**Jenkins**

**What is Jenkins:**

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration (CI), Continuous delivery (CD) and continue deployment purpose in software development.

**What is Continuous Integration (CI), Continuous delivery (CD) and continue deployment:**

It is a software development practices in which we will automatically build, test and deploy

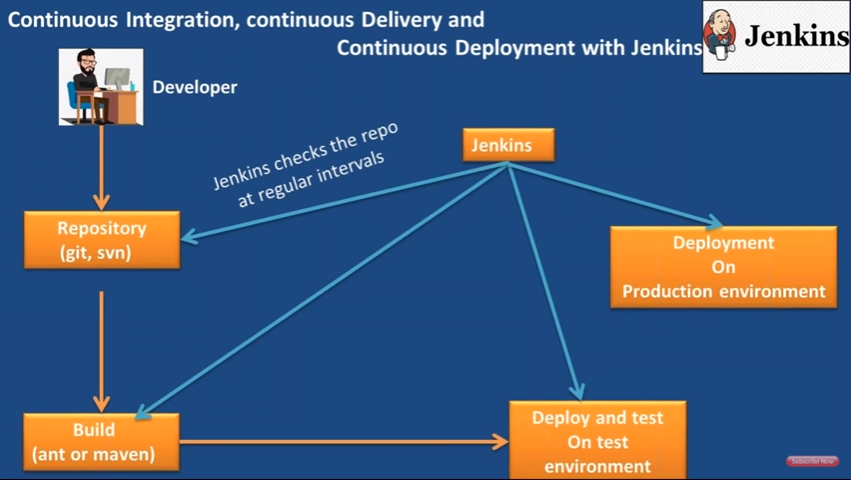
test/production environment.

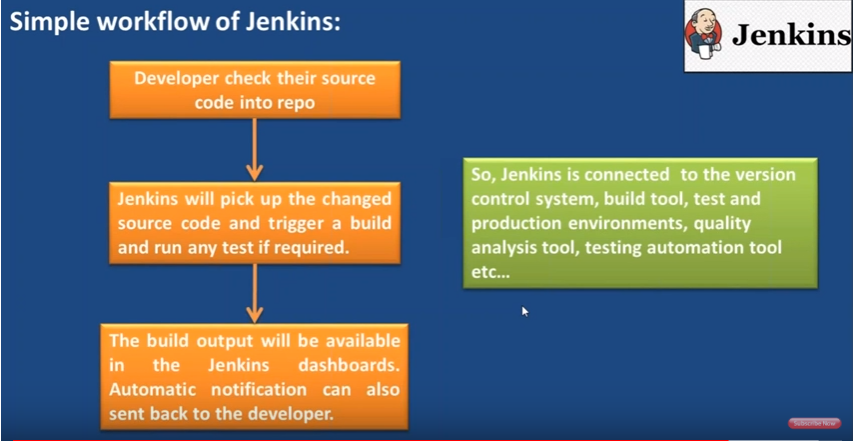
Software projects every time whenever a developer pushes code change to the repository.

In Continuous Integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment. If deployment is a success, the code is pushed to production. This commit, build, test, and deploy is a continuous process and hence the name continuous integration/deployment.

A Continuous Integration Pipeline is a powerful instrument that consists of a set of tools designed to **host**, **monitor**, **compile** and **test** code, or code changes, like:

* Continuous Integration Server (Jenkins, Bamboo, CruiseControl, TeamCity, and others)
* Source Control Tool (e.g., CVS, SVN, GIT, Mercurial, Perforce, ClearCase and others)
* Build tool (Make, ANT, Maven, Ivy, Gradle, and others)
* Automation testing framework (Selenium, Appium, TestComplete, UFT, and others)





|  |  |
| --- | --- |
| **Before Jenkins** | **After Jenkins** |
| Once all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed.  Code commit built, and test cycle was very infrequent, and a single build was done after many days. | The code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day  If the build is successful, then Jenkins will deploy the source into the test server and notifies the deployment team.  If the build fails, then Jenkins will notify the errors to the developer team. |
| Since the code was built all at once, some developers would need to wait until other developers finish coding to check their build | The code is built immediately after any of the Developer commits. |
| It is not an easy task to isolate, detect, and fix errors for multiple commits. | Since the code is built after each commit of a single developer, it's easy to detect whose code caused the built to fail |
| Code build and test process are entirely manual, so there are a lot of chances for failure. | Automated build and test process saving timing and reducing defects. |
| The code is deployed once all the errors are fixed and tested. | The code is deployed after every successful build and test. |
| Development Cycle is slow | The development cycle is fast. New features are more readily available to users. Increases profits. |

CI:

Automated build for every commit

C-delivery :

Automated buld and UAT for every commit.

C-Deplouement:

Automated build, uat release to production for every commit.

Continuous integration:

fast feedback to the developer

easy for the developers to find and fix issues

improves developers productivity

improves code quqlity, minimize defects

its automates integration flow

Continuous Delivery:

auatomates software release process

release software updated faster

improves developers productivity

Continuous Deployment: continuos deployments is completely automate process write from source code checkin till deployment to the production.

Continuous Delivery: continuos deployments is a automate process expact deploying the application to the

production based on its approval.

**Jenkins steps:**

**Declarative Pipeline fundamentals**

In Declarative Pipeline syntax, the pipeline block defines all the work done throughout your entire Pipeline.

Jenkinsfile (Declarative Pipeline)

pipeline {

agent any

stages {

stage('Build') {

steps {

//

}

}

stage('Test') {

steps {

//

}

}

stage('Deploy') {

steps {

//

}

}

}

}

Execute this Pipeline or any of its stages, on any available agent.

Defines the "Build" stage.

Perform some steps related to the "Build" stage.

Defines the "Test" stage.

Perform some steps related to the "Test" stage.

Defines the "Deploy" stage.

Perform some steps related to the "Deploy" stage.

|  |
| --- |
| script type:  pipeline {  agent any  stages {  stage('clone repo and clean it') {  steps {  sh "rm -r SwaggerWithRestApi"  sh "git clone https://github.com/NagendraMekala/SwaggerWithRestApi.git"  sh "mvn clean -f SwaggerWithRestApi"  }  }  stage('Test') {  steps {  sh "mvn test -f SwaggerWithRestApi"  }  }  stage('Deploy') {  steps {  sh "mvn package -f SwaggerWithRestApi"  }  }  }  }  Using git scm:  pipeline {  agent any  stages {  stage('---clean it---') {  steps {  sh "mvn clean"  }  }  stage('---test---') {  steps {  sh "mvn test"  }  }  stage('---deploy---') {  steps {  sh "mvn package"  }  }  }  } |

**Unix-based Jenkins commands**

You installed Jenkins on a Debian-based or a Fedora-based distribution, you can use the following commands:

$ sudo service jenkins restart

$ sudo service jenkins stop

$ sudo service jenkins start

Or in the latest distribution of Linux:

$ sudo systemctl start jenkins.service

$ sudo systemctl stop jenkins.service

$ sudo systemctl restart jenkins.service

**Windows**

You installed Jenkins as a service on Windows, you can either use the UI component Services manager (by running services.msc) or you can use the following command:

$ C:\Program Files (x86)\Jenkins>jenkins.exe start

$ C:\Program Files (x86)\Jenkins>jenkins.exe stop

$ C:\Program Files (x86)\Jenkins>jenkins.exe restart