(30-35%).

PROJECT-1

21/03/22

Monolithic application deployment by using Tool stack:

Integration and Deployment :-

Tool stack :-

GIT
Jenkins
Maven
SonarQube
Nexus
Ansible
Tomcat

Scripts (or) Automated code

Jenkins Pipeline
Ansible Deployment playbook
Sonar-Project.properties

Tools :-

Maven
Sonar scanner

Integrations

Git and Jenkins
SonarQube and Jenkins
Nexus and Jenkins

Deployment

Tomcat and Jenkins

Deployment Steps :-

Stop the services
Cleanup the existing content
deploy the latest code
Start the services

} Tomcat

8:02

sonar-project.properties will look for sonar.sources → job name.

This is the name we will give for pipeline too.

Properties name = Job name

In 'CD' process Jenkins will be used as a control machine.

So, install 'Ansible' on the Jenkins machine.

↓

var/lib/Jenkins

Now, on Jenkins machine switch to 'Jenkins user'

ex:- su Jenkins

22/03/22

Similary, GOL Repo (develop)

→ Creating Tomcat on High :-

If we want to re-start from a particular build then go to the build number (Restart from a stage) option.

Now, we can choose the option and can build.

* Always choose the private Ip's. So, we can enable easily.

⇒ To know if the job is success then

curl http://(tomcat url)/Jobname/

To revert the changes { change version = latest to version = number we want to revert to }.

23/03/22

multi-branch pipeline:-

* In this pipeline process there should be Jenkins file in each branch (or) else it will not pick the branch which doesn't have Jenkins file init.

→ Branch sources
  → Add source
    → Git
      → Repo

***
→ This Pipeline is only for required Jobs
→ configure
  → Branch sources
    → Add
      → Filter by name (wildcards)
        → include : master release staging develop.

User Management in Jenkins:-

→ manage Jenkins
  → users
    → manage users
      → creati user

Now, to give access to users for different Jobs

→ manage Jenkins
  → manage plugin

→ Available.

→ Role-based authorized strategy.

* Now, enable it.

→ Manage Jenkins.

→ Configure Global Security

→ Authorization.

⊙ Role-Based strategy.

→ manage Jenkins

→ manage & assign roles.

→ manage Roles. (creating groups).

→ Assign Roles (Assign people to groups).

→ manage Roles.

→ Role to add

☑ employee

→ Add

→ Assign Roles.

→ user/group to add.

☑ developer

In the same way create useres and give the access to the useres

→ manage Jenkins

→ manage useres.

→ create user.

→ Manage Jenkins

→ Manage and assign roles.

→ Item roles (on the Job level)

→ Role to add ex:- DevJob

→ Pattern ex:- Dev.*

Now, give permissions.

Similarly, → Role to add ex:- Test Job

→ Pattern ex:- Test.*

Assign Roles.

Global roles :- provide permissions.

User/group add

add ex:- developer & tester.

also → when jobs are not with pattern.

Ie, can give "Project-based matrix Authorization strategy".

Now, give permission.

Now, go to Job

→ configuration

→ Enable project based security.

⇒ we, will always use "Role based" authorization.

Slave creation :- (Node, agent) 8:15

→ Manage Jenkins
→ configure system
→ # of executer :- 5 (Real-time 7-8).

Adding the linux machine to Jenkins (Master nodes)

→ AWS
→ launch instance. (Jenkins PPT-46).
→ launch
→ linux-slave
* Now, install Java in the slave
* Add Java variables on the machine.
* Add the user to the slave machines.

user add Jenkins
visudo
→ add Jenkins (NOPASSWD=ALL).

on Jenkins machine

su Jenkins
cd
pwd
cd .ssh/
ls

cat id_rsa.pub

→ copy the Public-key and paste it in slave Machine

Now, on slave machine

su Jenkins

cd

pwd

mkdir .ssh

cd .ssh/

vi authorized_keys

⇒ Paste the public key.

chmod 600 authorized_keys

cd ..

chmod 700 .ssh

ls -la

Now, on Jenkins

ssh jenkins@ slave IP.

⇒ Now, after the ssh connectivity.

Go to Jenkins UI

→ manage Jenkins

→ manage Node and clouds

→ New Node ex:- linux-slave

→ Permanent Agent

→ Number of executors : 5

→ Remote root directory :- /home/Jenkins

→ ** Labels : linux-slave

→ usage :

→ launch method

Host (Place the slave IP).

credentials

→ kind

SSH username with private key.

→ ID

linux-slave

→ username

Jenkins

→ Private key.

→ Now, take the private key from Jenkins machine.

cat id_rsa. copy

→ Paste the (key Private)

→ Add

→ Now, create a new Job and save.

→ Job configure

→ General.

→ restrict where this project can run. (linux-slave)

→ apply save the workspace is created on slave machine in home/Jenkins

Windows - slave (.Net)

24/04/22

→ AWS
→ windows AMI
→ launch
→ connect
→ Download Remote desktop file
→ Get Password
→ Key pair path [Choose file] → .Pem file
→ Password
→ Now connect and paste password.

→ Now, windows machine needs Java (Download Java from Google).

→ Manage Jenkins
→ manage Nodes
→ Node name window-slave
→ cl. Jenkins
support a floder for workspace in windows files.
→ Lable window-slave

→ manage jenkins
→ Config Global security } web socket.
→ Agents
→ TCP
→ Random.
[Apply] [Save]

Now [Launch] → Jnlp file is downloaded.

copy and paste the file in windows file.

see the Jenkins URL is placed correctly in.

→ manage Jenkins

→ configure system

→ Jenkins URL

* TO Run a CLI command "Jar file" is required

* we can manage Jenkins with 'Groovy scripts'

***

Backups and Recovery:-

→ Take the Backup of Jenkins daily.

→ Jenkins related history is in /var/lib/Jenkins

→ Now, take this backup.

→ from linux we have scp to copy from one to another server (or) we can set crontjob.

from UI

→ Dashboard

→ manage Jenkins

→ manage plugins

→ Available

→ thinBackup.

→ Manage Jenkins

→ Incategorized

→ settings

→ Backup directory /var/lib/jenkins

→ Backup build results

Backup build archive

Backup plugin archive

→ Backup Now

Restore :- something deleted and we want to get it back then restore will help us. (it will create the Backup on same server).

To restart app url + restart

Jenkins app

Re-usability; (shared libraries). 25/03/22

Jenkins - shared - library
- vars → .groovy extension
- src → added to class path and custom groovy code.
- resources → non-groovy files are in this folder.

def call - A simple call function.

stage params.branch - it's the branch parameter.

shared librarie to call this.

Jenkins
→ manage Jenkins
→ configure System
→ Global pipeline libraries.
→ Add

Name : Jenkins - library
Default version : master.
Retrieval method : modern scm
Source code management : Git.
Url :- Git-repo frame-work (shankar)

Now, Go to sample pipeline (take code from pdf).

Paste in pipeline script

→ How to use common code :- Shared library

Process :-

We have to create a common code and save this common code with file called file.groovy and this groovy file we have to save it under vars folder of shared lib repository and what-ever the name we used to save the file will be function name, so we can use as function name in our pipeline by giving proper inputs.