**Jenkins**

===========

It is continuous integration tool.

Day-1

=====

This is a tool used for implementing CI-CD

Stage in CI-CD

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

Stage 1 (Continuous Download)

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

Whenever developers upload some code into the Git repository

Jenkins will receive a notification and it will download

all that code.This is called as Continuous Download

Stage 2 (Continuous Build)

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

The code downloaded in the previous stage had to converted

into a setup file commonly known as aritfact.To create this

artifact jenkins uses certain build tools like ANT,Maven etc

The artifact can be in the format of a .jar,.war..ear file etc

This stage is called as Continuous Build

Stage 3 (Continuous Deployment)

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

The artifact created in the previous stage has to be deployed into

the QAServers where a team of testers can start accessing it.

This QA environment can be running on some application servers like

tomcat,Weblogic etc.Jenkins deploys the artifact into these application

servers and this is called Continuous Deployment

Stage 4 (Continuous Testing)

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

Testers create automation test scripts using tools like selenium,UFT etc

JEnkins run these automation test scripts and checks if the application

is working according to clients requitrement or not,If testing fails

JEnkins will send automated email notifications to the corresponding

team members and developers will finx the defects and upload the modifed

code into Git,Jenkins will again start from stage 1

Stage 5 (Continuous Delivery)

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

Once the application is found to be defect free Jenkins will deploy it

into the Prod servers where the end user or clinet can start accessing it

This is called continuous delivery

Here the first 4 stages represent CI (Continuous Integration)

the laste stage represents CD (Continuous Delivery)

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

Day 2

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

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

Setup of servers for CI-CD

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

1 Create 3 AWS ubuntu instances

and name them Jenkinsserver,QAServer,ProdServer

Installing Jenkins

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

1 Connect to JEnkins server using Git bash

2 Update the apt repoistory

sudo apt-get update

3 Install jdk

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

4 Install git and maven

sudo apt-get install -y git maven

5 Download jenkins.war

wget https://get.jenkins.io/war-stable/2.263.4/jenkins.war

6 To start jenkins

java -jar jenkins.war

7 To access jenkins from browser

public\_ip\_of\_jenkinsserver:8080

8 Unlock jenkins by entering the first admin password

9 Click on Install suggested plugins

10 Create first admin user

11 Click on Continue--->Finish

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

Setting up tomcat on QA and ProdServer

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

1 Connect to QAServer AWS instance using git bash

2 Update the apt repository

sudo apt-get update

3 Install tomcat9

sudo apt-get install -y tomcat9

4 Install tomcat9-admin

sudo apt-get install -y tomcat9-admin

5 Edit the tomcat-users.xml file

sudo vim /etc/tomcat9/tomcat-users.xml

Delete the entire content of the file and copy paste the below content

<tomcat-users>

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

</tomcat-users>

6 Restart tomcat

sudo service tomcat9 restart

Repeat the above 6 steps on the Prodserver AWS instance

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

Day 3

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

Stage 1 (Continuous Download)

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

1 Open the dashboard of Jenkins

2 Click on New item---->Enter some item name (Developemnt)

3 Select Free style project--->OK

4 Go to Source Code Managment

5 Select Git

6 Enter github url

https://github.com/intelliqittrainings/maven.git

7 Click on Apply---->Save

8 Go to the dashboard of Jenkins

9 Go to the Developement job--->Click on Build icon

This job will download all the code created by the developers

Stage 2 (Continuous 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: package

7 Click on Apply--->Save

8 Go to the dashboard of jenkins and run the Development job

This job will create an artifact from the java code that was

downloaded and this artifact comes in the format of a war file

Stage 3 (Continuous Deployment)

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

1 Open the dashboard of Jenkins

2 Click on Manage Jenkins

3 Click on Manage Plugins

4 Go to Available section

5 Search for "Deploy to container" plugin

6 Click on Install without restart

7 Go to the dashoard of jenkins

8 Go to the Development job

9 Click on Configure

10 Go to Posy build actions

11 Click on Add post build action

12 Click on Deploy war/ear to containers

war/ear file: \*\*\\*.war

Context path: qaapp

Click on Add contianer

Select tomcat9

Enter tomcat credentials

Tomcat url: private\_ip\_qaserver:8080

13 Apply--->Save

14 Go to the dashboard of jenkins

15 Go to the development job---->Click on build

This job will deploy the artifact into the QAServers

and to access the application from the level of browser

public\_ip\_qaserver:8080/qaapp

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

Day 4

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

Stage 4 (Continuous Testing)

