10/6/2017

Linux

1. mkdir → To create folder or directory
2. CD → To open folder
3. CD .. → one step back
4. CD ../.. → Two steps back
5. Touch → To create file
6. Vi → To edit file
7. ESC :wq! → save & quit the file
8. ESC :q! → quit from file
9. ls → To check the list of folders & files
10. sh & ./ → To run any script in Linux
11. rm -rf → To remove folder & file
    eg: rm -rf file name
12. chmod 777 → To give access (all) permission
13. PWD → To check present working path & location
14. cat → It will display content
15. sudo su → switched user

ls -lart → To check the list of hidden files & folders

11/6/2017

SVN commands

1. svnadmin create (Repository name) → used to create empty repository
2. svn co URL → used to download the code from repository
   Eg:- svn co http://192.168.33.10:8080/subversion
3. svn ci -m "message" → pushing the code from local to svn repository
   Eg:- svn ci -m "created test folder"
4. svn update & svn up } To update the folders
5. svn status & svn st } To check the status
6. svn cp (or) copy
7. svn mv (or) move
8. svn delete
9. svn merge → To combine the code
10. svn log → To check the history

username → vagrant
password → vagrant

wget → command for getting the

18/6/2017

SVN Repository Creation

In putty all operations done

* SVN repository is a empty box where we can store source code & we can access source code

Putty
IP : 192.168.33.10
PWD → it will give /home/vagrant
↓
This is path of desktop

New Repository Creation

↳ Here we always creating repository in desktop and the desktop path is → /home/vagrant

After login
↓
PWD
↓
/home/vagrant ~ it is the path of desktop where we want to create repository
↓
sh svnadmin create /home/vagrant/qr
↳ It will give error because directory is no found that is svnadmin is not found
↓
CD subversion 1.9.4.2
↓
CD subversion
↓
CD bin
↓
ls → Here we can find svnadmin
↓ Step 1: creating repository
sh svnadmin create /home/vagrant/qr
↓

To check whether repository is created & not check in /Home/vagrant as we created in desktop

PWD

/Home/vagrant/subversion-1.9.4.2/subversion/bin

cd../..

cd..

PWD

/Home/vagrant

CD subversion-1.9.4-2

→ once repository is created some default folders are created

Configuration location is
/Home/vagrant/subversion-1.9.4.1/apache2/conf

step II → Add repository to configuration file

CD apache2

cd conf

vi httpd.conf

use shift+g to go to end ther edit

SVNPath "home/vagrant/gr"

→ place repository name instead of default name

esc :wq! (save & quit)

⇒ Ase we change configuration file after restart only the changes reflect into server

⇒ To restart subversion

After login

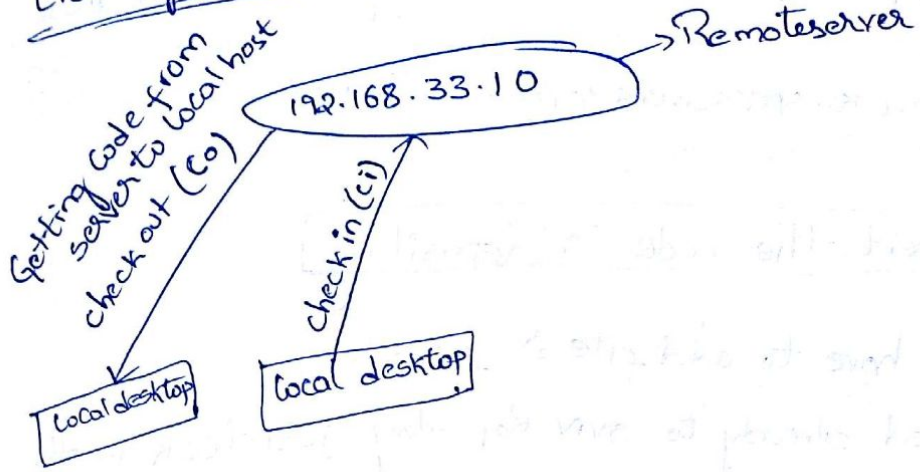
CD subversion-1.9.4.1

sh ctlscript.sh restart

→ After this process then goto from end & check it will show empty i.e., subversion – Revision 0:/

13/06/2017

Creating folders in Repository



steps

→ ① First create 1 folder directly in desktop
subversion 3
↳ open cmd

② Take the repository to folder
svn co http://192.168.33.10:8080/subversion

→ There is no direct interaction with server
(putty we are using i.e, only for login, it is backend)

→ repository downloaded (folder is created)

③ → CD subversion (Go into folder)

③ SVN · mkdir test { even we can use mkdir test bu
to add to subversion svn is use

A → added
? → you have to add using the svn add filena
comm
M → modified

④ SVN ci -m "created test folder"

→ To create new folder again repository download
is not required directly create & then check I
should be done.

Could you please create a SVN space for OCuloReferal Please call it "ServiceBus - OcculoReferal" and place it under

http://svn.uk.specsavers/oracledelivery/

How to insert the code in Repository

? → You have to add file & folders

A → Added already to SVN Repository just check in the code

M → Modified the code just check in the code

→ We have created 3 folders using above procedure

1. Trunk
2. Tag
3. Branch

* How to insert & add code in folders

step1

In windows open folder(eg Trunk we created already)

step2

create New Text file and name it. (any name)

step3

↳ Now open CMD

↳ C:\user\Gangireddy\Desktop\subversion\trunk

↳ press svn st it will show

? trunk/helloworld.txt

step4

↳ svn add helloworld.txt

step5

↳ do check in svn ci -m "code"

*copy code from trunk folder to branch folder

→ svn copy URL of source URL of destination -m "message"

eg:- svn copy http://192.168.33.10:8080/subversion/trunk
http://192.168.33.10:8080/subversion/branch/Java
-m "message"

it will create folder automatically

→ svn move URL of source URL of destination -m "message"

eg:- svn move http://192.168.33.10:8080/subversion/trunk/
http://192.168.33.10:8080/subversion/branch/Java
-m "message"

step-1
Go to putty & in home\vagrant location create 2 files
→ svn_users
→ svn_Access

step 2
Go to location CD/home/vagrant/subversion1.9.4.2/apache
& then sh htpasswd -m \home\vagrant\svn\bin
-users GR
↳ user ID
→ Enter password : GR
→ Re-enter the password : GR

sh htpasswd -m \home\vagrant\svn_users GR
↳ user ID

step 3
Go to location CD/home\vagrant\subversion1.9.4.2\apache2
\conf\ vi httpd.conf
Press shift+g and then write code after the

line where we have created one repository.

