Jenkins
==========
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 previosu stage had to converted
into a setup file commonly known ar aritfact.To create this
artifact jenkins uses certain build tools like ANT,Maven etc
The artifact can be inthe 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)

==========
Create 3 Ubuntu linux servers usung Vagrant
==========
1 Download and install Oracle virtual box

https://www.virtualbox.org/wiki/Downloads

2 Download Vagrant
https://www.vagrantup.com/downloads

3 Create an empty folder and copy the Vagrantfile into this folder

4 Open cmd prompt
Change directory to the folder that we created
cd path_of_folder_where_vagrantfile_is_copied

5 vagrant up

6 Username and password: vagrant

==========

Day 2

==========

Setup of Jenkins for CI-CD

==========

1 Create 3 AWS Ubuntu instances and name then as (Jenkinserver,Qaserver,Prodserver)

2 Connect to Jenkinserver using Gitbash

3 Update the apt repo
sudo apt-get update

4 Install jdk
sudo apt-get install -y openjdk-11-jdk

5 Install git maven
sudo apt-get install -y git maven

6 Download jenkins.war
wget https://get.jenkins.io/war-stable/2.332.1/jenkins.war

7 To start jenkins
java -jar jenkins.war

8 Unlock jenkins by entering the admin password

9 Click on Install suggest Plugin

10 Create a admin user

==========

Install tomcat9 on QA and ProdServer

==========

1 Connect to Qaserver using gitbash

2 Update the apt repo

```
  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
  cd /etc/tomcat9
  sudo vim tomcat-users.xml
  Delete all the content and add the below content
  <tomcat-users>
      <user username="intelliqit" password="intelliqit" roles="manager-script"/>
  </tomcat-users>

6 Restart tomcat9
  sudo service tomcat9 restart
  Repeat the above 6 steps on prodserver AWS instance
================================================================================
Day 3
================================================================================
================================================================================
Continuous Download
==============================
1 Open the dashboard of Jenkins
2 Click on New item---->Enter the item name as Development
3 Select Free style project-->OK
4 Go to Source code Management
5 Clcik on Git
6 Enter the github url where developers have uploaded the code
  https://github.com/intelliqittrainings/maven.git
7 Click on Apply--->Save

========================================================================
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 Top level maven targets
6 Enter the maven goal: package
7 Aplly--->Save

========================================================================
Continuous Deployment
============================
1 Open the dashboard of Jenkins
2 Go to Manage Jenkins
```

3 Click on Manage Plugins
4 Click on Availabl\e section
5 Search for Deploy to container plugin
6 Install it
7 Go to the dashboard of Jenkins
8 Go to the Development job--->Click on configure
9 Go to Post build actions
10 Click on Add post build action
11 Click on Deploy war/ear to container
   war/ear file: **/*.war
   Context path: testapp   (This is the name that testers will enter in browser to access the
   application)
   Click on Add container
   Select tomcat9
   Enter tomcat9 credentials
   Tomcat url: private_ip_qaserver:8080
12 click on Apply--->Save

==========
Day 4
==========
Continuous Testing
==========
1 Open the dashboard of Jenkins
2 Click on New item
3 Enter some item name (Testing)
4 Select Free style project
5 Enter the github url where testers have uploaded the selenium scripts
  https://github.com/intelliqittrainings/FunctionalTesting.git
6 Go to Build section
7 Click on Add build step
8 Click on Execute shell
  java -jar path/testing.jar
9 Apply--->Save

Linking the Development job with the Testing job
==========
1 Open the dashboard of Jenkins
2 Go to the dEvelopment job
3 Click on configure
4 Go to Post build actions
5 Click on Add post buuild actions
6 Click on Build another job
7 Enter the job the Testing
8 Apply--->Save
  This is called as upstream/downstream configurations

==========

Copying artifacts from Development job to Testing job
==========
1 Open the dashboard of Jenkins
2 Click on Manage Jenkins--->Manage plugins
3 Go to Availbale section--->Search for "Copy Artifact" plugun
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 Post build actions
8 Click on Add post build actions
9 Click on Archive the artifacts
10 Enter files to be archived as ***.war
11 Click on Apply--->>Save
12 Go to the dashboard of jenkins
13 Go to the Testing job---->Click on configure
14 Go to Build section
15 Click on Add build step
15 Click on Copy artifacts from other project
16 Enter project name as "Development"
17 Apply---->Save
==========
Stage 5 (Continuous Delivery)
==========
1 Open the dashboard of jenkins
2 Go to Testing job--->Configure
3 Go to Post build actions
4 Click on Add post build action
5 Click on Deploy war/ear to container
  war/ear files: ***.war
  contextpath: prodapp
  Click on Add container
  Select tomcat9
  Enter username and password of tomcat9
  Romcat url: private_ip_of_prodserver:8080
6 Apply---->Save