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

1 Open the dashboard of Jenkins

2 Click on New item--->Enter some item name(Testing)

3 Go to Source Code management---->Select Git

4 Enter the github url where testers have uplaoded the selenium code

https://github.com/intelliqittrainings/FunctionalTesting.git

5 Go to Build section

6 Click on Add build step

7 Click on Execute shell

java -jar path\_of\_testing.jar

8 Click on Apply--->Save

9 Go to the dashboard of Jenkins

10 Go to the Testing job--->Click on Build icon

This job will download all the selenium test scripts and execute them

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

Linking the Development job with the Testing job

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

1 Open the dashboard of Jenkins

2 Go to the Development job--->Click on Configure

3 Go to Post build actions

4 Click on Add Post Build action

5 Click on Build other projects

6 Enter project name as Testing

7 Click on Save

Now if we run the Development job it will finish the first 3 stages of CI

and then call the Testing job

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

Copy artifacts from Development job to Testing job

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

1 Open the dashboard of Jenkins

2 Click on Manage Jenkins

3 Click on Manage Plugins

4 Go to Available section

5 Search for Copy artifact plugin

6 Click on Install without restart

7 Go to the dashboard of Jenkins

8 Go to Development job--->Click on configure

9 Go to Post build actions--->Click on Add post build actions

10 Click on Archive the artifacts

11 Enter files to be archived as \*\*/\*.war

12 Apply--->Save

13 Go to dashboard of Jenkins

14 Go to Testing job--->Configure

15 Go to Build section--->Click on Add build step

16 Click on Copy artifacts from other projects

17 Enter project name as "Development"

18 Apply--->Save

Stage 5 (Continuous Delivery)

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

1 Go to the dashboard of Jenkins

2 Go to the Testing job--->Configure

3 Go to Post build actions

4 Click on add post build actions

5 Click on Deploy war/ear to container

war/ear file: \*\*/\*war

context path: prodapp

Click on Add container---->Select tomcat9

Enter username and password of tomcat9

Tomcat url: private\_ip\_prodserver:8080

6 Apply--->Save

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

Day 5

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

Alternate ways of setup of Jenkins

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

1 Update the apt repository

sudo apt-get update

2 Install jdk:1.8

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

3 Added the jenkins keys to the apt key repository

wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key

| sudo apt-key add -

4 Add the debain package repository to the jenkins.list file

sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/

> \ /etc/apt/sources.list.d/jenkins.list'

5 Update the apt repository

sudo apt-get update

6 Install jenkins

sudo apt-get install jenkins

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

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

User Administration in Jenkins

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

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-->manage jenkins

7 click on configure global security

8 check enable security checkbox

9 go to authorization section-->click on role based strategy radio 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 mange roles

15 go to global roles and create a role "employee"

16 for this employee in overall give read access

and in view section give all access

17 go to project roles-->Give the role as developer

and patter as Dev.\* (ie developer role can access

only those jobs whose name start with Dev)

18 similarly create another role as tester and assign the pattern as "Test.\*"

19 give all permissions to developrs and tester

20 apply--save

21 click on assign roles

22 go to global roles and add user1 and user2

23 check user1 and user2 as employees

24 go to item roles

25 add user1 and user2

26 check user1 as developer and user2 as tester

27 apply-->save

If we login into jenkins as user1 we can access only the development

related jobs and user2 can access only the testing related jobs

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

Day 6

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

Master / slave Architecture

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

When we want to run multiple Jenkins jobs parallely each job

has to compete for the hardware resources and this might create

performance related issues.To overcome this we can setup MAster/salve.