↳ Then sh ctlscript.sh restart

↳ open browser then it will ask user ID & password

SVN Merging

→ Merge is combining & adding example consider two folders ① trunk ② Branch

trunk	Branch
audio1.0	audio1.0
video1.0	video1.0
	audio1.1
	video1.1

→ we need to merge to trunk so in trunk we should get

audio1.0
video1.0
audio1.1
video1.1

steps

1. first create audio1.0 & video1.0 in trunk folder
   - → svn mkdir audio1.0 video1.0
   - → svn ci -m "message"
2. copy files from trunk i.e, audio1.0, video1.0 to branch/r1
   - → svn copy URLsource URLdestination -m "copy"
3. Create new folder in desktop eg:- new_dev & do checkout of r1
   - → svn co http://192.168.33.10/subversion/branch/r1
   - → it will down load r1 folder it contains audio1.0, video1.0
4) create audio1.1 & video1.1 in r1 folder
5) Now go to oldversion and go to trunk folder (where we need to merge)
   eg: subversion1
   svn merge http://192.168.33.10/subversion/branch/r1
6) svn ci -m "msg"
   after this command audio1.1, video1.1 are added in trunk folder

Branching strategy

* what is meant by base line & which strategy you are following in current organization

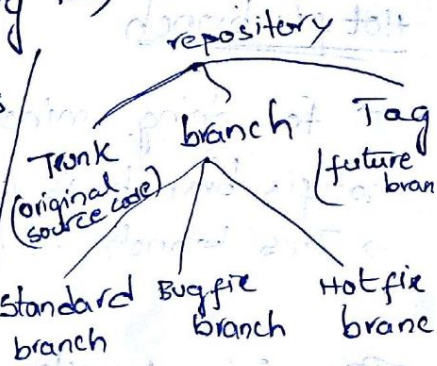
→ Currently our strategy is as per developer requirem

→ first step I will create repository

→ Then create folders in repository i.e, Trunk, Tag, branc

→ Again branches are of 3 types

1. standard branch
2. Bugfix branch
3. Hotfix branch



→ Trunk is using only for original source code

→ Tag is for future reference for particular version backup

* why Trunk, Tag, branch folders

→ As per SVN standards

→ original source code

Once code is completed with testing & code using by customers & once it go to production that is called original code which should copied in trunk. (This is used only for reference we will no do any modification to code

1. standard branch → It is like normal branch based on requirement

Here we can create master & Dev1 & Dev2, it is optional but in git only master we can create only master.

2. Bugfix branch → when ever there is issue in particular version then bug fix branch is create 1.0

→ Suppose on 20/8/2017 I released one version to customers and on 30/10/2017 1.1 version released to customer,

Currently working on 1.2 version, but a customer came with some bugs in 1.0 version.

→ To fix that bug if I work here only it is not good as 1.2 version is going on (Code is 1.2 related) to overcome this problem another ~~folder~~ branch is created

i.e, bugfixbranch-1.0#1

↳ #1 is first bug fix

Hot fix branch

→ for doing minor changes in code at that time hot fix branch is created

→ This branch is used in very rare cases

~~Tag~~ future branch

→ This is future branch

→ This is used whenever we are doing parllel development

Team A working on 1.1 — Team B working on 1.2 — at a time

Tag

A Tag is a copy of particular branch for future reference we are taking backup for particular branch.

→ What is future reference
for every version we will create Tag so if any error occurs in older version then the Tag code is used other than code in trunk.

SVN BACKUP

why backup?
for future reference

steps to create backup

1. First create new empty repository and donot do configuration for it

*first we will create backup & then load to required repository whenever we want we should give configuration then we can use

2. Create backup

sh svnadmin dump /home/vagrant/qa1 > /home/vagrant/backup.bkp

qa1 → current repository

backup.bkp → new file that create automatically name is as wish

↳ Above command done in home/vagrant/subversion1.9... /subversion/bin

3. Load

sh svnadmin load /home/vagrant/newrep < /home/vagrant/backup.bkp

newrep → new repository

④ Now configure the new repository created for backup if we want to use

SVN HOOKS : Hooks are conditions (To pass any conditions from svn we are using HOOKS concept)

In putty CD repository name

*> Hooks found in all repositories

*> Hooks are of 2 type

1. Pre-Commit hooks

*when ever developer check in the code to svn repository first it will check condition like minimum 200 characters required, minimum 100 lines required in code etc

(&)

code is to be validated before it goes to repository

2. post-commit hooks

* once we checkin the code & during execution of code it will give error based on condition

SVN workflow

The whole process starting from svn repository creation is called svn workflow.

ANT → Another Neat Tool

Rules

① Always the file name should be Build.xml

② Always default name and Target name should be same

③ when you open <project> and <target> you have to close that </project> and </target>

* ANT is a build tool
* build is compiling the code, creating packages
* Maven is also build tool & most of the companies use.

Advantages

1. We can build the code using Ant
2. This is in the format of Build.xml → developer will give build.xml
3. Using build.xml we can build the code
4. Build → Compile → package → Run

sudo yum install java 1.7

5. package format
   - Jar
   - war
   - par
   - RPm
   - Ear
   - Zip

* Ant is used to build code this is completely developer's work, & developer gives build.xml and then devops engineer checks version.

ant → is the command used to run build.xml

Steps

1. Create new folder in desktop eg: newdev
2. Open textfile in above created folder and name it as build.xml and also select type of file save as

→ normally here notepad++ is used open notepad++ create new file → save with name build.xml and type as .xml.

→ always file name should be with .xml otherwise it will throw error

→ open cmd in this folder (e.g. folder newdev)

→ give ant command

it will show build successful.

(I) Code to create new folder

```
<project name="build" default="a">
    <target name="a">
        <mkdir dir="mydev" />
    </target>
</project>
```

default name & target name should be same

</target> → closing target

</project> → closing project

when we open target, project we should close compulsory

(II) Delete folder

```
<delete dir="mydev" />
```

↳ name of directory to be delete

remaining code is same as above

**To create & delete folder in a particular location we need to give path.

```
<deletedir = "C:/user/GangiReddy/Desktop/GR" />
<mkdir dir = "C:/user/GangiReddy/Desktop/GR" />
```

folder name

Every time use forward slash if we copy the path change blackward slash to forward slash

(III) Delete files

```
<fileset dir = "C:/users/GangiReddy/Desktop/final">
    <include name = "*.txt" />
    <exclude name = "**/*.java" />
```

↳ it will delet all .txt files

Fileset

It is a task used to select particular folders & file

It has 2 attributes

1. Include → delet only specified type of files (like *.txt) it will delete all txt files
2. Exclude → apart from specified file & folder remaing all will delete

Jenkins

```
<project name = "java tasks" default = "run">
    <target name = "compile" description = "Compile the javafiles">
        <mkdir dir = "build/classes"/>
        <javac srcdir = "src" destdir = "build/classes"/>
    </target>
```

srcdir → where javafile present

destdir → this tells where to put all classfiles after compilation

```
    <target name = "package" depends = "compile" description
                                = "packaging the files">
        <mkdir dir = "build/jars"/>
        <jar destfile = "build/jars/Factorial.jar" basedir = "build/classes">
            <manifest>
                <attribute name = "Main-Class" value = "Factorial"/>
            </manifest>
        </jar>
    </target>
```

it takes .class files from build/classes and creates .jar file in build/jars folder

Every java code has main class & subclass, we need give name same in file (publicclass Factorial) → mainclass name

↳ It is first line

```
    <target name = "run" depends = "package" description
                                = "run the package">
        <java jar = "build/jars/Factorial.jar" fork = "true"/>
    </target>
</project>
```

↳ command to run jarfile present in path

fork = "true" → ant standard command to execute any build.xml

↳ for this code first create one folder in desktop and then open it & create src folder & open src folder

↳ save factorial.java

↳ outside of src save build.xml

↳ open cmd & type ant it will build & gives o/p

Jenkins

Jenkins is a open source automation server which can be used to automate all tasksort of tasks such as building, testing, and deploying software.

winscp

winscp is for file transfer ie, secure file transfer between a local and a remote compute (using FTP, SFTP

FTP File transfer protocol

SSH File transfer protocol
secure shell

*Here jenkins.war file is transfered to remote server

Java -Jar Jenkins.war → Command to start jenkins

↳ After executing this command, we should not do anything in putty, if we want to do anything open duplicate session.

Configure java

open configure java in windows search & in security press medium → ok.