==========
Day 5
==========
==========
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 assgning
==========
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

==========
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
  curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null

```
4 Add the debain package repository to the jenkins.list file
  echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null

5 Update the apt repository
  sudo apt-get update

6 Install jenkins
  sudo apt-get install -y jenkins
================================================================================
Scheduling the job for a particular date and time
================================================================================

1 Open the dashboard of jenkins
2 Go to the configuration page of the job
3 Go to Build triggers
4 Click on Build periodically
5 Schedule the date and time
6 Click on Save

================================================================================================
Day 6
================================================================================================
MAster Slave Architecture of Jenkins
=============================================
This is used distribute the work load to additional linux
servers called as slaves.This is used when we want to run multiple
jobs on jenkins parallelly.

Setup
============
1 Create a new AWS ubuntu22 instance

2 Install the same version of java as present in the master
  sudo apt-get update
  sudo apt-get install -y openjdk-8-jdk

3 Setup passwordless SSH betwen Master and slave
  a) Connect to slave and set password to default user
     sudo passwd ubuntu
  b) Edit the ssh 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
  d) Connect to Master using git bash
  e) Generate the ssh keys
     ssh-keygen
  f) Copy the ssh keys
```

```
     ssh-copy-id ubuntu@private_ip_of_slave
     This will copy the content of the public keys to a file called
     "authorised_keys" on the slave machine


  Connect to slave using git bash
4 Download the slave.jar file
  wget http://private_ip_of_jenkinsserver:8080/jnlpJars/slave.jar

5 Give execute permissions to the slave.jar
  chmod u+x slave.jar

6 Create an empty folder that will be the workspace of jenkins
  mkdir newfolder

7 Open the dashboard of Jenkins

8 Click on Manage Jenkins--->Click on Manage Nodes and Clouds

9 Click on New node---->Enter some node name as Slave1

10 Select Permanant Agent--->OK

12 Enter remote root directory as /home/ubuntu/newfolder

13 Labels: myslave (This label is associated with a job in jenkins
   and then that job will run on that slave)

14 Go to Launch Method and select "Launch agent via execution of command on master"

15 Click on Save

16 Go to the dashboard of Jenkins

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

18 Go to General section

19 Check restrict where this project can be run

20 Enter slave label as myslave


==========================================================================
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('built-in')
{
   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 Go to the dashboard of jenkins
2 Click on New item
3 Enter item name as "ScriptedPipeline"
4 Select Pipeline--->Click on OK

```
node('built-in')
{
    stage('ContinuousDownload')
    {
        git 'https://github.com/intelliqittrainings/maven.git'
    }
    stage('ContinuousBuild')
    {
        sh 'mvn package'
    }
    stage('ContinuousDeployment')
    {
       deploy adapters: [tomcat9(credentialsId:
'8cc7d40a-bab0-438d-8dc2-f0d886815228', path: '', url: 'http://172.31.16.84:8080')],
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')
    {
        input message: 'Need approvals from the DM!', submitter: 'srinivas'
        deploy adapters: [tomcat9(credentialsId:
'8cc7d40a-bab0-438d-8dc2-f0d886815228', path: '', url: 'http://172.31.29.58:8080')],
contextPath: 'prodapp', war: '**/*.war'
    }




}
```

==========
==========
POLL SCM
==========
This is a process where Jenkins will check the remote github for any new commits
1 Open the dahboard of Jenkins
2 Go to the relavant job--->Click on configure

3 Go to Build triggers
4 Click on POLL SCM and in Schedule section: * * * * *
5 Click on 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

Declarative Pipeline
==========
pipeline
{
    agent any
    stages
    {
       stage('ContinuosDownload')
       {
          steps
          {
             git 'https://github.com/krishnain/SampleMaven.git'
          }
       }

       stage('ContinuosBuild')
       {
          steps
          {
             sh 'mvn package'
          }
       }

```
       stage('ContinuosDeployment')
       {
          steps
          {
             deploy adapters: [tomcat9(credentialsId:
'cd076526-1975-42c2-a9e1-b79f5c0cc500', path: '', url:
'http://172.31.23.179:9090')], contextPath: 'testapp', war: '**/*.war'
          }
     }
   stage('ContinuosTesting')
   {
       steps
       {
          git 'https://github.com/intelliqittrainings/FunctionalTesting.git'
          sh  'java -jar
/home/ubuntu/.jenkins/workspace/ScriptedPipeline1/testing.jar'
       }
   }
   stage('ContinuosDelivery')
   {
      steps
      {
      deploy adapters: [tomcat9(credentialsId:
'cd076526-1975-42c2-a9e1-b79f5c0cc500', path: '', url:
'http://172.31.23.149:9090')], contextPath: 'prodapp', war: '**/*.war'
      }
   }



