**JENKINS COMPLETE**

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

Needed softwares:

1. Java

2. Maven

3. Git

4. Set up various software environment

Build Triggers Methods:

**GIT ------- GIT HOOKS/ WEB HOOKS**

**GIT<-------POLL SCM**

**Pool SCM: Where Jenkins look for updates**

[minutes] [hours] [day of the month] [month] [day of the week]

**\* \* \* \* \* command to be executed**

**- - - - -**

**| | | | |**

**| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)**

**| | | ------- Month (1 - 12)**

**| | --------- Day of month (1 - 31)**

**| ----------- Hour (0 - 23)**

**------------- Minute (0 - 59)**

* The asterisk (\*) : This operator specifies all possible values for a field. For example, an asterisk in the hour time field would be equivalent to every hour or an asterisk in the month field would be equivalent to every month.
* The comma (,) : This operator specifies a list of values, for example: “1,5,10,15,20, 25”.
* The dash (-) : This operator specifies a range of values, for example: “5-15” days , which is equivalent to typing “5,6,7,8,9,….,13,14,15” using the comma operator.
* The separator (/) : This operator specifies a step value, for example: “0-23/” can be used in the hours field to specify command execution every other hour. Steps are also permitted after an asterisk, so if you want to say every two hours, just use \*/2.

(i) Access Control which includes authenticating users and giving them an appropriate set of permissions, which can be done in 2 ways.

Security Realm determines a user or a group of users with their passwords.

Authorization Strategy defines what should be accessible to which user. In this case, there might be different types of security based on the permissions granted to the user such as Quick and simple security with easy setup, Standard security setup, Apache front-end security, etc.

**Matrix-based security:**

Step 1. Go to Jenkins Dashboard -> Manage Jenkins -> Configure Global Security ->

Security realm : Select Allow users to sign up (under Jenkins own user database).

If account was locked and showing tAccess Denied <user> is missing the Overall/Read permission then

Open terminal and goto

1.Edit config.xml file, e.g.

sudo vi /var/lib/jenkins/config.xml

2. Change useSecurity element's value to false, e.g.

<useSecurity>false</useSecurity>

3. Remove authorizationStrategy block

4. Restart Jenkins: /etc/init.d/jenkins restart.

(ii) Protecting Jenkins users from outside threats.

-> Upload Plugin file extension

Two plugin formats

.hpi = hudson plugin interface

.jpi = jenkin plugin interface(new)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **Adding Node to Jenkins** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Jenkins Master & Node communication happens via SSH (Linux) via JNLP (Windows)

2. Jenkins Master to any other Linux Node

**Non Password communication(Key Base Authentication)**

-> Jenkins Master we have to generate key on jenkins user.(Private & Public)

-> Copy that key in the nodes for communication (Public Key $ ssh-copy-id)

-> Jenkins Master to any Windows Machine we will download a Jenkins Slave .jar(From Global Security) and install on windows Node for accessing.

-> For Nodes to work with master we have to install Java on every node

**Backup of Jenkins:**

Two methods

**1. CLI by just storing the back up of /var/lib/jenkins**

**2. Install ThinBackup plugin and conf the location**

**What are the ways to configure Jenkins node agent to communicate with Jenkins master?**

Answer: There are 2 ways to start the node agent –

**Browser** – if Jenkins node agent is launched from a browser, a JNLP (Java Web Start) file is downloaded. This file launches a new process on the client machine to run jobs.

**Command line –** to start the node agent using the command line, the client needs the executable agent.jar file. When this file is run, it launches a process on the client to communicate with the Jenkins master to run build jobs.

**Configure the default port of the Jenkins build server**

The default port number can be changed in the config file at

$ ***sudo vim /etc/default/jenkins*** or ***sudo vim /etc/sysconfig/jenkins*** ---> JENKINS\_PORT=”8123"

Here you can set a different port number, e.g. HTTP\_PORT=8082

Now you need to restart Jenkins with

$ ***service jenkins restart*** or ***sudo systemctl restart jenkins***

or by adding /restart at the end of the Jenkins URL, e.g. https://yourjenkinsurl/ci/restart.

**Declarative versus Scripted Pipeline syntax:**

A **Jenkinsfile** can be written using two types of syntax - Declarative and Scripted.

Declarative and Scripted Pipelines are constructed fundamentally differently. Declarative Pipeline is a more recent feature of Jenkins Pipeline which:

**Pipeline :** A Pipeline is a user-defined model of a CD pipeline. A Pipeline’s code defines your entire build process, which typically includes stages for building an application, testing it and then delivering it.

