Jenkins

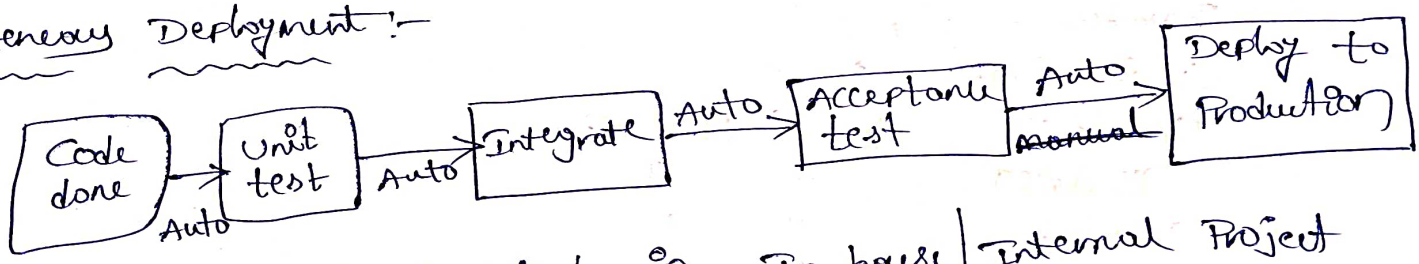
It is an opensource Continuous integration / Continuous deployment (CI/CD) automation software tool & written in Java Programing language. It supports multiple Programing languages.

Sonarqube also Java based tool but it's supports different Programing languages. maven & tomcat also Java based tool & it's supports only Java based Projects not other languages. In Jenkins we have 1000+ Plugins.

Continuous Integration :- CI is Process to automating build, test the Code, everytime a developer Push the Code.

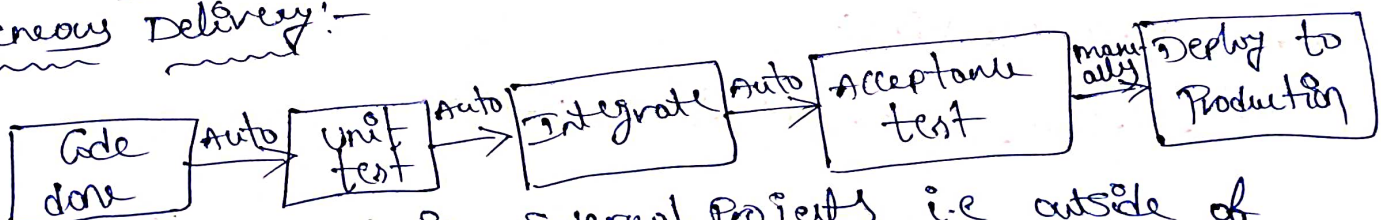
CI = C testing + C build

Continuous Deployment :-



this deployment is used in In house / Internal Project that means within organization.

Continuous Delivery :-



this CD is used in External Projects i.e outside of organization. if you want Deploy in Production you need

approval from the client. most of the deploy to Production on weekends.
Ex:- walmart | flipkart

Jenkins Can do:-

↳ Integrate with different version control systems (Github, CVS, SVN, TFS...)
↳ Generate test reports [Jacoco Plugin used]
↳ Push artifacts to various repositories [Nexus, JFrog]
↳ Notify stakeholders of build status (through Email)

Jenkins — Community Edition → if you getting any issue Community Edition not responding immediately.
Cloudees Jenkins — Enterprise Edition → if you getting any issue there are responding immediately here.

