**Version Controlling:**

* This is the process of preserving all the versions of code in remote server, All the team members upload their code(checkin) remote version controlling system, The vcs accepts the code uploads from multiple developers and create in integrated projects from these uploads, The next time developers download the code from the version controlling system, it will contain all the code created by the entire team,
* Vcs ‘s also maintain all different versions of the code so that team members can switch btw any version according to the requirement
* Vcs also keep a track of who is making what kind of changes
* There are 2 types of vcs 1.centralised 2.Distributed
* **Centralized version controlling:**

Here we have a remote server into which all the team members upload the code, version controlling happened only on the server.

On the individual developers machines only code is present.

**Code**

**developer 1**

**de**

**Code**

**developer 2**

* **Distributed version controlling:**

Here we have local repository installed in every developers machine, initially code created by developers is uploaded into local repository where version controlling happens at the level of individual developers, from here the code is pushed into the remote repository where version controlling happens at the level of entire team

**Remote Server for entire Application**

**Remote Server for PB**

**Remote Server for Laavi**

* Another advantage of distributed version controlling is we can maintain sub repositories at the level of individual smaller team, These repositories are called “ Bare repositories”
* Installation of Git on windows:

1. Open <https://git-scm.com/downloads>

2. Download git for windows-Install it

3. Once git is installed we get an application called git bash which is the command prompt of git

* Installation of Git on Linux:

1. Open terminal in a linux machine
2. Update the apt repository

Sudo apt-get update

1. Install git

Sudo apt-get install –y git

* Configuration of git into machine:

1. Setting username and email globally for all users

git config –global user.name “sai krishna”

git config –global user.email [selenium.sailsrishna@gmail.com](mailto:selenium.sailsrishna@gmail.com)

check global list

git config –global --list

* Git when working on the local machine uses three components

1. Working Directory or workspace
2. Stagging Area
3. Local Repository

* Working directory is the default location where the developer creates the code, Initially all the files present in the working directory are called untracked files
* Staging Area is an intermediate buffer zone into which files are initially moved before version controlling start. The files present here are called staged files
* Local Repository is the locations where version controlling happens at the level of the individual developer. The files present here are called committed files.

1. To initialize the working directory into a git repository

cd path of\_working\_directory

git init

The above command create a hidden folder called “.git” where it stores the configuration that the necessary for git to run.

1. To send the files from working directory to staging area

git add filename

1. To send multiple files into staging area

git add file1 file2 file3

1. To send all files and folders including subfolders into staging area

git add .(dot) – dot represents current working directory

1. To bring files back from staging to untracked sections

git rm –cached filename

git reset filename

1. To send files from staging area to local repository

git commit –m “some message”

1. To check the status of the untracked and staging areas

git status

1. To see the commit history of the local repository

git log

1. To see the commit history in oneline format(only few matter)

git log –oneline

10.To see the files in commit of local repository

git show commit\_id --name-only

Note: if any files staged/untracked/commited deleted by manually even git also performs the git operations on the each and every files in that particular folder

* .gitignore:

This is a special hidden configuration file which is used for storing private files info. The file names that we store .gitignore will no longer be accessed by git

1. Create few files in the working directory

Touch file1 file2 file3 file4 file5

1. Check the git status

git status

All the above 5 files will be shown as untracked files.

1. Imagine the first four files are private and they should not be accessed by git

Cat > .gitignore

file1 file2 file3 file4

To come out of cat command ctrl+d

1. Check the status of git now

git status

It will show only file5 and .gitgnore as untracked files file1-file4 are not longer accessed by git

* **Branching in git:**

This is the feature in git which is used by developers for creating code in different functionality of code on different branches by default, git performs all activities on master branch ,advantages of using branches is code can be created in unclutted way and later code can merged with master branch

1. To see the list of local branches

git branch

To see all the branches (local and remote)

git branch –a

1. To create a new branch

git branch branch\_name

1. To move into a branch

git checkout branch\_name

1. To create a new branch and also move into it

git checkout –b branch\_name

1. To merge a branch with master first move to master and then merge

git checkout master

git merge branch\_name

1. To delete a branch that is merged

Git branch –d branch\_name

This is also called as soft delete

1. To delete a branch that is not merged

git branch \_D branch\_name

This is called as hard delete

**Note:** Whenever a new branch is created whatever is the commit history of master till that point will be copied to the new branch

**Note**: Irrespective of where a file is created or modified git always considers only the branch from where it is commited and that file belongs only to the branch where the commit happened.

* **Git merging:**

Whenever branch merged with master branch the commit history present on the branch we get merged with master branch based on time stamp

1. Create few commits on master

touch f1 f2

git add .

git commit –m “a”

1. Create a new branch called test and create commits on it

git branch test

git checkout test

touch f3 f4

git add .

git commit –m “b”

1. Check the commit history of the test branch
2. Move to master branch and create few more commits

Git checkout master

Touch f5 f6

git add .

git commit –m “c”

1. Check out commit history of master branch
2. Merge test branch with master branch & delete test branch

git merge test

git branch –d master

1. Check commit history of master branch

* **Working on remote git repository(github):**

1. Open github.com--🡪sign up for a free account
2. Sign in into that account
3. Click on Plus on top right corner -🡪Click on new repository
4. Enter some name for repository---> select public

Click on create repository

1. Go to “Push and existing repository from command line”

Copy the 1st command

Paste it in git bash

This will create a link between the local repository and the remote repository

1. Copy the second command and paste

git branch –M main

git push –u origin main

Enter username and password of github account

The entire code from the local repository will be uploaded into the remote github repository. This is called checkin.

1. Upload the new commits(files) in existing remote repository in github

git push –f origin main

1. Adding the modified files into github from git bash

Git remote add origin “github\_url”

Ex: git remote add origin <https://github.com/sabithakota/ansible-playbooks.git>

1. Then push the modified files into the github from git bash

Git push –u origin master

1. Then check the updates into the github repository

* **Downloading code from remote git repository:**

This can be done in 3 ways.

1. git clone

2. git fetch

3. git pull

* **git clone:**

This will download the entire remote repository into the local machine irrespective of weather that code is already present on our local machine or not, git clone is generally one time activity where the team lead creates the basic folder structure and uploads into the remote github. All the team members run the git clone command to download the entire remote repository into their local machine.

Synax:

Git clone remote\_repository\_url

* **git fetch:**

This will work only when there are modifications in the code i.e the code present on the remote server is different from the code present on the local repository. Git fetch downloads only the modified data and it will place that data on a remote branch. We can checkin into that remote branch and see the modifications. If the modifications are acceptable we can merge then with the master branch

1. Open github.com
2. Click on the remote repository that we uploaded
3. Select some file to be modified-🡪click on it
4. Click on Edit icon-🡪 make some changes to the file
5. Click on commit changes
6. In git bash

git fetch

The above command will download the modified files and place it on a remote branch.

1. To see the list of all the branches

git branch –a

1. Move to the remote branch

git checkout remotes/origin/main

1. See the modified files and if they are acceptable merge with master

git checkout main

git merge remotes/origin/main

* **git pull:**

This will also work only when there are modified files on the remote server but it will merge those files directly with the master branch

1. Open github.com
2. Click on the remote repository that we uploaded
3. Select some files to be modified -🡪 click on it
4. Click on edit icon -🡪 make some changes to the file
5. Click on commit changes
6. In git bash

git pull

we can see the modified files directly on the master branch

* **git rebase:**

This is feature of git which is used for performing fast forward merge i. e commits coming from branch will be projected on the top most section of master branch head will point to the top most commit which comes from branch

1. Create few commits on master

touch f1

git add .

git commit –m “a”

touch f2

git add .

git commit -m “b”

1. Create a new branch called “test ”and create few commits on it

git checkout –b test

touch f3

git add .

git commit –m “c”

touch f4

git add .

git commit –m “d”

1. Move to master and create few more commits

git checkout master

touch f5

git add .

git commit -m “e”

touch f6

git add .

git commit –m “f”

1. Rebase test branch with master branch

git checkout test

git rebase master

git checkout master

git merge test

1. Check the commit history of master

git log --online

* **Git cherry pick:**

This is used for choosing which commits we want to take into the master branch generally when we perform “git merge ” or “git rebase ” all the commits of that branch will come into master branch

Cherry pick will allow us to select only those commits that we require and merge then with master

1. Create few commits on master

touch f1

git add .

git commit –m “a”

touch f2

git add .

git commit -m “b”

1. Create a new branch called “test ”and create few commits on it

git checkout –b test

touch f3

git add .

git commit –m “c”

touch f4

git add .

git commit –m “d”

touch f5

git add .

git commit –m “e”

1. Check the commit history of test

git log --oneline

1. Identify the commits that we want to copy to master
2. TO cherry pick those commits

Git checkout master

Git cherry-pick commit\_id1 commit\_id2

* **Git stash:**

This is a feature of git which is used for leaving unfinished work and start a new functionality related coding. Further commands of git should not touch the unfinished files

1. To stash the staged files

git stash

1. To stash the untracked files and staged files

git stash –u

1. To stash the .gitignore ,untracked files and staged files

git stash –a

1. To see the list of the stashes done

git stash list

1. To bring the latest stash out from the stash area

git stash pop

1. To bring an older stash out from the stash area

git stash pop stash@{stash\_number}

* **Git amend:**

Whenever we modify a file or create new files generally we create a new commit. Instead we can put the modifications in the existing commit itself rather than creating a new commit

This can be done using git amend command

1. Check the commit history

git log –oneline

1. Modify some file or create new files

touch f10 f11

1. Send them to staging area

git add .

1. Instead of creating a new commit we can add this to the top most commit

git commit --amend -m “To most commit message”

1. Check the commit history

git log – oneline

**Note:** git reflog to see the all commits in all branches like history

* **Tagging in git:**

Tags are used for placing bookmarks on commits. They are to specify info related to who tagged, when it was tagged and why it was tagged. Generally used for releases. This helps in understandings what are the commits that are related to specific releases

Tags are 2 types

1. Light weight tags

2. Annoted tags

1. To see the list of all tags

git tag

1. To create a lightweight tag

git tag tagname

1. To create an annoted tag

git tag –a tagname –m “some message”

**Note:** Tags are always created or the topmost commit

1. To create tags for an older commit

git tag –a tagname –m “some message” older\_commit\_id

1. To see tag info

git show tagname

1. To delete a local tag

git tag –d tagname

1. To push all tags in git remote repository

git push –tags

1. To delete tags from the remote git repository

git push origin :tagname

* **Git reset:**

Reset command can be used for moving files from staging area to working directory and also from local repository to staging area or working directory it prefer 3 ways soft ,mixed and hard

**Soft reset**: files present in topmost commit will be sent to staging area, git status will show these files which moved topmost commit early as coming back to staging area but working directory will not affected. i.e it will show files as they were present at time of latest commit.(one step back)

Syntax: git reset - -soft commit

**Mixed reset**: it will directed sent the files local repository to working directory they will shown either modified or untracked files but still files contain code as present in latest commit

(Two steps back)

Syntax: git reset - -mixed commit

**Hard reset:**it will move head to older commit and it will also make changes in file as they were present in the time of older commit(three steps back)

Syntax: git reset - -hard commit

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

**Jenkins:**

This is a tool for performing CI-CD

**Stages in CI-CD**

**Stage 1 (Continues download):**

Developers creates code and upload that code into the version controlling system like Git, Jenkins is integrated with the git version controlling system in such a way that whenever develops makes any modifications to the code and uploads into Git, Jenkins gets a notifications and it downloads that the code, This is called Continues Download

**Stage 2 (Continues Build):**

The code downloaded in the previous stage has to be built and we have to create an artifact. This artifact can be in the format of war.ear exe file etc, Jenkins performs this build process using plug-in like Maven plug-in, Ant plug-in, MsBuild plug-in etc, This stage is called continues build

**Stage 3 (Continues Deployment):**

The artifact created in previous stage has to be deployed in the QA Environment. The Testing servers might be running on some application servers like tomcat, jboss etc. Jenkins will now deploy the artifact into these applications servers. This is called Continuous Deployment

**Stage 4 (Continues Testing):**

Once the applications is deployed into the testing environment Jenkins run the automation testing programs (selenium) created by the testers and check if the applications is working correctly. If the selenium programs show any errors the developers again fix the defects and upload the modified code into the Git. Jenkins will start all the above 4 stages again.