Also, a pipeline block is a key part of Declarative Pipeline syntax.

**Node:** A node is a machine which is part of the Jenkins environment and is capable of executing a Pipeline.

Also, a node block is a key part of Scripted Pipeline syntax.

**Stage:** A stage block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" stages), which is used by many plugins to visualize or present Jenkins Pipeline status/progress. [6]

**Step:** A single task. Fundamentally, a step tells Jenkins what to do at a particular point in time (or "step" in the process). For example, to execute the shell command make use the sh step: sh 'make'. When a plugin extends the Pipeline DSL, [1] that typically means the plugin has implemented a new step.

pipeline{

agent any

stages{

stage('Checkout'){

steps{

git 'https://github.com/devops-trainer/DevOpsClassCodes.git'

}

}

stage('Compile'){

steps{

sh 'mvn compile'

}

}

stage('Test'){

steps{

sh 'mvn test'

}

post{

always{

junit 'target/surefire-reports/\*.xml'

}

}

}

stage('Package'){

steps{

sh 'mvn package'

sh 'mvn pmd:pmd'

}

}

}

}

Difference between Continuous Integration, Continuous Delivery, and Continuous Deployment:

CI

CodeBuild --AUTO--> Unit Testing --AUTO--> Staging/Integration --AUTO--> Acceptance Testing

latest copy of the source code at a commonly shared hub where all the developers can check to fetch out the latest change in order to avoid conflict.

CD Delivery

CodeBuild --AUTO--> Unit Testing --AUTO--> Staging/Integration --AUTO--> Acceptance Testing --MANUAL--> Production

Manual Deployment to Production. It does not involve every change to be deployed

Continuous Deployment:

CodeBuild --AUTO--> Unit Testing --AUTO--> Staging/Integration --AUTO--> Acceptance Testing --AUTO--> Production

Automated Deployment to Production. Involves every change to be deployed automatically

Continuous Delivery Workflow

Git Clone --> Compile --> Unit Test --> Packaging (war) --> Deploy to PROD

**Define the process of Jenkins.**

First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.

Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.

If the build fails, then the concerned team will be notified.

If the build is successful, then Jenkins deploys the build in the test server.

After testing, Jenkins generates feedback and then notifies the developers about the build and test results.

It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.

**Most popular environment variables**

$JOB\_NAME: Name of the project of this build. This is the name you gave your job.

$BUILD\_NUMBER: The current build number

$WORKSPACE: The absolute path of the workspace.

$NODE\_NAME

$BUILD\_URL

**Move a file from one server to the other:**

Simply copy the job directory and paste it in the other server. Before that ssh keygen copy and sshd config set passwords authen true

Goto var/lib/jenkins/jobs

$ scp -r <jobfile>/ ec2@<machine ip>:<dest loc>

$ scp -r raj1/ ec2@123.17.125:/root/jenkins/jobs/

Builds failing with OutOfMemoryErrors:

1.java.lang.OutOfMemoryError: Heap space

When you see this, you need to increase the maximum heap space. You can do this by adding JVM arguments -Xmx200m where you replace the number 200 with the new heap size in megabytes.

2. java.lang.OutOfMemoryError: PermGen space.

you need to increase the maximum Permanent Generation space, do this by adding the following to your JVM arguments -XX:MaxPermSize=128m where you replace the number 128 with the new PermGen size in megabytes.

3. Maven2/3 project type, you can set the -Xmx or -XX:MaxPermSize in your Jenkins global configuration.Maven2/3 project type, you can set the -Xmx or -XX:MaxPermSize in your Jenkins global configuration.

Name: MAVEN\_OPTS

Value = -Xmx500m -XX:MaxPermSize=500m

Did you automate anything in your project? Please explain

**What is input directive?**

The input directive on a stage allows you to prompt for input, using the input step.If the input is approved, the stage will then continue. Any parameters provided as part of the input submission will be available in the environment for the rest of the stage.

**Jenkins Cloud Bees**

Ans: CloudBees Core is a centralized management solution that controls Jenkins Masters providing scalable security, compliance, and efficiency of Jenkins in Enterprises.