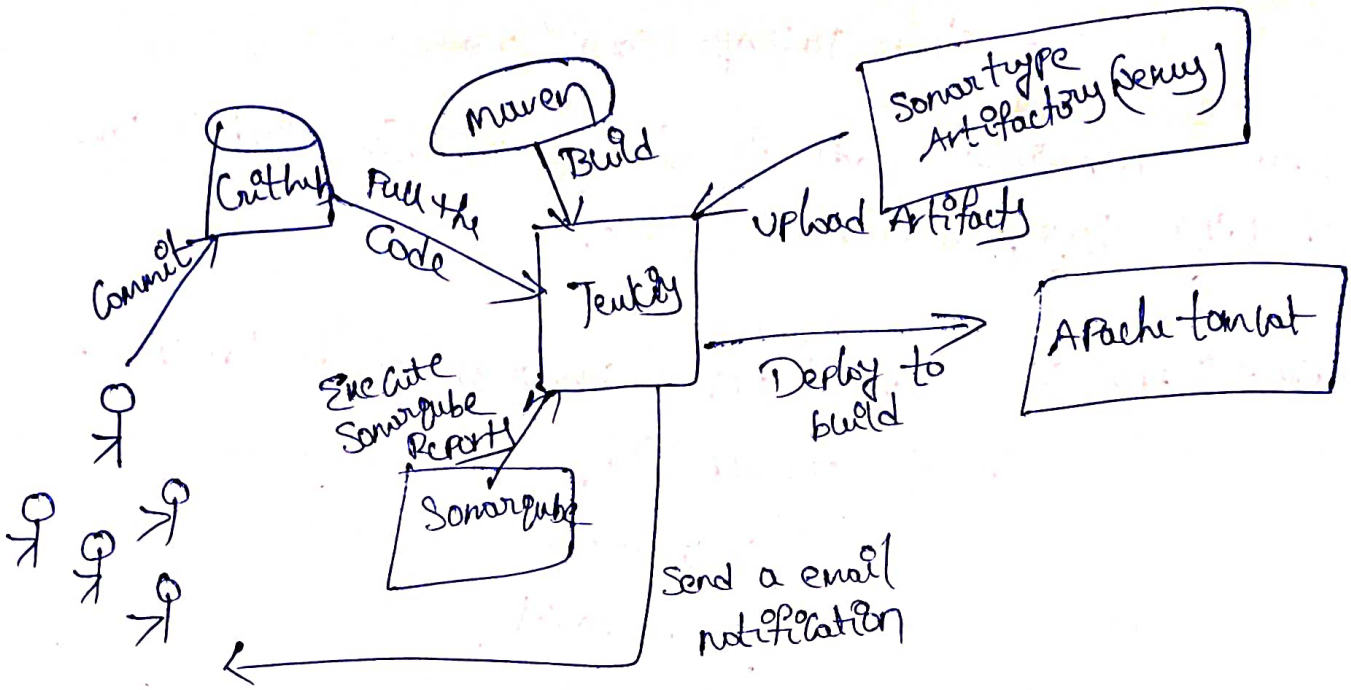
Product	opensource?
Bamboo	NO
Cruise Control	yes
Travis CI	yes & Paid also
Circle CI	" "
Gitlab	" "
Teamcity	" "

JIRA, Bamboo, Bitbucket → Atlassian Products

Installing Jenkins:-

↳ through Ubuntu, Linux, Redhat

6.



free style Job :- You can create Job for any of language,

if you install maven in Linux Server only one version is coming but when you install in Global tool Configuration we have chance to install diff version. it's better way also. you can choose any of one version while creation of Job.

i) Git clone :- give URL & Credentials of Github & branch name.
Install git in Linux Server.

ii) Maven Package :- Install maven using Global tool Config. Select 'Invoke top level maven'.
Goal - clean Package
Package path → /var/lib/Jenkins/workspace/target/*.jar

iii) Sonarqube test :- Create one Sonarqube Server in another Instance.
Install 'Sonarqube Scanner plugin' & Configure details in Configuration.
Create one token → Administrator → Security → Give name & Generate here.
→ Edit Pom.xml file enter 'Sonarqube Server IP' & 'token' under the Properties.

Goals - clean package sonar:sonar

here application code is Coming in Sonarqube Server. checking bugs, vunarabilities .. etc.

iv) nexus Repos :- Create (install) nexus server on one instance. here Create one repository.

↳ snapshot
↳ releases.
↳ .. etc

these snapshot & releases URLs Paste in Pom.xml file. & enter nexus username, Password in settings.xml file.

Cd /var/lib/Jenkins/tools/maven3.6.2

↓

ls (bin, boot, Conf)

↓

Cd Conf

↓

ls (logging, settings.xml)

↓

vi settings.xml

↓

Search here "<servers>"

↓

here enter username, Passwd with in <servers> tag.

```
<servers>
  <id> nexus </id>
  <username> admin </username>
  <password> admin </password>
  </servers>
```

Goal – clean sonar:sonar ~~deploy~~

here uploaded the artifact in nexus Repository.

v) Deploy into Artifact: first install 'Deploy to Container' Plugin.

Create & install tomcat Server in one instance.

under the Post build Action :-

Select "Deploy war/ear to a Container"

↓

war/EAR file = **/maven-web-apps.war

Context Path = "Give .war file Path if in Customise Path" & leave it.

