JENKINS

Before C.I:-

Developers push their code into source code repository and sends a mail to build team. Now build team will clone entire repo and using build tools like maven/ant they create executable files like war/ear and deploy manually on to the server like testing, staging, etc... Build team informs to developer if build fails. Now build team will generate email to testing team, if testers tests application and notifies to developer if bug exists. And finally the application is deployed onto production server. It takes so much time to identify which developer code failed the build. Each and every step is manual which delays the product release

But using continuous integration we can automate the entire process. We integrate jenkins with github, maven, tomcat container, email. C.I server polls github at regular interval. Whenever c.i server finds new commit in source code repository it builds the code and do unit test and notifies if build is failed and deploy on the server

Drawbacks:

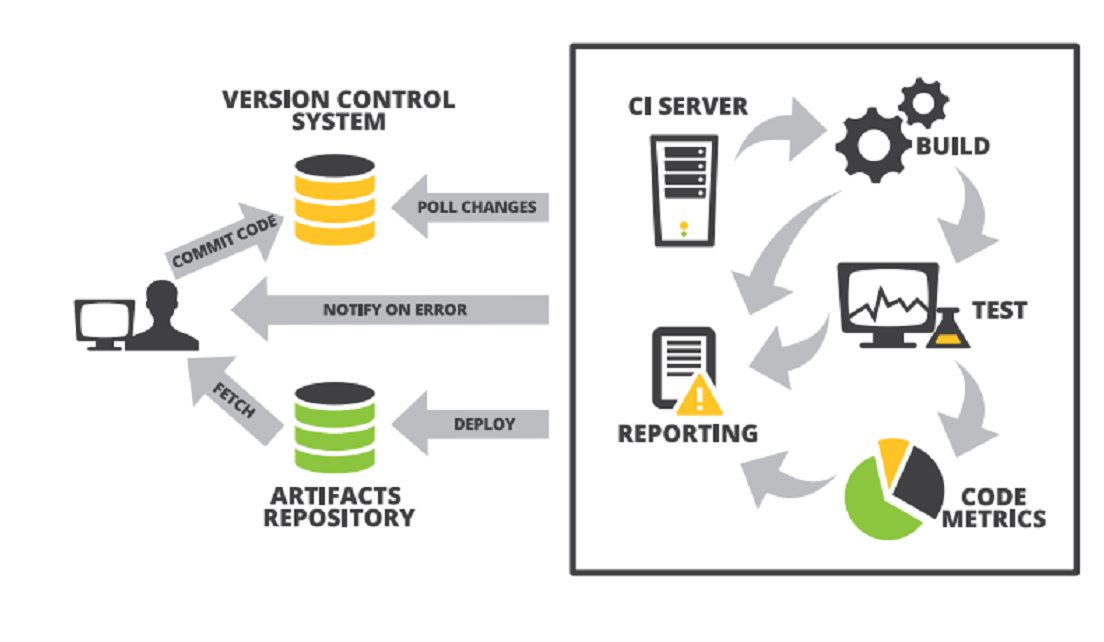
1. No integration
2. Building is manual and time consuming process

Product release lifecycle:

1. Developer develops the code
2. Commits the code into source code repository(Git/Svn)
3. Source code is built(maven)
4. Build application will be deployed on test servers
5. Tester test the application and notifies if any bug exists
6. Else will be deployed in production servers

Drawbacks:

1. Developers has to wait for testers result
2. Developers has to go through entire source code
3. Software delivery is slow
4. No continuous feedback from testing team



Continuous Integration (C.I):

It is a practice where developers are required to push their code into source code repositories several times a day. C.I server will pull that code from scm and does build, test and deploy and notify to developer if the build is failed

C.I servers does

1. checking new version code
2. build new version code
3. running tests
4. deploying application

C.I speeds development process and minimizes the risk of critical issues in production server

Using this we can have complete history of what happened, why it happened and who exactly did this

C.I Benefits:-

1. immediate bug detection
2. a deployable system at any given point time
3. record of evaluation of project

C.I Tools:

1. Jenkins 2. Hudson 3. Bamboo 4. Buildbot

JENKINS:

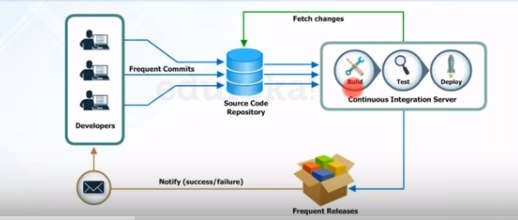
* Jenkins downloads the code from repository, resolve dependencies, builds the code and then deploys it.

* Jenkins is an open source tool with lot of plugins built for continuous integration
* Jenkins merges all code continually in a central build. Jenkins can merge, monitor, regulate, compare and maintain your project aspect

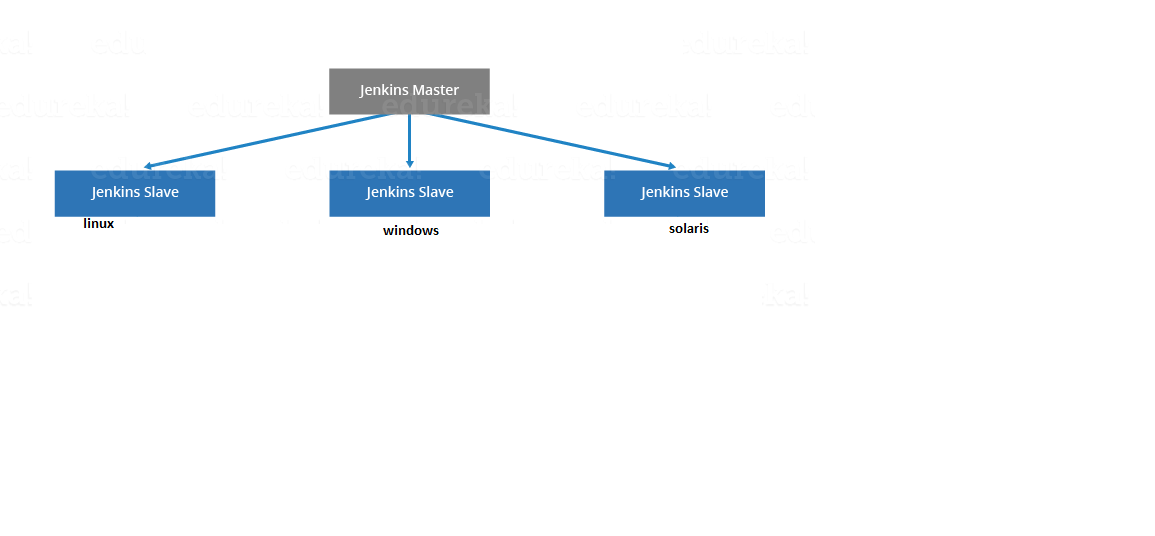
Advantages:

1. Lot of plugins
2. Installation is easier
3. Open source
4. Pipeline support
5. One click deploys
6. Fast
7. Rolling updates

Workflow:-



Architecture:-



In Jenkins we have one master server and many slave nodes. We will try to configure jobs either on master node or slave nodes

In master server after installing Jenkins we give some details like

1. Creating jobs
2. Gives the path on which machine should job run
3. Which plugins to use

Config files:-

1. /etc/sysconfig/jenkins
2. Config.xml saves our modified values
3. /var/lib/jenkins

C.I phases:-

We have three phases in continuous integration

1. Repository:-

Storing all the changes that developer has pushed. Now C.I server will periodically looks for any new commits. If new commit id is found then C.I server will do incremental build. If now new commit id is found, no further action will be taken

1. Integrate:- If any commit id is found then C.I server will integrate current changes, then does integration
2. Build:-

* Compiling code
* Testing code
* Packaging code
* Deploying code
* Code coverage analysis

JENKINS DASH BOARD OPTIONS:-

ITEM:-

1. PROJECT:-

General:-

1. Description:-

give description of the project what you are doing for

1. Discard builds:-

to delete old builds

1. strategy:-

all the old build will be compressed ie…, log rotation

1. days to keep builds:-

how many days should I keep the build, later the builds will be deleted

1. Max # of builds to keep :-

How many max builds should be there, remaining builds will be deleted

1. Github project:-

Enter the github project url

1. The project is parameterized:-

Taking a value from user and starting build. There are string parameter, credentials parameter, Boolean parameter…..

1. Throttle builds:-
2. Disable this project:-

Deleting the jobs/projects

1. Execute concurrent builds if necessary:-

Executing builds simultaneously

scm:-

1. Git:-

This option will be enabled only if you install plugin. This scm is used to integrate Jenkins with git

Similarly you have “svn” option

Build triggers:-

How the Build should trigger like periodically/scripting/remotely….(looks for a change in code)

1. Trigger builds remotely:-

we build job using remote url

1. Build after other projects are built:-

This option is used to build at last

1. Build periodically:-

we are building job periodically using cron job (we would go with this for night builds)

1. Poll scm:-

Whenever developer commits new code then automatically build will be triggered (realtime)

1. Github hook trigger for gitscm polling:-

Build environment:-

There are some build environments like delete workspace before build starts, add timestamp to console output, abort if build takes more time

Build:-

To say how we should we execute build using shell/batch/maven/ant/etc…..

Post-build actions:-

After build is done what next step to be taken like sending email to developer if build is failed, delete workspace when build is done, build other projects, publish junit test result reports, archive artifacts……..

1. CONFIGURE:-

If you want to edit project settings, you can do by simply clicking configuring

1. DELETE PROJECT:-

If you want to delete project for some reason. Choose this option

1. BUILD NOW:-

After creating the project you need to start it by “build now” option.

Click on the build number and see console output

1. CHANGES:-

This option is used if there is change in any of the builds

1. WORKSPACE:-

We have all jobs running in our workspace

/var/lib/jenkins/workspace

BUILD HISTORY:-

Build history shows the graph of failure and success percentage of builds

PEOPLE:-

All known users will be shown here

CREDENTAILS:-

Here all user/system credentials will be shown and also you can setup here

Go to system🡪global credentials🡪add credentials🡪give username, password, id, description🡪click ok

\* you can also add your domains and add domain credentials

VIEWS:-

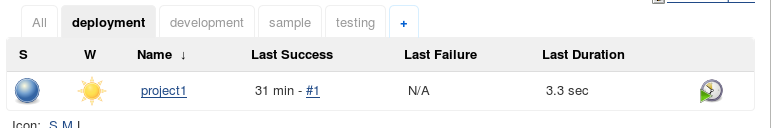
Since it looks messy on the dashboard displaying all projects/jobs. We segregate jobs into categories called as views (or) we group same kind of job and each group will be given as name called view

Ex;- all testing jobs will be put in testing view

We segregate Sample jobs under sample view

We put all Continuous integration jobs comes under c.i views

We put all deployment jobs in deploy view



1. Click on my view
2. Enter view name
3. Select list view/my view/global view
4. In status filter select one
5. All selected jobs
6. Enabled jobs= shows running jobs
7. Disabled jobs=disable jobs
8. Select jobs
9. Select regular expression (rpm\*= all jobs starting with rpm)

Columns :-

1. status
2. weather
3. name
4. last success

there will be many views like project based view maintenance based view…

MANAGE JENKINS:-

1. CONFIGURE SYSTEM:-
2. System message:-

That message will be displayed on dashboard

1. # of executors:-

How many parallel jobs you want to execute. Should be as low as possible

1. labels:-

Labels are group of nodes, we will group some servers and assign a label. When we specify Jenkins to where to run we give label so that Jenkins will check which one is running and assign job to it

1. usage:-

How is your usage, can we use as much as possible or limit to particular node in the label

