What is Jenkins?
++++++++++++++++

=> Jenkins is an open source automation tool for CI and CD

=> Jenkins tool developed using Java

=> Jenkins is part of Hudson Project

=> Intially it is called as Hudson then later it renamed to Jenkins

About CI CD
++++++++++++

=> CI and CD are two most frequently used terms in modern development practises and DevOps practises.

=> CI stands for Continuous Integration. It is fundamental DevOps best practise where developers frequently merge code changes to central repository where automated builds and tests runs.

=> CD means Continuous Delivery or Continous Deployment.

=> Jenkins is a self-contained, open-source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

++++++++++++++++++++++++++++++
Build & Deployment Process
++++++++++++++++++++++++++++

1) Take latest source code from repository

2) Compile source code

3) Execute Unit tests (Junits)

4) Perform Code Review

5) Package code as war file

6) Deploy the war file into server

Note: All the above build and deployment tasks can be automated using Jenkins tool.

++++++++++++++++++++++++
Jenkins Installation:
++++++++++++++++++++++++

1) Create an EC2 instance with Amazon Linux 2 AMI

2) Connect to your EC2 instance using MobaXterm

3) Update all packages
```
$ sudo yum update -y
```

4) Install Java
```
$ sudo yum install java
```

5) Add Jenkins repo to your yum repository
```
$ sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat/jenkins.repo
```

6) Import a key file from Jenkins-CI to enable installation from the package
```
$ sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
```

```
7) For Amazon Linux 2
        $ sudo amazon-linux-extras install epel

8)  Install Jenkins
        $ sudo yum install jenkins -y

9) Start and enable Jenkins service
        $ sudo systemctl start jenkins
        $ sudo systemctl enable jenkins
        $ sudo systemctl status jenkins

10) Access Jenkins Server in browser using below URL

                URL : http://ec2-public-ip:8080/

        Note: Enable 8080 port in security group

11) Get the initial administrative password
        $ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

pwd : 5fe6ddcc9db244cab6aca5ccf2d6a83a

-> Provide pwd which we have copied to unlock jenkins

-> Select "Install Suggested Plugins" card (it will install those plugins)

-> Create Admin account

++++++++++++++++++++++++++++++++++++++++++++++++++
Creating First Job in Jenkins
++++++++++++++++++++++++++++++++++++++++++++++++++

1) Goto Jenkins Dashboard

2) Click on New Iteam

                -> Enter Item Name (Job Name)
                -> Select Free Style Project & Click OK
                -> Enter some description
                -> Click on 'Build' tab
                -> Click on 'Add Build Step' and select 'Execute Shell'


3) Enter below shellscript

echo "Hello Guys,"
touch ashokit.txt
echo "Hello Guys, Welcome to Jenkins Classes" >> ashokit.txt
echo "Done..!!"

4) Apply and Save

Note: With above steps we have created JENKINS Job

5) Click on 'Build Now' to start Job execution

6) Click on 'Build Number' and then click on 'Console Ouput' to see job execution details.


=> Jenkins Home Directory in EC2 : /var/lib/jenkins/workspace/

                $ cd /var/lib/jenkins/workspace/

7) Go to Jenkins home directory and check for the job name --> check the file created inside the job
```

```
++++++++++++++++++++++++++++++++++++++++++++++++
Jenkins Job with with GIT Hub Repo + Maven - Integeration
++++++++++++++++++++++++++++++++++++++++++++++++

Pre-Requisites : Java, Maven and Git client

# Git installation In EC2 VM
$ sudo yum install git -y

++++++++++++++++++++++
JDK Installation In Jenkins
++++++++++++++++++++++

Jenkins Dashboard -> Manage Jenkins -> Global Tools Configuration -> Add JDK -> Choose JDK 8 ->
Configure Oracle account crdentials to download JDK.

++++++++++++++++++++++++++++++
Maven Installation In Jenkins:
++++++++++++++++++++++++++++++

Jenkins Dashboard -> Manage Jenkins --> Global Tools Configuration -> Add maven


Sample Git Repo URLS For Practise
++++++++++++++++++++++++++++++++++

Git Hub Repo URL-1  : https://github.com/ashokitschool/JAVA-MAVEN-WEB-APP.git
Git Hub Repo URL-2  : https://github.com/ashokitschool/maven-web-app.git

++++++++++++++++++++++++++++++++++++++++++++++++++
Steps To Create Jenkins Job with Git Repo + Maven
++++++++++++++++++++++++++++++++++++++++++++++++++

1) Connect to EC2 instance in which jenkins server got installed

2) Start Jenkins Server

3) Access Jenkins Server Dashboard and Login with your jenkins credentials

4) Create Jenkins Job with Git Hub Repo

                -> New Item
                -> Enter Item Name (Job Name)
                -> Select 'Free Style Project' & Click OK
                -> Enter some description
                -> Go to "Source Code Management" Tab and Select "Git"
                -> Enter Project "Git Repo URL"
                -> Add Your Github account credentials
                -> Go to "Build tab"
                -> Click on Add Build Step and Select 'Inovke Top Level Maven Targets'
                -> Select Maven and enter goals 'clean package'
                -> Click on Apply and Save

Note: With above steps we have created JENKINS Job

5) Click on 'Build Now' to start Job execution

6) Click on 'Build Number' and then click on 'Console Ouput' to see job execution details.

=> Jenkins Home Directory in EC2 : /var/lib/jenkins/workspace/

=> Go to jenkins workspace and then go to job folder then goto target folder there we see war file
created.

-----------------------------------------------------------------------------------------------
```