   }



================================================================================
Deployments using scp without the requirement of "dEploy to container" plugin
================================================================================
1 Establish passwordless ssh between Jenkins and Qaserver and Jenkins and Prodserver

2 On both QA and Prodserver change the permissions of tomcat
  sudo chmod o+r  -R /var/lib/tomcat9


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
           {

               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 with post conditions
================================================
```

```
pipeline
{
    agent any
    stages
    {
        stage('ContinuousDownload')
        {
            steps
            {
                git 'https://github.com/krishnain/mavenab.git'
            }
        }
        stage('ContinuousBuild')
        {
            steps
            {
                sh 'mvn package'
            }
        }
        stage('ContinuousDeployment')
        {
            steps
            {
                deploy adapters: [tomcat9(credentialsId:
'376e01e8-e628-40d2-aaec-6452f707a3ff', path: '', url:
'http://172.31.20.211:8080')], contextPath: 'qaaapp', war: '**/*.war'
            }
        }
        stage('ContinuousTesting')
        {
            steps
            {
                git 'https://github.com/intelliqittrainings/FunctionalTesting.git'
                sh 'java -jar
/home/ubuntu/.jenkins/workspace/DeclarativePipeline2/testing.jar'
            }
        }

    }
    post
    {
        success
        {
            input message: 'Required approvals', submitter: 'srinivas'
                deploy adapters: [tomcat9(credentialsId:
'376e01e8-e628-40d2-aaec-6452f707a3ff', path: '', url:
'http://172.31.21.226:8080')], contextPath: 'myprodapp', war: '**/*.war'
        }
        failure
        {
```

```
            mail bcc: '', body: 'Continuous Integration is giving a failure msg',
cc: '', from: '', replyTo: '', subject: 'CI Failed', to:
'selenium.saikrishna@gmail.com'
        }

    }




}
```

```
================================================================================
Exception Handling
========================
This is the process of overcoming a potential error and continuing the execution
of the program,This is implemented using try,catch
The section of code that can generate an error is given in the try block
if it generates an error the contol comes into the catch sextion

try
{

}
catch(Exception e)
{

}
================================================================================
Declarative Pipeline with execption handling
================================================
pipeline
{
    agent any
    stages
    {
        stage('ContinuousDownload')
        {
            steps
            {
                script
                {
                    try
                    {
                        git 'https://github.com/krishnain/mavenab.git'
                    }
                    catch(Exception e1)
                    {
                        mail bcc: '', body: 'Jenkins is unable to download from the
```

```
remote github', cc: '', from: '', replyTo: '', subject: 'Download Failed', to:
'git.admin@gmail.com'
                        exit(1)
                    }
                }

            }
        }
        stage('ContinuousBuild')
        {
            steps
            {
                script
                {
                    try
                    {
                         sh 'mvn package'
                    }
                    catch(Exception e2)
                    {
                        mail bcc: '', body: 'Jenkins is unable to create an artifact
from the downloaded code', cc: '', from: '', replyTo: '', subject: 'Build Failed',
to: 'dev.team@gmail.com'
                        exit(1)
                    }
                }

            }
        }
        stage('ContinuousDeployment')
        {
            steps
            {
                script
                {
                    try
                    {
                       sh 'scp
/home/ubuntu/.jenkins/workspace/DeclarativePipeline3/webapp/target/webapp.war
ubuntu@172.31.20.211:/var/lib/tomcat9/webapps/testapp.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.team@gmail.com'
                        exit(1)
                    }
                }
```

```
            }
        }
        stage('ContinuousTesting')
        {
            steps
            {
                script
                {
                    try
                    {
                        git
'https://github.com/intelliqittrainings/FunctionalTesting.git'
                        sh 'java -jar
/home/ubuntu/.jenkins/workspace/DeclarativePipeline3/testing.jar'
                    }
                    catch(Exception e4)
                    {
                        mail bcc: '', body: 'Selenium scripts are showing a failure
status', cc: '', from: '', replyTo: '', subject: 'Testing Failed', to:
'qa.team@gmail.com'
                        exit(1)
                    }
                }

            }
        }
        stage('ContinuousDelivery')
        {
            steps
            {
                script
                {
                    try
                    {
                         input message: 'Required approvals', submitter: 'srinivas'
                          deploy adapters: [tomcat9(credentialsId:
'376e01e8-e628-40d2-aaec-6452f707a3ff', path: '', url:
'http://172.31.21.226:8080')], contextPath: 'myprodapp', war: '**/*.war'
                    }
                    catch(Exception e5)
                    {
                        mail bcc: '', body: 'Jenkins is unable to deploy into tomcat
on the prodservers', cc: '', from: '', replyTo: '', subject: 'Delivery Failed', to:
'delivery.team@gmail.com'
                    }
                }

            }
        }
    }
```

```
}
```

==========