Imp

Can you explain master slave concept & build machine Configuration & Node Configuration & Lable Configuration

→ After executing command Java -Jar Jenkins.war
↓
open browser and search 192.168.33.10:8080
↓
Manage Jenkins (present in left side of page in Jenkins box)
↓
Manage Nodes
↓
New Node
Here we need to give node name eg:- node2
↓
o Dumb slave → It is used whenever a fresh node is created

o copy Existing Node → copy from already created node
↓
OK
↓
No of executors → 5
↓
Remote FS root → It is a home directory for the master on the slave machine.
Goto desktop → createfolder → Take path → paste
↓
launch Method : Launch slave agents via java web start
↳ this is for windows
→ for Linux → select linux
↓
save

Manage Jenkins
↓
Configure system (Here we need to add Tools
↓
JDK
JDK installations
Name : Java-home
☐ install automatically (unselect)

↓

ANT

ADD installations

ANT
Name: ant-home

☐ Install automatically (unselect) if we select it will download latest version of ant, it will not support older version if developer wrote code using older version

↓

save

↓

Again go to Manage Jenkins

↓

☑ Tool locations

list of tool locations

Name: (ANt) ant-home

Home: give the path of

Go to thispc → properties → Environment variable → path → edit → Edit text/copy path → paste in text editor & copy path upto bin & paste.

Name: (JDk) Java-home

Home: give path of c drive → Programfiles → java → JDK.

↓

save Download window will come ok

↓

Launch close

↓

Launch

↓

Another window will come → accept → run

↓

It will open new window just minimize it & work

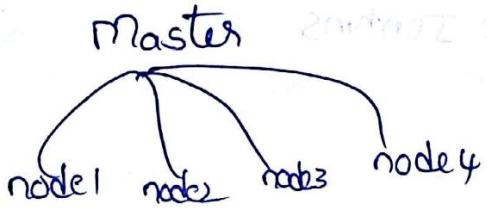
Master/slave

→ suppose one developer will write code at 10 am & build it will take 1 hour

→ Then another developer will write code at 10:30 am and need to build, it will take ½ hour to start build as above build is running

(Example 1 hour some time if it takes more time, and if client said to build urgent, it will create problem)

→ To avoid above problem slave is used to maintain load, by distributing load



Continuous Integration

Continuous integration is a process in which all development work done by different developers is integrated as early as possible

Job creation (or) New Item creation

Jenkins
↓
New Item
item Name : ci-src-trunk
⊙ Build a free-style software project
↓
OK
↓
☑ Restrict where this project can be run
Load Expression : node1 (▽ Dropdown box will come select)
↓
Source code management
↳ select subversion
Repository URL : Here select Repository URL

↓
Build
↳ select invoke ant

↓
post build action
↳ Archive the artifacts
↳ give **/*.jar

↓
save

↓
Build Now

Before this
Copy build.xml
&
src folder which contain java file

update in SVN repository and give URL to build in jenkins

Build Triggers

Scheduling jobs

Jenkins can run our Job either on demand or at a scheduled-time using the options shown below

(i) **Poll SCM**

* whenever a particular change occur in SVN and after doing checkin, build will be done automatically
* It is used whenever we need to build automatically if any change occur in SVN repository

select Job
↓
poll SCM → Enable
↓
give * * * * *
↓
save

open New folder
↓
Do checkout from SVN by getting URL from Jenkins Job configure where we gave while creating job

crontab

1st * 2nd * * * * *

* ---> year range: (1900 - 3000)
* ---> Day of the week range: (1-7, 1 standing for monday)
* ---> Month of the year range: (1-12)
* ---> Day of the month range: (1-31)
* ---> Hour range: (0-23)
* ---> Minute range: (0-59)

(ii) Build periodically

→ Build periodically builds the project periodically, here we need to give crontab

Ex:- * * * * * → every minute

If we give above expression build will happens for ever minute

(iii)

Quiet period

↳ To hold the build for particular duration Quiet period is used

↳ For example if we will give 120 sec then build will happen after 120 sec

↳ It is used in weekly release in real time

↳ If we set quiet period is 7am & 300 sec then Next build start at 7:05

↳ Files will build upto 7am then after 7am files start at next trigger i.e, 7:05

Build is parameterized

→ To pass any parameters & any IP address from jenkins we are using the option build is parameterized

→ There are different parameters available but most of the time we will use 1. String Parameter
↓
Here string name should pass

2. Test parameter
↓
Here Ip address should pass
&
Deploy condition should pass

Name : test
Default: test
Description : Job name

Name : IP address
default: 192.168.33.10, 192.168.33.11
Description : fontsize = 2, seperated by comma

Name : Deploy
Default : yes
Description : fontsize = 2, seperated by comma

Jenkins
↓
ci-src-tag
↓
Configuration
↓
☑ This build is parameterized
↓
save
↓
check now build now will replace will build with parameters

→ As per requirement we can change ip address it can be changed outside (ie, in jenkins tab) & inside (ie, configuration)

security

- Complete jenkins access security
- Jenkins access for particular job

Manage Jenkins
↓
Configure global security
↓
☑ Enable security
↓
☑ project-based Matrix Authorization strategy
↓
user/group to add : gangireddy Add
↓
It will come in red colour as we did'nt added http

Plugins :- → Jenkin compatable for plugin, it contain some plugin.
→ If we need to add extra functionality to existing software plugins are used

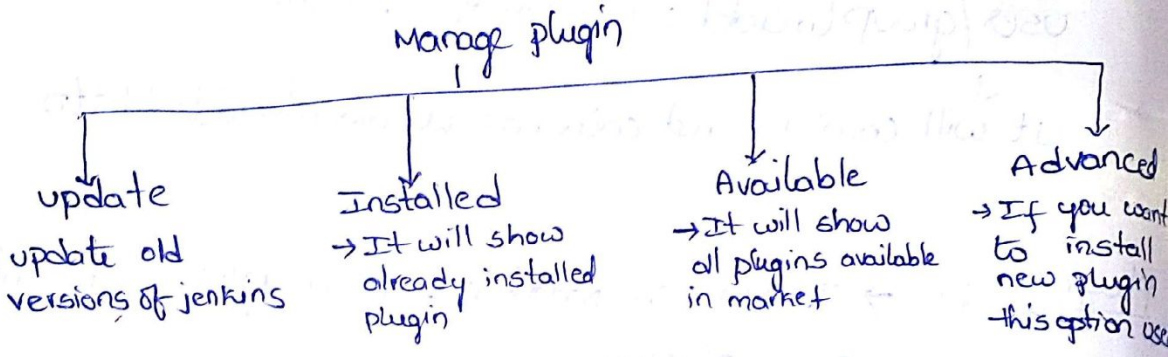
some plugins are:-

1. Green ball plugin
2. Embeddable build status
3. RHN push
4. M2 Release plugin
5. Sonar Qube
6. Rundeck
7. Multijob configuration
8. Configuration metrix
9. Docker
10. Git

How to Install plugin

open google
↓
search Greenball plugin
↓
open 1st one → wiki
↓
click on archives
↓
1.15 (version) → check version that is suitable (Compatible) for our core jenkins version
↓
download .hpi file
↓
Now open jenkins
↓
Manage jenkins
↓
Advanced
Manage plugin
↓
Upload .hpi file
↓
Restart (if required)

will get in right bottom of jenkins page.



chef

chef installation

Create 3 Machines with ip address 192.168.33.10
192.168.33.11
192.168.33.12

Then follow installation steps in chef installation file

→ In Jenkins we generated .jar, .war etc files, then next step is deployment, to do deployment automatically we will use chef.

