Overview of CI/CD:

CI CD Pipeline implementation or the Continuous Integration/Continuous Deployment software is the backbone of the modern DevOps environment. You can find the requirement of ***Continuous Integration & Continuous Deployment skills*** in various job roles such as Data Engineer, Cloud Architect, Data Scientist, etc. CI/CD bridges the gap between development and operations teams by automating build, test and deployment of applications. In this blog, we will know What is CI CD pipeline and how it works.

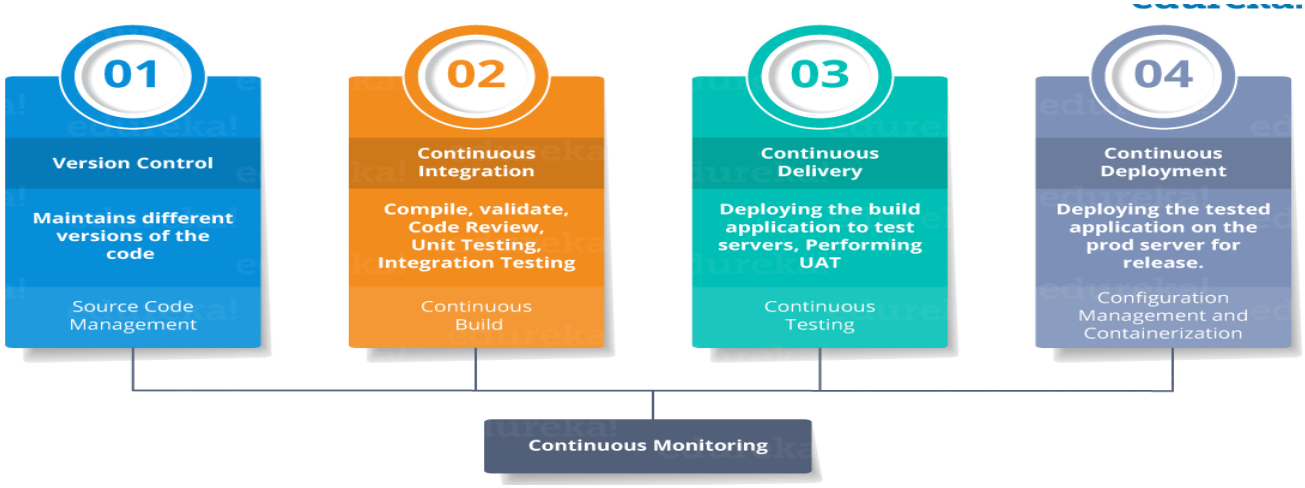
Before moving onto the CI CD pipeline's working, let's start by understanding DevOps.

What is DevOps?

DevOps is a software development approach which involves continuous development, continuous testing, continuous integration, continuous deployment and continuous monitoring of the software throughout its development life cycle. This is exactly the process adopted by all the top companies to develop high-quality software and shorter development life cycles, resulting in greater customer satisfaction, something that every company wants.

DevOps Stages

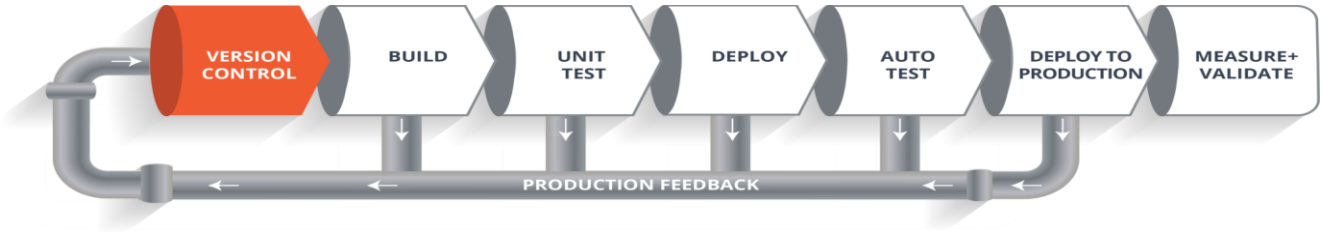
Your understanding of what is DevOps, is incomplete without learning about its life cycle. Let us now look at the DevOps lifecycle and explore how they are related to the software development stages.