**Stage 5 (Continues Delivery):**

If the automations testing programs pass Jenkins will now deploy the applications into the production environment where the enduser or client can start accessing it .This deployment into production environment is done by Jenkins after taking approvals from the delivery team. This stage is called continues delivery.

Note: The first 4stages are called continuous integration the 5th stage is called continuous delivery

**Set and Installation of Jenkins:**

1. Update the apt repository

sudo apt-get update

1. Install java 8

sudo apt-get install –y openjdk-8-jdk

1. Install git and maven

sudo apt-get install –y git maven

1. Download Jenkins.war

wget http://mirrors.jenkins .io/war-stable/latest/Jenkins.war

Note: (<https://jenkins.io/download/> -----🡪Long term support ----🡪Generic java package)

1. To start Jenkins

Java –jar Jenkins.war

1. To access Jenkins

Launch any browser

Public ip of Jenkins\_server:8080

1. In the unlock Jenkins screen enter the initial admin password -🡪 continue
2. Click on install suggested plugins
3. Create first admin user -🡪continue
4. Click on start using jenkins

**Setup of tomcat on QA Server and Production Server:**

1. Connect to QA server instance using gitbash
2. Update the apt repository

sudo apt-get update

1. Install tomcat8

sudo apt-get install –y tomcat8

1. Install tomcat8-admin

sudo apt-get install –y tomcat8-admin

1. Open tomcat-users .xml file and set the credentials

sudo vim /etc/tomcat8/tomcat-users.xml

go into insert mode by pressing i

Enter the below statement

<user username=”intelliq” password=”myintelliq” roles=”manager-script”/>

Save and quit Esc : wq enter

1. Restart tomcat8

sudo service tomcat8 restart

**Stage -1 (Continues Download):**

1. Open the dashboard of Jenkins
2. Click on New item --🡪 enter item name as “Development”
3. Select Free style project --🡪 ok
4. Go to Source Code Management—select git
5. Enter github url where developer has upload the code

https://github.com/selenium-saikrishna/maven.git

1. Click on Apply --🡪save
2. Go to the dashboard of Jenkins---🡪 go to Development job ---🡪click on build icon

The above job will download all the code uploaded by the developer into the github repository

Check the code will download or not in location.

**Stage -2 (Continues Build):**

1. Open the dashboard of Jenkins
2. Go to the Development job----🡪Click on Configure
3. Go to Build section
4. Click on Add build step
5. Click on Invoke top level maven targets
6. Enter the maven goal as “package”
7. Click on Apply --🡪save

The above job will create an artifact from the code that was downloaded in stage 1. This artifact comes in the format of a war file

1. Location of Build file(war file):/home/ubuntu/.jenkins/workspace/Development/webapp/target/webapp.war

**Stage-3(Continuous Deployment):**

1. Open the dashboard of the Jenkins
2. Click on Manage Jenkins --🡪Click on Manage Plugins
3. Go into Available Section
4. Search for “Deploy to Container” plugin
5. Install it
6. Go to dashboard of Jenkins
7. Go to Development job
8. Click on configure
9. Go to Post Build actions
10. Click on Add post build actions
11. Click on Deploy war/ear to a Container
12. Select the path of the war file or \*\*\\*.war in war/ear field
13. Select the “qaenv” in Context path
14. Click on Add Container--🡪 select the flavor of tomcat as tomcat8
15. Click on Add—>select Jenkins
16. Go to Add Credentials-🡪 enter username and password of the tomcat server which we created in QA server in tomcat users.xml file(user username=”intelliq” password=”myintelliq”)
17. Tomcat url (http://private\_ip:8080)
18. Apply-🡪save
19. Check the browser with QA server public ip (http://public\_ip:8080/qaenv)

i.e context path which is given in deploy to container

**Stage-4 (Continues testing):**

1. Go to Jenkins dashboard create new job as Testing
2. Click on Testing job
3. Click on configure go to source code management section
4. Enter the git testing repository url in
5. Click on apply—save
6. Build job(run) the testing job
7. Go to output console check the testing file will download or not in the location( /home/ubuntu/.jenkins/workspace/Testing)
8. Go to Dashboard of Jenkins—click on Testing job
9. Click on configure
10. Go to build section
11. Click on add build
12. Click on Execute Shell
13. Write “java -jar location of testing file

(/home/ubuntu/.jenkins/workspace/Testing/testing.jar)”

Note: instead of testing file location enter (echo “Testing passed”)

1. Click on output console check weather testing passed or not

**Linking continues development with continues Testing:**

1. For continuous integration with development and testing jobs
2. Go to dashboard of Jenkins
3. Click on development job go to configure
4. Go to post build action sections
5. Click on add build
6. Go to another project
7. Write “Testing,”
8. Click on apply --save

Note: Execute the development job, testing job also executes automatically then complete the continuous integration

**Copying artifacts from Development job to testing job**

1. Open the dashboard of Jenkins
2. Click on manage Jenkins---manage plugins
3. Go to Available section-🡪search for “copy artifact” plugin
4. Click on install without restart
5. Go to the dashboard of Jenkins
6. Go to the development job-🡪click on configure
7. Go to the build actions-🡪 click on add post build action
8. Click on Archive the artifacts
9. Enter the files to be archived as \*\*\\*.war
10. Apply---save
11. Go to the dashboard of Jenkins
12. Click on testing job—click on configure
13. Click on add build step
14. Click on copy artifacts from other project
15. Enter project name as development
16. Apply-🡪save

**Stage-5 (Continues delivery):**

1. Open the dashboard of Jenkins
2. Go to the Testing job
3. Click on configure
4. Go to the post build actions
5. Click on add post build actions
6. Click on Deploy war/ear to container
7. War/ear files: \*\*\\*.war

Context path: prodenv

Click on add container --🡪select tomcat9

Enter tomcat credentials

Tomcat url: private\_ip\_prodserver:8080

1. Apply-🡪save
2. Go to he dashboard of Jenkins
3. Go to development job and click on Build icon

All the 5 stages of CI-CD would be triggered

**Creating users in Jenkins:**

1. Open the dashboard of Jenkins

2. Click on manage Jenkins

3. Click on manage users

4. Click on create users

5. Enter user credentials

**Creating roles and assigning:**

1. Open the dashboard of Jenkins
2. Click on manage Jenkins
3. Click on manage plugins
4. Click on role based authorization strategy plugin
5. Install it
6. Go to dashboard 🡪 mange Jenkins
7. Click on configure global security
8. Check enable security checkbox
9. Go to authorization section🡪click on role based strategy radion button
10. Apply -🡪save
11. Go to dashboard of Jenkins
12. Click on manage Jenkins
13. Click on **manage and assign roles**
14. Click on manage roles
15. Go to global roles and create a role “employee”
16. For this employee is overall give read access and in view section give all access
17. Go to item roles -🡪Give the role as developer and patter as “Dev .\*” (i.e developer role can access only those jobs whose name start with Dev)
18. Similarly create another role as tester and assign the patters as “Test .\*”
19. Give all permissions to tester and developer
20. Apply--🡪save
21. Click on **assign roles**
22. Go to global roles and add user1 and user2
23. Check user1 and user2 are employees or not
24. Go to item roles
25. Add user1 and user2
26. Check user1 as developer and user2 as tester
27. Apply--🡪save
28. We login into the Jenkins as user1 we can access only development related jobs and user2 can access the testing related jobs

**Notifications when uploading/downloading or any modifications occur:**

1. Open <https://catlight.io/downloads>
2. Download catlight for windows---🡪 Install it
3. Select Jenkins from the list of CI tools
4. Enter public ip of the server where Jenkins is running
5. Enter the Jenkins credentials
6. Select the jobs for which we need notifications
7. Run that job from Jenkins, we will get a popup message about the status of the Jenkins job

**Email Notifications:**

1. Open the dashboard of Jenkins
2. Go to the job for which we want to send notifications --🡪 Click on configure
3. Go to Post Build Actions
4. Click on Add post build action
5. Click on Email Notifications
6. Enter email ids of team members separated by a space
7. Apply -🡪save
8. Go back to the dashboard of Jenkins
9. Click on manage Jenkins
10. Click on configure system --🡪 search for email notifications

**Master Slave concept:**

When we want to run multiple Jenkins job at the same time it will affect the performance of the jobs since all of them Triggered on same server we can avoid this problem by using master slave setup

Slave machines are additional server which takes up the load from the master machine

Note: the main machine Jenkins will run is called as master

**Steps for creating master slave setup:**

1. Create a new AWS ubuntu 18 instance-🡪 name it as slave
2. Establish passwardless ssh between master and slave
3. Connect to the slave machine and set password for default

Ubuntu user

Sudo passwd ubuntu

1. Open /etc/ssh/sshd\_config file

Sudo vim /etc/ssh/sshd\_config

Go into insert mode by pressing i

Search for”password authentication” change it from no to Yes

Save and quit Esc:wq enter

1. Restart ssh

Sudo service ssh restart

1. Connect to master machine and generate the ssh keys

Ssh-keygen

Cd .ssh

ls

1. Copy the public keys

Ssh-copy-id ubuntu@private\_ip of slave

The above command will copy the content of the public keys and save it in a file called “Authorised\_keys” on the slave machine

1. Then connect the slave server in master server without password

Ssh ubuntu@private\_ip of slave machine

1. Then login the slave server and install java in that

Sudo apt-get install openjdk-8-jdk–y

1. Then check the version of the java in both master and slave machines
2. Download the slave.jar into the slave machine

Connect to slave

Wget <http://private_ip> of the master:8080/jnlpJars/slave.jar

1. Give execute permissions on the slave.jar

chmod u+x slave.jar

1. Create a directory which will work like workspace

mkdir workspace

1. Open the dashboard of Jenkins
2. Click on manage Jenkins🡪manage nodes
3. Click on new node-🡪 enter some name (slave)--🡪 select Permanent Agent
4. Click on new node—select configure
5. Remote root directory-- 🡪/home/ubuntu/workspace
6. Labels --🡪myslave
7. Usage-🡪use this node as much as possible
8. Launch method--🡪 Launch agent via execution of command on the master
9. Launch command : ssh ubuntu@private \_ip of the slave machine java –jar slave.jar (ssh [ubuntu@172.31.90.200](mailto:ubuntu@172.31.90.200) java –jar slave.jar)
10. Click on save
11. Go to dashboard of Jenkins
12. Identify the job which run on slave---click on configure
13. Go into general section---check on Restrict when this project can be run
14. Enter the “myslave” in Label Expression
15. Click on apply🡪save
16. Then execute the jobs for checking

**Build Pipeline Plugin:**

1. Open the dashboard of Jenkins
2. Click on Manage Jenkins
3. Click on Manage Plugins
4. Go to Available section
5. Search for Build Pipeline plugin-🡪Install it
6. Go to dashboard of Jenkins
7. Go to the section where all the jobs are present
8. Click on New View icon
9. Enter some view name--🡪Select Build Pipeline view--🡪Ok
10. Go to upstream/downstream config
11. Select the initial job as Development---🡪Apply-🡪Save

**Pipeline as code:**

This is the combination of plugin which help in CI- CD with the help of code,

Jenkins pipeline created through a file which is called as Jenkins file, This Jenkins file generally checked in version controlling system along with development code.