* chef is a Configuration Management tool
* By using Code Managing Nodes is called chef
* chef will support scalability if, we can increase & decrease nodes
* chef invented in 2009 → 2013 in usage
* If an admin want to install particular Package in 10 servers admin needs to login to every server & install. If 10,000 servers it will become hard task to overcome this chef is used.
→ Chef Contains an workstation, there will be communication between workstation and nodes, this is also called bootstrapped
→ From Workstation 10,000 servers are bootstrapped.
→ Any changes made in workstation will effect in remaining servers connected.

	launch	started
Puppet	2005	→ 2008
chef	2009	→ 2013
Rundeck		
Ansible		

Difference b/w above tools

Puppet	chef	Rundeck or Ansible
50,000 servers at a time can deploy	50,000	Can deploy only 100 servers

Deployment :- Taking packages & move then install
↳ for manually installing java we use sudo yum install java 17

Rundeck probem is if there is 10 servers
→ upto 4th server build successful & in 4th an issue occur
an issue found, it will stop to build till we kill job
→ Rundeck used for small integration

chef Configuration

192.168.33.11
chef-dk

192.168.33.10
chef-server
chef-manage

192.168.33.12

chef server :- Used to do communication b/w different nodes

chef manage :- for managing complete process chef manage is used.

chef dk :- It is workstation.

chef-Installation

step:1
* First check 1gb memory available & not
Command: free -m.

step:2
* If not increase, for increase go to vagrant text file remove comment for

```
config.vm.provider "virtualbox" do |vb|
  vb.memory = "1024"
end
```

step:3
- Every server should have hostname
- chek it using the command hostname, it will give localhost.local
  change it to chef.mycompany.com (shown in step 4)
  this is our wish, but it should be same in every server otherwise connectivity fails.

step:4
* copy chef server from local using winscp. (to 10th machine chef server core 1120)

step:5
↳ vi /etc/hosts
↳ Go to insert mode & type 192.168.33.10 chef.mycompany.com

step:6: save & quit
↳ Go to path vi /etc/sysconfig/network & change the file

HOSTNAME = chef.mycompany.com

step7: sudo reboot

step8:
→ After this open browser & give 192.168.33.10 we will login as chef

step9:

* Before we copied chef-server & chef-manage now extract using the command rpm -ivh package name

sudo rpm -ivh chef-server-core-12.3.0-1.el6.x86_64.rpm

step10:

* Then reconfigure sudo chef-server-ctl reconfigure

step11:

* Now extract chef manage & reconfigure

sudo rpm -ivh chef-manage-2.2.0-1.el6.x86_64.rpm

sudo chef-manage-ctl reconfigure

step12:

As we already did browsing 192.168.33.10, it will ask userId & password so we need to configure for that run

Command → chef always use pem file & pem format is

chef create username firstname lastname emailid, password --filename /home/vagrant/user.pem

It will create automatically

↓

once setup organization

↓

It will download zip file. / extract zip file so we will get chef-repo then copy to 11th machine

↓

After download using winscp copy to 192.168.33.11

11th machine

↓

Extract dk

↓

Check chef --version

If we want to deploy from 11th machine to 12th machine server is required i.e, 10th machine was server. So to establish connection b/w them we change hostname → as shown above

↓

Now copy certificates from 10th machine → to 11th machine path is shown below

open 10th machine in putty

*go to path → cd /var/opt/opscode/nginx/ca → path ssl certificates

* give ls here we will get chef.mycompany.com.crt this is the certificate need to copy to 11th Machine

what is meant by certificate?

↳ certificates are used for authentication purpose of any website

↳ consider an example
open google.com in Internet explorer there we will see 🔒 symbol i.e, certificates

↳ If we need to access particular site we have to copy this certificates into backend server then only there is communication b/w server & backend.

↳ In this only we will mention valid date, befor the end of valid date they need to extend to work otherwise website will fail

*↳ So chef server also will have some certificates, these certificates need to copy to development kit, so that only we can access server & communicate b/w server & workstation

↓

for copying we need to create folder for this go to path

mkdir CD /chef-repo/.chef

↳ after this give ls -lart it will show hidden files (.chef) go into .chef

↓

mkDir trusted-certs (In .chef folder)

↓

In 10th machine

SCP chef.mycompany.com.crt vagrant@192.168.33.11:

secure copy — username — ip address — ↳ home location in 192.168.33.11 machine

↓

Go to 11th machine & move file copied in above

mv chef-mycompany.com.crt /home/vagrant/chef-repo/.chef/trusted-certs

↓
go to → trusted-certs location of check using ls whether chef.mycompany.com.crt is moved & not

Knife ssl check → Command to check certificates verified & not

↓
go to • chef (folder) & chef

Bootstrapping :- Establishing connection & install package in required Server

Recipes :- performing the action
single action

Cookbook :- Collection of actions & recipes

→ Here files will be saved in .rb format
ruby

open 12th server
sudo service httpd status
↳ we will get unrecognized service → means there is no httpd package
↳ usually to install package we will use sudo yum install httpd
No I dont want to install manually i want to install from chef server for this ↙

open browser 192.168.33.10
we will get this site not working

* Deploying httpd (apache) from chef server to 12th server

we need to write recipe
go to 11th machine
↓
ls
↓
go to inside chf-repo we will get one file cookbooks
↓
CD cookbooks
↓
sudo chef generate cookbook LeadApp
↓
ls → LeadApp will come
↓
CD LeadApp

↓
CD recipes

↓
sudo vi default.rb → Here we need to write some code
press Enter 5 times & write

```
package 'httpd
service 'httpd' do
        action [:enable, :start]
end
```

:wq!

for uploading

sudo knife cookbook upload LeadApp ["if it will not come"]

[sudo knife cookbook delete LeadApp] ↓ for deleting

(change path to cookbooks)

↓
sudo chef-client --local-mode --runlist 'recipe[LeadApp]'

↓
go to browser → go to chef server & refresh the page a node will come.

↓
go to path /home/vagrant/chef-repo

sudo knife bootstrap 192.168.33.12 --ssh-user vagrant --ssh-password 'vagrant' --sudo --use-sudo-password --node-name node1 --run-list 'recipe[LeadApp]'

file structure

Create → cookbook

↓
recipe → default.rb

↓
run → list

what ever file written in .rb compile then execute - If anything wrong it will stop there only

↓
upload

Upload cookbook to chef server

↓
Bootstrap

After uploading to chef server then only it will bootstrap to servers.

Build pipeline plugin

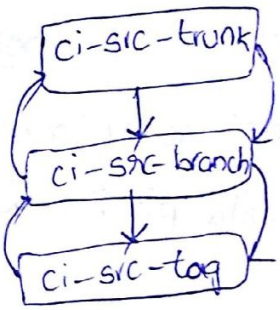
Install build pipeline

Manage Jenkins
↓
Manage plugin
↓
click on Available
search by typing Build pipeline
↓
Install without restart

Pipeline

Create 3 Jobs
↓
Configure same node1 for all 3 (individual also we can give)
↓

for ci-src-trunk
down stream is ci-src-branch
and no upstram



for ci-src-branch
upstream → ci-src-trunk
downstream - ci-src-tag

For ci-src-tag
upstream → ci-src-branch
downstream → ~~ci-src~~ No downstram.