1. quiet period:-

If you specify some amount of time then jenkins master will execute job only after that much time (in case of network latency)

ex:- 5 sec is the job, jenkins master will start job after 5 sec

1. scm checkout retry count:-

How many times Jenkins should retry scm

1. default view:-

by default the jobs should be of which view

1. restrict project naming:-

Project naming is default or pattern type

1. environment variables:-

These environment variables are global, if you want you can set java, maven env\_var

When we install java, we define environment variables in system properties by right clicking on my computer

echo $JAVA\_HOME--- to know path of java

Value: path

Name: our wish

1. email notification:-

We have to notify the result to all the concerned people

Email suffix= @gmail.com, @yahoo.co.in, @hotmial

1. Git plugin:-

Here you can add git config username and email

1. CONFIGURE GLOBAL SECURITY:-

As a user who uses jenkins, who should work and what permissions should they have

Access control:-

1. Security realm:-

It says who should access jenkins in terms of username and passwords. You can either use jenkins user database/ldap/unix user groups

1. Jenkins user database is genrally used for testing purpose
2. Single sign on (sso) = LDAP. Using single account we are able to connect to different machines and applications
3. Authorization:-

Giving permissions to the users so that they have privileged access

1. Anyone can do anything = no authorization for users, so any user can do anything
2. Matrix based security = sudo, using this we give limited privileges what they really needed

* If you want to give matrix based authorization, then select, add user and assign permissions

1. CONFIGURE CREDENTIALS:-
2. GLOBAL TOOL CONFIGURATION:-

Global tool configuration is used to set path of tools like git, docker, maven, and java

1. MANAGE PLUGINS:-

Plugins are third party scripts which are integrated with Jenkins for better functionality

You can install, uninstall, and update plugins

Ex: - github plugin integrating with Jenkins for poll scm

1. MANAGING NODES:-

We can create many remote nodes. So that Jenkins master server will execute jobs on remote nodes

1. MANAGING USERS:-

We can create many user from this section

1. SYSTEM INFORMATION:-

To see where Jenkins is running, java version, environment variables, properties….

1. SYSTEM LOG:- you can see all the logs, build reports, mainly used for troubleshooting

1. JENKINS CLI:-

Jenkins can also be used in command line

1. LOAD STATSTICS:- memory, disk, cpu utilization of the node

===========================================================

Downstream dependency:-

There are scenarios where job2 depends on job1, means if job1 is executed then job will be triggered this is called downstream dependency

There are some conditions how and when second job should trigger

1. Whenever the first job is triggered successfully completed then trigger second job (stable)
2. Even if first job is failed complete the second job (unstable)

So post build comes into picture

1. Build other project
2. Second job name
3. Condition of the job

Continous.Integration:-

The process of pulling the code from scm and building the application automatically is called C.I

c.i = build + unit test + stage

for every commit developer does, we immediately build, do unit test, put in staging environment

Continous.Delivery:-

The process of pulling the code from scm and building the application automatically and deploying automatically on to only test servers and manually deploying the application to production server is called C.Del

c.d = build + unit test + stage + automated test-------manual------- +deploy

Continous.Deployment:-

The process of pulling the code from scm and building the application automatically and automatically deploying on to only test servers and automatically deploying the application on to production server is called C.Dep

c.d = build + unit test + stage + automated test-------automatic------- +deploy

we majorly go for c.del, we go for c.dep in some business cases(requirement)

Automated.Deployment:-

Automating the deployment process in a continuous delivery system (which is build🡪deploy🡪test🡪release)

Best practice:-

1. Secure jenkins by changing port number, permissions,
2. Plan disk usage, setup jenkins which has more disk space
3. Don’t run too many jobs on master server
4. Make sure you have backup plugin
5. Integrate tightly with issue tracking system like jira, bugzila
6. Avoid scheduling all jobs at once
7. Make jenkins url short
8. Discard old builds and keep latest builds

Popular plugins:-

1. Git & Github:- used to integrate scm with jenkins
2. Github integration:- used for webhooks
3. Maven:- used to build maven jobs
4. build pipeline plugin:- using this plugin we can create our own workflow
5. artifactory:- used to deploy artifacts to jfrog repo
6. pipeline multibranch:- used for multibranch
7. email:- used to send email in case of failure
8. Role based Authorization strategy:- is used for managing user’s permissions
9. Thin backup:- used for taking backups
10. deploy to container:- is used to deploy war/ear files on running application server like tomcat/jboss at the end of build
11. Build timeout plugin:-this plugin allows you to stop the jobs which take more time
12. Job config history plugin: - saves a copy of config file (config.xml) for every change made in system configuration. Can revert to previous config if you want
13. Import jobs plugin:- if you want to import all jobs from one node to another node
14. Green ball plugin:-
15. Disk usage plugin:-

----------------------------------------------------------------------------------------------------

How to configure a job:-

1. go to “item” 🡪 give a project name 🡪 select “type of project” 🡪 click ok
2. select the options you need and click on apply and save
3. now on the dash board click on the project and select build now

----------------------------------------------------------------------------------------------------

Managing Jenkins services :-

1. To shutdown Jenkins

<http://localhost:8080/exit>

1. To restart jenkins (force restart allows you to stop the jobs)

<http://localhost:8080/restart>

1. To safe restart jenkins (allow all running jobs to complete and new jobs will be in quee)

<http://localhost:8080/saferestart>

(or)

Using cli:-

#service jenkins start/restart/stop

After installing plugins we sometimes restart Jenkins

----------------------------------------------------------------------------------------------------

How to create users in jenkins:-

1. go to “manage jenkins”🡪 “manage users” 🡪 “create users”
2. give “username, password, full name”
3. click on “create users”
4. configure a particular user

----------------------------------------------------------------------------------------------------

How to install plugins:-

1. go to “manage jenkins” 🡪 “manage plugins”
2. go to “available section” and search for the plugin you want
3. select the plugin and click on “install without restart”
4. if you need you can update the plugins

