DevOps Lifecycle

DevOps is a practice that enables a single team to handle the whole application lifecycle, including development, testing, release, deployment, operation, display, and planning. It is a mix of the terms "Dev" (for development) and "Ops" (for operations). We can speed up the delivery of applications and services by a business with the aid of DevOps. Amazon, Netflix, and other businesses have all effectively embraced DevOps to improve their customer experience.

DevOps Lifecycle is the set of phases that includes DevOps for taking part in Development and Operation group duties for quicker software program delivery. DevOps follows positive techniques that consist of **code, building, testing, releasing, deploying, operating, displaying, and planning. DevOps lifecycle** follows a range of phases such as non-stop development, non-stop integration, non-stop testing, non-stop monitoring, and non-stop feedback. Each segment of the DevOps lifecycle is related to some equipment and applied sciences to obtain the process. Some of the frequently used tools are open source and are carried out primarily based on commercial enterprise requirements. DevOps lifecycle is effortless to manipulate and it helps satisfactory delivery.

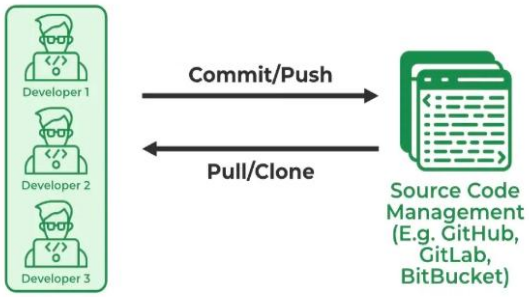
7 Cs of DevOps

1. Continuous Development
2. Continuous Integration
3. Continuous Testing
4. Continuous Deployment/Continuous Delivery
5. Continuous Monitoring
6. Continuous Feedback
7. Continuous Operations

The **DevOps lifecycle integrates** development and operations for seamless project delivery. To explore each phase of the DevOps lifecycle in-depth, from planning to continuous deployment, the **DevOps Engineering – Planning to Production** course offers a structured roadmap to mastering the entire lifecycle.

1. Continuous Development

In Continuous Development code is written in small, continuous bits rather than all at once, Continuous Development is important in DevOps because this improves efficiency every time a piece of code is created, it is tested, built, and deployed into production. Continuous Development raises the standard of the code and streamlines the process of repairing flaws, vulnerabilities, and defects. It facilitates developers' ability to concentrate on creating high-quality code.

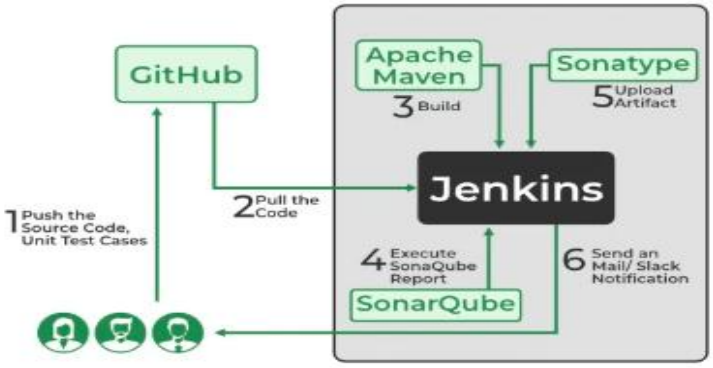


2. Continuous Integration

Continuous Integration can be explained mainly in 4 stages in DevOps.

They are as follows:

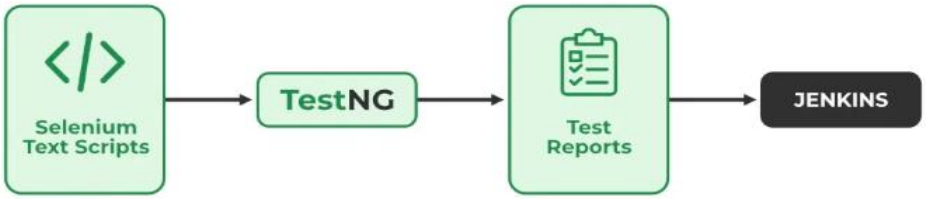
1. Getting the SourceCode from SCM
2. Building the code
3. Code quality review
4. Storing the build artifacts



The stages mentioned above are the flow of Continuous Integration and we can use any of the tools that suit our requirement in each stage and of the most popular tools are **GitHub for source code management(SCM)** when the developer develops the code on his local machine he pushes it to the remote repository which is GitHub from here who is having the access can Pull, clone and can make required changes to the code. From there by using **Maven we can build** them into the required package (war, jar, ear) and can test the Junit cases.**SonarQube performs code quality reviews** where it will measure the quality of source code and generates a report in the form of HTML or PDF format. **Nexus for storing the build artifacts** will help us to store the artifacts that are build by using Maven and this whole process is achieved by using a Continuous Integration tool Jenkins.