↓
To do this go to particular job
↓
Configure
↓
ADD post-build action → select Trigger parameterized build on other projects

This option will come only if we install updated build pipeline plugin.

projects to build : ci-src-tag or branch or trunk
Trigger when build is : stable
Trigger build without parameters : ☑ → save

Now goto Jenkins

↓

Here we will get + symbol press on it

All +

ci-src-trunk

ci-src-tag

ci-src-trunk

↓

give name : mypipeline → select initial job ci-src-trunk

↓

◉ Build pipeline view

↓

number of displayed builds : 3

↓

OK

↓

Run

(Our wish, it is just build history)

If we need to do automatically we will use build Triggers. Like poll SCM

If we configure poll scm in Trunk & if any change occur build pipeline will start automatically.

MAVEN

Difference between Ant and Maven

ANT	MAVEN
1. We will write build.xml	1. We will write pom.xml
2. By using Ant we can build the code	2. By using maven we can build and release the code
3. Backup is not possible	3. Backup is possible [nexus]
4. Not possible to reuse the code	4. We can reuse the code
5. No life cycle	5. It has life cycle
6. No plugin	6. It support plugin (M2-Release plugin
7. Ant doesn't has formal conventions, so we need to provide information of the project structure in build.xml file	7. Maven has convention to place source code, compiled code etc. so we don't need to provide information about the project structure in pom.xml file
8. ANT is procedural, you need to provide information about what to do and when to do through code. You need to provide order	8. Maven is declarative, everything you define in the pom.xml file

POM.XML

POM → Project Object Model is an xml file which contains information about the project and configuration details used to build a project with maven along with its dependences:

→ POM also contains the goals and plugins. While executing a task, Maven look for the POM in the current directory.

It reads the POM, gets the needed configuration information and then executes the goal. Some of the configurations that can be specified in the pom are

- Project dependencies
- plugins
- goals
- build profiles
- build ...
- project versions
- developers
- mailing.

in maven 1 → project.xml then in later versions it renamed as pom.xml
now maven version is maven 3

Pom.xml structure

① <project> (parent path)

Parameters:
- G → Group ID → is path
- A → Artifact ID → Particular folder
- V → version → always versioning mention in snapshot like 1.1-snapshot, 1.2-snapshot

</project> → sometimes developers write dependencies

② <SCM>
192.168.33.10:8080/subversion
</SCM>

③ Distribution Management → we want to specify 3rd party repository we will use distribution management
<Nexus URL>

Installing Maven

→ first copy maven files
→ open apach-maven-3.3.9 → bin & copy path → bin also and paste in path of environment variable
This PC → properties → environment variable → path (system variable 2nd part)
→ edit then at last give ; & paste path

→ In above (new)
variable name : JAVA_HOME
variable value : path of java without bin

New

variable name : M2_HOME

variable value : Maven path without bin

↓

OK

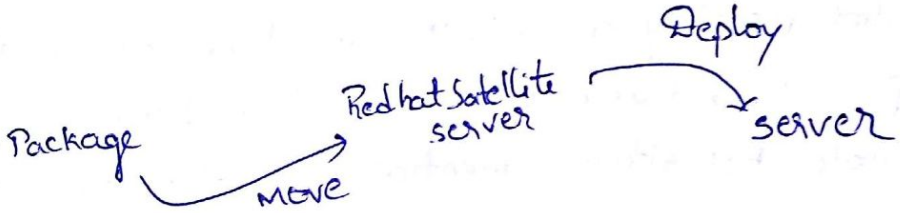
↓

OK.

Now open cmd → check mvn --version

it will give Apach Maven 3.3.9

Release



Release is moving package to Red hat satellite server & Deploying in required server

Repositories

Local Repository
M2 Repository
3rd party Repository
Backup Repository } News

* maven searches for the dependencies in the following order
* local repository then central repository then Remote Repository

Local Repository

→ It is a folder location on our machine. It gets created when you run maven command for the first time.

→ When we run a maven build, the maven automatically downloads all the dependency jar's into local repository. It helps to avoid references to dependencies stored on remote machine every time a project is build.

→ Maven local repository by default get created in User-Home directory

Central Repository

→ Maven central repository is a repository provided by the maven community. It contains a large number of

Commonly used libraries.

→ When maven does not find any dependency in local repository it starts searching in central repository

Third Party Repository

→ Also called as backup repository

→ i.e, Nexus

→ To get 3rd party repository we need to install plugin

* Maven does not find a mentioned dependency in central repository as well. It will then search in 3rd party repository, Maven will download dependency from Remote Repositories mentioned in the same pom.xml

* Some times developers will write dependencies

* Whenever developer gives pom.xml
we need to check (i) source code management tag
↓ i.e, SCM URL they should mention

How to check is copy URL from ticketing tool search in pom.xml it should reflect

(ii) Version → Current version
Next development version

(iii) Snapshot version

If any version mismatch developer request to modify by us only then check out pom.xml modify & check in

Snapshot

→ Snapshot is a backup of any version

→ Snapshot is created in 2 condition
i) Befor upgrade
ii) After upgrade

→ Indicated in -SNAPSHOT

If Current release version is 1.0 then next development release version is 1.1

If Current release version is 1.1 then next development release version is 1.2

M2 Release Plugin

Instal perform maven release plugin

→ Used to release packages
How I release is once M2 Release plugin then we will get
↳ perform maven release
we will get Login
password

* How to check current maven release version & next development version is by using perform maven release option
* If I release 1.0 in today, next development version is 1.1 (i.e pom.xml)
* Always Jenkins release version < maven version
  this is current release version ↓ this is next development i.e, in pom.xml

Daily release

Release Request

1. Build URL : Not applicable
2. SCM URL : 192.168.3.10:8080
3. If branch required - branch name : Not required
4. Release version : 1.0
5. Next Development version : 1.1
6. Artifacts Required in the release : src-trunk-1.0-SNAPSHOT
   20160720171217.noarch.rpm
7. Do you need Jenkins instance after release? yes

* As we know that pom.xml code ~~can~~ is reused. only snapshot version & URL will changes in most of the case

Daily release → only no dependencies
Weekly release → only 1 dependencies
Monthly release → More dependencies

Monthly Release

Release process is same but dependencies different, we are following the release structure

R2.5.(10) ← This is standard
(10) → It will change for every release

R2.5.10
R2.5.11
R2.5.12
R2.5.13

R2.5.13#1 ↓ In 13th version had bug so we have released another version.

Dependency → No → no need to release
Yes → If they mention yes we need to release.

How to release

Developers will give sheet

↓

perform maven release option

↓

check the version whether they mention properly or not

Here current release version 54 (once we release)

Next development version is 55 and then file the credentials and then build

→ In ticket only they will give Jenkins URL & SVN URL

Release types	Duration	SCM URL	Build URL	Dependency	version
Daily Release	5-10 min	192.168.33.10:8080 subversion/branch/ or	192.168.33.10:8080/Jenkins/job	NO	NO
		"	"	NO	"
		"	"	NO	"
Weekly Release	3-4 hours	"	"	yes	$R_2.4.10$
		"	"	NO	$R_2.4.11$
		"	"	yes	$R_2.4.11\#1$
Monthly Release	2 days	"	"	yes	$R_2.4.12$
		"	"	yes	$R_2.4.13$