----------------------------------------------------------------------------------------------------

How to install plugins through hpi file/manually:-

1. search in the Google for desired plugin
2. you will see that plugin wiki ex:-thin backup
3. download .hpi file of desired version
4. now go to manage jenkins--->manage plugins---->advanced section
5. go to “upload plugin” section, click on choose file and click on upload

\*we will use this when we don’t have access to intenet connection or if we want to upload our custom plugins

----------------------------------------------------------------------------------------------------

How to change port number of Jenkins:-

1. stop jenkins service

# systemctl stop jenkins

1. go to /etc/sysconfig/jenkins and change prt number

#vi /etc/sysconfig/jenkins

1. start jenkins service

#systemctl start jenkins

1. now access jenkins url using new port number

<http://localhost:8090>

----------------------------------------------------------------------------------------------------

How to change jenkins home directory:-

When you install Jenkins by default Jenkins home directory will be created in /var/lib/jenkins

Since /var is o.s related filesystem. We need to change home directory

1. check jenkins path #which jenkins
2. stop jenkins service #systemctl stop jenkins
3. create a directory #mkdir /home/jenkins
4. change ownership of newly created directory

#chown jenkins:jenkins /home/jenkins

1. now take backup from old to new jenkins directory

#rsync -av /var/lib/jenkins /home/jenkins

1. usermod -d /home/jenkins
2. cat /etc/passwd |grep –i jenkins
3. now edit jenkins home directory path in /etc/sysconfig/jenkins

vi /etc/sysconfig/jenkins

Jenkins\_Home=“/home/jenkins”

1. now start jenkins service

----------------------------------------------------------------------------------------------------

How to backup Jenkins:-

The location where Jenkins is installed is called as Jenkins home directory, it contains jobs, nodes, builds, plugins, logs…

You can use backup in jenkins by three methods

1. manual backup using cron with shell script
2. you can backup to cloud
3. thin backup plugin:-
4. create a backup directory

ex:- mkdir /home/jenkins\_backup

1. change ownership of backup directory to jenkins user

ex:- chown jenkins:jenkins /home/jenkins\_backup

1. go to “manage jenkins” 🡪 “manage plugins” 🡪 “install thin backup client”
2. go to settings
3. give “backup directory name”
4. give “backup schedule” (how do you schedule backup like night/morning using cron job format)
5. give “differential backup schedule” (full backup tarwatha malli eppudu backup start avvali)
6. give “max number of backup sets” (how many backup copies you need)
7. give “file excluded” (if you want to any exclude any files during backup)
8. wait until jenkins is idle to perform a backup (jenkins backup wont start if jobs are running)
9. select “options” (only if you want)
10. click save
11. now go to “manage jenkins” 🡪 “thin backup” 🡪 click on “backup now”
12. select “restore backup” from a particular date and click on “restore”

----------------------------------------------------------------------------------------------------

Creating email notification:-

1. go to “manage jenkins”🡪”manage plugins”
2. install “email, email -ext, notification” plugins
3. click “install without restart”
4. go to “manage jenkins”🡪”configure system”
5. ask your infrastructure team for “smtp server name, port nmber”
6. now enter “smtp mail”
7. now enter “suffix mail” (@gmail.com, @tcs.com)
8. tick the checkbox of “use smtp authentication”
9. give “username and password”
10. tick “ssl” checkbox
11. give “test mail” (sample mail id for testing purpose)
12. click “apply and save”
13. now in the project go to “post build action”🡪 select “email notification”, give email recipients of developers to notify if build is failure

----------------------------------------------------------------------------------------------------

How to assign roles using role-based-strategy:-

1. go to “manage jenkins”🡪 “manage plugins”🡪install “Role-based Authorization Strategy” plugin
2. go to “manage jenkins”🡪 go to “configure global security”🡪click on “enable security”🡪 choose “role based strategy” 🡪save
3. go to “manage jenkins”🡪 click on “manage and assign roles”🡪 click on “manage roles”
4. now go to “global role”🡪 in “role to add” section, add a role ex:-employee
5. now assign permissions to employee roles
6. similarly go to “project role”🡪in “role to add” section, add a role ex:- developer, tester.. (where pattern[dev.\*] is like developer should access development projects)
7. go to “manage jenkins”🡪 click on “manage and assign roles”🡪 click on “assign roles”
8. now under global roles add users and assign global level access ex:- employee/admin level access to a particular user
9. in project roles add users and assign particular project to particular user ex:- user1 for developer role
10. click on save

----------------------------------------------------------------------------------------------------

How to pass parameterized job in jenkins:-

If user wants to run a job with parameters, we will go with parameterized job

Ex:- if a job has to be executed in dev, test, uat, pro environments. So that we can choose one env before running a job

1. String Parameter:-
2. click on “freestyle project”🡪 in general section, click on “this project is parameterized”
3. click on “add parameters”🡪choose “string parameter”
4. give name, default value, description

ex:- name=country, default value=india

1. now go to “build section” and choose “execute shell/windows”
2. in “command section” ex:- echo %parameter% 🡪 for windows

echo ${parameter}🡪 for linux ex:-echo "the country displayed is:$country"

1. apply save

\*whenever you click build now button. You will be prompted with parameters

1. Choice Parameter:-
2. Click on “freestyle project”🡪 in general section, click on “this project is parameterized”
3. click on “add parameters”🡪choose “choice parameter”
4. give name, choices, description

ex:- name=tools, choices=ansible, git, docker, jenkins

1. now go to “build section”🡪choose execution type based on o.s
2. in command section ex:- echo %tools% 🡪 for windows echo $tools$ 🡪 for linux ex:- echo "building $tools "
3. apply save
4. now click on build with parameters and choose one value

----------------------------------------------------------------------------------------------------

