**DevOps Introduction**

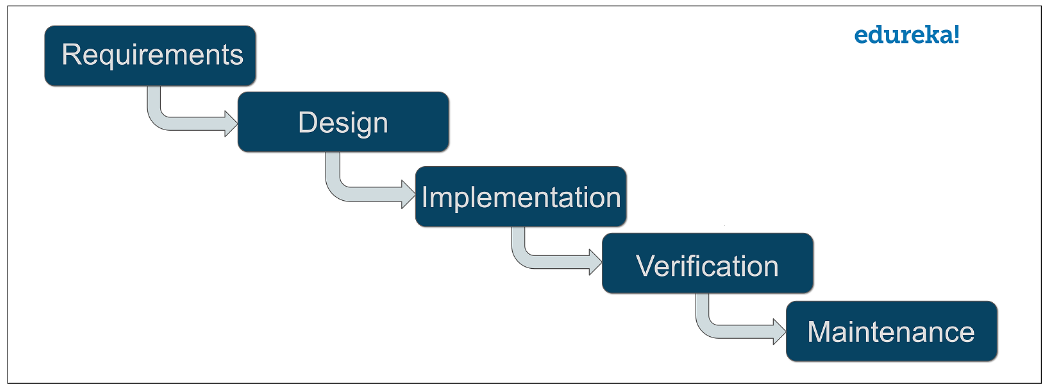
DevOps combines development and operations to increase the efficiency, speed, and security of software development and delivery compared to traditional processes.

**Why DevOps?**

Before we know what DevOps is, it is very important to know how DevOps came into existence. Before DevOps, we had the waterfall model and the agile model for software development. So let us have a look at the waterfall model.

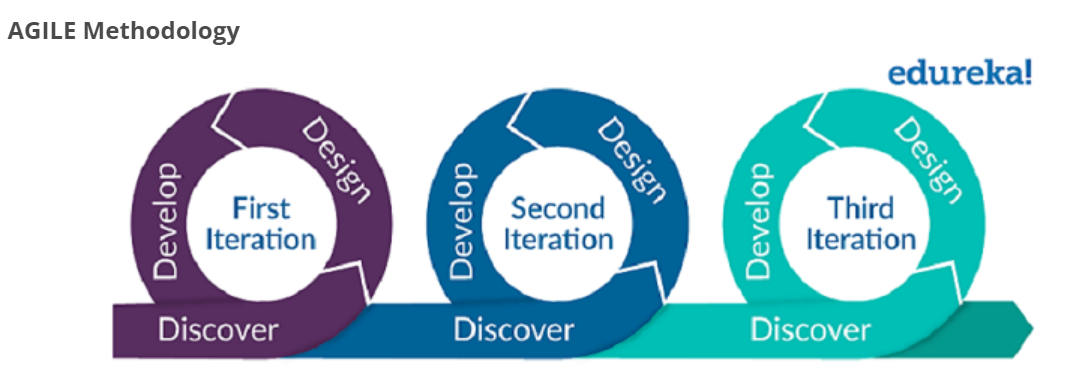
**Waterfall Model**

The waterfall model can be defined as a sequential process in the development of a system or software that follows a top-down approach. This model was a straight forward and linear model. The waterfall model had various phases such as Requirement Definition, Software Design, Implementation, Testing, Deployment, and Maintenance.



Software development companies that used the waterfall model approach had to spend a lot of time to get their product right. That is because unless you complete a particular phase, you could not proceed to the next phase. Also, the working software was delivered only after the final phase of this model.

This model was only suitable for projects which had stable requirements. By stable, I mean that requirements will not change with the time. But in today’s world, this is a very unlikely thing because requirements keep on changing from time to time. These were a few drawbacks of the waterfall model.

**AGILE Methodology**

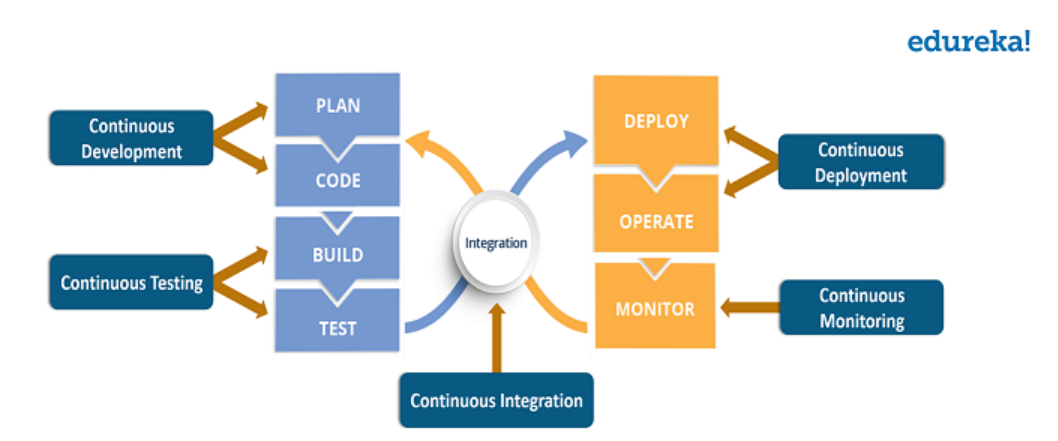
Next came, the agile methodology of software development. Agile methodology is a practice that promotes continuous iteration of development and testing throughout the software development life cycle of the project. Both development and testing activities are concurrent, unlike the Waterfall model. While the Agile approach brought agility to development, it was lost on Operations which did not come up to speed with agile practices.