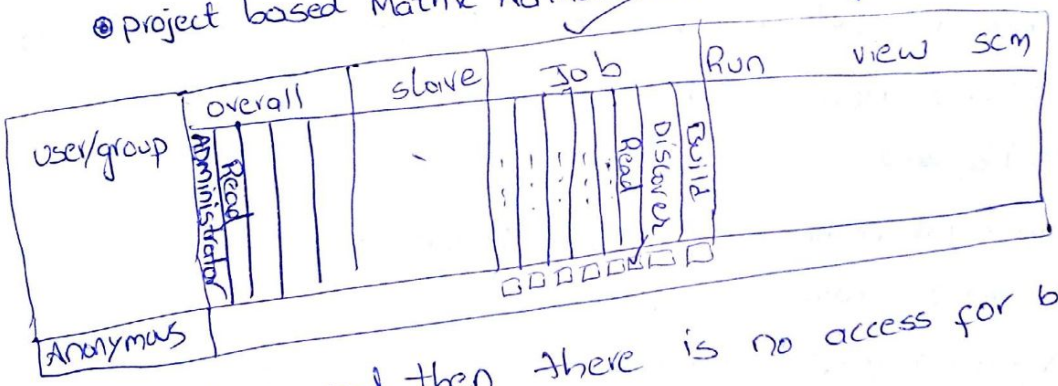
Jenkins
User Access

Usually in Jenkins we will give access in 2 ways
1. Access for particular Job
2. Access for Complete Jenkins

1. Access for particular Job

* For this we had one plugin i.e., Security Matrix
using this we can provide particular access to user
what kind of access 1. Read
or
2. Build
or
3. Configure
or
4. workspace

Manage Jenkins
↓
Configure global security
☑ Enable security
Authorization
◉ project based Matrix Authorization strategy



If we select read then there is no access for build, Configure - - -

User/group to add: [ ] add
↓
Here we need to user/group name

Blue colour will come if we add any name, It means he is in our project/team
Red colour → left Job (or) not exist

2. Access for complete Jenkins

→ Here only Devoops engineer, AM & TM had Admin access
→ If we provide admin access to Developer they do not

about some tools so they can delete any required things. It will create problem.

→ Difference b/w both is for Access for particular Job there is some access rigths & for complet Jenkins had permission to access everything.

Discard Old builds

→ Delete old builds

→ Everytime we build, a workspace will be created in backend (it will occupy memory). To delete that we will use discard old builds

→ Here will give how many days to keep build.

BUILD Request

* There will be some particular teams to raise a request in mail

Hi

This request is below R1503267 Request numbe
EMS - create - CI - Job

This request number need to copy & search in service now ticketing tool.

Details Required

1. Required job name? : CI-SIAB F-UK29
2. SVN URL? : 192.168.33.10:8080
3. OS version? : NA } Means any version is compatible
4. JAVA version? : NA }
5. Any specific configuration if required? request to create a new job for the build for below path
6. Path: http://svn.uk.specssavers.com/projects/SIAB/branches/F-UK-PRB225
   ↳ this is branch

For this

URL open & check if they gave pom.xml or not
↳ this URL open in citrix only

Next step is copy Job name

Go & search in Jenkins whether this type of job is there or not

Here same job is there with different name

F-UK-PRB28 we want F-UK-PRB29

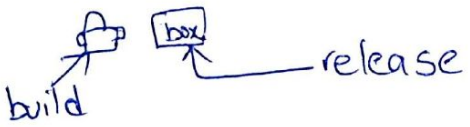
so while creating Job copy from and do modify

⊙ Copy existing item

↓

Build Now

In build



Here create Job → Build → Release

Build & Release

Ticket

1. Build URL : not applicable
2. SCM URL
3. If branch required : branch name : not required
4. Release version 1.0
5. Next development version 1.1
6. Artifacts required : ..... rpm
7. Do you need jenkins instance after release ? Yes
8) whas is / are the RHN channe - - - -
Redhat servers

ant life cycle

Compile
↓
Package
↓
Run

MAVEN LIFECYCLE

validate
Compile
test
package
verify
Install
deploy

maven support to rundeck for deployment

Java versions

→ If they mention any version just install & give path like already we did we can install old versions (JDK)

maven version

Root POM : POM.xml

clean & deploy
↳ previous build clean

RHN plugin
↳ Move package to server

release
↳ Redhat satellite server

Deployment

Hi

Please rise an request by pushing Job 4499 into test environment 10.34.104 (It has to build & deploy in 10.34.104.1)

for doing it Manually

Login to winscp using 10.34.104.1

go to Jenkins go to package & download

Then take package & copy to server

Login to putty

rpm -ivH package

10.34.104.1 Check in RHER

NEXUS

Nexus version is 2.7.0

Nexus came from opensource project called proximity

→ First release made on 2006

→ Nexus used for back up of packages

→ Nexus is a third party repository

* How to upload packages to nexus repository?
↳ It has 2 ways
1. Manual upload
2. Automation upload.

1. Manual upload

first developer rise a request and they will give

GroupID
ArtifactID
versioning
} Parameter

and also zip file

Hi
Could you please upload a zipfile located in the link below for order status to news and provide us with the path

URL: - - - -

The code has been uploaded in the following SVN repository for order status

URL: - - - -

GroupID
ArtifactID
Versioning

→ First download zip file from link

open Nexus
↓
Artifacts upload
↓
G
A
V
} give parameters take from above

some times they will not give GAV parmeters and they will give old version 1.2.0 & replace new 1.2.1. so open 1.2.0 by search & take GAV & do.

packaging : select zip
if zip is not there just type zip

↓
select Artifacts to upload
↓
select downloaded package
↓
ADD Artifacts
↓
upload.

HOW to check file is uploaded or not ?

if they gave com.specs.sis copy and replace dot with / & then search. com/specs/sis

Browse storage
↓
they will give groupID just copy & search
search:

Artifact ID is folder
↳ click on folder → open version

(or)
search folder by folder like they will give com.specs.sis
com → specs → sis

② Automation upload.

How you will configure nexus in pom.xml?

```
</distribution management>
<repository>
< id > release - profile - </id >
<name > Internal Release Repository </name >
<URL >http: // - - -  release </URL >
</repository>
<snapshot repository>
</id > snapshot - profile </id>
<name > Internal snapshot repository </name>
<URL>http: // - - - - - </URL>
</snapshot repository>
</distribution management>
```

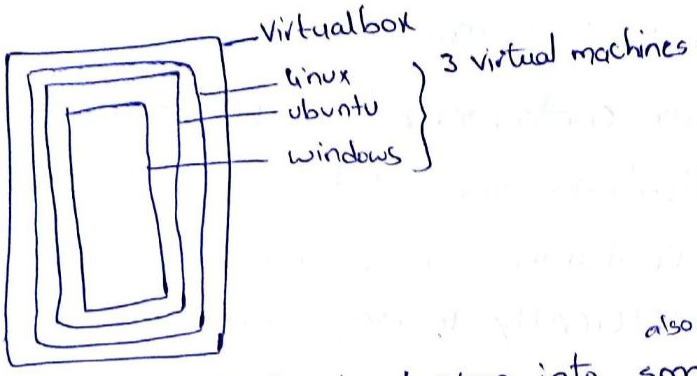
→ Configure URL in Jenkins & perform maven release
Install distribution management plugin in Jenkins

Docker

Problems before docker

1. There is a developer who developed an application, that works fine in his own environment but when it reach production, it will get some issues, this is because of defference in computing environment b/w Development & production

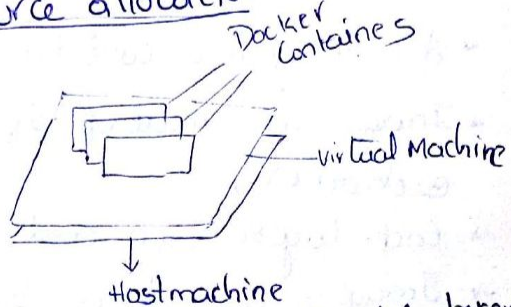
2. virtual machine / virtualbox used to operate multiple os



Consider an application which is broken into smaller services (also called as micro servics) because smaller services are each to build & maintain & also if any error occur it will not reflect whole app.