Multi branch pipeline:-

Creates new builds for every branch in remote repository

1. Go to “manage jenkins”🡪 “manage plugins” 🡪 install “pipeline multibranch” plugins
2. Click on “new item”🡪enter “project name”🡪click on “multibranch pipeline”🡪click ok
3. Go to “branch source” section🡪in drop down list select your “scm”🡪 now copy the scm project url🡪choose your “credentials”
4. Click on save
5. As soon as you save this plugin will do branch indexing
6. now click on your project and you will see multiple branches

\* It works with jenkins pipeline file

\* Now this pipeline searches for remote branches in github repo and creates a build for every branch ex:-if there are 2 builds developer and master branch respectively then 2 builds will be created

----------------------------------------------------------------------------------------------------

Launching linux/windows slave to run jobs on remote node:-

Pre-requisites:

1. Must have java installed on remote node
2. Jenkins up and running, must be login as admin
3. ssh keys must be copied if it is linux node

setup:

1. go to “manage jenkins”
2. go to “manage nodes”
3. click on “new node”
4. give name to that new node
5. select “permanent agent” or if you already have a node created then go with “copy existing node”
6. click ok
7. now give “name”, enter “description”
8. give “number of executors” (means how many jobs should run simultaneously on the node)
9. specify “remote root directory” [ensure permission for the directory]
10. enter “label” name (label is nothing but group of nodes, if one node in the group is down then jenkins will assign to another node)
11. select “usage” from dropdown list
12. select “launch method” (use ssh if remote node is linux else use java web start if it is windows node)

for linux---🡪ssh

1. give remote node hostname/ipaddress
2. add credentials
3. known hosts file verification strategy

for windows

1. go to remote node and open browser and enter jenkins master url
2. go to manage jenkins🡪 manage nodes🡪click on newly created node-🡪click on launch
3. now a “slave.jar” file will be downloaded
4. that’s it
5. select “availability” (how should the node be like keep agent always online)
6. give “node properties”
7. click apply and save
8. launch the agent

----------------------------------------------------------------------------------------------------

Launching jenkins slave on ec2:-

===========================================================

JENKINS PROJECT PIPELINE

DEVELOPER CODE---🡪JIRA---🡪GIT---🡪SONARQUBE---🡪MAVEN---🡪JUNIT---🡪JAR---🡪JFROG---🡪ANSIBLE---🡪ACCEPTANCE TESTING---🡪JACOOC---🡪RELEASE

---------------------------------------------------------------------------------------------------- How to integrate jenkins with jira:-

Jira has to be integrated with ldap for raising tickets

----------------------------------------------------------------------------------------------------

How to integrate jenkins with git and github:-

1. go to “manage jenkins”🡪 “manage plugins” 🡪 install “ github” plugins
2. go to “item”
3. select a “freestyle project” click ok
4. give name, description, label,
5. come to “scm” section under that select git
6. now paste the github link and give credentials
7. give “pollscm” using cron job format
8. save

\*git should be installed on that node to run Jenkins job for poll scm

----------------------------------------------------------------------------------------------------

Git webhooks:-

After each and every commit made by the developers into a specific repository. There will be a new builds automatically starting. This is called web hooks

1. go to “manage jenkins”🡪 “manage plugins”🡪install “github integration”
2. click on “new item”🡪enter “name”, “freestyle project”🡪 ok
3. in general section click on “github project” and paste the github project url ex:- <https://github.com/kvvmanikanth/webapp-master>
4. in scm section paste the git clone repo

Ex: - <https://github.com/kvvmanikanth/webapp-master.git>

1. in build trigger section choose “github hook trigger for git scm polling”
2. click on save
3. now go to github webpage and go to your repository ex:-webapp-master
4. click on settings 🡪on left side choose “webhooks”🡪 click on “add webhooks”
5. now in “payload url” copy jenkins url ex:- <http://jenkins.com/github-webhook/>
6. click on save
7. whenever there is a change commited to github, automatically new build is started in jenkins

----------------------------------------------------------------------------------------------------

How to integrate jenkins with sonarqube:-

----------------------------------------------------------------------------------------------------

How to integrate jenkins with maven:-

1. go to “manage jenkins”🡪 “global tool configuration”🡪go to maven section add maven path
2. go to “item”🡪select freestyle project
3. now under “build ” section and choose “invoke top level maven target”
4. select “maven version”🡪select “Goals”
5. click save

---------------------------------------------------------------------------------------------------

How to integrate jenkins with ssh:-

----------------------------------------------------------------------------------------------------

How to integrate jenkins with jfrog artifactory:-

1. go to “manage jenkins”🡪 “manage plugins”🡪install “artifactory” plugin
2. go to “manage jenkins”🡪 “configure system” 🡪 go to “artifacts section”
3. Add “artifactory server”🡪give server id(name), url(<http://localhost:8081//artifactory>), credentials.
4. now click on “test connection”
5. now go to “new item”
6. a) if you choose “maven project” in post build action choose “deploy artifacts to artifactory”, now choose your artifactory server and click on “refresh”

b) if you select “freestyle project” 🡪 in build trigger section choose “Generic-Artifactory Integration/maven3-Artifactory Integration(if maven repo is set in jfrog server)”

1. click on save

----------------------------------------------------------------------------------------------------

How to integrate Jenkins with apache tomcat:-

1. go to “manage jenkins”🡪 “manage plugins”🡪install “deploy to container”
2. click on new item🡪choose freestyle project
3. in post build action choose “deploy to container”
4. in WAR/EAR files give “\*\*/\*.war”
5. context path is “file name without war extension”
6. choose “tomcat version”
   1. give “username”
   2. give “password”
   3. paste “tomcat url”
7. click on save

----------------------------------------------------------------------------------------------------

C.I-C.D job:-

Pre-requisites:-

Ensure java, maven tomcat is installed and set their environment variables in node as well as in Jenkins and install github, git, deploy to container, maven plugins.