=> Access below URL in browser to stop Jenkins Server

URL : http://EC2-VM-IP:8080/exit

(Click on Retry using Post button)

Steps To Create Jenkins Job with Git Repo + Maven + Tomcat Server

---

1) Go to Tomcat server folder and configure below users in "tomcat-users.xml" file (it will be available in tomcat -erver conf folder)

```
<role rolename="manager-gui" />
<role rolename="manager-script" />
<role rolename="admin-gui" />

<user username="tomcat" password="tomact" roles="manager-gui" />
<user username="admin" password="admin" roles="manager-gui,manager-script,admin-gui"/>
```

2) Go to Jenkins Dashboard -> Manage Jenkins --> Manage Plugins -> Goto Available Tab -> Search For "Deploy To Container" Plugin -> Install without restart.

3) Create Jenkins Job

-> New Item
-> Enter Item Name (Job Name)
-> Select Free Style Project & Click OK
-> Enter some description
-> Go to "Source Code Management" Tab and Select "Git"
-> Enter Project "Git Repo URL"
-> Add Your Github account credentials
-> Go to "Build tab"
-> Click on Add Build Step and Select 'Inovke Top Level Maven Targets'
-> Select Maven and enter goals 'clean package'
-> Click on 'Post Build Action' and Select 'Deploy war/ear to container' option
-> Give path of war file (You can give like this also : **/*.war)
-> Enter Context Path (give project name)
-> Click on 'Add Container' and select Tomcat version 9.x
-> Add Tomcat server credentials (give the username & pwd which is having manager-script role)
-> Enter Tomact Server URL (http://ec2-vm-ip:tomcat-server-port)
-> Click on Apply and Save

4) Run the job now using 'Build Now' option and see see 'Console Output' of job

5) Once Job Executed successfully, go to tomcat server dashboard and see application should be displayed.

6) Click on the applicaton name (it should display our application)

++++++++++++++++
Jenkins Pipeline
++++++++++++++++

-> Sequence of Jobs execution is called as Pipeline

-> For our application, we will have multiple environments like below

-> DEV
-> SIT
-> UAT

-> PILOT
-> PROD

-> For every environment, one JENKINS Job will be created

JENKINS_DEV_JOB -------> DEV Environment (Dev Server)

JENKINS_SIT_JOB -------> SIT Environment (SIT Server)

JENKINS_UAT_JOB ------> UAT Environment (UAT Server)

JENKINS_PILOT_JOB -----> Pilot Environment (Pilot Server)

JENKINS_PROD_JOB ------> Prod Environment (Prod Server)

-> If we want to deploy code changes to all environments then it is recommended to create Build Pipeline

-> By Using Build Pipeline we can execute JENKINS Jobs sequentially.

Requirement
++++++++++++

-> If code commit happend in git hub then deploy code into DEV Server

-> If DEV Server Deployment successful, then deploy code into SIT environment

-> If SIT Server deployment successful, then deploy code into UAT environment.