**Advantages:**

1. Pipeline is implemented as code as checked into the version controlling system it gives the teams the ability to edit and review the pipeline process
2. Pipeline scripts can survive both planned and unplanned restarts of Jenkins master
3. Pipeline scripts can stop wait for human approval before continuing further steps
4. Pipeline scripts support complex real world scenarios where it we can implement conditional statements, loops etc
5. Pipeline as code can implemented in two ways scripted pipeline, declarative pipeline

**Scripted Pipeline:**

node('master/slave')

{

stage('stage name in CI-CD')

{

Goovy code for implementing this stage

}

}

**Declarative Pipeline:**

pipeline

{

agent any

stage

{

stage('stage name in CI-CD')

{

steps

{

Goovy code for implementing this stage

}

}

}

}

**Scripted Pipeline In CI-CD:**

1. Open the dashboard of jenkins
2. Click on new item
3. Enter some name for new item (ScriptedPipeline)
4. Select pipeline----click on ok
5. Go to pipeline section
6. Generate pipeline code

node

{

stage('ContinuesDownload')

{

git'https://github.com/selenium-saikrishna/maven.git'

}

stage('ContinuesBuild')

{

sh label: '', script: 'mvn package'

}

stage(‘ContinuousDeployment’)

{

sh label: ' ', script: 'scp/home/ubuntu/.jenkins/workspace/Scriptedpipeline/webapp/target/webapp.war ubuntu@private ip of qa server:/var/lib/tomcat8/webapps/qaenv.war

}

stage('ContinuesTesting')

{

git 'https://github.com/selenium-saikrishna/TestingNew.git'

sh label: '', script: 'echo "Testing passed"'

}

stage(ContinuesDelivery)

{

input message: 'Waiting for Approval fromDeliverymanager!',submitter:'Prasad'

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/Scriptedpipeline/webapp/target/webapp.war ubuntu@private ip of Prod server:/var/lib/tomcat8/webapps/prodenv.war'

}

}

Note: Here prasad is admin for giving permissions to delivery

Create user as prasad give admin permissions

Scriptedpipeline:

QA server:

1. Set password for instance

sudo passwd ubuntu

enter new password

sudo vim /etc/ssh/sshd\_config

make passwd authentication yes

Jenkin server:

1. Copy the ssh keys in QA server

Ssh-keygen

Ssh-copy-id ubuntu@ip of QA server

Cat > newfile

Enter some data in that file

1. Copy the newfile from jenkin server to QA server named as newfile to file1

scp newfile ubuntu@ip of QA server:/tmp/file1

1. Go into QA server check the file1 having content or not

cd /tmp

ls

1. Copy the below code into the Jenkins scripted pipeline page

scp /home/ubuntu/.jenkins/workspace/scriptedpipeline/webapp/target/webapp.war ubuntu@private\_ip of QA server:/var/lib/tomcat8/webapps/qaenv.war

1. Go into QA server give write permissions to tomcat8 file

cd /var/lib

ls –ld tomcat8/

sudo chmod –R o+w tomcat8/

1. browse the QA server DNS name:9090/qaenv

**Declarative pipeline:**

pipeline

{

agent any

stages

{

stage('ContinuesDownload')

{

steps

{

git 'https://github.com/selenium-saikrishna/maven.git'

}

}

stage('ContinuesBuild')

{

steps

{

sh label: '', script: 'mvn package'

}

}

stage(‘ContinuousDeployment’)

{

steps

{

sh label: ' ', script: 'scp /home/ubuntu/.jenkins/workspace/Scriptedpipeline/webapp/target/webapp.war ubuntu@private ip of

server:/var/lib/tomcat8/webapps/qaenv.war

}

}

stage('ContinuesTesting')

{

steps

{

git 'https://github.com/selenium-saikrishna/TestingNew.git'

sh label: '', script: 'echo "Testing passed"'

}

}

}

post

{

success

{

input message: 'Waiting for Approval from Delivery manager!',submitter:'Prasad'

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/Scriptedpipeline/webapp/target/webapp.war ubuntu@private ip of

Prod server:/var/lib/tomcat8/webapps/prodenv.war'

}

failure

{

mail bcc: '', body: 'Jenkins Job Failed',cc: '', from: ''

replyTo: '', subject: 'Jenkins Build Status',

to: 'ganadam.saikrishna@gmail.com'

}

}

}}

**Pushing Jenkins file along with development code into the remote git hub repository:**

1. Open dash board of Jenkins

2. Click on new item

3. Enter item name

4. Select pipeline--click on OK

5.Go to pipeline section---in definition section select pipeline script from scm

6. In scm section select git----enter git hub repository url wr developer uploaded the code

7.Go to build triggers-----click on Poll SCM----enter 5 start (\*\*\*\*\*)

**Multibranch pipeline:**

Developers create separate branches for different functionalities each branch will contain the code related to that functionality this specific functionality code has to be downloaded it should be build and tested.

In order to do this we create separate Jenkins files for every branch and this Jenkins file will contain steps that are necessary for performing CI on that branch

**Developer’s activity:**

1.git clone https://github.com/selenium-saikrishna/maven.git

2. cd ro the cloned\_maven\_project

3.rm -rf .git

4. Initialise a new git repo

5. Send all files into the stagging area and local repository

git add .

git commit -m "a"

6. Create a new branch

git branch master

7. Make some minor modifications to the jenkins file on master

vim jenkinsfile

Make changes like changing the stage tittle etc

8.Commit the changes

git add .

git commit -m "b"

9. Move to loans branch ans make changes to the jekinsfile

git checkout loans

vim jenkinsfile

Make minor changes

10.Commit the change on the loans branch

git add .

git commit -m "c"

11.Move to the master

git checkout master

12. Create a remote repository and push the all branches

a)Openthe github.com

b)Create a new repository

c)Link the local repository with remote repository

d)To puch all the branches

git push -u origin –all

**Jenkins admin activity:**

1. Open the dashboard of the jenkins ---create a job give some name(multibranch pipeline)---click on OK

2.Go into branch sources ----click on add source ---select git---enter the git hub repository URL

3.Go to Scan multi branch pipeline triggers---check periodically----select intervel as 1 minute--click apply---save

**Docker:**

**Virtualization:**

Here we have a bare metal (new H/W) where we install a host OS and on top of the Host OS we install a software called hypervisor (Vm ware Esxi,Citrix Xen,MicroSoft HyperV etc).On the top of the hypervisor we install the guest OS and on top of the guest OS we install the applications that we want. The problem here is these applications have to pass through lot of layers to access the hardware resources.

**Jenkins Oracle**

**Guest OSGuestOS**

**Windows RHEL**

**Hyper visor**

**Host OS**

**Hard Ware (Bare metal)**

**Containerization:**

Here we have bare metal on which we have the host OS and on the hosts OS, we install a s/w application called docker engine. On the docker engine we can run any application in the form of containers, Docker performs “Process isolation” i.e it remove the dependency that an application has on OS and make it directly run in the docker engine. These applications have to pass through less no of layers to access the H/W resources and no fixed amount of h/w is allocated for each container based on the process running on it

**Jenkins Oracle**

**Docker Engine**

**Host OS**

**Hard Ware (Bare metal)**

Docker can be used at all the stages of build--🡪ship---🡪run

i.e we can use docker in development environment, QA environment, Stagging and production environment creation.

Docker comes in 2 flavors

1. docker ce (community edition)
2. docker ee (enterprise edition)

**Installing docker on Windows:**

1. Open https://docs .docker.com/docker-for-windows/install/
2. Install docker for windows
3. Open power shell and execute the docker commands

**Installing docker on linux:**

**Docker images and containers**:

An image is collection of binaries and libraries which are necessary for one software application to run it contain basic configuration file that are required for that application.

**Container:** Running instance of image is called as container**,** any number of containers can be created from one image

**Docker Host:** The window or linux server where docker is installed and where we run the containers is called docker host.

**Docker client:** This software is used for accepting docker commands and pass them to background process called docker daemon,on window machine powershell act like docker client on linux machine the terminal acts like docker client

**Docker daemon:** This is background process which is responsible for accepting the docker commands and pass these commands to either docker images or containers or docker registry

**Docker registry:** This is server where all the docker images are stored, These are 2 types public and private

**Public registry:** Public registry cloud site of docker(hub.author.com), Where we create an account and upload docker images

**Private registry:** Private registry is created within our organization servers and it can be access only by our team members.

**Working on docker images:**

1. To download a docker image

docker pull image\_name

1. To upload a docker image into docker registry

docker push image\_name

3. To tag a docker image with a localregistry

docker tag image name localregistry\_name

1. To search for an image on docker hub

docker search image\_name

1. To delete an image

docker rmi image

6.To create an image from a container

docker commit container\_name/container\_id new\_image\_name

7.To create an image from a docker file

docker build -t new\_image\_name

8.To delete all images,networks not associated with any containers

docker system prune -a

9.To see the list of all images in docker host

docker images or docker image ls

**Working on containers:**

1.To see the list of all running container

docker container ls

2.To see the list of all containers running and stopped

docker ps -a

3.To stop running container

docker stop container\_name/container\_id

4.To start a stopped container

docker start container\_name/container\_id

5.To restart a running container

docker restart container\_name/container\_id

6.To restart a container after 10 seconds

docker restart -t container\_name/container\_id

7.To delete a stopped container

docker rm container\_name/container\_id

8.To delete a running container

docker rm -f container\_name/container\_id

9.To stop all running containers

docker stop $(docker ps -aq)

10.To delete all stopped containers

docker rm $(docker ps -aq)

11.To delete all containers running and stopped

docker rm -f $(docker ps -aq)

12.To get detailed info about a container

docker inspect container\_name/container\_id

13.To see the logs generated

docker logs container\_name/container\_id

14.To see the port used by the container to communicate with docker host

docker port container\_name/container\_id

15.To go into bash terminal in a container which went into background

docker attach container\_name/container\_id

16.To execute any command in a container from outside the container

docker exec -it container\_name/container\_id command\_to\_be\_executed

Eg: To launch bash shell in a container

docker exec -it container\_name/container\_id bash

17.To create a container from a docker image

docker run image\_name

**Run command options:**

--name=Used for giving a name for the container

-it=used to open interactive terminal in the container

-v=used to attach a external directory or device as a volume to the container

--volume-from =used for sharing volume between different containers

-e = used for passing environment variables

-p = used for port mapping

Ex: -p 8080:80 Here 8080 will be docker host port(external port) and 80 will be container port(internal port)

