96) git rebase (branch name)

→ We need to be on the traget branch and then rebase the data from one branch to another

ex:- we are on branch (target) feature

git rebase master.

Rebase is used when Hotfix comes into the scene so, to keep the develop branch with latest data we will use rebase to merge hotfix data to develop.

97) git rebase -i ~~(1)~~HEAD~3 → for the record to select.

→ We can rearrange the data

98) git rebase -i HEAD~3

→ merging multiple commits ← vi will open.

ex:- PICK ____ line
s ____ line } Squash commit.
s ____ line

Maven set-up :- Build Tool

POM files :- When you execute a maven command you give maven a POM file to execute the commands on.

→ artifact :- converted file

* .Java = Maven
* .Net = MSBuilder.
* Python - PyBuilder.

Maven Repositories :-

* Local repo
* Remote repo
* central repo

→ Laptop → under organization internal network → internet.

Maven life cycle :-

i) Validate - necessary information is available.
ii) compile - compile the source code (the 'c' files)
iii) test - Run the unit test cases
iv) Package - Create the package (artifacts) ex :- warfiles, jarfiles etc.,
v) install - To store the artifacts in Local.
vi) deploy :- To store the artifacts in to Remote (or) central.

install Java :-

```
sudo amazon-linux-extras install epel
sudo yum update
sudo yum install java-1.8.0-openjdk
sudo yum install java-1.8.0-openjdk-devel
java -version

ls -la
cd /etc/
pwd
ls
cat profile
echo 'export JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk' |
                                    sudo tee -a /etc/profile.

cd /usr/lib/jvm
ls
cd
cat /etc/profile.
echo 'export JRE_HOME=/usr/lib/jvm/jre' | sudo tee -a /etc/
                                    profile

cat /etc/profile
env
```

Source /etc/profile

echo $ JAVA_HOME

Echo $ JRE_HOME

Maven installation :-

wget

wget (Path)

ls

sudo tar -xf apache-maven-3.6.3-bin.tar.gz -C /usr/local

cd /usr/local/

ls

sudo ln -s apache-maven-3.6.3 maven

ls

ls -l

cd

cd /etc/

cd profile.d/

ls

vi maven.sh

→ adding the maven path

export M2_HOME=/usr/local/maven

export PATH=${M2_HOME}/bin:${PATH}

ls

Sourc /etc/profile

cd

echo $M2_HOME

mvn -v

→ When working with maven we need

* Source code
* POM file

mvn archetype:generate (to get sample files)

mvn validate

mvn compile → (Java class file are created).

mvn test target file is created in

mvn Package. the class file are created

ls

cd target/

ls

cd classes/

ls

cd Javaapp

ls

cd com/

cd samplejava/

pwd

ls

mvn test

→ Runs test case and generate reports.

ls

cd target

ls

→ Here we can find the reports.

cd surefire-reports/

ls

cat javaapp.com.AppTest.txt

cd..

cd..

ls

Mvn package.

→ Here the artifact is created

cd target/

ls

→ artifact ex :- Sample Java-1.0-SNAPSHOT.jar

if we are not storing and re-run then the existing will be gone.

mvn install

→ for installation we should be in the POM file location

→ .m2 is the maven local repo.

```
ls
ls -a
pwd
cd ..
cd .m2/
ls
cd repository/
ls
cd javaapp/
ls
cd com/
ls
cd sampleJava/
ls
cd 1.0-SNAPSHOT/
ls
cd
ls
```

Jenkins's :- Automation Tool [8080]
Install Java integration

cd

sudo wget (Path)

ls /etc/yum.repos.d/

cat /etc/yum.repos.d/jenkins.repo

sudo rpm --import https:// (Path) → key.

Yum install Jenkins

— on instance —

* launch-wizard

Actions

Edit inbound rule.

Add rule.

All traffic Custom IPv4

Save rule.

service Jenkins status

ps -ef | grep Jenkins.

netstat -nltp

service Jenkins start.

Service Jenkins status

netstat -nltp.

ps -ef | grep Jenkins

Administrator Password:

```
cd /var/lib/jenkins/
ls
cd secrets/
cat initialAdminpassword
ls
```

→ copy and paste the Password. (continue)

→ Installation of plugins

* Creat Admin

Username :- vamshi
Pass :- vamshi
confirm pass :- vamshi
full name :- vamshi
E-mail :- vamshikalurla1229@gmail.com.

