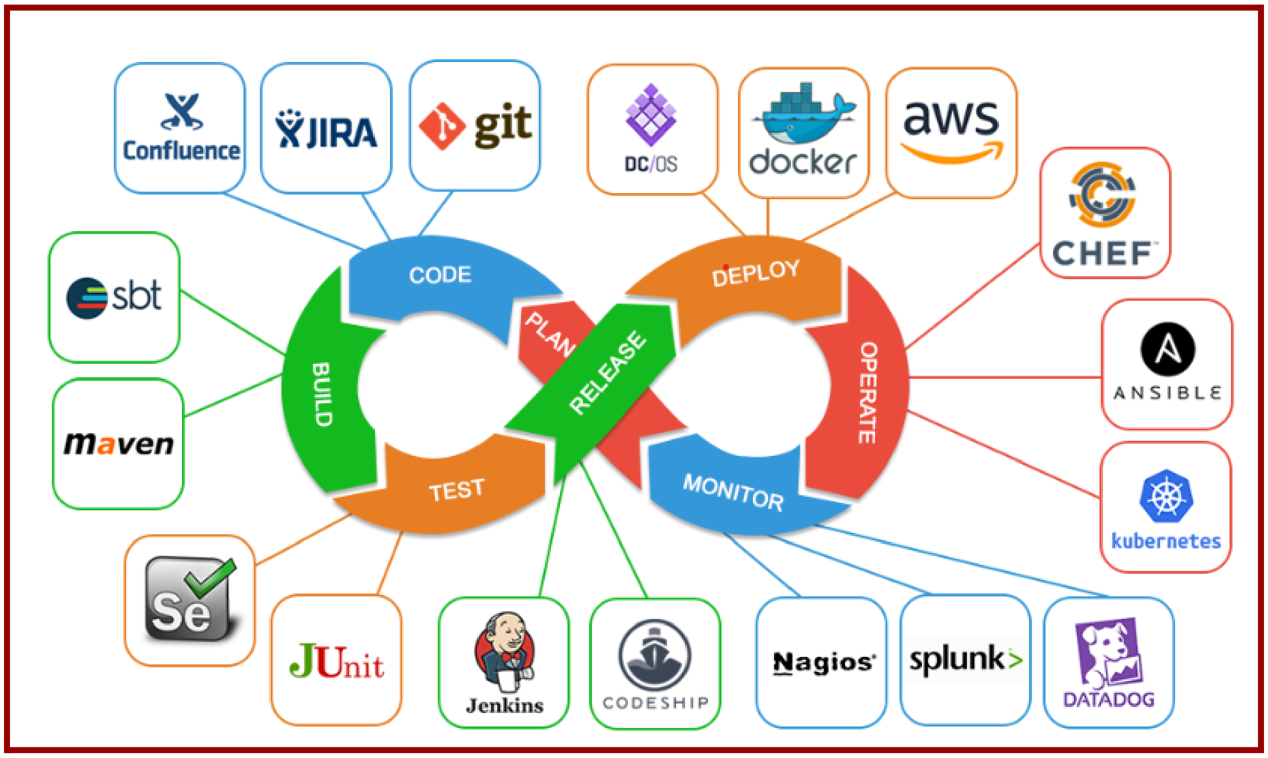
- Today, Netflix is the world’s largest streaming platform that operates in 190 countries with approximately 222 million subscribers globally.

- The different stages present in the DevOps cycle are as follows:
    - **Plan**: In this phase, teams identify the business requirements, have discussions, collect end users’ feedback and create a project roadmap. They also use project management tools such as the Kanban board. For example, while creating a feature, the development team might have to interact with the cybersecurity team to ensure no loopholes are created while adding this feature.
    - **Code**: In this phase, the developers who might be in different geographical locations write the code. For example, two developers

might be working on two different components of the feature while sitting in different time zones.

- **Build**: In this phase, developers push their code to source code repositories such as GitHub after ensuring it is running as expected.
- **Test**: In this phase, the developed feature is tested thoroughly against different test use cases.
- **Release**: In this phase, the entire engineering team works on scheduling releases and creating release packages(encapsulating all the external libraries and dependencies required by an application).
- **deploy**: In this phase, the release package is deployed into the production environment, where it is accessible to end users.
- **operate**: In this phase, the engineering team manages the provisioning(setting up) and configuring of the infrastructure.
- **monitor**: In this phase, different performance metrics, such as the number of requests, and the server response time, among others, are monitored.