-P =Used for automatic port mapping. The internal port of container will be linked with an external port that is automatically assigned by docker

and this number will be greater than 3000

-d =used to run the container in detached mode as a background daemon

--link =used to link multiple containers to create multi container architecture

--network =used to start a container on a specific network

-rm =used to delete a container on exit

--memory = to specify an upper limit on the amount of memory that can be used by a container

--cpus =To specify the no of cpus that can be used by container

**Working on networking:**

1.To see the list of available networks

docker network ls

2.To create a new network

docker network create --driver network\_type network\_name

3.To delete a network

docker rm network\_name/network\_id

4.To get detailed info about a network

docker network inspect network\_name/network\_id

5.To make a running container to connect to a specific network

docker network connect network\_name/networl\_id container\_name/container\_id

6.To remove a container from a network

docker network disconnect network\_name/network\_id container\_name/container\_id

**Docker Volumes:**

1. To see the list of docker volumes

docker volume ls

2.To create a volume

docker volume create volume\_name

3.To delete a volume

docker volume rm volume\_name

4.To get detailed info about a volume

docker volume inspect volume\_name/volume\_id

**Usecase-1:**

* Start tomcat as a container,run it in detached mode and also perform port mapping.

docker run --name mytomcat -d -p 7070:8080 tomcat

* To see the hom page of tomcat

Launch any browser

public\_ip\_of\_dockerhost:7070

**Usecase-2:**

* Start Jenkins as a container and perform automatic port mapping

docker run --name myjenkins -d -p Jenkins

* To check the port of jenkins

docker port myjenkins

* To see the home page of jenkins

Launch any browser

public\_ip\_of\_dockerhost:port\_number captured from above command

Note: to open interactive terminal in app

Docker exec –it contain\_id /bin/bash

**Usecase-3:**

* Start contos as container and open interactive terminal in it

docker run --name mycentos -it centos

* To come out of the centos container

exit

**Usecase-4:**

* Start mysql as a container and create some tables in it

docker run --name mydb -d -e MYSQL\_ROOT\_PASSWORD=intelliq mysql:5

* To open interactive terminal in the container

docker exec -it mydb bash

* To login into the mysql as root user

mysql -u root -p

Enter password: intelliq

* To see the list of available databases

show databases;

* To switch into anyof the above displayed databases

use db\_name

* To create emp and dept tables here

Open https://justinsomnia.org/2009/04/the-emp-and-dept-tables-for-mysql/

copy the code for emp and dept table creation and paste in mysql container

* To see the records in these tables

select \* from emp;

select \* from dept;

**Linking of containers:**

Multiple docker containers can be liked with each other to create

Multicontainer architecture this can be used for setting up dev environment, testing environment, production environment etc

This can be done in following ways:

1. Using --link option

2. Using docker -compose

3. Using docker networking

4. Using python scripting

**Using --link option:**

This is the deprecated way of linking docker containers and it might not support

Newer versions of docker images

**Use case:**

Start 2 busybox containers and link both of them check if they can ping each other

1. Start busybox as a container and name it c1

docker run --name c1 -it busybox

2.Come out of the c1 container without exit

ctrl+p,ctrl+q

3.Start another busybox container

docker run --name c2 -it --link c1:c1-alias busybox

4 In the c2 container

ping c1

It should ping.

* Create dev env where mysql is linked with wordpress container

wordpress developer can create

1 start mysql as a container

docker run --name mydb -d -e MYSQL\_ROOT\_PASSWORD=intelliq mysql:5

2 start wordpress as a container and link with mysql container

docker run --name mysite -d -p 8888:80 --link mydb:mysql wordpress

3 To access the home page of wordpress

Launch any browser

public\_ip\_of\_dockerhost:8888

4 Install wordpress ---->A developer should be able to create a wordpress website

**Usecase:**

Create an environment where Jenkins container is linked with 2 tomcat containers 1 for QA and 2 is prod server

1. Start jenkins as container

docker run --name devserver -d -p 5050:8080 jenkins

2.To launch the home page of jenkins

open any browser

public\_ip\_dockerhost:5050

1. Start tomcat as a container and name it as qaserver,link with jenkins server

docker run --name qaserver -d -p 6060:8080 --link devserver:jenkins tomcat

4 .To launch the home page of tomcat qaserver

open any browser

public\_ip\_of\_dockerhost:6060

5.Start tomcat as a container and name it as prodserver,link with jenkins server

docker run --name prodserver -d -p 7070:8080 --link devserver:jenkins tomcat

6 .To launch the home page of tomcat qaserver

open any browser

public\_ip\_of\_dockerhost:7070

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

**Master slave concept in Docker:**

1 .Start jenkins as a container

docker run --name master -d -p 5050:8080 jenkins

2. To launch the home page of jenkins

Open any browser

public\_ip\_of\_dockerhost:5050

3. Identify the ip address of jenkins container

docker inspect master

4. Create a ubuntu container andlink with jenkins container

docker run --name slave -it --link master:jenkins ubuntu

5. In the ubuntu slave container

a) ubuntu the apt repository

apt-get update

b)Install wget

apt-get install -y wget

c)wget <http://ipaddress_of_jenkins_container:8080/jnlpJars/slave.jar>

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

**Create architecture for implementing LAMP:**

Where mysql container is linked with tomcat container and php container

1. Start mysql as container

docker run --name mydb -e MYSQL\_ROOT\_PASSOWRD=intelliq -d mysql:5

2. Start tomcat as a container

docker run --name apache -d -p 9090:8080 --link mydb:mysql tomcat

3. Start php as a container

docker run --name php --link mydb:mysql --link apache:tomcat -d php:7.1-apache

4. To check if php container is linked with tomcat container and mysql container

docker inspect php

**Use case:**

Create testing environment where selenium hub container is linked with 2 node containers, One node with Firefox and another node with chrome

The tester should be able to perform selenium cross browser testing in this network

1. Start selenium hub as container

docker run --name hub -d -p 4444:4444 selenium/hub

2. Start a chrome node and link with the hub container

docker run --name chrome -d -p 5901:5900 --link hub: selenium selenium/node-chrome-debug

3. Start a firefox node and link with the hub container

docekr run --name firefox -d -p 5902:5900 --link hub: selenium selenium/node-firefox-debug

1. The above 2 containers are GUI containers

To view then

a) Install VNC viewer

b)Enter public\_ip\_of dockerhost:5901 or 5902

c) Enter password: secret

**Docker volumes**:

Containers are ephemeral(temporary) but the data that it processes should be

Permanent, This can be done using volumes

Volume is external folder or device which is mounted on container and whatever data is created within the volume will persist even after container is deleted, volumes can we implemented in 2 ways

1. Simple docker volumes
2. Docker volume containers

**Simple Docker Volumes**:

These are used only we want preserve data and this data cannot be reused in some other containers

1 .Create a directory called /data and Mount it as volume on to ubuntu container

mkdir /data

2 .Create an ubuntu container and mount /data as volume

docker run --name c1 -it -v /data ubuntu

3 .In the ubuntu c1 container create some files in the mounted directory

cd data

touch f1 f2

exit

4. Identify the mounted location of the container

docker inspect c1

Go to the "Mounts" section and copy the"source" path

5. Delete the container

docker rm -f c1

6. Check is the files are still present

cd "source\_path\_copied\_from\_step4"

ls

**Docker volume containers:**

These are sharable docker volumes, these volumes can be used for sharing between multiple containers, they can be implemented using run command option called --volumes-from

**Use case:**

1. Create a directory /data

mkdir /data

1. Start centos as container and mount /data on it

docker run --name c1 -it -v /data centos

1. In the c1 container go into /data and create some files

cd data

touch f1 f2

Come out of the c1 container without exit ctrl+p,ctrl+q

1. Create another centos container c2 and this should be the volume used by c1

docker run --name c2 -it --volumes-from c1 centos

1. In the c2 container go into data folder and Create some files

cd data

touch f3 f4

Come out of the c2 container without exit ctrl+p,ctrl+q

1. Create another centos container c3 and this should use the volume used by c2

docker run --name c3 -it --volumes-from c2 centos

1. In the c3 container go into data folder and create some files

cd data

touch f5 f6

come out of the c3 container without exit ctrl+p,ctrl+q

1. Go into any of the three container we can see the all files

docker attach c1 or c2 or c3

exit

1. Identify the mounted location

docker inspect c1

Go to "Mounts" section and copy the "Source-path"

1. Delete all the three containers

docker rm -f c1 c2 c3

1. Still the data will be present on the host

cd "Source\_path\_copied\_from\_step\_9"

ls

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

**Docker Compose:**

This is used for creating multi container architecture using YAML files,The advantage of using docker compose over regular commands is reusability, we can create necessary Dev ,QA or Testing environment from the level of single YAML file rather than using individual docker commands

**Installing docker compose:**

1. Open https://docs.docker.com/compose/install/

2.Go to linux section

3. Copy and paste the below 2 commnds

sudo curl -L

"https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$ (uname-m)" -o/usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

4. To check the version of docker compose

docker-compose –version

**Docker-compose commands:**

1. To start the service docker compose

docker-compose up -d

2. To stop the services

docker-compose stop

3.To start multiple docker compose files

Docker-compose –f filename.yml up -d

**Sample YML file**:

----

intelliq:

trainers:

sai: Devops

sheshi: AWS

coordinators:

laxmi: Devops

priyanka: AWS

.....

* docker compose file for create dev environment where mysql container can be linked with wordpress container

vim docker-compose.yml

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

version: '3'

services:

mydb:

image: mysql:5

environment:

MYSQL\_ROOT\_PASSWORD: intelliq

mysite:

image: wordpress

ports:

-5050:80

links:

-mydb:mysql

....

* create docker compose file for implementing CI-CD setup where Jenkins containeris linked with 2 tomcat containers

vim docker-compose.yml

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

version: '3'

services:

devserver:

image: jenkins

ports:

- 6060:8080

qaserver:

image: tomcat

ports:

- 7070:8080

links:

- devserver: jenkins

prodserver:

image: tomcat

ports:

- 9090:8080

links:

- devserver:jenkins

....

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

* Create a docker compose file for setting up testing environment where selenium hub container is linked with 2 node containers, one with Firefox installed another is chrome installed.

vim docker-compose.yml

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

---

version: '3'

services:

hub:

image:selenium/hub

ports:

- 4444:4444

chrome:

image:selenium/node-chrome-debug

ports:

- 5901:5900

links:

- hub:selenium

firefox:

image: selenium/node-firefox-debug

ports:

- 5902:5900

links:

- hub:selenium

...

* Docker file is implementing LAMP architecture:

Vim docker-compose.yml

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

version: '3'

services:

mydb:

image: mysql:5

environment:

MYSQL\_ROOT\_PASSWORD: intelliq

apache:

image: tomcat

ports:

- 9988:8080

links:

- mydb:mysql

php:

image:php:7.1-apache

links:

- mydb:mysql

- apache:tomcat

...

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

* Create a docker compose file for setting up testing environment where selenium hub container is linked with 2 node containers 1 with Firefox installed another is chrome installed.

Version: ‘3’

services:

hub:

image:selenium/hub

ports:

- 4444:4444

chrome:

image:selenium/node-chrome-debug

ports:

- 5901:5900

links:

- hub:selenium

firefox:

image: selenium/node-firefox-debug

ports:

- 5902:5900

links:

- hub:selenium

....