**Promoting a build means**

Keeping multiple ptojects in downstream and upstream. Here CI is done trigger CD. For this process we used promote build to deploy to production and run production release job

I have different Jenkins pipeline jobs for Build, Deploy, QA Automation tests and Release to Production. I am building a Jenkins Delivery pipeline to run these jobs in a single pipeline. The pipeline should be like once QA certification is done, a build job should be triggered for approval and then Release preparation jobs can run further. Flow should be like:

Run the Build, Junit and SonarQube Analysis in single pipeline job using Jenkinsfile on Develop branch

Run the QA Deployment job after the above job is stable or success.

Run the QA Automation job on QA environment

Here, I want a promote a build for approval before start the Release process on Release branch

Once the promotion is approved, release versioning/tagging job is triggered

Run the Production release job.

(OR)

For this we uses promoted builds plugin "General" section, click on "Promote builds when"

name and select a star and "Promote immediately when build is " or "Manually Approve" and built the job.

**Upstream and down stream**

The upstream job is the one that is triggered before the actual job is triggered. The downstream job is the one that is triggered after the actual job is triggered. We can configure the actual job not to be triggered if the upstream job is failed.

**Jenkins Directory structure**

/var/lib/jenkins or /var/lib/jenkins\_home

JENKINS\_HOME

+- config.xml (jenkins root configuration)

+- \*.xml (other site-wide configuration files)

+- userContent (files in this directory will be served under your http://server/userContent/)

+- fingerprints (stores fingerprint records)

+- nodes (slave configurations)

+- plugins (stores plugins)

+- secrets (secretes needed when migrating credentials to other servers)

+- workspace (working directory for the version control system)

+- [JOBNAME] (sub directory for each job)

+- jobs

+- [JOBNAME] (sub directory for each job)

+- config.xml (job configuration file)

+- latest (symbolic link to the last successful build)

+- builds

+- [BUILD\_ID] (for each build)

+- build.xml (build result summary)

+- log (log file)

+- changelog.xml (change log)

Jenkins Plugin Strategies

Workf flow lib in jenkins

**Security Structure in Jenkins**

1. **Access Control**: users are authenticated and appropriate set of permissions are given to them.

Security Realm and Authorization Strategy

Login to Jenkins with your credentials.

Go to "Manage Jenkins".

Go to "Configure Global Security".

In the option "Authorization" select "Project-based Matrix Authorization Strategy" and put the privileges as appropriate. Permissions are given to these topics : Overall, Slave, Job, Run, View, SCM.

2. **Protect users of Jenkins from other threats**

Slave To Master Access Control, Securing JENKINS\_HOME

Disabling Security:

Stop Jenkins (the easiest way to do this is to stop the servlet container.)

Go to $JENKINS\_HOME in the file system and find config.xml file.

Open this file in the editor.

Look for the <useSecurity>true</useSecurity> element in this file.

Replace true with false

Remove the elements authorizationStrategy and securityRealm

Start Jenkins

When Jenkins comes back, it will be in an unsecured mode where everyone gets full access to the system.

**Change default Port of jenkins**

The default port number can be changed in the config file at

$ sudo vim /etc/default/jenkins or sudo vim /etc/sysconfig/jenkins ---> JENKINS\_PORT=”8123"

Here you can set a different port number, e.g. HTTP\_PORT=8082

Now you need to restart Jenkins with

$ service jenkins restart or sudo systemctl restart jenkins

or by adding /restart at the end of the Jenkins URL, e.g. https://yourjenkinsurl/ci/restart.

Continuous Integration vs Continuous Delivery vs Continuous Deployment flow

How many servers you are using --🡪 around 10-15

**How to Setup Webhooks method**

GitHub webhooks in Jenkins are used to trigger the build whenever a developer commits something to the master branch.

GIT HUB: Goto your project repository --> "settings" ---> "webhooks." ---> "Add webhooks" add Payload URL from jenkins.

Jenkins: Manage Jenkins ---> Configure System ---> GitHub Pull Requests ---> Published Jenkins URL and add repo link

Now go to the Jenkins pipeline and select "GitHub hook trigger for GITScm polling."

**Post-build Actions in Jenkins**

Steps that are executed after build are called postbuild actions

Execute Windows batch command, Execute shell, Invoke Ant, Invoke Gradle script, Invoke top-level Maven targets, E-mail Notification, Delete workspace when build is done

Archive the artifacts, Build other projects

**How To Send Email Notification In Jenkins?**

Using Email Extension Plugin – This plugin lets you configure every aspect of email notifications.

Using Default Email Notifier – This comes with Jenkins by default. It has a default message consisting of a build number and status.