The different tools used in various phases of the DevOps cycle are as shown below:

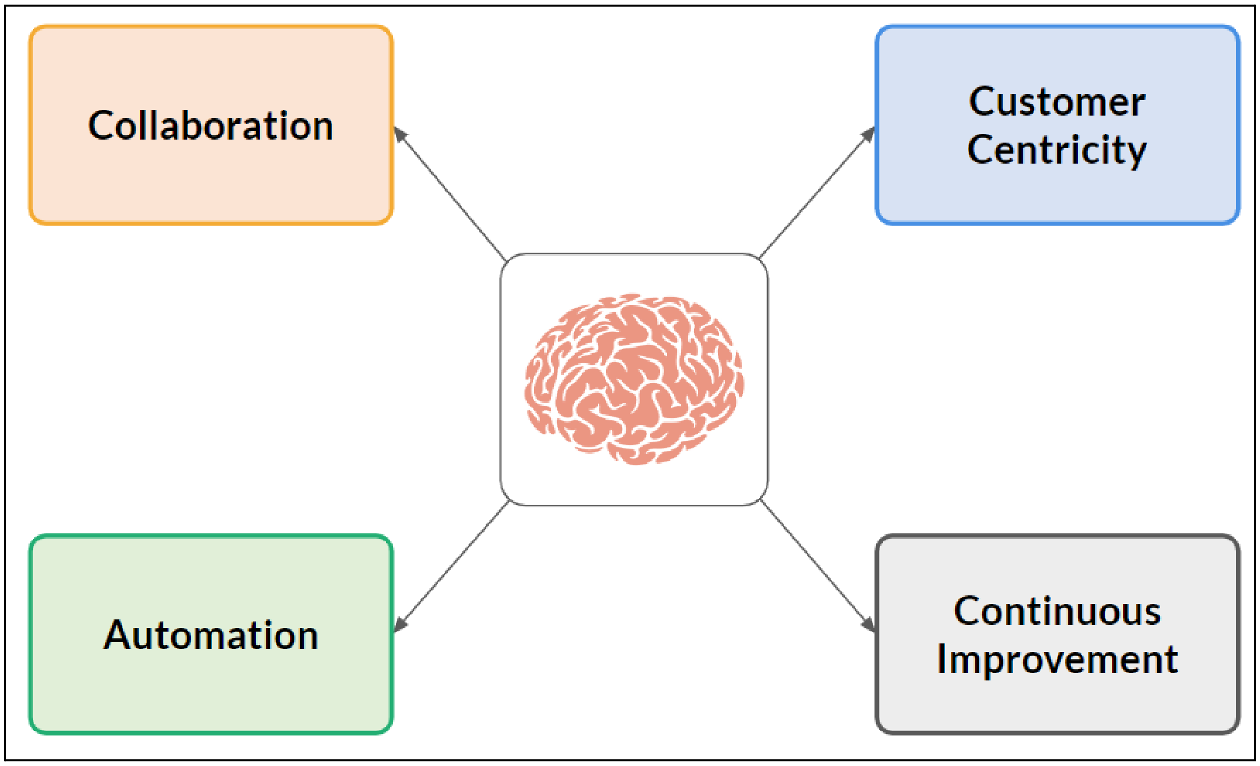


This complete DevOps cycle can be divided into two parts:

- **Continuous Integration(CI)**: It comprises of the plan, code, build and test phases. In these four phases, the code written by developers is pushed continuously, which triggers automatic test cases to run. If the code fails, the developers are notified immediately; otherwise, the code passes to the next step.
- **Continuous Delivery(CD)**: It comprises the release, deploy, operate and monitor phases. Once the code is tested properly, it is converted into release packages and deployed into the production environment, where the application is monitored continuously.

1.3 DevOps Principles

Whenever an organisation starts implementing DevOps practices, they look for the following fundamental key principles.



- **Collaboration**: DevOps increases collaboration among different teams working on a feature. The development and operation teams work together, share continuous feedback and communicate throughout the entire development cycle. Therefore, high-quality products are created.

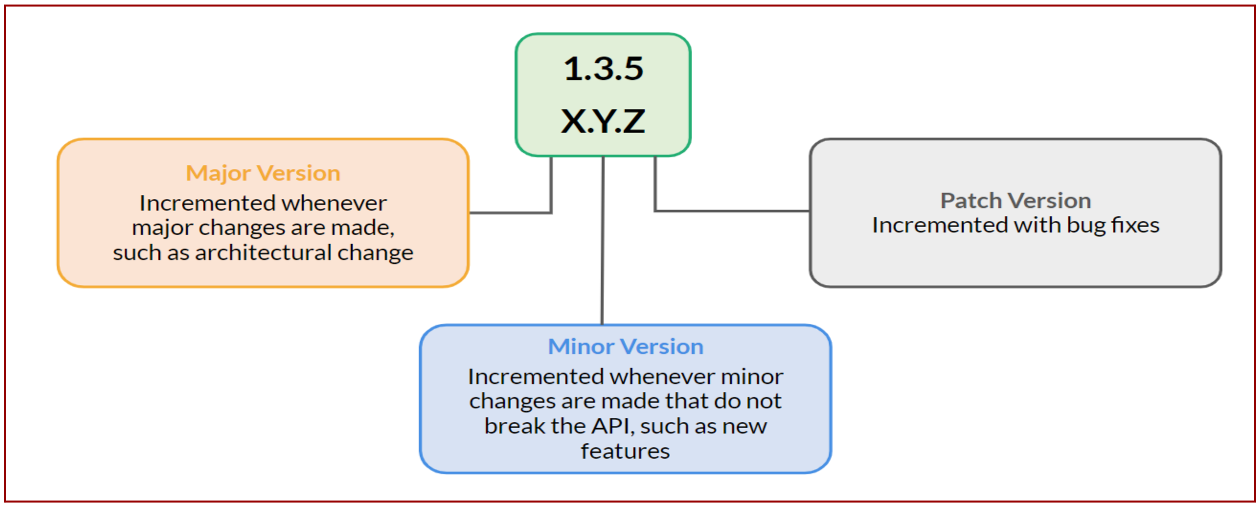
- **Customer-centricity**: In DevOps, development takes place in short time sprints; teams collect continuous feedback from end users and make improvements in the product. The continuous feedback from end users also helps us to analyze whether the expected impact is created or not.