* Docker file is implementing LAMP architecture:

---

version: '3'

services:

mydb:

image: mysql:5

environment:

MYSQL\_ROOT\_PASSWORD: intelliq

apache:

image: tomcat

ports:

- 9988:8080

links:

- mydb:mysql

php:

image:php:7.1-apache

links:

- mydb:mysql

- apache:tomcat

...

**Creating docker customized docker images:**

This can be done in 2 ways

1. docker commit command

2 .docker file

**Use case:**

Docker commit :

Start ubuntu as container and install git in it,exit from the container ,save the container as image

Create the new container from the newly created image and we should find git already install in it.

1. start ubuntu as container

docker run --name c1 -it ubuntu

2 In the c1 container install git

apt-get update

apt-get install -y git

exit

3. save the container as in image

docker commit c1 ubuntu-git

4. Delete the c1 container

docker rm -f c1

5. Check is the ubuntu-git images is created or not

docker images

6 create a new container from the newly created image

docker run --name c1-it ubuntu-git

7. check if git is present or not

git --version

**Docker file:**

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

Docker images can also be created using docker file

docker file uses predefined keywords for created these images

1. FROM ---used to specify the base image from which docker file has to be created

2. MAINTAINER -------this represents name of the organization or the author who created this docker file

3. CMD --this is used for executing any command whenever the container starts, we can have only 1 cmd instructions in docker file

4. ENTRYPOINT ----every docker container has default process that is triggered whenever the container starts, entry point is used for specified that default process

5. RUN ----this is used for executing linux commands within container it is generally useful for upgrading the repositories, installing software’s, delete software’s etc.

6. EXPOSE ---this is used for specify which port should open in container, this will become internal port of the container

7. ENV ---this is used to specify what variables would be passed as env variables for container

8.VOLUME ---used to attach default volume to the container, whenever the container starts this volume will be activated without using the -v option

9.USER ---used to specify who should be the default user, who should login into the container

10. WORKDIR ----used default working directory of the current user

11.LABEL --used for attaching default name for the container

12. STOPSIGNEL ---used specify the key sequences of the

13.COPY --- used to copy the files from docker host to the container

14.ADD ---this can also be used coping files from host to container,it can also be used for downloading files from remote server

Directly into the container,

**Use case:**

Create dockerfile using nginx as base image and specify intelliq as maintainer

1. vim dockerfile

FROM nginx

MAINTAINER intelliq

save and quit :wq

2. To build an image from the above dockerfile

docker build -t mynginx .(dot)

**Use case:**

1. vim dockerfile

FROM centos

MAINTAINER intelliq

CMD ["ls","-la"]

save and quit :wq

1. To build an image from the above dockerfile

docker build -t mycentos .(dot)

3. To start a container from the above image

docker run -it mycentos

It should display ls -la output as excuted in the container

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

Create a docker file from ubuntu based image

update apt-repository and install git in it

if we start container from this image we should find git preinstalled in it

**Use case:**

1. vim dockerfile

FROM ubuntu

MAINTAINER intelliq

RUN apt-get update

RUN apt-get install -y git

save and quit

1. To build an image from the above dockerfile

docker build -t myubuntu .

3. To start a container from the above image

docker run -it myubuntu

4. Check if git is present or not

git –version

**Cache busting:**

Whenever we create images from dockerfile, docker read those instructions as stores them in the docker cache, next time

if we try to create image from the same docker file ,docker will read the instructions from the cache and it will not re-execute them

and this is time saving mechanism, the disadvantage of this process is we might end up installing packages from a repository which is updated long time back.

Ex:

FROM ubuntu

RUN apt-get update

RUN apt-get install -y git

When we build an image from above dockerfile, docker stores these steps in cache later if we modify this file by adding this statement

RUN apt-get install -y maven

It will execute only latest instruction this might result in installing git and maven from repository which is updated long time back

To overcome this problem we can use cache busting by using double Ampercent symbol (&&),

Create docker file from ubuntu base image and using cache busting update the apt repository install git and maven

FROM ubuntu

MAINTAINER intelliqit

RUN apt-get update && apt-get install -y git maven

**Use case:**

Create a docker file from Jenkins base image and make default user as root install maven in it

Steps:

1. Create the dockerfile

vim dockerfile

FROM jenkins

MAINTAINER intelliqit

USER root

RUN apt-get update

RUN apt-get install -y maven

Save and quit :wq

2. Build an image from the above dockerfile

docker build -t myjenkins .

3. Create a container from above image

docker run --name j1 -d -p 8899:8080 myjenkins

4. To open interactive terminal in the container

docker exec -it j1 bash

5.Check who is the default user and maven is installed or not

whoami

mvn –version

Note:directly open interactive terminal in Jenkins

Docker run –name j2 –p 8899:8080 –it Jenkins bash

* Create a dockerfile from ubuntu base image

and download jenkins.war into it.

steps:

1. vim dockerfile

FROM ubuntu

MAINTAINER intelliqit

ADD http://mirrors.jenkins.io/war-stable/latest/jekins.war /root

save and quit :wq

2. create an image from this dockerfile

docker build -t myubuntu .

3. create a container from the above image

docker run --name u1 -it myubuntu

4. in the ubunutu container in /root we should find jenkins.war file

cd root

ls

* Create a docker file from tomcat base image and expose 9090 as internal port of the container

1.vim dockerfile

FROM tomcat

MAINTAINER intelliqit

EXPOSE 9090

save and quit :wq

2.create and image from this dockerfile

docker build -t mytomcat .

3.create a container from the above image

docker run --name t1 -P mytomcat

4. check the ports that are opened

docker port t1

* Create a docker file from ubuntu base image install jdk download jenkins.war file and make execution of java -jar jenkins.war as the default process of container

steps:

1.vim dockerfile

FROM ubuntu

MAINTAINER intelliqit

RUN apt-get update

RUN apt-get install -y openjdk-8-jdk

ADD http://mirrors.jenkins.io/war-stable/latest/jekins.war /

ENTRYPOINT ["java","-jar","jenkins.war"]

save and quit

2. create an imge from this dockerfile

docker build -t myubunut .

3. create a container from the above image

docker run --name u1 -it myubuntu

weshould see the logs of "java -jar jenkins.war" coming up

**Docker Networking:** docker supports 4 types of networking

Bridge network: when multiple docker containers run on the same host machine they use bridge network.

Host only network: when we want create a container which can communicate with the only host machine and not with other containers, then we can use host only network.

Null or none network: This is used for create isolated containers which can’t communicate with host machine or with other containers, it is generally useful in docker security for preserving confidential data.

Overlay network: When docker containers are running in distributed environment that is on multiple docker host then we can use overlay network, This is generally used in docker swarm for performing container

Orchestration.

**Use case:**

1. Create 2 bridge networks intelliq1 and intelliq2

2. Create 3 busy box containers c1, c2 and c3

3. c1 and c2 should be running on intelliq1 network, so they can communicate with each other

4. c3 should running on intelliq2 network so it can’t communicate with both c1 and c2

5. Now put c2 container on intelliq2 network because c2 is on both intelliq1 and intelliq2 network it should be able to communicate with both c1 and c3 containers but c1 and c3 should not communicate directly with each other

1. Create 2 bridge networks intelliq1 and intelliq2

docker network create --driver=bridge intelliq1

docker network create --driver=bridge intelliq2

1. Check the list of available networks

docker network ls

1. Crete a busybox container of c1 on intelliq1 network

docker run --name c1 -it --network intelliq1 busybox

come out of the c1 container without exit ctrlp+p, ctrl+q

1. Identify the ip\_address of c1 container

docker inspect c1

5. Create another busybox container c2 on intelliq2 network

docker run --name c2 -it --network intelliq1 busybox

1. In the c2 container check if it can ping to c1

ping ip\_address of c1 (it should ping)

1. Create another busybox container c3 in intelliq2 network

docker run --name c3 -it --network intelliq2 busybox

1. In the c3 container check if it can ping to c1 and c2

ping ipaddress of c1 (it should not ping)

ping ipaddress of c2 (it should not ping)

come out of the c3container without exit ctrl+p, ctrl+q

1. Now attach c2 container to intelliq2 network

docker network connect intelliq2 c2

c2 is running on both intelliq1 and intelliq2 networks

1. Since c2 is on both the network it should ping to both c1 and c3

docker attach c2

ping ip\_address of c1 (it should ping)

ping ip\_address of c3 (it should ping)

come out of the c2 container without exit ctrl+p, ctrl+q

1. But c1 and c3 should ping each other

docker attach c1

ping ip\_address of c3 (it should not ping)

**Note:** To open interactive terminal in existing container

docker exec –it container\_name sh

**Docker registry**:

Registry is the location is where all the docker images can be stored,

This is 2 types

1. Public registry:

Public registry is hub.docker.com which is maintained by docker organization,

1. Private registry:

Private registry is created within our local server only our team members can access these images

**Uploading customized docker images into docker public registry**:

Create customized ubuntu image with java installed, push this image into docker.hub.com

**Steps:**

1. Open hub.docker.com --->signup for free account

2. Create a customized ubuntu image

a) start ubuntu as a container

docker run --name u1 -it ubuntu

b) In the ubuntu container install java

apt-get update

apt-get install default-jdk

exit

c) save the container as an image

docker commit c1 intelliqit/ubuntu\_java

3. Login into hub.docker.com

docker login

enter username and password

4. Push the customized docker image

docker push intelliqit/ubuntu\_java

**pushing the docker images to hub.docker.com**

1. login to the hub.docker.com by enter the username and password

2. go to create repository then give name to the repository then create

3. identify the image which has to be push

Docker tag image\_name user\_id/repo\_name

4.then push the image

Docker push user\_id/repo\_name

**Working on docker local registry:**

docker local registry can be created using an image called registry when we start This registry image as container it will allows us to upload or download our Customized docker images and only our team members can access these images

**Use case:**

Start docker local registry and upload an alpine image into the registry

Steps:

1. Start a local registry

docker run --name lr -p 5000:5000 -d registry

1. Download an alpine image

docker pull alpine

1. Tag the alpine image with the local registry

docker tag alpine localhost:5000/alpine

1. Push the image into the local registry

docker push localhost:5000/alpine

**Container orchestration:**

This is the process of run in docker containers in distributed environment and all these containers should communicate with each other and scale to gather, container orchestration can be implemented in

1. Docker swarm

2. Kubernates

3. Mesos

**Advantages:**

1. Load balancing of containers

2. Scaling of containers

3. Performing rolling updates of containers

4. Handling fails over scenarios

**Docker swarm:**

This is product of docker for implementing container orchestration the main machine where swarm is initialized is called as manager the other machines are called as workers, manager initial as service and distribute the workload to the worker machine

Setup of docker swarm:

1. Create 3 aws ubuntu instances

2. Name one machine as manager 2nd machine as worker1 and 3rd machine as worker2

3. Install docker in all of them

**Optional:**

open /etc/host mane and edit the hostname

vim /etc/hostname

delete the existing data and enter the manager or worker1 or worker2

restart the machine

init 6

**To initialize the docker swarm:**

1. Go to manager machine

2. Docker swarm init --advertise-addr private\_ip of manager

This command will convert the current machine as a docker swarm manager and it will also generate the token id that can be used for joining workers copy that token id and paste in all the worker instances

**Load balancing:**

When concurrent users try to access the resources parallel it generally slows down the performance of the application to handle the problem we can performing load balancing where we can distribute the load of same application on to multiple containers