Slaves are additional servers which takeup the work load from Master

Setup of a slave

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

1 Create a new AWS ubuntu instances

2 Install java

sudo apt-get update

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

3 Setup passwordless between Master and slave

On Slave

--------

a) Setup password for default ubuntu user

sudo passwd ubuntu

b) Edit the sshd config file

sudo vim /etc/ssh/sshd\_config

Search for "PasswordAuthentication" and change it from no to yes

c) Restart ssh

sudo service ssh restart

On Master

---------

d) Generate the ssh keys

ssh-keygen

This will create 2 keys called as public and private in .ssh folder

e) Copy the ssh keys

ssh-copy-id ubuntu@private\_ip\_of\_slave

This will copy the content of the public keys and paste in a file

called "authorised\_keys" on the slave machine

4 Connect to slave using git bash and download the slave.jar file

wget http://private\_ip\_of\_master:8080/jnlpJars/slave.jar

5 Give execute permissions on the slave.jar

chmod u+x slave.jar

6 Create an empty folder which jenkins can use as workspace on slave

mkdir workspace

7 Open the dashboard of Jenkins--->Click on Manage jenkins

8 Click on Manage nodes and clouds

9 Click on New node--->Enter some node name--->Select Permanant Agent

10 Enter Remote root directory: /home/ubuntu/workspace

11 Labels: myslave (This name will be linked with a job in Jenkins)

12 Go to Launch Method-->Click on Launch agent via execution of command

on Master

ssh ubuntu@private\_ip\_of\_slave java -jar slave.jar

13 Click on Save

14 Open the dashboard of Jenkins

15 Go to the job that we want to run on slave---->Click on Configure

16 Go to General section

17 Click on Restrict where this project can be run

And enter the slave label:myslave

18 Apply--->Save

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

Day 7

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

Pipeline as Code

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

This is the process of implementing all the stages of CI-CD

from the level of a Groovy script file called as the Jenkinsfile

Advantages

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

1 Since this is a code it can be uploaded into git and all the

team members can review and edit the code and still git will

maintain multiple versions and we can decide what version to use

2 Jenkinsfiles can withstand planned and unplanned restart of the

Jenkins master

3 They can perform all stages of ci-cd with minimum no of plugins

so they are more faster and secure

4 We can hanlde real world challanges like if conditions,loops

exception handling etc.ie if a stage in ci-cd passes we want

to execute some steps and it fails we want to execute some other

steps

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

Pipeline as code can be implemented in 2 ways

1 Scripted Pipeline

2 Declarative Pipeline

Syntax of Scripted Pipeline

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

node('master')

{

stage('Stage name in ci-cd')

{

Groovy code to implement this stage

}

}

Syntax of Declarative Pipeline

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

pipeline

{

agent any

stages

{

stage('Stage name in CI-CD')

{

steps

{

Groovy code to implement this stage

}

}

}

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

Scripted Pipeline

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

1 Open the dashboard of JEnkins

2 Click on New item

3 Enter some item name (ScriptedPipeline)

4 Select Pipeline--->OK

5 Go to the Pipeline section and generate the below code

node('master')

{

stage('ContinuousDownload')

{

git 'https://github.com/intelliqittrainings/maven.git'

}

stage('ContinuousBuild')

{

sh 'mvn package'

}

stage('ContinuousDeployment')

{

deploy adapters: [tomcat9(credentialsId: '7d1a1f42-858a-4c50-93cb-b2919f4b059f', path: '', url: 'http://172.31.23.20:8080')], contextPath: 'testapp', war: '\*\*/\*.war'

}

stage('ContinuousTesting')

{

git 'https://github.com/intelliqittrainings/FunctionalTesting.git'

sh 'java -jar /home/ubuntu/.jenkins/workspace/ScriptedPipeline/testing.jar'

}

stage('ContinuousDelivery')

{

deploy adapters: [tomcat9(credentialsId: '7d1a1f42-858a-4c50-93cb-b2919f4b059f', path: '', url: 'http://172.31.28.60:8080')], contextPath: 'prodapp', war: '\*\*/\*.war'

}

}

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

Day 8

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

In lot of cases the Jenkinsfile is uploaded by the

developers into the remote git repository and from there

it will trigger all stages of CI-CD

Developers activity

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

1 Cone the remote git repository

git clone https://github.com/intelliqittrainings/maven.git

2 Move into the mave fodler and delete .git folder

cd maven

rm -rf .git

3 Initlise a new git repository

git init

4 Create a file called as Jenkinsfile and paste the Pipeline code into it

vim Jenkinsfile

Copy Paste the Grrovy pipeline code here

5 Send the files to stagging and local reposiotry

git add .

git commit -m "a"

