**Branch :- Computer Science and Engineering Class :- III Year**

**Subject :- C-Skill Lab-IV Sem :- VI**

**Teacher Manual**

**PRACTICAL NO. 3**

**Aim:** Study DevOps life cycle.

**Theory:**

**DevOps**

* The term DevOps is a combination of two words namely Development and Operations. DevOps is a practice that allows a single team to manage the entire application development life cycle, that is, development, testing, deployment, operations.
* The aim of DevOps is to shorten the system’s development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.
* DevOps is a software development approach through which superior quality software can be developed quickly and with more reliability.

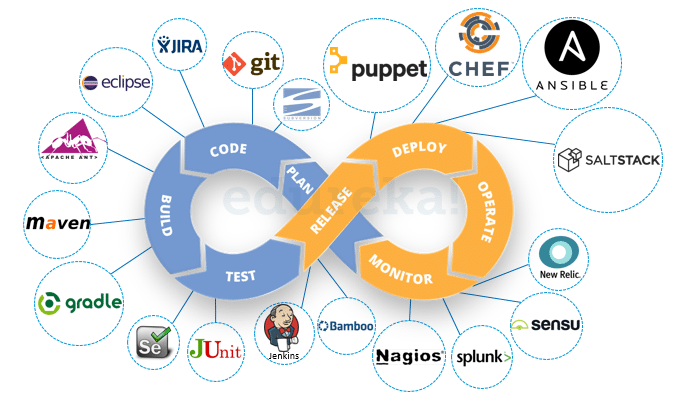


Fig: DevOps Lifecycle

**DevOps Life cycle**

* It consists of various stages such as continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring.

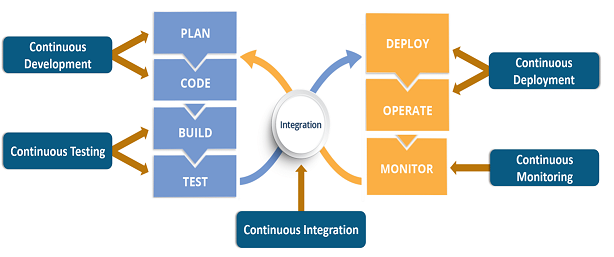


Fig: DevOps Lifecycle Stages

#### Continuous Development

* This phase involves ‘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](https://www.edureka.co/blog/devops-tools" \t "_blank) that are required for planning, but there are a number of tools for maintaining the code.
* The code can be written in any language, but it is maintained by using Version Control tools.
* Maintaining the code is referred to as Source Code Management.
* The most popular tools used are Git, SVN, Mercurial, CVS, and JIRA.
* Also, tools like Ant, [Maven](https://www.edureka.co/blog/create-selenium-maven-project/" \t "_blank), Gradle can be used in this phase for building/ packaging the code into an executable file that can be forwarded to any of the next phases.

#### Continuous Testing

* This is the stage where the developed software is continuously tested for bugs.
* For Continuous testing, automation testing tools like Selenium, TestNG, JUnit, etc are used.
* These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality.
* In this phase, Docker Containers can be used for simulating the test environment.
* Selenium does the automation testing, and the reports are generated by [TestNG](https://www.edureka.co/blog/selenium-webdriver-tutorial" \t "_blank). This entire testing phase can be automated with the help of a Continuous Integration tool called Jenkins.
* Suppose you have written a selenium code in Java to test your application. Now you can build this code using ant or maven. Once the code is built, it is tested for User Acceptance Testing (UAT). This entire process can be automated using [Jenkins](https://www.edureka.co/blog/jenkins-tutorial/" \t "_blank).
* Automation testing saves a lot of time, effort and labor for executing the tests instead of doing this manually. Besides that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. We can also schedule the execution of the test cases at predefined times.
* After testing, the code is continuously integrated with the existing code.

1. **Continuous Integration**

* 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 a weekly basis. Every commit is then built and this allows early detection of problems if they are present.
* Building code not only involves compilation but it also includes code review, unit testing, integration testing, and packaging.
* The code supporting new functionality is [continuously integrated](https://www.edureka.co/blog/continuous-integration/" \t "_blank) with the existing code. Since there is continuous development of software, the updated code needs to be integrated continuously as well as smoothly with the systems to reflect changes to the end-users.
* Jenkins is a very popular tool used in this phase.

1. **Continuous deployment**

* This is the stage where the code is deployed to the production servers.
* It is also important to ensure that the code is correctly deployed on all the servers. Configuration management and [Containerization tools](https://www.edureka.co/blog/docker-tutorial" \t "_blank) help in achieving Continuous Deployment (CD).
* [Configuration Management](https://www.edureka.co/blog/what-is-puppet/" \t "_blank) is the act of establishing and maintaining consistency in an application’s functional requirements and performance.
* In simpler words, it is the act of releasing deployments to servers, scheduling updates on all servers and most importantly keeping the configurations consistent across all the servers.
* Since the new code is deployed on a continuous basis, configuration management tools play an important role in executing tasks quickly and frequently. Some popular tools that are used here are Puppet, [Chef](https://www.edureka.co/blog/what-is-chef/" \t "_blank), Salt Stack, and Ansible.
* Containerization tools also play an equally important role in the deployment stage. Docker and Vagrant are the popular tools used for this purpose.
* These tools help produce consistency across Development, Test, Staging and Production environments. Besides this, they also help in scaling-up and scaling-down of instances swiftly.
* Containerization tools help in maintaining consistency across the environments where the application is developed, tested and deployed.
* Using these tools, there is no scope of errors/ failure in the production environment as they package and replicate the same dependencies and packages used in the development/ testing/ staging environment. It makes application easy to run on different computers.

1. **Continuous Monitoring**

* In this stage performance of application is continuously monitored.
* In this stage vital information about the use of the software is recorded.
* This information is processed to recognize the proper functionality of the application.
* The system errors such as low memory, server not reachable, etc are resolved in this phase.
* The root cause of any issue is determined in this phase. It maintains the security and availability of the services.
* Also, if there are network issues, they are resolved in this phase. It helps us automatically fix the problem as soon as they are detected.
* This practice involves the participation of the Operations team who will monitor the user activity for bugs or any improper behavior of the system.
* The popular tools used for this are [Splunk](https://www.edureka.co/blog/what-is-splunk/" \t "_blank), [ELK Stack](https://www.edureka.co/blog/elk-stack-tutorial/" \t "_blank), [Nagios](https://www.edureka.co/blog/nagios-tutorial/" \t "_blank), NewRelic and Sensu.
* These tools help to monitor the application’s performance and the servers closely and also enable you to check the health of the system proactively.
* They can also improve productivity and increase the reliability of the systems, which in turn reduces IT support costs. Any major issues if found are reported to the development team so that it can be fixed in the continuous development phase. This leads to a faster resolution of the problems.

These DevOps stages are carried out on loop continuously till it achieves the desired product quality. Therefore, almost all of the major IT companies have shifted to DevOps for building their products.

**Result:** Thus, I have studied DevOps life cycle.