**Use case:**

Start tomcat with 5 replicas and check if they are distribute manager and workers

1. Start tomcat with 5 replicas

docker service create --name webserver -p 7070:8080 --replicas 5 tomcat

1. To check if tomcat is running with 5 replicas

docker service ps webserver

1. To access the home page of any of these replicas

Launch any browser

public\_ip of the manager/worker1/worker2:7070

**Use case:**

Start mysql with 3 replicas in docker swarm and check if they are distribute manager and workers

1. Start mysql with 3 replicas

docker service create --name mydb -e MYSQL\_ROOT\_PASSOWRD=intelliq --replicas 3 mysql:5

2. To check if 3 replicas are running or not

docker service ps mydb

**Scaling of replicas in swarm:**

When business requirements increases we can increase the number of

replicas without experience in any downtime

1. Start nginx with 5 replicas

docker service create --name appserver -p 9090:80 --replicas 5 nginx

2. To check if 5 replicas of nginx are running

docker service ps appserver

3.Scale the replicas from 5 to 8

docekr service scale appserver=8

4. Check if 8 replicas of nginx are running

docker service ps appserver

Note:

* This scaling is done by docker swarm without having any downtime
* To see the list of services running in swarm

docker service ls

* To get detailed info about a service

docker service inspect service\_name

The above command will display the output in JSON file format

* To get the output in the simple English format

docker service inspect service \_name --pretty

**Perform rolling updates:**

The services which are deployed in docker swarm can be upgraded from one version to another version or they can be downgraded and all these can be done without experiencing any down time,

we can also perform rolling rollback where we can switch back from current version earlier version

**Use case:**

Start redis 3 as service in docker swarm with 5 replicas

perform rolling update and change the version of redis from

redis 3 to redis 4 later perform roll back and go back to redis 3 version

1. Start redis:3 with 5 replicas in swarm

docker service create --name myredis --replicas 5 redis:3

1. Check if redis:3 is running with 5 replicas

docker service ps myredis

1. Perform a rolling update to move from redis:3 to rediss:4

docker service update --image redis:4 myredis

1. Check if redis:3 replicas are shutdown and 5 replicas of redis:4 are running

docker service ps myredis

1. Perform a rolling rollback is move back redis:4 to redis:3

docker service update --rollback myderdis

1. Check if redis:4 replicas are shutdown and 5 replicas of redis:3 are running

docker service ps myredis

**Handling failover scenarios**:

Docker swarm always ensure that the necessary count of replicas is always

Maintained if one or more replicas containers shutdown immediately docker swarm will ensure that those replicas are recreated similarly if any of the worker machine shut’s down or get disconnected from the network docker swarm will ensure those replicas are migrated into manager or other workers

**Note:** if manager machine shuts down the entire docker swarm will collapse, to overcome this we always maintain multiple managers in docker swarm

**Use case:**

Start httpd replicas in swarm, delete one replica check if it is recreated,

drain worker1 from swarm check if all the replicas running on worker1 have moved back to manager and worker2,rejoin worker1,Go to worker2 machine and make it leave swarm check if all the replicas migrated manager and worker1

1. Start httpd with 5 replicas

docker service create --name webserver -p 8899:80 --replicas 5 httpd

1. Check if 5 replicas of httpd are running on manager,worker1 and worker2

docker service ps webserver

1. Delete one replica running on manager

docker container ls

Identify the container id of the httpd replicas running on manager

docker rm -f container\_id

1. Check if 5 replicas are still running

docker service ps weserver

1. Drain a Worker from swarm

docker node update --avaialbility drain worker1

1. Check if 5 replicas are still running on manager and worker2

docker service ps weserver

1. Make worker1 rejoin swarm

docker node update --avaialbility active worker1

1. Go to worker2 and make it leave swarm

docker swarm leave

1. Check if 5 replicas are still running on manager and worker1

docker service ps webserver

Note: To stop the services in docker swarm

Docker service rm service\_id/service\_name

**Removing workers from docker swarm**:

This can be done in 2 ways 1.draing a worker 2.worker leaving

1. Draing worker:

The manager machine can remove any worker from docker swarm using the drain option and it can also make it rejoin the swarm using the active option.

1. For manager to drain a worker1 from the swarm

docker node update --avaialbility drain worker1

1. To make this worker1 rejoin swarm

docker node update --avaialbility active worker1

1. For a worker to leave swarm

Go to worker machine

docker swarm leave

1. For a manager to leave the swarm

docker swarm leave –force

**Promoting commands:**

1. In manager machine To generate token id for a node to join in swarm as a worker

Docker swarm join-token worker1

1. To generate the token id for a node to join in swarm as manager

Docker swarm join-token manager

1. To make “worker1” node as manager

Docker node promote worker1

1. To make “worker1 ” node back to worker status

Docker node demote worker1

**Overlay network:**

docker swarm by default uses overlay network when we have multiple services deployed in docker swarm all these will run on same default overlay network and they will be able to communicate with each other

to isolate the services in docker swarm so that they cannot communicate with each other we can use virtual overlay network for each service

**use case:**

1 create 2 overlay networks

docker network create –driver overlay intelliq1

docker network create – driver overlay intelliq2

1. Check the list of available networks

Docker network ls

1. Start tomcat with 5 replicas in swarm

Docker service create –name webserver –p 6060:8080 –replicas 5 tomcat

1. Perform a rolling network update and make it run on intelliq1 network

Docker service update –network-add intelliq1 webserver

1. To check if tomcat has be updated to the intelliq1 network

Docker service inspect webserver –pretty

1. Start httpd in swarm on intelliq2 network

Docker service create –name appserver –replicas 5 –p 9090:80 –network intelliq2 httpd

1. To check if httpd is running on intelliq2 network

Docker service inspect appserver –pretty

**Docker stack:**

Stack is the group of interrelated services that share depencies with each other so that they can be orchestrated to gather single stack is capable of defining the functionality of entire application

1. To deploy a stack in swarm

Docker stack deploy –c docke-compose/stack\_filename stackname

1. To see the list of stacks

Docker stack ls

1. To see the list of tasks in a stack

Docker stack ps stackname

1. TO delete a stack

Docker stack rm stackname

1. To see the list of services in a stack

Docker stack services

Docker compose + docker swarm ===docker stack

**Use case:**

Create docker stack file where 3 replicas of workpress are linked with 1 replica of mysql

1.vim stack.yml

Version :3.1

Services:

Wordpress:

Images:wordpress

Ports:

-8899:80

deploy:

replicas:3

db:

image: mysql

environement:

MYSQL\_ROOT\_PASSWORD:intelliq

1. Deploy this file in docker swarm

Docker stack deploy –c stack.yml wordpress

1. To check if 3 replicas of wordpress and 1 replica of mysql is running or not

Docker stack ps mywordpress

Maven is build tool of java and it is used for compiling the java code and later creating an artifact ,maven also works like security mechanism where developers can use 3rd party api s in secure way ,maven is product of apache and it runs on community servers all the open source organizations upload their updated api s into this maven global repository we connect to the maven global repository download those artifacts that are necessary for the project and stored them on separate server called maven local repository all the developers connect their projects to this maven local repository and perform their development activity in this way the development code stored separately on git server and 3rd party api s required for development are stored on maven local repository this will tell can protecting code present in git from any viruses that might be present in these open sources

**Maven life cycle:**

1.prepare\_resources-

In this stage the component that are neccesary for maven project are created,

this is default stage of maven and it is executed automatically

2.validate-

In this stage maven will check whether the maven projects is created correctly

according to the filter id that was choosen

3.compile-

When java programers create a projects the default extenstion of these files will

be .java, here these .java files compile and it will create .class files,these .class fiels

also known widecode files

4.Test-

The unit test programs that created by the developers will be compile in this stage

5package-

In this stage jenkins will create artifacts which can come in the formate

of jar war/ear etc

6.install-

The artifact created by our develers can be installed into the maven local repository

7.deploy-

Here the artifact will be deployed into applications server like tomcat, jboss etc

**Configuration management:**

This is the process of configuring remote servers, which are present in geographically different locations

**Advantages:**

1. Provisioning of server:

The application that has to be installed on servers can be performed

Very easily using configuration management tools

1. Create in replica env:

In the case of disaster recovery when we can use entire data center due to natural calamity or war we can create replica data centers very easily in very less amount of time using configuration management tools

1. Handling snowflake:

Over a period of time all the servers present in data center will start behaving like snowflakes i.e they might be running on slightly different hardware and software Configurations these server information can be captured in simple setup files which will Help us creating similar servers.

1. Idempotent:

Configuration management tools are used to bring the server to particular stage called as desired stage if server is already in the desired stage configuration management tools will not Reconfigure that server

**Disadvantage:**

1. We cannot perform OS installation using configuration management tools

Popular configuration management tools:

Ansible,chef,puppet

**Ansible:**

It is open source configuration management tool, created using python

ansible will install in only 1 machine that machine is called as controller

the remote machines that we try to configuration are called as managed nodes or managed ports.

Note: ansible in created using python we should 1st install python on controller and manage nodes

**Setup of ansible:**

1. Sign in into Aws

2. Create 4 ubuntu18 instances

3. Name them controller, server1, server2 and server3

4. Connect to controller using git bash

5. Install python

sudo apt-get update

sudo apt-get dist-upgrade

sudo apt-get install -y python-pip python2.7

6.To check if python is installed or not

python --version

7.Setup the passwordless ssh in all the manage nodes

a)Connect to 1st managed node via git bash

b)Set a new password for ubuntu user

sudo passwd ubuntu

c)Open sshd\_config file and make some changes

vim /etc/ssh/sshd\_config

Go into insert mode by pressing i

Search for passwordauthentification and change yes

save and quit :wq

d)Restart ssh

sudo service ssh restart

Repeat the steps a-d on all the managed nodes

repeat the above steps on all the aws instances

8.Connect to controller via gitbash

9.Generate the ssh keys

ssh-keygen

10.Copy the ssh keys

ssh-copy-id ubuntu@ip address of managed nodes

Repeat the above step in all remaining 2 managed nodes

11.Installing ansible

a)Connect to controller

b)Up date the apt-repository

sudo apt-get update

c)Install software-properties-common

sudo apt-get install software-properties-common

d)Add the latest version of ansible ppa repository

sudo apt-add-repository ppa:ansible/ansible

e)Update the apt-repository

sudo apt-get update

f)Install ansible

sudo apt-get install -y ansible

12.Ansible stores all remote server information in file

called inventory file default location of the inventory file is

/etc/ansible/hosts

in this file we should store the remote managed nodes ip address

sudo vim /etc/ansible/hosts

Go into insert mode py pressing i

copy and paste the private ip's of all managed nodes

save and quit :wq

Ansible performs remote configuration of servers in 3 different ways

1. Adhoc commands

2. Playbooks

3. Roles

**Imp modules in Ansible:**

command:This is the default module of ansible and it is used for running

basic linux commands on the managed nodes

shell:This module is used for running commands which involved redirection and pipping, it can also be used for running shell scripts on managed nodes

ping:This is used to check if remote server can be pinged from the controller

copy:This is used for coping files and folders from controllers to managed nodes

fetch:This is used to copy files from managed nodes to controller

apt:This is used for software package management on managed nodes like installing software’s, Upgrading, uninstalling etc

yum:This is used for package management but it works on red hat linux, centos, fedora etc flavors of linux

user: This is for creating user accounts assigning home directories, assigning passwords etc

service:This can be used for starting ,stopping and restarting services

uri:This is used to check remote url is reachable from the managed nodes

git:This is used for performing git version controlling on remote managed nodes

file:This is used for creating, deleting files and directories on the managed nodes

replace:This is used for replacing specific string with some other string

stat:This is used to get detailed information about files and directories present on managed nodes

debug:This is used for getting detailed output for any module

include:This is used to call child playbooks from parent playbooks

setup:Used for capturing the remote manage nodes software and hardware information

ec2:Used for creating aws ec2 instances

docker\_container:This is used for managing docker containers on the remote servers

docker\_image:This is used for managing docker images on the remote servers

docker\_login:This is used for logging in docker registry from managed node

**Sintax of Adhoc commands:**

ansible all/group\_name/ipaddress -i inventory\_file\_path -m module\_name -a 'arguments'