Credentials = "Add here"

Tomcat URL = PublicIP:Port

Give Permission in tomcat Server along with

Roles { manager-gui
admin-gui

need Add this Roles { manager-script

So login to tomcat Server,

↓

cd /opt/tomcat9/Conf

↓

ls [server.xml, web.xml, tomcat-users.xml]

↓

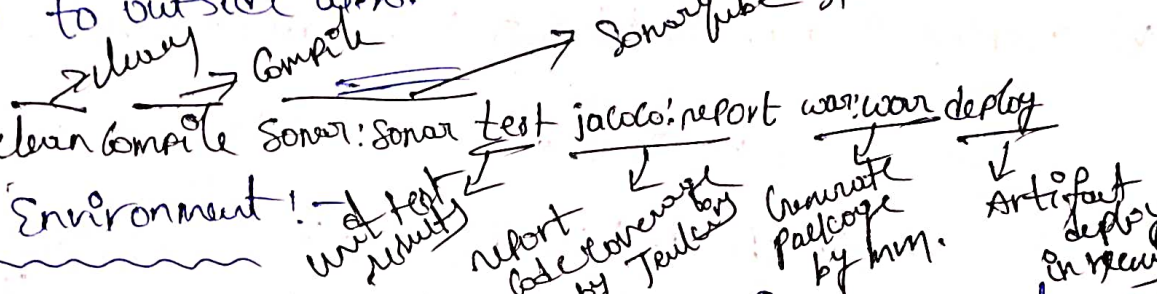
vi tomcat-users.xml

↓

under the <users> tag, Add this 'manager-script' Role.

~~Goal – clean sonar:sonar deploy~~

Here application is deployed in tomcat. if you want to see then goto 'managed apps' & checket. this application can access to outside also.



✓ Add timestamps to Console output → this Gives a Console output after building done along with time.

✓ Delete workspace before build starts → if you select this workspace is deletes before new build starts.

✓ Discard old builds → if you want to keep only last '5' builds or more. Choose this option & Give number here only maintain last '5' build.

Max # of builds to keep = 5

" Artifacts = 5

"

Poll SCM!- only if change is done in Source Code management then we are going see in build by history [Poll SCM triggers] based on Crone expression. polling by

But ~~Go~~ Poll SCM Everytime run based on Crone Expression.

Build Periodically!- Based on Crone expression it's Every time runs. But if changes done or not in SCM then we are going to see in build by history.

* * * * *

min Hours Day of Month Month Day of week

Webhook:- Goto Projects → select settings → select webhooks → select "Jenkins URL" → select Push event:
whatever code changes by developers & pushed to SCM then webhook will intimates to Jenkins & the Jenkins will takes the code from SCM.

Jenkins URL [http://180.90.170.90/github-webhook/]

✓ Disable Project:- if any servers are (nexus, sonarqube) Scheduled maintanance then we are going to use this option because if 'build now' clicking then build going to fail due maintanance of Servers.

✓ JACOCO Plugin:- After installing this plugin, Goto Post build action → Jacoco Coverage report → Enter coverage value

☑ change build status according the threshold

Instruction	% Branch	% Complexity	% Line	% method
80	80	80	80	80
80	80	80	80	80

☑ fail the build Coverage degrades more then delta threshold.

80	80	80	80	80

↓

this plugin is used for generate the reports. if code coverage not more then 80%. then Consider as build failure & stops the deployment but Sonarqube will not stops the deployment.

Configure Email functionality :- Jenkins Sends Notification to Email.

Goto manage Jenkins → Configure Systems →

Extend Email notification ⇒ Enable here (or)

Email notification ⇒ Enable here.

while Creation of Job, under the Post Build Actions

↓

Select 'Email notification' & Extend Email notification'

& give Configuration for sending the notification.

Jenkins Directory Structure :-

↳ Home directory of Jenkins, /var/lib/Jenkins

↓

ls (logs, nodes, Jobs, tools, secrets, users, workspace, secret, updates)

↳ for seeing builds for Jobs,

/var/lib/Jenkins/Jobs

↓

ls (Jobs shows)

↓

cd 'Job directory'

↓

ls (builds, config.xml, nextBuildnumber, SCM-Polling.log)

Job Configuration all we have in this file.

↓

cd builds

↓

all builds numbers shows here & Goto build & Check logs.

↳ for seeing Job file for particular Job,

/var/lib/Jenkins/workspace

↓

ls (Jobs shows here).