1. go to “manage jenkins”
2. go to “global tool configuration” add jdk, maven, select git, save
3. click “new item” 🡪enter “project name” 🡪 “freestyle” 🡪ok
4. now “enter description”,
5. go to “scm”, select “git”
6. give “github url”, add “credentials” only if scm is private
7. go to “build triggers” give “poll scm” using cron format
8. go to “build” 🡪invoke top level maven target

select goals (goals is whether you want to create package or you want to clean or ….)

1. now go to “post build actions” 🡪 select “deploy to container”
   1. war/ear files \*\*/\*.war

\*war/ear files will be in workspace🡪cronjob🡪target🡪yourfiles

* 1. context path ||mvn-hello-world||

context path means file name without .war extension

* 1. select container (glassfish/jboss/tomcat)
     1. enter container version
     2. give credentials
     3. container url

1. click apply and save

\*for deployment we can use many ways like

(i) manually copying the war file

(ii) through scripts

(iii) deploy to container

(iv) publish over ssh plugin

(v) docker

(vi) ansible

----------------------------------------------------------------------------------------------------

Upstream and downstream jobs:-

Assume there are two jobs job1 and job2 respectively, if job2 is dependent on job1, then job2 is downstream project and job1 is upstream project

1. Create a new job ie.., job1 in freestyle project—under scm paste github link
2. Now go to post build actions choose “build other projects”
3. Select downstream job
4. Click on save
5. Now create job2 under pollscm give cron format and save
6. Now execute job1, if job1 is executed successfully then job2 will be triggered

----------------------------------------------------------------------------------------------------

Build pipe line:-

1. Create a separate jobs and configure them like job for jira, job for git, job for sonarqube, job for maven, job for junit, job for jar, job for jfrog, job for tomcat.
2. go to “manage jenkins”🡪 “manage plugins”🡪install “build pipeline” plugin
3. now go to job1🡪click on configure🡪click on post build actions🡪choose “build other projects” && “notify via email”🡪give job2 name🡪save
4. now come to job2 and add job2 in post build action similarly do it for all
5. click on “+”🡪enter “view name”🡪select “build pipeline view”🡪ok
6. now select the initial job which has to be triggered first and later all dependent jobs will be kicked off
7. now run the pipeline
8. jobs must execute sequentially, if any job failed then it turns into red color

----------------------------------------------------------------------------------------------------How to integrate Jenkins with docker:-

DYNAMICALLY PROVIDING JENKINS BUILD SLAVES IN A DOCKER CONTAINER

In order to integrate Jenkins with docker you must enable docker remote api. This remote api is used for Jenkins to connect docker

A) Install Jenkins and docker on same/different host

B) On the docker host enable remote API

C) Add Jenkins user to the docker group

D) On the Jenkins host

1. Go to manage Jenkins🡪 manage plugins🡪available🡪click on “docker” plugin

2. Go to manage Jenkins🡪click on configure system🡪click on “cloud “section, choose “docker”

a) Name your docker cloud (DOTCLOUD) b) Enter docker host Uri(tcp://dockerhostipadress:4243)

c) Tick enable

d) Click on "test connection"

\*if correctly configured you will see docker api version

Now click on docker agent template

a) Assign a name to label (docker-jenkins) b) Name of docker template (docker-agent-template)

c) Docker image (evarga/jenkins-slave)

d) Instance capacity (1)

e) Remote filesystem(/home/jenkins)

f) Usage (build with label expression)

g) Idle timeout

h) Connect method (ssh)

i) sshkeys (inject sshkey)

j) User (jenkins)

k) Tick remove volumes

l) Pull strategy (pull once update latest)