Manage Jenkins-> Configure System --> email notification section

using Gmail then type smtp.gmail.com for the SMTP server.

Use SMTP authentication --> mail username and password --> SSL option --> port number as 465.

post {

always {

emailext body: 'A Test EMail', recipientProviders: [[$class: 'DevelopersRecipientProvider'], [$class: 'RequesterRecipientProvider']], subject: 'Test'

}

**How can I access Jenkins environment variables in workspace?**

Manage Jenkins -> Configure System -> Global Properties -> Environment Variables -> Add

BUILD\_ID, BUILD\_NUMBER, BUILD\_TAG, JENKINS\_URL, NODE\_NAME

Ex: echo "${env.JENKINS\_URL}" or echo ${BUILD\_NUMBER

**Jenkins Memory Issue. So i want to save last 5 five builds. How?**

Discard old build: Set -Days to Keep Build, Max Builds to Keep. Max it displays 30. To see all builds goto /var/lib/jenkins/jobs/myjob/builds

**How to run Docker in jenkins pipeline**

Ans: As Agent Docker

**In maven project if containers are deleted every time it has to fetch the data. Which is time consuming. How to over come?**

Using Vol. Agent as Docker and specify a vol by mapping local M2 to container Vol. If i can Local M2. with Container M2 I can overcome.

**MultiBrach Pipeline**: SCM repository in which we have three branches these are Master, Develop and Feature. Master branch have complete CI/CD pipeline script, Develop branch have Only CI pipeline script and feature branch have online unit test pipeline script.

To perform complete CI/CD pipeline for the master branch and Only CI pipeline for the develop branch. You can do this with the help of multibranch pipeline project.

**Commit Based Job in Jenkins? What configurations you need to do in jenkins and github**

Poll SCM cron tab and Triggering a build using Git hooks

**How to set jenkins build to fail based specific word in console output?**

Text-finder plugin .\*(?i)failed.\*|.\*(?i)error.\* would find the words "failed" or "error" in a case insensitive manner. This will cause any build which outputs "failed" or "error" to fail in Jenkins.

**Active and reactive parameters (Dynamic Parameterization) in jenkins**

Jenkins build need to have inputs like check-boxes, radio buttons, multi-select values, etc. So to achieve this use case there is a plugin called Active Choice Plugin

**How to customize the build number to something else in jenkins job page**

Manage Jenkins -> Script Console by writing updateNextBuildNumber(1) or Next Build Number Plugin

**What is shared library in jenkins**

Jenkins Shared Libraries are written in Groovy and allow you to create common sets of logic, and share that among teams/projects/organizations.

Here you can simply load a library in to Jenkins and every pipeline job on that Jenkins master has access to that shared library.

The pipeline code is declared in global variable. When we call the variable entire code is runned and deployed.

jenkins-shared-library

vars ----global shared library code with .groovy and .txt

src ---- regular java source directory, added to the classpath during every script compilation.

resources -----non-groovy files like Json,

**How to Clear Jenkins build history**

ANS:1. GUI--- Goto build history and click on delete build.

2. Login to Jenkins command-line.

$ ***rm -rf jobs/${JOB\_NAME}/builds/${BUILD\_NUM}\****

(or)

$ ***ls jobs/jobname/builds***

$ ***rm -rf jobs/jobname/builds/97\* jobs/jobname/builds/13\****

Note: After removing jobs or builds from file system you need to go to Manage Jenkins in the Jenkins GUI and click Reload Configuration from Disk.

Note: The Jenkins home directory might be in /var/jenkins\_home or /var/lib/jenkins or another location.

**Integrate Jenkins with Github (Private repo):**

1. Generate your ssh keys using this blog by Github.

2. Adding generated keys to the private Github repository created before.

Go to your private Github repo -> Settings -> Deploy keys.

3. Jenkins configuration to access private repo:

Go to Jenkins dashboard -> Credentials -> System -> Global credentials -> Add credentials.

Add ssh username with a private key

4. Jenkins job setup: New item -> Freestyle Project. After this go to source code management -> git. Copy the ssh URL of your Github repo. As you have used ssh key you need to use ssh URL instead of https URL.

5. Add the credentials you created before to the credentials section in source code management under the above Repository URL

1. How to create multiple free style jobs with different names with same job configurations.
2. Can we have different nodes for each stage
3. Types of build triggers and diff b/w Build Periodically and Poll SCM
4. Create and deletion of containers will be taken care by jenkins or You
5. In jenkins once the build is success what will happen to that container.