↳ user information do Jenkins shows,

/var/lib/Jenkins/users

↓

ls (files shows)

↓

Cat 'userfile' (user.xml)

↳ when we are installing Jenkins server the token can Generates that is available in 'secrets' folder,

/var/lib/Jenkins/secrets

↓

ls (shows here)

↳ when we are installing any tools using Global tool Config that all information stores in 'tool' directory.

/var/lib/Jenkins/tools

↳ whatever updates will having that information will available in 'updates' directory

/var/lib/Jenkins/updates

↳ all the master & worker nodes information will available in,

/var/lib/Jenkins/nodes

↳ Jenkins logs checking if anything happens with Jenkins,

/var/log/Jenkins/Jenkins.log

Maven Project:- first initially create a one plugin & install it

↓
maven integration

Similar into the free style Project but have more features we have & its only supports to the Java based Projects.

i) Git clone
ii) mvn Package
iii) Sonarqube test
iv) nexus Repos
v) Deploy into Artifact

Under the Post build Action Select Email notification & Give Configuration.

Plugin managment:-

↳ Deploy to Container → used for deploying the application into tomcat & JBOS ..etc

↳ Deploy weblogic & Deploy websphere → used for deploy into weblogic server or Deploy websphere.

↳ maven integration → for seeing maven Project while Creation of maven Job.

↳ SafeRestart → used for restart the Jenkins Server after all running Jobs are finished.

http://JenkinsURL:8080/restart → for running Jobs
http://JenkinsURL:8080/safeRestart → Stops when restart of Jenkins

↳ next Build number → this plugin used for if you want to run specific build number (50, 70 anything) then directly we have option after installing of this plugin
under Jobs, Set next build number → 50 & 70

↳ Email Extension → this plugin used for Send the notification under the Post-build Actions while creation of Job.

↳ SSH Agent →

↳ Sonarqube Server → if you dont have POM.xml then this plugin used to integrate the Sonarqube Server.

* After that under Post build Action, Clicke sonarqube Server & Give info. later Execute marts in serve. Credentials here.

Adding Sonarqube Credentials in Jenkins Credentials. under the 'Configure Systems' Integrate.

↳ Audit Trail Plugin → After Installing this plugin, GoTO

Configure Systems → Audit trail

↓

log location = /var/lib/Jenkins/Audit.log
log file size MB = 10
log file Count = 5

↓

5 files creates Each one is 10MB in that Particular path.

this plugin used for who perform the Jenkins operations about all info. GoTO that path & check once.

tail -f Audit.log

↓

here shows all info. about Job

↳ Schedule Build → used for we want to build the Job at Particular time then Go for this plugin.

↳ Artifactory plugin:→ if you want to integrate with nexus & Jfrog then Go for this plugin & install it & Give add Credentials in Jenkins Credentials.

↳ thin Backup → used for Backup the Jenkins server.

↳ Build name & Description Setter → This plugin is going to set the Build name with number. Install this plugin.

GoTO Configure → under the build environments →

☑ Set Buildname

Buil name = dev-#${BUILD_NUMBER}

↓

start 'Build now' then here shows name with build number.

↳ Blue ocean → this is Give separate Dashboard for creation of Job After installing of this plugin

↳ Convert To Pipeline → used for Convert the Job from free-style Project to pipeline Project.

NOTE:-

↳ if you forgot Passwd the Jenkins Password then Password can easily recovery in,

/var/lib/Jenkins/config.xml

↓

<useSecurity> true/false </useSecurity>

↓

restart the Jenkins.

one Password okay then keep 'true' & restart Again.

↳ if you want to change port number, Jenkins home path, Jenkins user name you can go & Edit here,

~~/var~~ /etc/default/Jenkins

↓

Jenkins file change here & restart it.

✓ Build with Parameters → while Creation Job select this →

Add parameter (choice, StringParameters.. etc)

ChoiceParameter: name: BranchName
choices: Master
UT
Dev
Description:

StringParameter:-
name: name
Defaultvalue: vasu

Under SCM: "select Git URL"

Branch: /${BranchName}

this option is used for instead of creating '3' Jobs for 3 Branches, you can Create one Job & within this Job you Create & running '3' environments.

→ Creat new → this option used for in Jenkins Server you have 200+ Jobs. In this 50 ERP & 150 Jobs Amazon. for Sortlisting Particular domain Jobs this is very useful.

Jenkins Dashboard → [+U] + → click & Create Flipkart & Amazon Domain

↓

Add here whatever Job related to flipkart & amazon.

Jenkins user Security :-

[Create user] → manage Jenkins → manage Jenkins user & nodes

→ Create user.