→ There micro services are using 3 VM's wherewe need to allocate RAM, Diskspace to each and every machine, suppose if one machine is serving & other 2 are sleep state at this time we wil get disadvantage in resource allocation i.e, waste of ram & diskspace. Wastage of resource allocation

→ Let us see how docker solves this problem.



There will be one host machine on the top of the host machine there will be virtual machine and on the top of VM docker containers are exist.

→ Docker containers are light light weight alternatives of VM

→ In docker containers we dont need to preallocate any ram & diskspace. It will take the ram & diskspace according to the requirement of application

→ Why we are using again VM here is the host

machine should be a linux & unix machine in order to run Docker containers, but if host machine is windows we will use virtual machine & also we can configure memory.

* 1st problem is solved by using the Docker containers throughout software development life cycle.

What is docker?

Docker is a opensource container management service. Initial release of docker was in march 2013.

Docker is a containerization platform that packages all our applications and all its dependencies together in the form of containers to ensure that our application works efficiently in any environment

uses:-

Container :- Container is a runtime instance of a docker image

Image:- Image is a standard template & empty template where all os files are formed as one unit

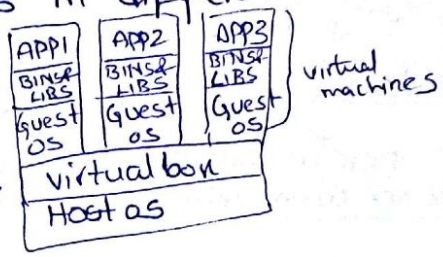
→ using one image we can create more containers

* An image is a collection of files + some meta data
* Images are made up of layers, conceptually stacked on top of each other.
* Each layer can add, change & remove file
* Images can share layers to optimize disk usage, transfer time
* Image is a Read only file system

⇒ while creating Image only we need to configure system & Docker Image is a standard template. They are the building blocks of Docker Containes

Virtualization

Virtualization is the technique of importing guest operating system on the top of a Host operating system. This allows to use multiple os in different VM's all running on the same host



Adv of Virtualization & Virtual Machines

i) Multiple os can run on the same machine
ii) Maintainance and recovery were easy in case of failure conditions.

Disadv

* Running multiple VM's leads to unstable performance
* Boot up process is long and takes time

Containerization

Containerization is lightweight alternative to full machine virtualization Containerization is however more efficient because there is no guest os here and it utilizes a host's operating system, share relevant libraries & resources as and when needed unlike virtual machines.

Adv

i) Containers on the same os kernel are lighter & smaller
ii) Better resource utilization compared to VMs
iii) Boot up process is fast

Docker Installation

open docker hub
open google & search docker
↓
create account in docker
↓
After that do Login
↓
search Jenkins in search bar

Jenkins official → Details

Here we will get command docker pull Jenkins

open putty 11th machine

sudo su

↓

docker pull Jenkins

↓

docker run -p 8080:8080 -p 50000:50000 jenkins

↓

Here we will get one password

Docker commands

Docker images → To check the images

Docker ps -a → To check image with details

Docker create -i -t [image name] → to create image

Docker start -i -a [Image ID] → To start the image

service docker status → gives docker running & not

service docker start

service docker stop

↓

Docker ps -a

↓

Docker start -i -t [ID]

↓

Then open browse 192.168.33.11:8080, it will ask password as we got in above step copy & paste then we will get Jenkins page

How to create Image & push

In putty sudo su then

docker create -i -t centos

Image name

↓

docker start -i -t ImageID

It will get in above step

↓

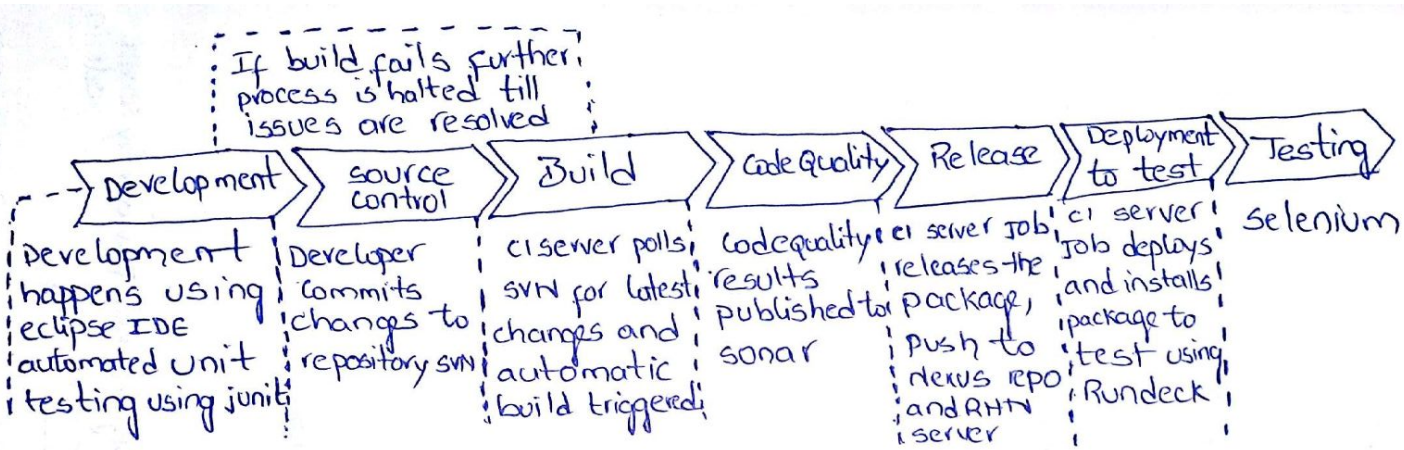
yum Install Java

↓

yum Install Git

↓

yum Install Ant



Code Commit – SVN
Build phase – Maven
Code quality – sonarqube
Release – maven
uploading channel – Redhat satellite server
Deployment – chef, Rundeck

Development
Devops
Environment
Testing
production

→ Devops team creates packages & upload to server
→ only devops team will have rights to Nexus other teams cannot access as they donot know about these, if they delete any files, it will create problem.
→ Some time environment team will do deployment by downloading packages from server. (or) upgrading like vm ware maintainance, server
→ Next team download packages from server.

git
↓
docker tag centos gangireddy/centos
↓
docker push gangireddy/centos

RHN satellite server

↳ This is deployment server

↳ we can use Tomcat also

* In RHSS we have channels
* channels are nothing but folders
* we used to locate channels & move packages
* channels are created by client & developer will provide in the ticketing tool
* Uploading packages to server in 2 ways
  1. Automation
  2. Mannual

RHN push :- uploading packages from Jenkins to Redhat satellite server.

First take channel name from ticketing tool

↓

In the server dashboard → channel

there will be so many channels, sear our folder as they provided in ticketing tool

check our folder CEH

↓

open CEH

↓

packages

↓

1441

then go to Jenkins & do configure

satellite server : 10.4.4.12.0 [IP address]

↓

if we call we need to do login so

username : test_push

password : ------

RPM Path [ ] signing key provided by server admin

channel to publish [ ]

↓

Here we need to give channels seperated by [space] How many

channels we need to push. (like if they gave 6 channels (channel, channel2, channel3,)

* Every time we need to configure Jenkins it will take more time so we will use templates.