1. Ansible command to see the memory information on all the managed nodes

ansible all –i /etc/ansible/hosts -m command -a 'free'

Note:/etc/ansible/hosts is the default inventory file and when using it giving -i option is not mandatory

ansible all -m command -a 'free'

Note:command module is the default module and when using it giving -m is not mandatory

ansible all -a 'free'

1. This is used for running shell scripts on managed nodes

Ansible command to install docker on all managed nodes and check which version has been installed

ansible all -m shell -a 'curl -fsSL https://get.docker.com -o get-docker.sh'

ansible all -m shell -a 'sh get-docker.sh'

ansible all -m shell -a 'docker\_version'

1. Copy: This module is used for copying file and folders into managed nodes,ansible command to copy a file from controller to managed nodes

ansible all -m copy -a 'src=file1 dest=/home/ubuntu'

a)Create multiple directories within the other and copy all of them into all the managed nodes

Create multiple directories

mkdir -p d1/d2/d3/d4

copy all this into the managed nodes

ansible all -m copy -a 'src=d1 dest=/tmp'

b) copy module can also be used for changing the permission ownership group ownership also

ansible command to change the ownership, group owner of files

ansible all -m copy -a 'src=file1 dest=/tmp owner=root group=ubuntu mode=740' –b

c) copy module can also can edit the content of the file in all managed nodes

ansible all -m copy -a 'dest=file1 content="Welcome to intelliq"'

4) Fetch: this is used for copy files and directories from managed nodes to controller

ansible to copy /etc/passwd from all managed nodes to controller

ansible all -m fetch -a 'src=/etc/passwd dest=/home/ubuntu'

5 )user:This is used for creating user accounts on managed nodes

ansible command create user in all managed nodes

ansible all -m user -a 'name=Ravi password=intelliq' -b

1. using user module we can also set user id,group id,comment,home directory,shell etc

ansible all -m user -a 'name=sabitha password=intelliq uid=1234 home=/home/ubuntu/sabitha

shell=/bin/bash comment="A regualar user"' –b

Note: To know about the module details means which arguments would pass command

Ansible-doc <module name>

Ex: ansible-doc copy

Ansible-doc user

6. apt: this module is used for installing software package management on managed nodes

a) ansible command install tree on all managed nodes

ansible all -m apt -a 'name=tree state=present' -b

state=present for installation

state=absent for uninstallaion

state=latest for upgrading the current package to latest version

1. ansible to install jdk on all the managed nodes

ansble all -m apt -a 'name=default-jdk state=present' -b

c)using apt module it is also possible to update apt repository

update\_cache=yes

ansible command to install tomcat on all managed nodes after updating the apt repository

ansible all -m apt -a 'name=tomcat8 state=present update\_cache=yes' -b

7.service :This is used for stating,stoping and restarting services

ansible command to stop tomcat on all managed nodes

ansible all -m service -a 'name=tomcat8 state=stopped' -b

state=stopped for stopped the services

state=started for starting the services

state=restarted for restarting the service

8.Git:This is used for performing git version controling on the managed nodes

ansible command to clone remote git repository into all managed nodes

ansible all -m git -a 'repo=https://github.com/selenium-saikrishna/maven.git dest=/home/ubuntu/mygit'

9.replace:This is used for changing a specific value in the file

ansible command change the default port number of tomcat8 from 8080 to 9090

ansible all -m replace -a 'path=/etc/tomcat8/server.xml regexp=8080 replace=9090' -b

restart tomcat8:

ansible all -m service -a 'name=tomcat8 state=restated' -b

10.uri:This module is used to check if remote url

ansible command check wether google is reachable from managed nodes or not

ansible all -m uri -a 'url=http://google.com status=200'

status 200 represent reachable and -1 represents unreachable

11.file:This is used for creating files and directories on the managed nodes

a)ansible command to create a file on all the managed nodes

ansible all -m file -a 'name=/tmp/file100 state=touch'

state=touch to create files

state=directory to create directory

state=absent will delete the files and directories

b)ansible command to delete the file in all managed nodes

ansible all -m file -a 'name=mygit state=absent'

**imp note :ansible command to copy files from 1 managed node to another managed node**

1.create a file in one managed node

2.Fetch that file into contoller

ansible controller\_ip -m fetch -a 'src=newfile dest=/home/ubuntu'

3.copy the file into managed node2

ansible manage node\_ip -m copy -a 'src=/home/ubuntu/managed node\_ip/newfile dest=/tmp'

**Configuring tomcat8 on all the managed nodes:**

1. install tomcat8

ansible all -m apt -a 'name=tomcat8 state=present' -b

1. Replace the port of tomcat from 8080 to 9090

ansible all -m replace -a 'regexp=8080 replace=9090 path=/etc/tomcat8/tservice.xml' -b

1. To restart tomcat8

ansible all -m restart -a 'name=tomcat8 state=restarted' -b

1. Check if tomcat8 is reachable in all managed nodes or not

ansible all -m uri -a 'url=ip\_server1:9090 state=200'

ansible all -m uri -a 'url=ip\_server2:9090 state=200'

ansible all -m uri -a 'url=ip\_server3:9090 state=200'

**Configuring apache2** **on all the managed nodes:**

1. Install apache2

ansible all -m apt -a 'name=apache2 state=present' –b

To see the hompage elinks http://private\_ip\_server

1. Edit index.html file

ansible all -m copy -a 'content="Intelliq page\n" dest=/var/www/html/index.html'-b

1. To restart apache2

ansible all -m service -a 'name=apache2 state=restrted' -b

1. check if apace2 is reachable in all managed nodes or nor

ansible all -m uri -a 'url=ip\_server1 state=200'

ansible all -m uri -a 'url=ip\_server2 state=200'

ansible all -m uri -a 'url=ip\_server3 state=200'

5.uninstall the apache2 and delete the configuring files also

Ansible all –m apt –a ‘name=apache2 state=absent purge=yes’ -b

**Ansible playbooks:**

Create playbook directory and go into the directory then open the playbook file to write content in that

1. mkdir playbooks
2. cd playbooks
3. vim playbook1.yml

write content in that yml file then save

1. run the playbook command to execute the file

ansible-playbook playbook1.yml -b

1. create playbook for configuring tomcat8 in all managed nodes

---

-name: configuring tomcat8

hosts: all

tasks:

-name:Install tomcat8

apt:

name:tomcat8

state:present

-name:change port from 8080 to 9090

replace:

regexp:8080

replace:9090

path: /etc/tomcat8/server.xml

-name:restar the tomcat8

service:

name:tomcat8

state:restarted

-name:check tomcat is reachable serve1

uri:

url:ip\_server1:9090

status:200

-name:check tomcat is reachable serve2

uri:

url:ip\_server2:9090

status:200

-name:check tomcat is reachable serve3

uri:

url:ip\_server3:9090

status:200

...

---

-name: configuring apache2

hosts: all

tasks:

- name: Install apache2

apt:

name: apache2

state: present

- name: Edit index.html file

copy:

content: "Welcome to intelliq"

dest: /var/www/html/index.html

- name: retart the apache2

service:

name: apache2

state: restarted

- name: Check url responce of servers

uri:

url: http://private\_ip of server

status: 200

**variables in ansible:**

Ansible uses 3 type of variables

1. Global scope variables

2. Host scope variables

3. Play scope variables

**Global scope variables**: these variables are pass

from command prompt and they have the highest priority

they pass --extra-vars

* **ansible variables for installing and uninstalling software packages**

---

name:Installing/uninstalling s/w packages

hosts: all

tasks:

-name: Install/uninstall/upgrade

apt:

name:"{{a}}"

state:"{{b}}"

update\_cache:"{{c}}"

...

To execute this playbook:

ansible-playbook playbooks5.yml --extra-vars "a=default-jdk b=present c=no" -b

ansible-playbook playbooks5.yml --extra-vars "a=git b=absent c=no" -b

* **ansible plybook for creating user and creating files and directories in user home directory**

---

-name: creating users and creating files/direc in user home directory

hosts:all

tasks:

-name:creating user

user:

name:"{{a}}"

password:"{{b}}"

home:"{{c}}"

-name:creating files in user home directory

file:

name:"{{d}}"

state:"{{e}}"

...

ansible-plybook playbook6.yml --extra-vars "a=anu b-intellq c=/home/anu d=/home/anu/file12 e=touch" -b

ansible-plybook playbook6.yml --extra-vars "a=hari b-intellq c=/home/hari d=/home/hari/dir1 e=directory" -b

**Play scope variables**: these variables are designed to work on single play and they cannot affect multiple places present in playbook, these last levels of priority

* **Ansible playbook for installing apache2**

---

-name: install apche2 using playscope variables

hosts:all

vars:

- a: apache2

- b: present

- c: no

tasks:

- name:Install apache2

apt:

name: "{{a}}"

state: "{{b}}"

update\_cache: "{{c}}"

...

Note: The above playbook works like template those default behaviors to install apache2 but we can bypass that behaviors and make it work on

Some other software by passing global scope variables we can use the above playbook for uninstalling git in all the playbooks use below command

ansible-playbook playbook7.yml --extra-vars "a=git b=absent c= yes" –b

**Creating groups in inventory file:**

Open inventory file and group the ip addresses using square brackets sudo vim /etc/ansible/hosts

[web server]

ip\_server1

ip\_server2

[dbserver]

ip\_server1

[appserver]

ip\_server1

ip\_server3

[india:children]

webserver

dbserver

**Host scope variables:** These variables classified 2 types,

1. Variables work on group of host

2. Variables work on single host

1. variables work on group of host:

These variables are created group\_vars this folder is created in the directory where playbooks are present, in this group\_vars folder we create a file whose name same as group name from the inventory file

1. Go to the folder where the playbooks are present

cd path\_playbooks\_folder

1. Create a directory called group\_vars and move into it

mkdir group\_vars

cd group\_vars

1. Create a file whose name is same as a group name in inventory file

vim webserver

---

a: Ramesh

b: intelliq

c: /home/ubuntu/Ramesh

d: 5432

e: /bin/sh

f: firewalld

g: absent

h: no

...

1. Go back to the folder where the playbooks are present

vim playbook8.yml

---

- name: User creation using host scope variables

user:

name: "{{a}}"

passowrd: "{{b}}"

home: "{{c}}"

uid: "{{d}}"

shell: "{{e}}"

...