Here by default user have Administrative access you Give necessary Permissions to users.

(or)

you can create users through LDAP server also after integrating the LDAP server to Jenkins server.

↳ Give necessary Permissions to user, GoTo

Configure Global Security

↓

Authentication (whether knows to system or not)

Security Realm

- ☑ Jenkins' own user database
- ☐ Allow users to sign up
- ☐ LDAP
- ☐ Unix user/group database

Authorization (execution is Performing)

- ☐ Anyone can do anything
- ☐ Logged-in users can do anything
- ☑ Project-based matrix Authorization Strategy
- ☐ Role based Strategy (if you install Plugin)

Afterwards, Add users & give necessary Permissions to particular users. if you want to check GoTo & verify also.

↳ After giving Permissions to users, Every user able to see all Jobs. But my requirement only specific one user only see 'flipkart Job' & another specific user only & see 'maven Jobs'. for this while creation of Job (flipkart & maven). select & add user.

☑ Enable Project based Security

↓

Add user & Give Permissions also.

↓

if you want to check then GOTO & login to the user & check it.

Build Process of .Jar files Stores in S3 & Jenkins access to S3:-

↳ 1st Create a one bucket in S3

↳ this Role Can attach to Jenkins Server Instance

↳ In Jenkins Server Install plugin (S3 Publisher).

↳ Choose IAM Role in amazon S3 Profile in Configuration systems.

GOTO Jenkins Server → manage Jenkins → Configure system → choose Role in Amazon S3 Profile

↳ Select "Publish artifacts to S3 bucket" option in Post build action while Creation of Job in Jenkins Server But here give source & destination & Region.

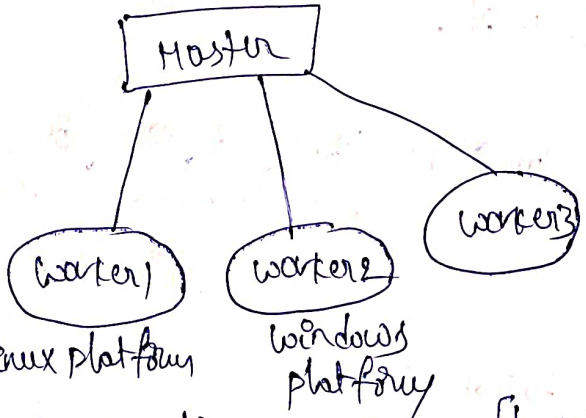
source → target/*.jar

Destination → 'Bucketname/dir'

↓

now Build the Job (periodically or pollSCM) whatever .jar file is generated on Every build, that will stored in Amazon S3. if you want to check then GOTO S3 in Console & see folder & check here.

Jenkins master-slave Architecture:- the Jenkins master act as schedule the Jobs, assign slaves & send builds to slaves to execute the Jobs.



- STEP(1):- first create one instance(t2.micro) & give name as "worker". here whatever u need those all softwares install (Java, mvn, git, docker.io) except Jenkins.

STEP(2):- Connect server through Putty & create one directory.

mkdir worker

↓

Pwd { /home/ubuntu/worker
/root/work }

STEP(3):- GOTO Jenkins Server & Create a New node in manage Jenkins.

Select newnode

↓

Give (name, Description, slavename)

↓

Give home directory Path or Remote directory
(/home/ubuntu/worker)

Select "via ssh through username & key"

[Give Credentials username, Privat Pemkey]

Host = Pvt IP

Launch method :- Select "manually trusted verification strategy"

usage :- select "as soon as possible"

↓

Save & apply

Initially node be in "x" state after some time it is coming to "✓" state.

Step(9):- Create one Job with free style Project or Pipeline.

↓

Select option "Restrict where this Project can run" & Select node (worker) here.

(or)

But in Pipeline Project Edit Jenkins file

Agent "label 'worker'" instead of "any" then Job is going to be run in Particular node instance.

↓

Save & apply

↓

Build now

↓

GOTO Server(worker) & check the Jobs are coming or not & see .Jar file in target folder

↓

if you want to run "Java -Jar target/*.Jar".

Build manually Java application without Jenkins:-

first create Instance & Connect to Putty.

Sudo su -

↓

APt update

↓

Crive (keys & Binarys)

↓

Sudo apt-get update

↓

Sudo apt install openjdk-11-jdk

↓

apt-get update

↓

Sudo apt-get install git & mvn & docker.io

↓

→ Sudo usermod -aG docker Jenkins
→ Chmod 777 /var/run/docker.sock
→ Docker pipeline plugin install

(git --version)
(mvn --version)

if you want to build through Automation then install Jenkins & check status.