- **Continuous Improvement**: Since DevOps supports rapid deployment and continuous feedback, the application developed keeps continuously improving.

- **Automation**: The customer demands continuously change, and teams deploy their code very frequently; therefore, manually checking for errors, creating packages, deploying a feature, etc. can be done in an automated fashion such that whenever a developer pushes the code, build and test steps are triggered automatically. If the software passes all the tests, it gets deployed into production. Alternatively, if the code fails, developers are notified immediately so that they can check the failed build. So, the advantage of automation is that developers can focus on writing code rather than worrying about other steps.

1.4 Semantic Versioning

With the advent of DevOps, the frequency at which developers deploy the code has increased and with the continuous feedback from end users, the applications have been improving continuously. With each new release, bug fix or modification in the existing code, a unique number is assigned to the newer release. This process of maintaining different versions(releases) of a particular software is called software versioning and the method by which a release is numbered is called semantic versioning. Semantic Versioning uses three numbers to represent different software versions that follow the structure of X.Y.Z(Major.Minor.Patch), as shown in the image below.


1.3.5
X.Y.Z
Major Version
Incremented whenever major changes are made, such as architectural change
Patch Version
Incremented with bug fixes
Minor Version
Incremented whenever minor changes are made that do not break the API, such as new features

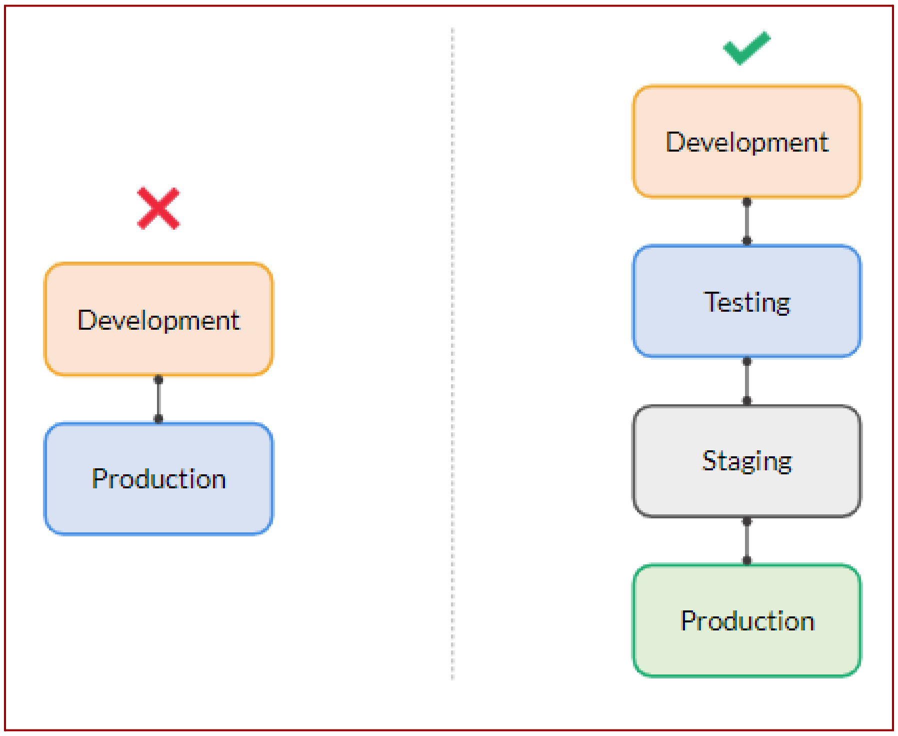
Now, let’s understand the meaning of each component:

Major(X): The leftmost integer is incremented only when some major change takes place, such as an architectural change. If the major version is incremented, then the minor and the patch versions are set to 0.

Minor(Y): The centremost integer is incremental only when a minor change occurs, such as adding a new feature. If the minor version is incremented, the patch version is set to 0.

Patch(Z): The rightmost integer is incremented in case any bug or issue is fixed.

1.5 Different Environments Used in Software Testing



A piece of code never goes directly from development to production. It first passes the development environment, where developers write code, to the testing environment, where testing takes place. Then, once the code passes the testing environment, it goes to the staging environment. A staging environment is similar to a production environment. It helps us understand how an application will work in an actual production environment. A sample group of users use the application in the staging environment and provide feedback. The application is then moved to the production environment if the feedback meets the criteria.