**vim playbook9.yml**

---

-name: Uninstalling firewall using host scope variables

hosts: webserver

tasks:

-name: Uninstall firewall

apt:

name: "{{f}}"

state: "{{g}}"

update\_cache: "{{h}}"

**2. Variables work on single host**

This variable only work on one single host, they are created in the file whose name is same as ip address of that host this file will be created in folder called host\_vars and the host\_vars folder should be created in the same folder where playbooks are present

1. Create a folder host\_vars in the folder where all playbooks are present

cd path\_playbooks\_folder

mkdir host\_vars

1. Move into this host\_vars folder and create a file whose name is same as

an ip\_address of a host in the inventory file

cd host\_vars

vim ip\_server1(172.19.0.1)

---

a: /tmp/file1

b: touch

...

1. Come out this host\_vars folder and create the playbook

cd ..

vim playbook10.yml

---

-name: Creating files using host scope variables

hosts: ip\_server1

tasks:

-name: File creation

file:

name: "{{a}}"

state: "{{b}}"

...

1. Execute the playbook

ansible-playbook playbook10.yml –b

**Loops in ansible:**

Any module can be executed for specific number of times using loops

anisble supports 2 types of loops with \_items, with\_sequences

* ansible playbook for installing multiple software’s

---

- name: Install s/w packages

hosts: all

tasks:

- name: install s/w's

apt:

name: "{{item}}"

state: present

update\_cache: no

with\_items:

- tree

- git

- default-jdk

- apache2

...

* ansible playbook for installing and uninstalling and upgrading software packages

---

- name: Installing, uninstalling, upgrading s/w packages

hosts: all

tasks:

- name: install/uninstall/upgrade s/w 's

apt:

name: "{{item.a}}"

state: "{{item.b}}"

update\_cache: "{{item.c}}"

with\_items:

- {a: tree, b: present, c: no}

- {a: git, b: absent, c: yes}

- {a: default-jdk, b: latest, c: no}

- {a: apache2, b: present, c: yes}

...

**Handlers:** handlers are executed only if some other module is executed successfully and it has made some changes handlers are always executed only after all the tasks are over, handlers are executed in the order that they are mentioned in handler section and not in the that they called in the task section even if handler is called multiple times in the task section it will be executed only once.

vim tomcat-users.xml open this file and enter some data in that file like below

<tomcat-user>

<user username="sabitha" password="sabi" roles="manager-script"/>

</tomcat-user>

---

- name: Configuring tomcat8

hosts: all

tasks:

- name: Install tomcat8

apt:

name: tomcat8

state: present

update\_cache: no

notify: Modify tomcat-users.xml

- name: Change port of tomcat from 8080 to 9090

replace:

regexp: 8080

replace: 9090

path: /etc/tomcat8/server.xml

notify: Restart tomcat8

handlers:

- name: Restart tomcat8

service:

name:tomcat8

state: restarted

- name: Modify tomcat-users.xml

copy:

src: /home/ubuntu/tomcat-users.xml

dest: /etc/tomcat8

...

**Error handling**:

Whenever any module in ansible fails the execution of the playbook stops at that point if we want to continue the playbook execution even after encountering an error then we can use error handling

The piece code which might generate an error should be given in the block section if this code generate an error the control comes into rescue section always section is executed every time irrespective of whether the block code is successful or not.

---

- name: Install apache

hosts: all

tasks:

- block:

- name: Install apache1

apt:

name: apache1

state: present

rescue:

- name: Install apache2

apt:

name: apache2

state: present

always:

- name: restart apache2

service:

name: apache2

state: restarted

**when conditions**:

--

-name: Implementing when conditions

hosts: all

vars:

- a: 10

tasks:

- name: install tree

apt:

name: tree

state: present

when:

a== 10

**stat module**:

this module is used to check information about files and folder present on managed nodes

**debug module with when condition:**

ansible playbook check directory is presents and it is present create file in it

---

- name: check is a dir is present and if it present create file in it

hosts: all

tasks:

- name: check if dir is present

stat:

path: /tmp

register: results

- name: Display theoutput of stat module

debug:

var: results

- name: Create a file

file:

name: /tmp/file50

state: touch

when:

results.stat.exitst == true

**Calling child playbooks**:

using include module in ansible it is possible to call child playbooks in parent playbook

child playbook:

---

-name: creating user

user:

name: Anusha

password: admin

home: /home/Anusha

parent playbook:

---

- name: calling child playbook for user creation

hosts: all

tasks:

- name: call child playbook

include:

child\_ playbookname.yml

**Ansible Roles:**

Role is predefined folder structure which is used configuring one software application, everything that is necessary to configure that application will be available in role main advantage is reusability since all the roles use same folder structure it becomes easy for anyone to understand roles created by some other person

**Creating roles:**

1. Move to /etc/ansible/roles folder and create a role

cd /etc/anisble/roles

sudo ansible-galaxy init role\_name –offlline

2.enter tree to show the role

Folder structure of roles:

README.md- this is text file which contain information about the role

defaults-this is used for specifying info about the application that is being configured in this role it also contain defaults variables in roles

files- this is used for storing all static configure files that are necessary for configuring this application

handlers-it is the piece of code which is executed only some other module executed successfully and it has make changes all this handlers will be stored in this folder

meta-data about the data is called meta data ex:who created role,

here info about role in yml format it contains info like who created the role, when it was created which platforms it supports etc

tasks-all tasks that are to be done on remote servers should be stored in this folder

templates-this is used for storing dynamic configuration files

tests-all the yml scrips which are used for checking if configurations are done correctly or not should be stored in this folder

vars- this is storing variables and these variables has higher priority then the variables present in this folder

**Creating role for configuring apache2**:

1. Go into the /etc/ansible/role floder

Cd /etc/ansible/roles

1. Create a new role for apache2

Ansible-galaxy init apache2 –offline

1. Check the tree structure of the role that we are created

Tree apache2

1. Go to taks folder in role and create that task for configuring apache2

Cd apache2

Vim main.yml

* Include: install.yml
* Include : configure.yml
* Include: check\_url\_response.yml

Save and quit

1. Sudo vim install.yml

-name: install apache2

apt:

name: apache2

State: present

Save and quit

1. Sudo Vim configure.yml

-name: copy index.html

Copy:

Src: index.html

Dest: /var/www/html/index.html

Notify:

Restart apache2

Save and quit

1. Sudo vim check url response

-name: check url response

Uri:

url: “{{item}}”

status: 200

with\_items:

* Private\_ip of managed node

1. Go into the files folder create index.html file

Cd files

Cat >index.html

Welcome to new page

1. Go into the handlers folder and create handler for restart the apche2

Cd handler

Sudo vim main.yml

-name: restart apache2

Service:

Name: apache2

State: restarted

Save and quit

1. Create parent playbook to call the roles

Enter cd

Sudo vim apache2\_role.yml

-name: implementing roles for apache2

hosts: all

roles:

* Apache2

Save and quit

1. To execute the role

Ansible-playbook apache-role.yml –b

**Creating roles for tomcat**

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

1 cd /etc/ansible/roles

2 ansible-galaxy init tomcat --offline

3 Create tasks for tomcat

a) cd tomcat/tasks

b) sudo vim main.yml

---

- name: Calling child playbooks

include: "{{item}}"

with\_items:

- install.yml

- configure.yml

- restart.yml

...

Save and quit

c) sudo vim install.yml

---

- name: Installing tomcat8 and tomcat8-admin

apt:

name: "{{item.a}}"

state: "{{item.b}}"

update\_cache: "{{item.c}}"

with\_items:

- {a: "{{pkg1}}",b: "{{state1}}",c: "{{cache1}}"}

- {a: "{{pkg2}}",b: "{{state1}}",c: "{{cache2}}"}

...

d) sudo vim configure.yml

---

- name: Copy tomcat-user.xml

copy:

src: "{{file1}}"

dest: "{{destination1}}"

- name: Change port of tomcat from 8080 to 9090

replace:

path: "{{path1}}"

regexp: "{{port1}}"

replace: "{{port2}}"

notify:

- check\_url\_response

...

e) sudo vim restart.yml

---

- name: Restart tomcat8

service:

name: "{{pkg1}}"

state: "{{state3}}"

...

4) Create the handlers

cd ..

cd handlers

sudo vim main.yml

---

# handlers file for tomcat

- name: check\_url\_response

uri:

url: "{{item.a}}"

status: "{{item.b}}"

with\_items:

- {a: "{{server1}}",b: "{{status1}}"}

- {a: "{{server2}}",b: "{{status1}}"}

...

5) create static files

cd ..

cd files

a) sudo vim tomcat-users.xml

<tomcat-users>

<user username="intelliq" password="myintelliq" roles="manager- script"/>

</tomcat-users>

Save and quit

6) Define the variables

cd ..

cd vars

sudo vim main.yml

---

# vars file for tomcat

pkg1: tomcat8

pkg2: tomcat8-admin

state1: present

state2: absent

state3: restarted

cache1: yes

cache2: no

file1: tomcat-users.xml

destination1: /etc/tomcat8

server1: http://172.31.87.8:9090

server2: http://172.31.84.59:9090

status1: 200

status2: -1

path1: /etc/tomcat8/server.xml

port1: 8080

port2: 9090

...

7 Come out of the tomcat roles

cd ../..

8 Create a playbook to call that role

sudo vim configure\_tomcat.yml

---

- name: Configuring tomcat using roles

hosts: all

roles:

- tomcat

...

Ansible playbook for creating dev environment mysql container should be linked with wordpress container :

---

- name: implementing docker compose using ansible-playbook

hosts: all

tasks:

- name: Start mysql as a container

docker\_container:

image: mysql:5

name: mydb

env:

MYSQL\_ROOT\_PASSWORD: admin

- name: Start wordpress and linked with mysql

docker\_container:

image: wordpress

name: mysite

ports:

- 8899:80

links:

- mydb:mysql

Ansible playboo for creating centos and attach volume

---

- name: Start centos as container and attach volume

hosts: all

tasks:

- name: start centos container

docker\_container:

image: centos

name: c1

interactive: yes

tty: yes

volumes:

- /data

Ansible playbook for creating docker compose of Jenkins container linkes with 2 tomcat containers

---

- name: Implementing docker compose using ansible

hosts: all

tasks:

- name: start jenkins container

docker\_container:

image: jenkins

name: devserver

ports:

- 5050:8080

- name: Start tomcat(qaserver and prod server) linkes with devserver

docker\_container:

image: "{{item.a}}"

name: "{{item.b}}"

ports:

- "{{item.c}}"

links:

- deverver:jenkins

with\_items:

- {a: tomcat,b: qaserver,c: "6060:8080"}

- {a: tomcat,b: prodserver,c: "7070:8080"}

Ansible playbook for push the customized docker images into the docker hub registry

---

- name: upload a docker image into dockerhub using ansible

hosts: ip\_managed\_node

tasks:

- name: Login into docker hub

docker\_login:

username: sai

password: admin

email: sai@gmail.com

- name: push the image

docker\_image:

name: <dockerhub\_id>/image\_name

push: yes

**Ansible vault**: This is feature of ansible which is used for creating playbooks in secure way

1. To create a vault playbook

ansible-vault create playbook\_name.yml

2. To see the content of a valult playbook

ansible-vault view playbook\_name.yml

3. To edit the vault playbook

ansible-vault edit playbook\_name.yml

4.To convert an ordinary playbook into the vault playbook

ansible-vault encrypt playbook\_name.yml

5.To convert a vault playbook into ordinary playbook

ansible-vault decrypt playbook\_name.yml