↓

git clone "springpet"

↓

ls (shows ~~all files of~~ springpet)

↓

Cd springpet

↓

ls (all files of springpet)

initially target folder not having

↓

for creating .Jar file → ./mvnw package

↓

ls (shows target folder)

↓

cd target

↓

ls -al [shows .Jar file]

↓

if you want to run → Java -Jar target/*.Jar

[Paste URL in & see homepage]

if you want run as container then follow,

↓

ls (shows SpringBoot folder along with Dockerfile)

↓

for creating docker image → docker build -t Spring-app .

Spring-app ↓ imagename

. ↓ current directory of Dockerfile

↓

docker images

↓

run as container → docker run -d -P 9081:9090 Spring-app

(or)

docker run -name mySpring-app -d -P 9081:9090 Spring-app

mySpring-app ↓ Give any name

Spring-app ↓ Imagename

↓

Paste the PublicIPv4:9081 in URL you are getting Home page application.

↓

docker tag spring-app:latest harisettivasu/spring-app:latest

↓

docker images [Shows tags images also]

↓

docker login [Enter username & Passwd of dockerhub]

↓

docker push harisettivasu/spring-app:latest

↓

Refresh the Docker hub account then able to see Docker images of spring-app.

↓

delete Container → docker rm Container-id

delete image → docker rmi imagename & Containerid

↓

for seeing Process of Container → ps -ef | grep docker

↓

for giving history whats happens → docker log Container-id

Pipeline:- Jenkins pipeline is a suite of plugins which supports implementing & integrating continuous delivery pipelines into Jenkins. A definition of Jenkins pipeline is typically written into text file is called as Jenkins file.

- there are two types of pipelines.
  - i) Declarative pipeline
  - ii) Scripted "

these two pipelines are totally diff in their constructions. Dec Declarative pipeline are more recent features of Jenkins pipeline that provides better syntactical features compared to scripted pipeline syntax.

Script:-

```
node
{
def mavenHome = tool name: "maven3.6.2"
```

↓ Give name as per Global tool Config.

Install Git in Server

```
stage('checkoutCode')
{ git URL }
```

Install mvn in Global tool

```
stage ('Build')
{ sh "${mavenHome}/bin/mvn clean Package" }
```

↓ mvn Homedirectory in bin

Enter Sonar qube token & Server IP in Pom.xml

```
stage ('execution SonarQubeReport')
{ sh "${mavenHome}/bin/mvn Sonar:Sonar" }
```

If you don't have Pom.xml file then you need to install 'Sonarqube Server' plugin & configure details in

stage ('upload Artifact into nexus')

{ sh "${mavenHome}/bin/mvn deploy" }

Snapshot & release IP URLs upload in Pom.xml & username, Password update in setting.xml file. In Pom.xml version tag we have so Artifact uploads in direct snapshot & release based on tag of version.

stage ('Deploy into tomcat')

{ Generate Pipeline Script for tomcat Server username & Pemkey using Jenkins Credentials.

{ sh "scp -o StrictHostKeyChecking=no target/maven-web-application.war ec2-usr@PblicIPoftomcat:/opt/apache-tomcat-9.0.39/webapps/" }

when deploy time not asking yes/no

tomcat server username

tomcat webapps Path

}

Permission for 'webapps' directory → chmod -R 777 webapps.

stage ('send notification')

{ Generate Pipeline Script through Email Configuration (or) Email notification in Configure System.

}

Commented.

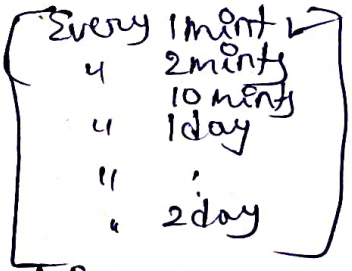
/* —— */

chown -R Jenkins:Jenkins <filename>

Multi Pipeline Branch:- It is the enables you to implement different Jenkinsfiles for different branches of the same object.

if one repository you having a multiple branches (main, stag dev..etc). At this time if you want execute Jobs in all Jobs of branches (4 Jobs) in single time you go for better way is multipipeline Project. or when we go for freestyle Project here you need to create 4 Jobs at 4 times.

while creation of multipipeline Project → Select "git URL", Select Build



based on build time Jenkins will triggery. Suppose Something update is done on 'dev' branch then only Another build starts & execute in 'dev' branch not in all branches.