Steps To Create Jenkins Pipeline
++++++++++++++++++++++++++++++++

1) Create EC2 VM and install Tomcat Server (Dev Server VM)

2) Create EC2 VM and install Tomcat Server (SIT Server VM)

3) Create EC2 VM and install Tomcat Server (UAT Server VM)

4) Create EC2 VM and install Jenkins (Jenkins Server VM)

5) Install JDK, Maven, Git, Deploy To Cotainer in Jenkins Server VM

6) Create Jobs in Jenkins Server

DEV-Job ----> Dev Server

SIT-Job ----> SIT Server (SIT Job should execute if DEV-JOB is stable)

UAT-Job ----> UAT-Server (UAT job should execute if SIT-JOB is stable)

7) Create Jenkins Build Pipeline to execute Jobs in sequence

**** If we forgot Jenkins Password, then how to recover it ? *****

-> Go to /var/lib/jenkins/

Open : config.xml file

-> Set Value for useSecurity as false

ex: <useSecurity>false</useSecurity>

-> Re-start jenkins and try to access

******* How to change Jenkins Port Number **********

-> Go to root directory

-> Go to /etc/sysconfig

-> Open jenkins file and change Jenkins port number

-> Restart Jenkins server

How to Create Jenkins Jobs with Build Parameters

-> Create New Item
-> Enter Item Name & Select Free Style Project
-> Select "This Project is parameterized" in General Section
-> Select Choice Parameter
-> Name : Branch Name
-> Choices : Enter every branch name in nextline
-> Branches to Build : */${BranchName}

++++++++++++++++++++
Creating Users in Jenkins
++++++++++++++++++++++++

-> Manage Jenkins -> Manage Users

-> Create Users

-> Configure Global Security For Users

-> Manage Roles & Assign Roles

Note: By default admin role will be available and we can create custom role based on requirement

-> In Role we can configure what that Role assigned user can do in jenkins

-> In Assign Roles we can add users to particular role

Excercise
+++++++++

-> create 1 account for DevOps team member

-> Create 1 account for Development team member

-> Configure Roles for DevOps team member and Development team member

---

Jenkins Master and Slave Configuration

When we build the Jenkins job in a single Jenkins master node then Jenkins uses the resource of the base machine and If no executor is available then the jobs are queued in the Jenkins server.

Sometimes you might need several different environments to test your builds. This cannot be done by a single Jenkins server.

It is recommended not to run different jobs in the same system that required a different environment. In such scenarios where we need a different machine with a different environment that takes the specific job from the master to build.

On the same Jenkins setup, multiple teams are working with their jobs. All jobs are running on the same base operating system and the base operating system has limited resources.

To overcome this problem, Jenkins provided Distributed Architecture i.e Jenkins Master Slave Architecture

Jenkins uses A Master-Slave architecture to manage distributed builds. The machine where we install Jenkins software will be Jenkins master and that runâ€™s on port 8080 by default. On the slave machine, we install a program called Agent. This agent requires JVM. This agent executes the tasks provided by Jenkins master. We can launch n numbers of agents and we can configure which task will be run on which agent server from Jenkins master by assigning the agent to the task.

Jenkins Master

Your main Jenkins server is the Master. The Masterâ€™s job is to handle:

a) Scheduling build jobs.

b) Dispatching builds to the slaves for the actual execution.

c) Monitor the slaves (possibly taking them online and offline as required).

d) Recording and presenting the build results.

e) A Master instance of Jenkins can also execute build jobs directly.

Jenkins Slave

A Slave is a Java executable that runs on a remote machine. Following are the characteristics of Jenkins Slaves:

1) It hears requests from the Jenkins Master instance.

2) Slaves can run on a variety of operating systems.

3) The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.

4) You can configure a project to always run on a particular Slave machine or a particular type of Slave machine, or simply let Jenkins pick the next available Slave.