Install template plugin

New item
↓
Job name → Free style project
↓
select use as Allow this job to be used as template
↓
configure remaining things
↓
save

* Next time create project & there we will get one option → Allow this job to be used as a template.

Rundeck

Rundeck is also configuration management tool

To deploy packages to the server there are 2 ways

1. Automation
2. Manual

rundeck installation

In 11th Machine

```
sudo su
yum install java-1.8.0
rpm -Uvh http://repo.rundeck.org/latest.rpm
yum install rundeck
service rundeckd start
```

↓

```
CD /etc/rundeck
```

↓

```
vi rundeck-config.properties
```

Here change localaddress name to our Ipaddress

```
↳ rails.serverURL=http://192.168.33.11:4440
↳ wq!
```

↓

Then run the command reboot

open browser & run the IP 192.168.33.10:4440

username : admin
password : admin

↓

Then click on new project

↓

project name

ARtch

↓

Description

ARtech

↓

Then we need to give SSH key storage path for this

go to putty

SSH-keygen

press enter

press enter

press enter

press enter

Then one random image will come

cd ~

cd .ssh/

vi id_rsa.pub

Copy total key & paste @ in above SSH key storage path

again in browser

click on create

click on ARTech Job

Create a new Job

Job name : qr-Job

Description : qr-Job

Enter the entire script to execute

script for creating Job from Jenkins to rundeck

```
deploy = @option.deploy@
if [ $deploy = "YES" ]
then
echo "The package has been pushed. waiting for two
      minutes to reflect in the channel."
sleep 120
/root/install.sh
else
```

echo "you have not opted to deploy"
fi
↓
Node save
↓
Nodes ◉ Dispatch to Nodes
Node filter [▼ .]
↓
Finally click on save below screen you will get UUID press on defination
gr_Job
Prepare and Run... Defination

UUID : [- . - - -]
↓
Copy UUID

Jenkins
☑ This build is parameterized
☐ string parameter
Name [testJob]
Defaultvalue [font size = 2]
☐ Test parameter
Name [IPaddress]
Defaultvalue [192.168.33.108, 192.168.33.11]
Description [font size = 2]

Rundeck
Job Identifier [UUID]
Job options [number = $UtilsNumber
deploy = $Deploy]
Node filters [name = $IPAddress]
wait for Rundeck Job to finish? ☑
↓
save.

steps

Install Rundeck
↓
Login to Rundeck
↓
Create the Job in rundeck
↓
write the script to deploy
↓
Copy the UUID Code
↳ go to Jenkins install Rundeck plugin
↳ In post build action step & select the rundeck

Green up
Red down

Manual deployment

open Jenkins & search Job number
↓
Download package
↓
Login to the required Machine using Winscp
↓
copy the package from local to remote
↓
yum install package name
↳ extract

Automation

First check server up & not
↓
then go to Jenkins Configure IP in Build is parameterized
↓
☑ mask password → If we want to manually run maven release to automatically Jenkin release use command

mvn -B -f $WORKSPACE/pom.xml -Ddevelopmentversion = 1.39-SNAPSHOT -Dreleaseversion = 1.33 -Dusername = username -Dtag = Release-Devops-1.34 - Dresume = false release: prepare release: perform -pre

SONAR QUBE

→ checks the code quality

→ Install sonarqube plugin in jenkins

→ Add post build action → sonarqube

↓

Just save configuration then it will reflect into your job

↓

whenever there is code issue it will stop build

Blocker, critical } very high issues

Issues with sonar (normall will get)

(1) maven issue

cant release because of local modification issue with version already release

solution

manually go to nexus delete existing package

(2) Unable to tag scm whenever we are trying to create same package again

solution

we have to change tag name (&) delete old tag → to rename svn.oldurl newurl

Both are same only last name change in new URL

(3) Error deploying the artifacts failed

whenever we are trying to upload package to nexus if package is already exists in nexus we will get this error

(4) Java.long null point exception

this is fixed by developer only

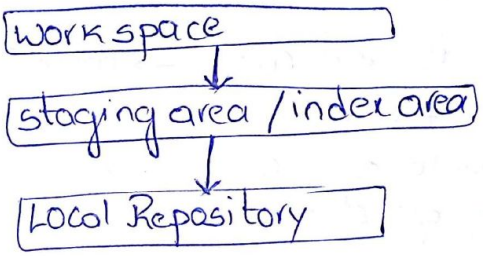
(5) Jenkin Job

If getting error so stop build & send email.

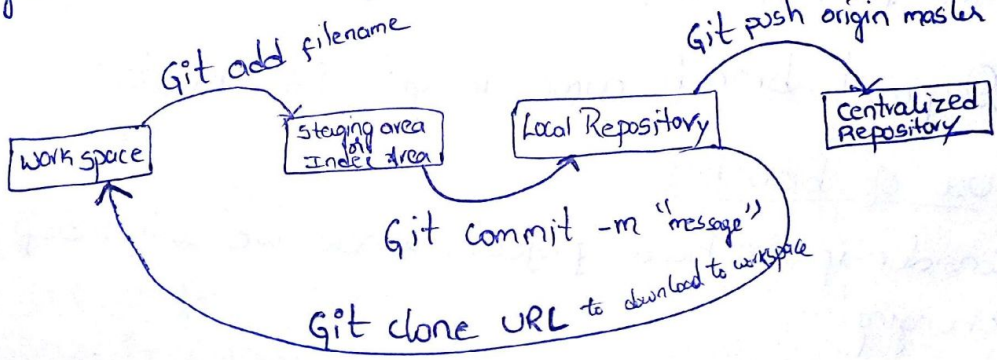
GIT

Difference between SVN and GIT

SVN	GIT
1.) SVN is a centralized repository we are directly involved in the centralized repository	1.) GIT is a distributed repository Here first we are working on laptop & then transfered to centralized repository
2.) As we are connected directly to centralized repository, if any network issue o/n w/ failure happens then we can't work.	2.) Git has 3 phases 1) work space 2) staging area / index area 3) Local Repository
3. Developers directly interact with centralized repository can create some issues	
4. There is no reviews direct checkin	



* Here we develop the code in workspace then we need to transfer to staging / Index area then to local repository
* All these stages in our local machine only after these stages we need to push to centralized repository



Git Commands

① git init → for initializing git after this it will show master, when we install git it will automatically create 1 branch i.e., master

② git clone URL

③ git push origin master

④ git commit -m "message"

⑤ git log → to check comitted files ex: git log -n 10 to see last 10 commits

⑥ git pull → It is like svn update

⑦ git ADD

⑧ git status

⑨ git branch → gives list of branches

use git status

if the file is red colour → means it is in workspace

Eg: vi test
write some code & save
check git status

if the file is in green colour → file is in index area

* In git files are stored in the form of objects
* In centralized repository also git files are stored in the form of objects.

* Defaut branch name in git is master

use of branches

Consider if we had project, where we will keep on changing

git branch master1
creating new branch

How to change branch

git checkout (branch name)

merge

Consider if we had 2 branch

master	master1
audio1.0	audio1.0
	video1.0

↓

Here do git merge master1

so we will get video1.0 also in master

Launch file

(i) Cat > file name

add data

then press ctrl+d

save

(ii) Cat >> file

add data

ctrl+d

add data to above line we added using cat > file name

To remove branch

git branch -d branch name

To create & enter into branch

git checkout -b master-2

** How to give access our repository to others

go to repository name

↓

settings

↓

Colaborations

↓

give username of required person

↓

↓

ADD collaborators

↓

a verification email will send to user

after verify you can push.