Jenkins Backup:-

1st install 'thin Backup' plugin & check it

manage Jenkins → See here option 'thin Backup' option

Restore

Configure the here

where the Path Backup= /var/lib/Jenkins/Backup

Back schedule for full backups (*****)

Max no. of backup = 5

Old backup zip ☑ ☑ Backup build results

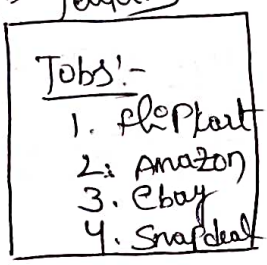
Backup plugin Content ☑ ☑ Backup build Archives

used for restore the data but not given Jobs history data. if you want Job history data you need to Configure 'Job Config history' plugin.

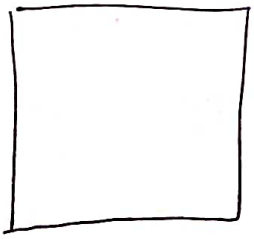
if you not given Crone Schedule then manully click "Restore" option. if you are given Crone Schedule then based on that it has to be given backups. that is also only upto 5 backups.

Jenkins migration (Jobs migrates from one jenkins to other Jenkins Server):-

1st way:- 1st Jenkins



2nd Jenkins



↳ for doing migration, In target server install Jenkins Same version of 1st Jenkins Server.

↳ Install 'Job import' plugin in target Jenkins Server

↳ Add source Jenkins configurations in 'Configuration Systems'.

Job import plugin:-

Name:- Jenkins_Source
URL:- "Jenkins source URL"
Credentials:- "source Jenkins Credentials Add here"

↳ click 'Job import plugin' option in target Jenkins server.

↓

Jenkins Server:- Jenkins_Server
Remote folder:- Empty or Enter if you want Custom path
Search into folder ☑
Query!

↓

Copy of folder = Empty

Install required Plugins ☑

Import?	Disable?	Name
☑		flipkart
☑		amazon
☑		ebay
☑		Snapdeal

Import → click import then all Jobs & Installed Plugins Comes to target Jenkins Server.

2nd way:-

If you have a linux machine access then you can copy the Jobs from Source to target Jenkins Server also through the Command line linux Server.

from source mc:-

↳ scp -r /var/lib/Jenkins/ root@IPAddr:/root/var/lib/Jenkins

- /var/lib/Jenkins/ → from Source path of Jenkins
- IPAddr → destination server IP
- /root/var/lib/Jenkins → path where to copy.

then GoTO destination Jenkins Server & refresh it then all Source Jobs Comes here.

manage Jenkins → Reload Configuration from Disk

Why copy the JIS Config file also to destination.

Jenkins CLI :- It is the command line to interact with the Jenkins Server Dashboard. With the help of CLI you can Create, delete, Build & Configure the Jobs also..etc

In URL: http://180.60.190.87/cli

↓

you able to see JenkinsCLI.Jar file you can Download this & all the CLI Commands of Jenkins & one main command to connecting Jenkins server from Gitbash.

↓

for executing Jenkins CLI Commands, GoTO 'JenkinsCLI.Jar' file path & Execute the main Command,

↳ Java -Jar Jenkins-cli.Jar -auth bavisettivasu:vasu@1431 -s http://Jenkins URL/help

- -auth → authentication
- bavisettivasu → username of Jenkins server
- vasu@1431 → password of Jenkins server (By Instead of this you can enter token by creation of in configure system)
- -s http://Jenkins URL/help → help for seeing all Commands

Scope

↓

Here you can execute a Command for Job related Configuration (Build, delete, create..etc)

1) Git clone → Add Github Creds.

2) mvn → clean Compile Package

3) Static code →

if you don't have Pom.xml

Install 'Sonarqube Server' plugin

↓

Add creds in Configuration Systems (or) Jenkins Creds

↓

In Post build action Give Configuration while creation of Job.

if you ~~don't~~ have Pom.xml

Add token of Sonarqube in Pom.xml

↓

& "ServerIP" & also add in Pom.xml

↓

In Post build action Give Configuration while creation of Job.

4) Nexus Repos →

Install 'Artifactor' plugin

↓

Add creds in Configuration Systems or Jenkins Creds

Add 'Release & Snapshot' URLs in Pom.xml

↓

Add nexus ~~Re~~ Credentials in Setting.xml.