6 Open github.com--->Create a new remote repository

and push the code into it

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

JenkinsAdmin Activity

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

1 Open the dashboard of Jenkins

2 Click on New item--->Enter some item name--->Select Pipeline Project

3 Go to Pipeline section

4 In Definition select Pipeline script from SCM

5 Select SCM as Git

6 Enter the github url where developer has uploaded the code

7 Go to Build triggers

8 Click on POLL SCM and in Schedule section: \* \* \* \* \*

9 Click on Apply--->Save

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

Day 9

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

Declarative Pipeline

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

pipeline

{

agent any

stages

{

stage('ContinuousDownload')

{

steps

{

git 'https://github.com/intelliqittrainings/maven.git'

}

}

stage('ContinuousBuild')

{

steps

{

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

}

}

stage('ContinuousDeployment')

{

steps

{

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/DeclarativePipeline/webapp/target/webapp.war ubuntu@172.31.31.15:/var/lib/tomcat8/webapps/testwebapp.war'

}

}

stage('ContinuousTesting')

{

steps

{

git 'https://github.com/intelliqittrainings/FunctionalTesting.git'

sh label: '', script: 'java -jar /home/ubuntu/.jenkins/workspace/DeclarativePipeline/testing.jar'

}

}

stage('ContinuousDelivery')

{

steps

{

input message: 'Waiting for Approval from the DM!', submitter: 'naresh'

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/DeclarativePipeline/webapp/target/webapp.war ubuntu@172.31.26.41:/var/lib/tomcat8/webapps/prodwebapp.war'

}

}

}

}

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

Declarative Pipeline

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

pipeline

{

agent any

stages

{

stage('ContinuousDownload')

{

steps

{

git 'https://github.com/intelliqittrainings/maven.git'

}

}

stage('ContinuousBuild')

{

steps

{

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

}

}

stage('ContinuousDeployment')

{

steps

{

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/DeclarativePipeline/webapp/target/webapp.war ubuntu@172.31.31.15:/var/lib/tomcat8/webapps/testwebapp.war'

}

}

stage('ContinuousTesting')

{

steps

{

git 'https://github.com/intelliqittrainings/FunctionalTesting.git'

sh label: '', script: 'java -jar /home/ubuntu/.jenkins/workspace/DeclarativePipeline/testing.jar'

}

}

}

post

{

success

{

input message: 'Waiting for Approval!', submitter: 'naresh'

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/DeclarativePipeline/webapp/target/webapp.war ubuntu@172.31.26.41:/var/lib/tomcat8/webapps/prodwebapp.war'

}

failure

{

mail bcc: '', body: '', cc: '', from: '', replyTo: '', subject: 'Jenkins CI-CD Failed', to: 'gandham.saikrishna@gmail.com'

}

}

}

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

Declarative Pipeline

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

pipeline

{

agent any

stages

{

stage('ContinuousDownload')

{

steps

{

script

{

try

{

git 'https://github.com/intelliqittrainings/maven.git'

}

catch(Exception e1)

{

mail bcc: '', body: 'Jenkins is unable to download from remote github', cc: '', from: '', replyTo: '', subject: 'Download failed', to: 'gitadmin@outlook.com'

exit(1)

}

}

}

}

stage('ContinuousBuild')

{

steps

{

script

{

try

{

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

}

catch(Exception e2)

{

mail bcc: '', body: 'Jenkins is unable to create an artifact from the code', cc: '', from: '', replyTo: '', subject: 'Build failed', to: 'developers@outlook.com'

exit(1)

}

}

}

}

stage('ContinuousDeployment')

{

steps

{

script

{

try

{

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/DeclarativePipeline/webapp/target/webapp.war ubuntu@172.31.31.15:/var/lib/tomcat8/webapps/testwebapp.war'

}

catch(Exception e3)

{

mail bcc: '', body: 'Jenkins is unable to deploy into tomcat on the QaServers', cc: '', from: '', replyTo: '', subject: 'Deployment failed', to: 'middleware@outlook.com'

exit(1)

}

}

}

}

stage('ContinuousTesting')

{

steps

{

script

{

try

{

git 'https://github.com/intelliqittrainings/FunctionalTesting.git'

sh label: '', script: 'java -jar /home/ubuntu/.jenkins/workspace/DeclarativePipeline/testing.jar'

}

catch(Exception e4)