What is CI CD Pipeline?

CI stands for Continuous Integration and CD stands for Continuous Delivery and Continuous Deployment. You can think of it as a process which is similar to a software development lifecycle.

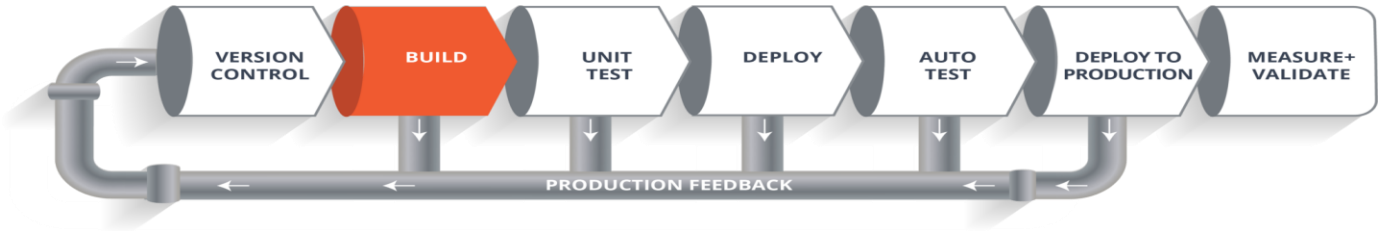
Now let us see how does it work.



The above pipeline is a logical demonstration of how a software will move along the various phases or stages in this lifecycle, before it is delivered to the customer or before it is live on production.

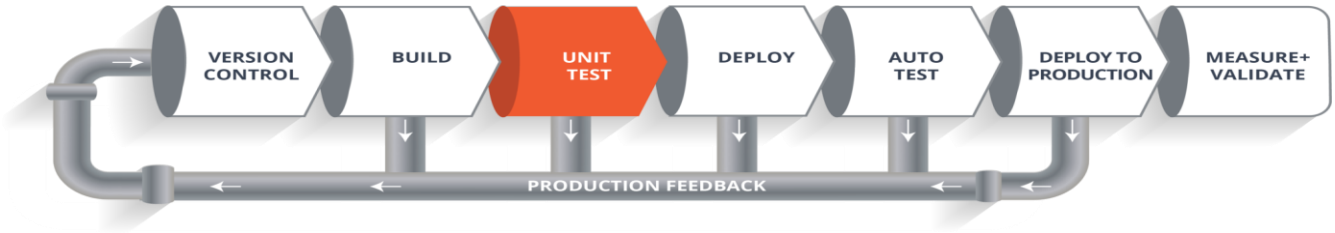
Let's take a scenario of CI CD Pipeline. Imagine you're going to build a web application which is going to be deployed on live web servers. You will have a set of developers who are responsible for writing the code which will further go on and build the web application.

Now, when this code is committed into a version control system(such as git, svn) by the team of developers. Next, it goes through the **build phase** which is the first phase of the pipeline, where developers put in their code and then again code goes to the version control system having a proper version tag.