<server> Id => <username> => <password> => </server>

IntelliPaat

master slave architecture in Jenkins:-

1) Create 3 instances in AWS (master, slave-1, slave-2).

2) Install Jenkins in master & open the Jenkins dashboard in URL.

3) Install Java, docker.io in slave-1, slave-2

4) In master → Goto manage Jenkins → Configure Global Security → In Agents select Random.

TCP Ports for JNLP agent ☐ fixed ◉ Random

5) then create nodes for slave-1, slave-2,

→ Goto manage Jenkins → manage nodes → create nodes for slave-1, slave-2

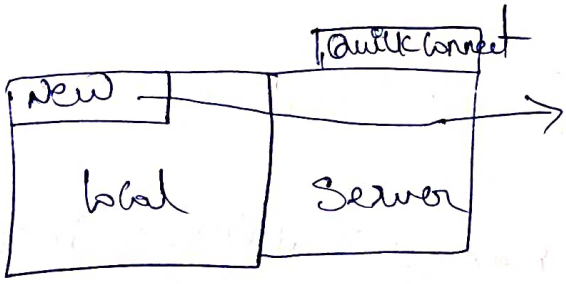
name = slave-1
Description=
Renote root directory = /home/ubuntu
label = slave-1
Launch method = Launch Agent via Java web start
↳ Custom WORKDIRPATH = /home/Ubuntu
↓
Save & apply

Like slave-2 also same.

6) Open the slave-1 node initially (x) position. now download the launch Agent [↓launch], Agent.Jar file to local system.

& we have link also [Java -Jar agent.Jar ....]

7) using filezilla, winscp you can download those files from local to Remote server.



Give username (ubuntu)
Host - Public IPv4 (slave-1)
Port - 22

Goto Edit → SFTP →
Select PPK file → ok
(apt update)

(agent.Jar, slave-agent.JNLP)
& then files (two) drag & drop from local to remote server.

8) Goto slave-1 m/c server & check "ls" two files coming or not. then " here Paste the link & then Connected to this slave-1 to Jenkins Server.

9) Like slave-2 m/c also same setup.

10) & in Jenkins master the nodes is [✓] position.

master node Connection through ssh-keygen:-

1) Create two instances (master, slave-1).

2) Connect to Putty of master node & install [Java, Jenkins].

& in worker node install [Java].

3) In master node, GoTO .ssh → ls (authorized_keys) →

↓

ssh-keygen

↓

cd .ssh

↓

ls

↓

id_rsa(Penkey) id_rsa_Pub (Publickey)

this Public key can Paste into the slave-1 mlc in authorized_keys.

4) In slave-1, GoTO sudo nano authorized_keys

[Ubuntu] → ↓ scroll down

" Paste it Publickey of master"

↓

Save

↓

Sudo su

↓

[root] → sudo nano authorized_keys

↓ scroll down

" Paste it Public key of master"

Save

↓

↓

service sshd restart

5) In master check Connection.

↳ ssh ubuntu@IPv4 Public (slave-1)

↳ you are logged into the slave-1 from master.

↓

Exit (master)

6) In master node,

Go to manage Jenkins → Configure Global Security

→ TCP & JNLP Port make it random ☐

7) Go and Create node for slave-1,

Remote Root directory = /home/ubuntu

label = slave-1

Launch method = launch Agent via ssh.

Host = Pvt IP of slave1

& Credentials (username, Pvt key Enter)

↓

id_rsa of master

↓

Save & apply

then Agent [✓] position. that means Connection is established.

Creating freestyle Project Jobs (Test, Produ) :-

1) Create one free style Job for Test & through Github URL & run in slave-1 m/c (Restrict where this Job can run).

for removing all Containers

↳ sudo docker rm -f $(sudo docker ps -a -q).

↓

check the Job(Test) in slave-1 machine & see home directory Jenkins [/var/lib/Jenkins/workspace].

2) similarly create free style Job for Produ & run in slave-2 m/c.

↓

And check the Job(Produ) & see home directory do Jenkins.

we will be triggered "Produ" Job only when "Test" Job will Complete

1) Go to Test Job & click on Configure

↓

Add Post build Action

↳ Build other Projects

Project name: Produ,

↓

Save & apply

2) Install "Build Pipeline" Plugin without restart.

3) Goto Jenkins dashboard click [+] new view,

↓

Build Pipeline view

↓

Based on upstream & downstream relation ✓

upstream downstream config:

Select initial Job [Test]

↓

Save & apply

Now first initially "Test" Job runs later "Prod" Job runs.