→ Jenkins is ready.

```
cd
cd /var/lib/jenkins/
ls
cd jobs/
ls
cd plugins/
ls
```

How to change port NO:-

cd
cd /etc/sysconfig
ls
vi jenkins

→ Jenkins_port = "8080"
- "9090"
:wq

find / -type f -name jenkins*

service jenkins restart

Jenkins :- 'UI'

Manage Jenkins :- To manage.

New Item :- To create Job

freestyle project :- can be used with UI

Pipeline :- using code.

Folder and et.,

* Free style project :-

Build :- Main task are execute.

Add build step
execute shell

command

```
echo "hello all"
```

cd /var/lib/jenkins/.

on instance after the execution of the job "workspace" is created.

Build history:-

* To see the Previous triggers (build history).
* To know the outcome of build to click on build no.

Here, console output is present.

→ To modify any job go to the job then 'configuration' and Add build step 'execute shell' to create multiple commands.

Maintain Build history:- (Test job). (To Delete)

Configure

→ Build General :-

→ Discard old builds ☑

→ Days to keep builds :- 1

max# of builds to keep :- 1

To know individual info :- (exact time).

→ Build environment :-

→ Add timestamps to the console output.

Apply

Save

Clean up activities:- Build with parameters & Assignment

To accept user input :- Sampe-Job

-> Dashboard

-> Config

☑ This project is parameterized

Bookean parameter

Choice parameter ✓

Name:
ENV.

Choice:
Dev
QA
UAT
PROD

Description:
select the enviornment.

(clear cache).

df -h to see the Sewer space

if wanted to add another parameter then
- String parameter :-

Connecting to Version control tool :-
Source code management :- (scm)

-> by dfault we will get 'Git'

To pull the data from Git-repo to Jenkins first install git on git bash then take the code in https/ from git repo and paste in the Jenkins (scm)

Workspace :- workspace is created after the build Job outcomes will be placed in workspace.

Changes :- Which build number no for which change.

Build Triggers :-

1) ☐ Trigger builds remotely (from scripts).

→ Auth TOKEN :- Build

→ Jenkins-URL/job/CI-CD-Job/build?token=TOKEN_NAME

↓ (our) → Change URL

↓ Ex:- (Build) Whatever name given.

→ Paste this URL in browser, Hence we can trigger job by using remote host (browser).

2) ☐ Build after other projects are built (Chain jobs)

→ once after Parent job is successful then only the child job will be triggered

Ex:- dev - (first Job) — [Post-build] — QA-Job. ① option trigger only if stable

QA - (B+OP) - Dev-Job. ② option.

UAT -

Here :-	Dev	DEV	QA
	QA	QA	UAT.
		UAT	

3) ☐ Build periodically (scheduling). (crontab).

Ex:- * * * * *

Drawback :- it will create many duplicates.

so, it is used as (backup)

** 4) GITHUB hook trigger for GIT scm polling :-

i) one change from version control tool (Jenkins)

ii) one change from GITHUB

GITHUB :- Repo settings (hello-world-servlet).

webhooks :- Add webhook.

Payload URL *

i.e., http://server details/github-webhook/

ex:- http://3.141.40.212:9090/github-webhook/

Add webhook.

(Then commit changes and push to Github and merge the features and on Jenkins job is triggered).

5) Poll scm (schedule).

It will trigger the job when there is a change on Version control tool

ex:- * * * * *

To install any Tool from Jenkins :-

Dash board

* Manage Jenkins

⇒ Global Tool configuration

→ Gradle

→ Ant

→ Maven.

Add maven.

Name

ex:- maven.

Auto install.

version

[Save] [APPly.]

CI-CD-Job.

Build.

⇒ Add build steps.

→ invoke top-level maven targets

maven version

⇒ Goals.

Clean Package.

Post-build Actions:-

→ Publish Junit test result report

↓

Test report xml

**target/surefire-report/*.xml

To see Test results report

Save | Apply

⇒ SONARQUBE (9000)

code Quality checking :- (sonarcube). (Dependencises)

⇒ (Sonar cube) set-up:-

→ Java set-up (required).

→ Sonar setup

wget (path)

ls

cat sonar.repo

sudo mv sonar.repo /etc/yum.repose.d/sonar.repo

Yum install sonar.

→ Sonar installed

Now, open port

then start. → service start sonar

Port

→ Security group

Add rule. → All traffic IPv4

Then,

Copy and paste IP adress in browser and port number is '9000'

→ sonarcube Dashboard ←

H_2 Database for (Demo) – practice –

Sonar Supports 20+ computer Languages.

Log in

user : admin } default.
pass : admin

→ integrate sonar to Jenkins.

⇒ Jenkins.

→ Dash board

→ manage Jenkins

→ configure system.

→ manage Jenkins

→ manage pulgins.