3. Continuous Testing

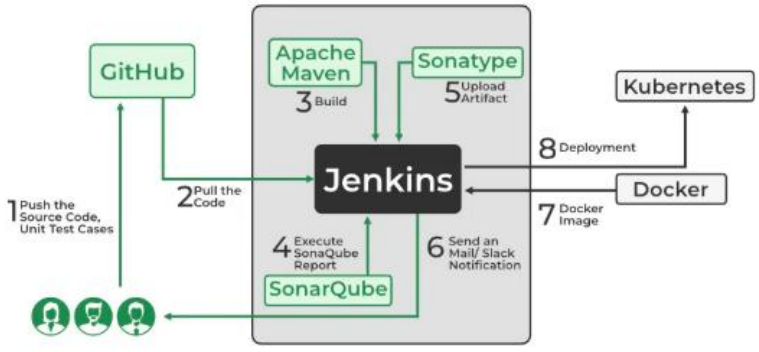
Any firm can deploy continuous testing with the use of the agile and DevOps methodologies. Depending on our needs, we can perform continuous testing using automation testing tools such as **Testsigma, Selenium, LambdaTest,** etc. With these tools, we can test our code and prevent problems and code smells, as well as test more quickly and intelligently. With the aid of a continuous integration platform like Jenkins, the entire process can be automated, which is another added benefit.



4. Continuous Deployment/ Continuous Delivery

Continuous Deployment: Continuous Deployment is the process of automatically deploying an application into the production environment when it has completed testing and the build stages. Here, we'll automate everything from obtaining the application's source code to deploying it.

Continuous Delivery: Continuous Delivery is the process of deploying an application into production servers manually when it has completed testing and the build stages. Here, we'll automate the continuous integration processes, however, manual involvement is still required for deploying it to the production environment.



5. Continuous Monitoring

DevOps lifecycle is incomplete if there was no Continuous Monitoring. Continuous Monitoring can be achieved with the help of Prometheus and Grafana we can continuously monitor and can get notified before anything goes wrong with the help of Prometheus we can gather many performance measures, including CPU and memory utilization, network traffic, application response times, error rates, and others. Grafana makes it possible to visually represent and keep track of data from time series, such as CPU and memory utilization.

6. Continuous 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 that which we are currently developing and can produce much more effective results for the end users also we reduce any unnecessary steps to deploy the application. Continuous Feedback can increase the performance of the application and reduce bugs in the code making it smooth for end users to use the application.

7. Continuous Operations

We will sustain the higher application uptime by implementing continuous operation, which will assist us to cut down on the maintenance downtime that will negatively impact end users' experiences. More output, lower manufacturing costs, and better quality control are benefits of continuous operations.

Different Phases of the DevOps Lifecycle

1. **Plan:** Professionals determine the commercial need and gather end-user opinions throughout this level. In this step, they design a project plan to optimize business impact and produce the intended result.
2. **Code** – During this point, the code is being developed. To simplify the design process, the developer team employs lifecycle DevOps tools and extensions like Git that assist them in preventing safety problems and bad coding standards.
3. **Build –** After programmers have completed their tasks, they use tools such as Maven and Gradle to submit the code to the common code source.
4. **Test** – To assure software integrity, the product is first delivered to the test platform to execute various sorts of screening such as user acceptability testing, safety testing, integration checking, speed testing, and so on, utilizing tools such as JUnit, Selenium, etc.
5. **Release** – At this point, the build is prepared to be deployed in the operational environment. The DevOps department prepares updates or sends several versions to production when the build satisfies all checks based on the organizational demands.
6. **Deploy** – At this point, Infrastructure-as-Code assists in creating the operational infrastructure and subsequently publishes the build using various DevOps lifecycle tools.
7. **Operate** – This version is now convenient for users to utilize. With tools including Chef, the management department take care of server configuration and deployment at this point.
8. **Monitor** – The DevOps workflow is observed at this level depending on data gathered from consumer behavior, application efficiency, and other sources. The ability to observe the complete surroundings aids teams in identifying bottlenecks affecting the production and operations teams' performance.