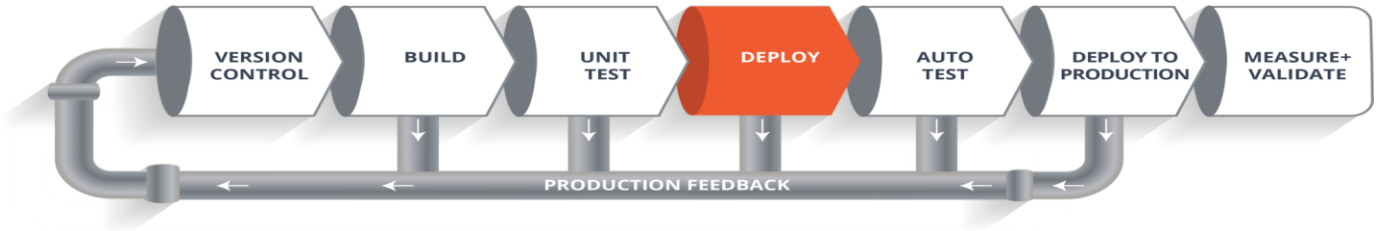
Suppose we have a DotNet code and it needs to be compiled before execution. So, through the version control phase, it again goes to build phase where it gets compiled. You get all the features of that code from various branches of the repository, which merge them and finally use a compiler to compile it. This whole process is called the **build phase**.

Testing Phase:



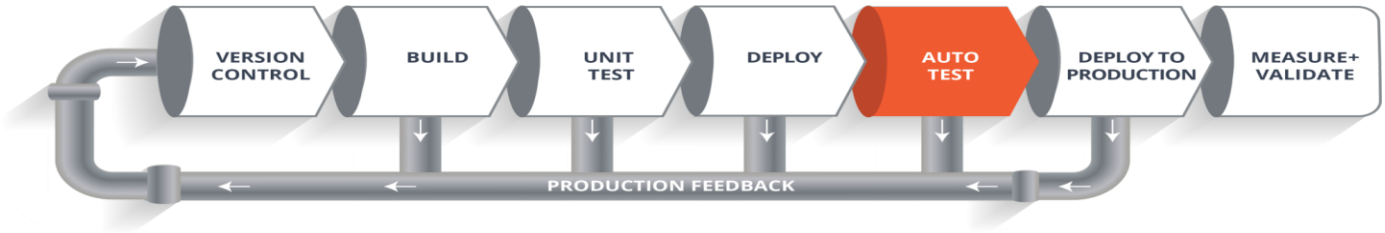
Once the build phase is over, then you move on to the **testing phase**. In this phase, we have various kinds of testing, one of them is the *unit test* (where you test the chunk/unit of software or for its sanity test).

Deploy Phase:



When the test is completed, you move on to the **deploy phase**, where you deploy it into a staging or a test server. Here, you can view the code or you can view the app in a simulator.

Auto Test Phase:

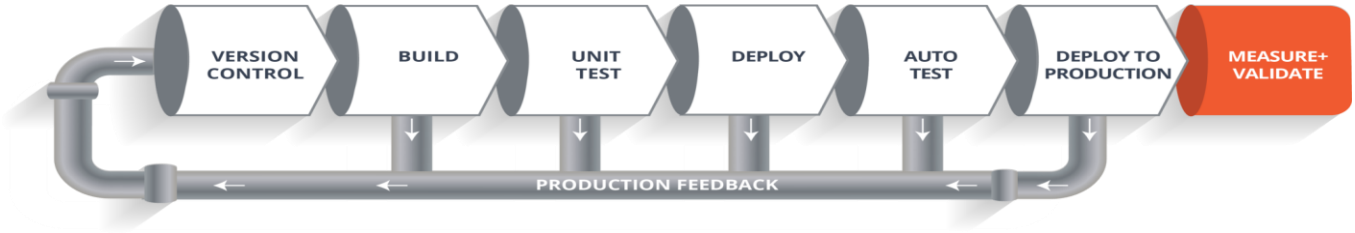


Once the code is deployed successfully, you can run another set of a sanity test. If everything is accepted, then it can be deployed to production.

Deploy to Production:

Meanwhile in every step, if there is some error, you can shoot a mail back to the development team so that they can fix them. Then they will push it into the version control system and goes back into the pipeline.

Once again if there is any error reported during testing, again the feedback goes to the dev team where they fix it and the process re-iterates if required.



So, this lifecycle continues until we get a code or a product which can be deployed in the production server where we measure and validate the code.