→ Available.

☑ SonarQube scanner

Download and install after install

→ manage Jenkins.

→ Configure System (integration).

Add SonarQube.

upload plugin (interview Q/A)

.hpi is the plugin extension (Advanced)

Name

Sonarqube

Server URL

provide sonarqube URL.

[Save] [Apply]

→ Sonarqube :-

Sonarqube scanner install this tool in Jenkins

→ Jenkins

→ manage Jenkins

→ Global tool config

→ sonarqube scanner

→ Add sonarqube scanner

→ Name

sonarqube

[Save] [Apply]

** Two Things :-

i) Integrating the SonarQube server with the Jenkins (i.e, configure system) then,

ii) Installing the analysis tool & SonarQube scanner on the Global tool config.

Build

→ Execute SonarQube scanner [imp]

→ Analysis properties.

From P.P.T (Shankar).

i.e. key =
Project name =
Version =
Source =
Java-binaries =

Save | Apply

→ When we stop and start the instance (public) then then the (IP address) changes, Hence the communication b/w Jenkins and SonarQube can't be reached.

→ SO, Here we can give (private IP) instance to the Jenkins in Jenkins server URL, Hence the private IP can't be changed then the communication b/w Jenkins and SonarQube is reached.

→ Sonarqube Quality metrics :- To see what are the Bugs, Vulnerability and code Smells.

Metrics :- We can ignore A and B metrics.
↓ minor ↓ info

Sonar Qube :-

→ Log in as a Admin to have full control

Quality Gate :-

→ Sonar way is the default Quality Gate

→ But we want create our own Quality gate then Log in as admin and create and add conditions for the project.

→ Since, sonar way is the default Quality gate. so, it will be applied when we go for build

but, if we want our own Quality gate to run then change sonarway default to our job as default

→ sonarway → Default (Built in).

our gate —

→ sonarway — (Built in)

our job → Default

TO give Quality gate to individual Jobs then go to Project then click on project name and on project level Administration go to Quality gate and change → Project

ex:- → CI-CD

→ Administration

→ Quality gate

Quality profiles:- sonarway (Default).

→ We can find all the set of rules

→ Sonarway (default) rule can't be modified. so, we have to create our own profile for that copy the profile and give name

ex:- Sonar way

Java

→ To change the issue type. So, click on the rule and change the severity of the rule and then make it as 'Default' to use for the particular job

→ Quality profiles.

Language. Quality profile.

→

To provide access to other folks.

→ Administration

→ security

→ create user.

→ security (restrictions). } And give permissions
→ Global permissions.

** Any setting of the app file (or) service side it is a config file.

cd /opt/sonar → Sonar related data is present here.

cd logs/

ls

cat sonar.log

ls

cd..

ls

cd extensions/ → (plugins are present).

ls

cd plugins/.

ls

cd ../..

ls

cd conf/.

ls.

cat sonar.properties.

If Quality is failed on sonar then fail it on Jenkins

Jenkins

→ manage Jenkins

→ manage Plugins

→ Available.

→ Sonar Quality Gate.

GOL + JOB

→ manage Jenkins

→ configure system

→ Quality Gate - System

* Name

* Server URL

GOL Job

→ Config system.

→ Post-build Action

→ Quality Gate SonarQube plugin

→ Project key (from properties)

→ Status when analysis fail.

[Save] [Apply]

→ Build.

→ Job failed.

[CI-CD-Job :-]

→ cobertura [old code-quality Tool].

CI-CD-Job

→ config.

Dashboard.

→ manage Jenkins

→ plugins.

→ Available.

→ cobertura.

CI-CD-Job

→ config

→ Build.

→ Post-build Actions. [Publish the cobertura coverage report]

→ Build ~~Trigger~~.

* maven

* Goal

clean package cobertura:cobertura

→ Post-build Action.

→ Publish Cobertura coverage Report.

→ **/target/site/cobertura/coverage.xml

Nexus :- NEXUS 07/03/22

(ARTIFATCS)

(8081/nexus)

1) Install Java

Yum install Java-1.8.0-openjdk

2) Create a directory app and switch into directory.

sudo mkdir /app && cd /app.

Pwd

3) Download nexus.

wget path. (tar file).

ls

4) extract the tar file

sudo tar -xvf nexus-2.11.02-03-bundle.tar.gz

ls

sudo mv nexus-2.11.2-03 nexus (mv to rename)

→ create a new user named nexus.

sudo adduser nexus.

ls

ls -l

→ provide permissions to newly created user.

```
sudo chown -R nexus:nexus /app/nexus
ls -l
sudo chown -R nexus:nexus /app/sonatype-work
ls -l
cd nexus/
ls
```

→ creating a shortcut to the service.