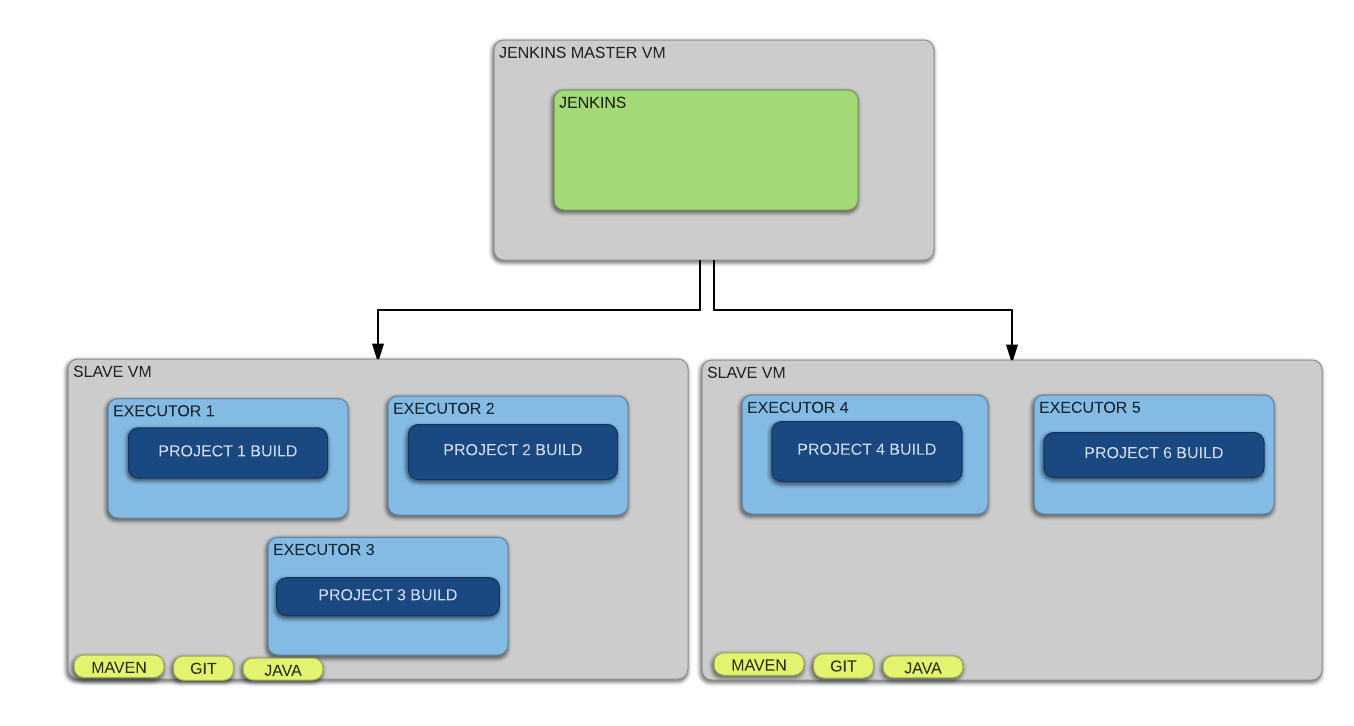
m) Pull time out

n) under container settings--->go to volumes---->paste "/var/run/docker.sock:/var/run/docker.sock"

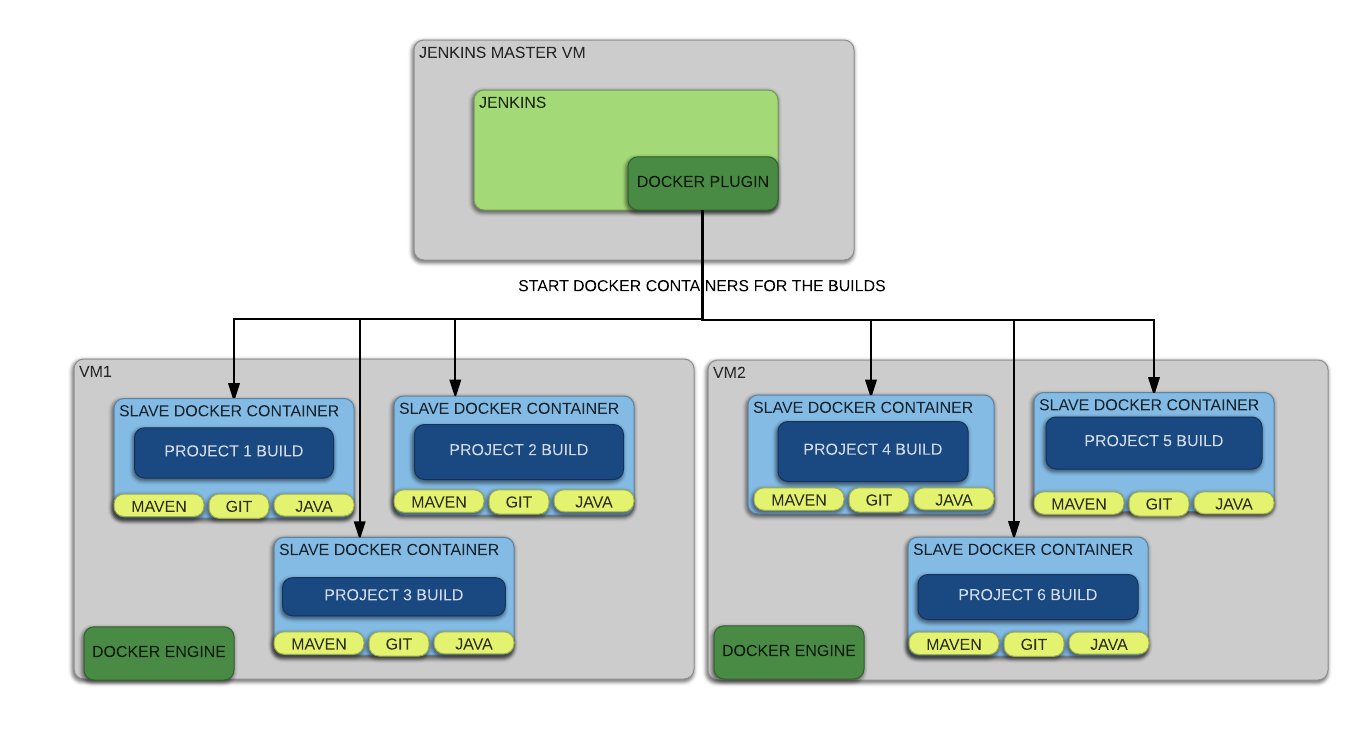
o) click on enabled under labels

1. click apply and save
2. click on new item---give project name----give label name------in build section "echo "hi""
3. click on save and apply
4. now click on build
5. you will see node launching, container launching and build executing

\*since we have many development groups within the enterprise we have different platforms, tools, libraries…. It is difficult to install each version of tools, configuring on Jenkins slave.



So we go for docker containers which dynamically build Jenkins slaves



----------------------------------------------------------------------------------------------------How to integrate Jenkins with ansible:-

We have HOST1 on which Jenkins and ansible is installed and HOST2 on tomcat is installed. We configured ssh mechanism between 2 hosts

1. On HOST1 go to "manage Jenkins"🡪"manage plugins"🡪install "ansible plugin"

2. Go to global tool configuration set ansible name & location by executing #which ansible o/p:- (/usr/bin)

3. In scm section give "githublink"🡪in build section click on "invoke top level maven target" and then click on "invoke ansible Playbook"

4. (i) Ansible installation🡪just name of ansible you defined in global tool configuration ex:-ansible2.4.2.0

(ii) Playbook path🡪location of playbook ex: - /ansible/jenkins-tomcat.yml

(iii) Inventory🡪choose file path (location of inventory) ex: - /etc/ansible/hosts

(iv) Host subset🡪a group of hosts from inventory file ex :-${demo}🡪demo is a group name

(v) Credentials🡪ssh credentials (here I used root name and password)

5. Click on apply and save

6. Click on build and check whether the war file is deployed on to tomcat or not

\*the playbook will be in JENKINS folder as jenkins-ansible

----------------------------------------------------------------------------------------------------How to integrate Jenkins with kubernetes:-

----------------------------------------------------------------------------------------------------

Jenkins file/pipeline as a code:-

A Jenkins pipeline allows you to define an entire application life cycle as code. The Jenkins code which is stored in a file is called Jenkins file. This Jenkins file is maintained in source code repository because if developers want to modify the file they can pull and modify it.

