**What is Jenkins?**

Jenkins is a powerful Continuous Integration Server (automation server) written in Java that allows continuous development, test and deployment of codes. It is a web application which can run on any web server.

Jenkins has got many plugins that allows to connect to all kinds of tools i.e software development, development, coding, deployment, connect to source code.

**What is continuous integration?**

The software development life cycle, the delivery happening in small sprints, may be three to four weeks delivery. There are bunch of developers working located in different location are working on same code base on same branch, If the code check-in doesn’t happen quickly/daily finding problems at later stage is difficult, earlier detection of problems is quick to resolve will not affect the software delivery schedules.

As part of CI , every developer he checks in code daily, end of the day we have automated server wakes up and pulls out the code and it builds on completely different server ( CI Server ) it got all the tools that are required to Compile it , Build it and Test it, (if we have automated test case, most of the Regression test suite) any breakages detected will notified( receive emails).

**Why Pipeline?**

* Pipeline is suite of Jenkins features, installed as plugins, which enable implementing and integrating continuous delivery pipelines into Jenkins.
* Various steps in software development life cycle approach.
* Once the build is ready, we will propagate and build moves to various phases (environments)
* The first phase will minimal testing (unit test) next integration test, functional tests, automated acceptance tests, user acceptance test and last release.
* Each of these tasks are jobs in Jenkins.

**How does Jenkins Pipeline work? (pre Jenkins 2.0)**

**Build Triggers**

* **To link the jobs/one job after other job.**

Post build action > build other projects.

To view all jobs in sequence we need to install > **delivery pipeline plugin.**

First provide view name and select **Delivery pipeline view** option.

Then **components** > **add name and provide initial job name** (first job in pipeline)

**Post Jenkins 2.0:**

Introduced a feature called **pipelines are written in form of scripts** using programming language.

* **Scripted Pipeline – groovy scripting**
* **Declarative Pipelines**

**node**: represents the agent or actual machine on which the job would be running.

**stage**: bunch of stages declared with multiple steps

**steps**: to define steps to be executed

**Scripted Pipeline:** it is based on Groovy script as their Domain Specific Language, one or more blocks does the core work throughout the entire pipeline

**Jenkinsfile (Scripted Pipeline)**

node {

stage(‘Build’){

////code

}

stage(‘Test’){

////code

}

stage(‘Deploy’){

////code

}

}

**Declarative Pipeline:** it provides a simple and friendly syntax to define a pipeline without the need of groovy script. Here, pipeline block defines the work done throughout the pipeline.

pipeline {

agent any

stages {

stage(‘Compile’){

steps{

echo 'complied successfully';

}

}

stage(‘Unit’){

steps{

echo 'unit test';

}

}

stage(‘Build’){

steps{

echo 'build successfully';

}

}

stage(‘Deploy’){

steps{

echo 'Deployed';

}

}

}

}

post {

always {

echo ‘This will always run’

}

success {

echo ‘This will run only if successful’

}

failure (

echo ‘This will run only if fails’

)

unstable{

echo ‘ This will run only if the run was marked as unstable’

}

changed{

echo ‘This will run only if the state of the pipeline has changed’

echo ‘for example if the pipeline was previously failing but is now successful’

}

}

}

Note : if the pipeline script is in Source code then we will select, pipeline script for SCM.

@@@@@@@@

**Build Triggers**

* To link the jobs or one job after other job.

Post build action > build other projects.

To view all jobs in sequence we need to install **delivery pipeline plugin.**

First provide view name and select Delivery pipeline view option.

Then components > add name and provide initial job name (first job in pipeline)

1. **Build Periodically:**

Corn jobs

\*\*\*\*\* --- every minute

1. **SCM Poll: to tell Jenkins to poll every 5 minutes for commits in repo.**

H/5 \* \* \* \*

1. **GitHub hook trigger for GitScm polling:**

Step1: got to Jenkins configure system> click on Github Server, copy the the url.

Step2: got GitHub > choose the project > setting > got web hooks> paste url.

Step 3: then build trigger for a job select> GitHub hook trigger for GitScm polling.

**E-mail notifications:**

Step 1:

Manage Jenkins> Configure System > E-mail Notification:

[Note:We need SMTP server, we need to mention SMTP Server name: for g-mail it is smtp.gmail.com ]

Step 2: select > use SMTP Authentication

Provide user name > emailed

Provide password > password

Select > use SSL and provide SMTP port number [ for gmail, it is 465]

Reply-To address > noreply@gmail.com

**Configuration of Jenkins Master-Slave:**

**Master and Slave/Agent**: (Means Jenkins instances are on different machines.)

A Jenkins master comes with the basic installation of Jenkins and in this configuration the master handles all tasks for your build system.

Jenkins supports the master-slaves architecture, i.e. many slaves works for a master. It is also known as Jenkins Distributed Builds.

Jenkins Agent/Slave: is just a small java "Client" processes that connects to the "Master" Jenkins instance over Java Network Launch Protocol. [JNLP: Java Network Launch Protocol.]

Jenkins Master role is to schedule the build jobs, assign slaves/agents and send builds to the slaves for the actual execution.

It also monitors the slaves (taking them online and offline as and when required), getting back the response of the build results from slaves then showing the build results on console.

All of the job results are collated on the master node for easy viewing.

It will also allow you to run jobs on different environments, like Linux, Windows, MacOS, etc

You might need to run same test case on different environments in parallel, that is where this distributed approach helps you to achieve the desired results quickly.

Once you configured few Jenkins slaves/agents, you might remove the executors on the Jenkins master in order to free up Jenkins master resources, but this isn’t necessary.

With this approach of Jenkins master-slaves, the actual workload of building projects are delegated to multiple “slave” nodes, which allows to run numerous projects and their jobs.

[Note: jdk version should be same on master and slave.]

**Step 1**: configure Agent in Global Tool Configuration to Random.

Manage Jenkins > Global Tool Configuration>Agents> select Random.

**Step 2**:

Manage Jenkins > Manage Node > New Node > provide Node Name>Permanent Agent>ok.

Then

>provide Description> Remote root directory (path to create folder where Jenkins for Jenkins workspace) > Labels (provide labels to node) >Launch Method (choose Launch agent by connecting it to the master) > save.

**Step 3**: Populate newly created node.

Copy the complete agent command line and download the agent.jar file.

**Step 4**:

Copy and paste the download jar file into the workspace folder.

Then navigate to the workspace through command line and execute the agent command line command to execute the jar file.

**Step 5**:

Build configuration > General > Restrict where this project can be run > provide the Agent name>save.

Then Build now.

Next remote:

[Note: Bunch of Plugins are provided for Jenkins to connect anything like Github, Aws,Docker instance]

**Artifacts**: Artifacts can be used to represent data created as a side-effect of running a Jenkins build. Artifacts are files which are associated with a single build. A build can have any number of artifacts associated with it.

[ By default, Jenkins archives artifacts generated by the build, are stored in the JENKINS\_HOME directory with all other elements such as job config files].

**Deployment artifacts**: is a process of putting an artifacts where it needs to be and performing any taks it need in order to fulfill its purpose.

pipeline{

agent none

stages{

stage(‘Non- Parallel Stage’){

agent{

label “master”

}

}

stage(‘Run Tests’){

parallel{

stage(‘Test on Window’){

agent{

label “Windows\_Node”

steps{ echo “Task1 on Agent”

}

}

}

stage(‘Test on Master’){

agent{

label “Master”

steps{ echo “Task2 on Master”

}

}

}

}

}

}

}