→ Any service executable file is located in 'bin'.

```
cd bin/
ls
pwd
cd
/app/nexus/bin/nexus status.
su nexus.
cd
pwd.
whoami
/app/nexus/bin/nexus status.
cd /etc/init.d/.
pwd
ls
```

→ It is one of location in our systems. (To use the service command) cd /etc/init.d/

cd

→ always be in root user when stepping up something.

sudo ln -s /app/nexus/bin/nexus /etc/init.d/nexus

cd /etc/init.d/

ls

ls -L

service nexus status

su nexus.

cd

pwd.

whoami

service nexus start.

→ now open the fire wall (8081).

publicIp:8081/nexus/

→ Login

user : admin

pass : admin123

→ To install plugins [Jenkins].

→ To work with nexus we need 2 plugins.

Nexus platform

Nexus artifact uploader.

Jenkins

→ manage Jenkins

→ manage Plugins

→ Available

→ nexus (download).

For nexus authentication is required.

Integration :-

Jenkins

→ manage Jenkins

→ configure System

(nexus).

(Nexus repo mager 2.x server).

* Display name · nexus

* Server ID 1234

* Server URL

http:// private IP / 8081 / nexus.

save | apply

Authentication:-

→ Manage Jenkins

→ security

→ manage credentials.

→ Add credentials.

(or) instant creation.

→ manage Jenkins

→ config system

(nexus).

Add credentials.

kind : username & pass

username : admin

pass : admin123

ID : nexus-cred

Descrip: nexus-cred.

Test connect

(Test connection) to check connectivity

Now,

Go to the 'Job' and config Job settings.

CI-CD-Job

→ config

→ Nexus repo manger publisher.

- Nexus instance : Nexus
- Nexus Repository : Release

Packages — (POM file) (GitHub repo)

→ Add packages

→ Group :- com.geekcap.vmturbo

Artifact :- hello-world-servlet-example

Version :- 1.0

Packaging :- war

(POM :- Project object model file)

Artifacts

→ Maven artifact

→ File path → (in workspace)

target / helloworld.war

Nexus :-

→ Repositories

→ Release

→ com

→ geekcap

→ vmturbo

} (or) Go TO 'Browse index'

Nexus

→ Configuration

→ Access settings

→ Deploy policy : Allow Redeploy

Save

Ex:- (Git+Hub repo).

Develop.

→ Redme.file.

→ Add content.

} Build.
Poll scm (trigger)

Jenkins

→ Build

{→ linking the change as Build number.}

→ manage Jenkins.

→ Status info.

→ System info

→ manage Jenkins.

→ config system (build) → search

→ $ BUILD_NUMBER

Dash board

→ Job

→ config

→ Build

→ version :- $ BUILD_NUMBER.

Change code once again from GitHub

***

How to track artifact version?

We are uploading the build number as tracking.

TOM CAT [8080]

Tom cat :- (Web server).

install Java

Sudo amazon-linux-extras install epel

Sudo Yum update.

Sudo Yum install Java-1.8.0-openjdk

Sudo Yum install Java-1.8.0-openjdk-devel

echo 'export JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk' | sudo tee -a /etc/profile.

echo 'export JRE_HOME=/usr/lib/jvm/jre' | sudo tee -a /etc/profile.

source /etc/profile

printenv.

→ Tom cat (browse).

copy tar.gz from browser

wget paste the path.

→ extract file.

tar -xvf apache-tomcat-8.5.76.tar.gz

ls

cd apache-tomcat-8.5.76/

ls

All executabe files are under 'bin'

cd bin/

ls

ls -l

chmod +x shutdown.sh

chmod +x startup.sh

ls -l

→ create a soft link.

ln -s /root/apache-tomcat-8.5.76/bin/startup.sh /etc/init.d/tomstart

ln -s /root/apache-tomcat-8.5.76/bin/shutdown.sh /etc/init.d/tomcatstop

cd /etc/init.d/

ls

ls -l

cd

service tomcatstart

/etc/init.d/tomcatstart

Now,

Open the port and add rule

port number of Tomcat is '8080'

To deploy the content into Tomcat 'manager app'.

```
cd apache-tomcat-8.5.76/
ls
cd webapps/
ls
cd host-manager/
ls
cd META-INF/
ls
vi context.xml
```

⇒ Valve block

```
<!--    (starting)
        -->   (ending)
```

→ Same ←

```
cd ../../..
cd webapps/
cd manager/META-INF/
ls
vi context.xml
<!--
        -->
```