There was a lack of collaboration between Developers and Operation Engineers and this slowed down the development process and releases. Software companies had begun to realize the need for better collaboration between the teams and faster delivery of software. This gave birth to the DevOps approach. DevOps enabled continuous software delivery with less complex problems to fix and faster resolution of problems.

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



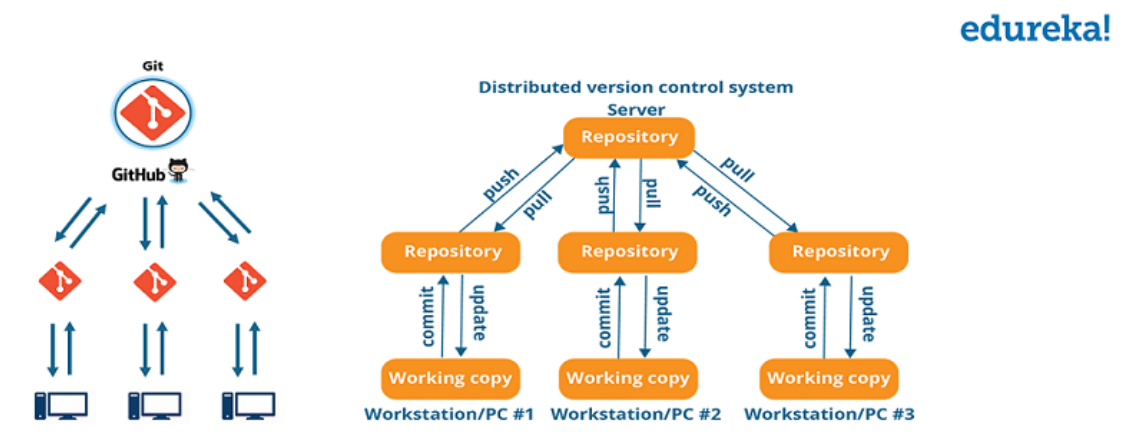
**What is DevOps Life cycle?**

As mentioned earlier, the various phases such as continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring constitute DevOps Life cycle. Now let us have a look at each of the phases of DevOps life cycle one by one.

**Continuous Development –**

This is the phase that 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) 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/), and 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.



Now let us try to know a bit more about Git.

* Git is a distributed version control tool that supports distributed non-linear workflows by providing data assurance for developing quality software. Tools like Git enable communication between the development and the operations team.
* When you are developing a large project with a huge number of collaborators, it is very important to have communication between the collaborators while making changes in the project.
* Commit messages in Git play a very important role in communicating among the team. Apart from communication, the most important reason to use Git is that you always have a stable version of the code with you.
* Hence, Git plays a vital role in succeeding at DevOps.

**Continuous Testing –**

This is the stage where the developed software is continuously tested for bugs. For [Continuous testing](https://www.edureka.co/continuous-testing-devops-training), 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.

Automation testing saves a lot of time, effort and labour 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.

#### Continuous Integration –

#### 

This stage is the heart of the entire DevOps life cycle. 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/) 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. Whenever there is a change in the [Git repository](https://www.edureka.co/blog/git-tutorial/), Jenkins fetches the updated code and it prepares a build of that code which is an executable file in the form of a war or a jar. This build is then forwarded to the test server or the production server.

#### 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. Before moving on, let us try to understand a few things about Configuration management and [Containerization tools](https://www.edureka.co/blog/docker-tutorial). These set of tools here help in achieving Continuous Deployment (CD).

[Configuration Management](https://www.edureka.co/blog/what-is-puppet/) is the act of establishing and maintaining consistency in an application’s functional requirements and performance. Let me put this 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/), [SaltStack, and Ansible](https://www.edureka.co/community/22254/ansible-vs-saltstack" \t "_blank).

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 your application easy to run on different computers.

#### Continuous Monitoring –

This is a very crucial stage of the DevOps life cycle where you continuously monitor the performance of your application. Here 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 behaviour of the system. The popular tools used for this are [Splunk](https://www.edureka.co/blog/what-is-splunk/), [ELK Stack](https://www.edureka.co/blog/elk-stack-tutorial/), [Nagios](https://www.edureka.co/blog/nagios-tutorial/" \t "_blank), NewRelic and Sensu. These tools help you 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 you achieve the desired product quality. Therefore almost all of the major IT companies have shifted to DevOps for building their products.

**DevOps stages & Tools:**

1. continuous development
   1. Plan – No DevOps tool
   2. Code – Git, GitHub
   3. Build – Maven(Spring boot package)
2. continuous integration
   1. Release – Jenkins, Ansible (Configuration management tool)
3. continuous testing
   1. Test – Junit, SonarQube
4. continuous deployment
   1. Deploy – tomcat, Docker, Kubernetes, Ansible
5. continuous monitoring
   1. Monitor – Splunk, Nagios

CICD 🡪 Jenkins