Benefits Of Pipeline as a code:-

1. versioning
2. reusability
3. ask for user inputs
4. can run jobs in parallel
5. developers can easily modify
6. can easily debug

There are 2 types of Jenkins files

1. scripted/traditional pipeline:-
2. introduced first
3. strict groovy syntax
4. typing code in Jenkins ui
5. code starts with node block

Ex: - node {

}

1. declarative pipeline:-
2. introduced recently
3. simpler syntax
4. writing code in a file and storing it in source code repository
5. code starts with pipeline block

Ex: - pipeline {

}

In this tutorial we go with declarative pipeline, it consists of following directives which are mandatory to use

1. Agent:-

Agent is nothing but the job that runs on it, it may be either host/container.., and Jenkins will distribute workloads to different agents

* Agent has few parameters like

1. none = runs pipeline on its own agent
2. any = runs the pipeline on available agent
3. label = runs the pipeline on labeled agent
4. docker = runs the pipeline in a docker container

Ex: - pipeline {

agent any

}

Ex: - pipeline {

agent {

docker{

image ‘ubuntu’

}

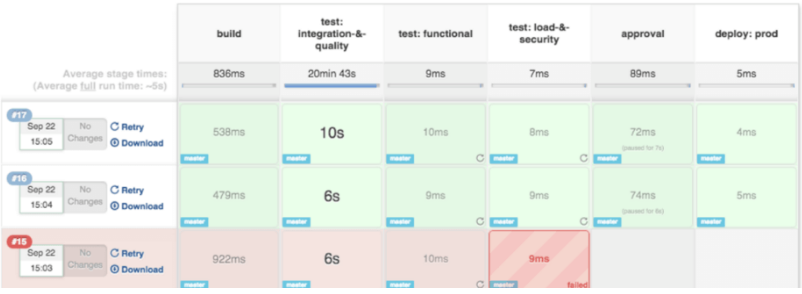
}

}

The job will execute on docker container

1. Stages:-

This section contains different stage, used for visualizing different segments



Ex: - pipeline {

agent any

stages {

}

}

1. Stage:-

Stage defines what task to be done

Here pulling code is a stage, deploying artifacts is a stage, building & packaging is a stage

Ex: - pipeline {

agent any

stages {

stage ('build') {

...

}

stage ('test') {

...

}

}

}

1. Steps:-

Steps are defined with in the stage section, where we mention the actual task

Ex: - pipeline {

agent any

stages {

stage ('build') {

steps {

sh 'echo "running build phase"'

}

}

}

}

There are many more directives like

1. Environment = can be defined at stage or pipeline level, if environment defined at stage level then the scope is only for that stage, if it is pipeline then scope of environment will be for all stages

pipeline {

agent any

environment {

OUTPUT\_PATH = './outputs/'

}

stages {

stage ('build') {

...

}

}

}

\* Here environment is defined at pipeline level

1. Input = Defined at stage level, used for prompting input

pipeline {

agent any

stages {

stage ('build') {

input{

message "Press Ok to continue"

}

}

}

}

\*Here we defined input ata stage level asking for prompt

1. Options = Defined at pipeline level, some of the options are retry, timeout, timestamps

pipeline {

agent any

options {

retry(3)

}

stages {

stage ('build') {

...

}

}

}

4. Parallel = two run parallel, you can either define in stages/stage that means can have multiple stages in parallel or multiple steps in parallel

* A stage directive can have either a parallel or steps directive but not both.
* A stage directive inside a parallel one cannot nest another parallel directive, only steps are allowed.
* Stage directives that have a parallel directive inside cannot have "agent" or "tools" directives defined.

pipeline {

agent any

stages {

stage ('build') {

steps {

parallel(

a: {

echo "Tests on Linux"

},

b: {

echo "Tests on Windows"

}

)

}

}

}

\* Here we have defined parallel option at steps level



1. Parameters = It should be defined at pipeline level, string and Boolean parameters can be used

pipeline {

agent any

stages {

stage ('build') {

steps {

parallel(

a: {

echo "Tests on Linux"

},

b: {

echo "Tests on Windows"

}

)

}

}

}

1. Post = if we want to know the status post the job like success/failure/unstable….. Can be defined at pipeline level or stage level

pipeline {

agent any

stages {

stage('Some steps') {

steps {

...

}

}

}

post {

always {

echo“ Pipeline finished”

bat. / performCleanUp.bat

}

}

}

\* Here post directive is defined at pipeline level

7. Script =

8. Tools = can be added at stage or pipeline level, it allows you to specify which jdk, maven versions to use (just how we define at global tool configuration)

pipeline {

agent any

tools {

maven 'apache-maven-3.0.1'

}

stages {

...

}

}

\* Here I have defined tools at pipeline level

9. Triggers = how to trigger the pipeline like cron/pollscm/….

pipeline {

agent any

triggers {

cron('0 \*/4 \* \* 1-5')

}

stages {

...

}

}

10. When = the steps in a stage executes only when condition matches

pipeline {

agent any

stages {

stage('Deploy stage') {

when {

branch 'master'

}

steps {

echo 'Deploy master to stage'

...

}

}

}

}

HOW TO USE DECLARATIVE PIPELINES IN JENKINS:-

1. click on new item
2. name the project & click on pipeline project and then click ok
3. click on pipeline tab
4. there you choose pipeline script from scm (for declarative pipeline)
5. choose scm and paste scm link and select your credentials
6. mention your Jenkins file path in script
7. click on save and apply

\* Note that the naming convention should be strictly followed like this Jenkinsfile