{

mail bcc: '', body: 'Functional testing of the app on QAServers failed', cc: '', from: '', replyTo: '', subject: 'Testing failed', to: 'testers@outlook.com'

exit(1)

}

}

}

}

stage('ContinuousDelivery')

{

steps

{

script

{

try

{

input message: 'Waiting for Approval!', submitter: 'naresh'

sh label: '', script: 'scp /home/ubuntu/.jenkins/workspace/DeclarativePipeline/webapp/target/webapp.war ubuntu@172.31.26.41:/var/lib/tomcat8/webapps/prodwebapp.war'

}

catch(Exception e5)

{

mail bcc: '', body: 'Unable to deploy into ProdServers', cc: '', from: '', replyTo: '', subject: 'Delivery failed', to: 'delevery@outlook.com'

}

}

}

}

}

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

Day 10

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

Notifications in Jenkins

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

Notifications can be sent to the team members from Jenkins

in two ways

1 Catlight Notifications

2 Email Notifications

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

Cat Light

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

This is a third party s/w that can be integrated with various

continuous integration tools.It is a clinet side s/w and it has

to be installed on all team members machines

1 Download and install catlight from

https://catlight.io/downloads

2 It will show the list of CI tools and out of them select Jenkins

3 Enter the public\_ip\_of\_jenkinsserver:8080

Enter username and password of Jenkins servers

4 It will display the jenkins jobs,select the job for which we

want notifiaction

5 Run that job in jenkins

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

Email Notifications

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

1 Open the dashboard of Jenkins

2 Click on Manage Jenkins

3 Click on Configure System

4 Go to Email Notifications

5 SMTP server: smtp.gmail.com

6 Click on Advanced

7 Check Use SMTP Authentication

8 Enter gmail username and password

9 Check Use SSL

10 SMTP port: 465

11 Check Test configuration by sending test e-mail

Enter gmail id and click on Test configuration

Note: In Gmail setting Turn on access to less secure apps

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

Day 11

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

Multi Branch Pipeline

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

Generally developers create multiple branches and upload

code related to various functionalities on these branches

We have to configure Jenkins in such a way that it triggers

CI-CD process for all these branches parallelly.

To do this we need to have a copy of JEnkinsfile on each branch

and then based on the instructuions in the Jenkinsfile all

the stages have to be triggered

Developers Activity

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

1 Clone the maven repository

git clone https://github.com/intelliqittrainings/maven.git

2 Move into this cloned repository and delete .git folder

cd maven

rm -rf .git

3 Initilise a new git repository

git init

4 Send the files into stagging area and local repository

git add .

git commit -m "a"

5 Create a jenkins file and put the stages of CI that should happen

on master branch

vim Jenkinsfile

6 Send it to stagging and local repository

git add .

git commit -m "b"

7 Create a new branch called loans and create a new Jenkinsfile

git checkout -b loans

vim Jenkinsfile

Use the CI instructions that should be done on Loans branch

8 Send this to stagging and local repoistory

git add .

git commit -m "c"

9 Open github.com---->Create a new repository

10 Push all the branches from local machine to remote github

git push origin --all

Jenkins Admin Activity

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

1 Open the dashboard of Jenkins

2 Click on New item---->Enter item name as MultiBranchPipeline

3 Select MultiBranchPipeline--->OK

4 Go to Branch Sources---->Select Git-->enter github url where developers

uploaded the code

5 Go to Scan Multi branch pipeline triggers---->Select 1 minute

6 Apply--->Save

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

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

Webhooks

===========

This is used to send notifications from github to jenkins

Whenever any code changes are done and that is checkdin into

github, webhook will send an immediate notifiction to JEnkins

and Jenkins will trigger the job

1 Open github.com---->Click on the repository that we are working on

2 On the right corner click on Setting ...

3 Click on Webhooks in the left pannel

4 Click on Add Webhook

5 In Payload URL: http://public\_ip\_jenkinsserver:8080/github-webhook/

6 In Content type select :application/json

7 Click on Add Webhook

8 Open the dashboaard of Jenkins

9 Go to the job that we want to configure

10 Go to Build triggers

11 Check GitHub hook trigger for GITScm polling

12 Click on Apply--->Save

Now if we make any changes to the code in github then github

will send a notification to jenkins and jenkins will run that job

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