++
Working with Jenkins Master Slave Architecture
++

Step-1 : Create Jenkins Master
++++++++++++++++++++++++++++++
1) Create EC2 instance
2) Connect EC2 using Mobaxterm
3) Install Git client
4) Install Java Software

5) Install jenkins server
6) Add Git, JDK and Maven Plugins
7) Enable Jenkins Port Number in Security Group
8) Access Jenkins Server in Browser and Login

Step-2 : Create Jenkins Slave
++++++++++++++++++++++++++++++
1) Create EC2 instance
2) Connect to EC2 using Mobaxterm
3) Install Git client
4) Install Java Software
5) Create one directory in /home/ec2-user (ex: slavenode)

Step-3: Configure Slave Node in Jenkins Master Node
++

1) Go to Jenkins Dashboard
2) Go to Manage Jenkins
3) Go to Manage Nodes & Clouds
4) Click on 'New Node' -> Enter Node Name -> Select Permanent Agent
5) Enter Remote Root Directory (/home/ec2-user/slavenode)
6) Enter Label name as Slave-1
7) Select Launch Method as 'Launch Agent Via SSH'
8) Give Host as 'Slave VM DNS URL'
9) Add Credentials (Select Kind as : SSH Username with private key)
10) Enter Username as : ec2-user
11) Select Private Key as Enter Directley and add private key

Note: Open gitbash from your .pem file location and execute below command to get private key from pem file

```
$ cat <key-pair-file-name>.pem
```

Note: It will display private key on Git Bash terminal (Just copy and paste in jenkins)

12) Select Host Key Strategy as 'Manually Trusted Key Verification Strategy'

13) Click on Apply and Save (We can see configure slave)

**********With above steps Master and Slave Configuration Completed**********

-> Go to Jenkins Server and Create Jenkins Job

Note: Under Generation Section of Job creation process, Select "Restrict Where This Project Can Run" and enter Slave Nodel Label name and finish job creation.

-> Execute the Job using 'Build Now' option

Note: Job will be executed on Slave Node (Go to Job Console Ouput and verify execution details)

++++++++++++++++++++
JENKINS - Pipeline
++++++++++++++++++++

Jenkins Pipeline is an automation solution that lets you create simple or complex pipelines.

Jenkins Pipeline is a combination of Plugins which automates number of tasks and makes the CI/CD

pipeline efficient, high in quality and reliable.

Jenkins provides two ways of developing a pipeline

1) Scripted
2) Declarative

Traditionally, Jenkins jobs were created using Jenkins UI called FreeStyle jobs.

In Jenkins 2.0, Jenkins introduced a new way to create jobs using the technique called pipeline as code.

In pipeline as code technique, jobs are created using a script file that contains the steps to be executed by the job.

In Jenkins, that scripted file is called Jenkinsfile.

++++++++++++++++++++++++++
What is Jenkinsfile?
++++++++++++++++++++++++++

Jenkinsfile is nothing but a simple text file which is used to write the Jenkins Pipeline and to automate the Continuous Integration process.

Jenkinsfile usually checked in along with the projectâ€™s source code in Git repo. Ideally, every application will have its own Jenkinsfile.

Jenkinsfile can be written in two ways â€“

1) Scripted pipeline syntax
2) Declarative pipeline syntax

++++++++++++++++++++++++++++++++
What is Jenkins Scripted Pipeline?
++++++++++++++++++++++++++++++++

Jenkins pipelines are traditionally written as scripted pipelines. Ideally, the scripted pipeline is stored in Jenkins webUI as a Jenkins file. The end-to-end scripted pipeline script is written in Groovy.

It requires knowledge of Groovy programming as a prerequisite.
Jenkinsfile starts with the word node.
Can contain standard programming constructs like if-else block, try-catch block, etc.

++++++++++++++++++++++++++
Sample Scripted Pipeline
++++++++++++++++++++++++++

```
node {
    stage('Stage 1') {
        echo 'hello'
    }
}
```

++++++++++++++++++++++++++++++++++
What is Jenkins Declarative Pipeline?
++++++++++++++++++++++++++++++++++

The Declarative Pipeline subsystem in Jenkins Pipeline is relatively new, and provides a simplified, opinionated syntax on top of the Pipeline subsystems.

The latest addition in Jenkins pipeline job creation technique.

Jenkins declarative pipeline needs to use the predefined constructs to create pipelines. Hence, it is not flexible as a scripted pipeline.

Jenkinsfile starts with the word pipeline.

Jenkins declarative pipeline should be the preferred way to create a Jenkins job as they offer a rich set of features, come with less learning curve & no prerequisite to learn a programming language like Groovy just for the sake of writing pipeline code.

We can also validate the syntax of the Declarative pipeline code before running the job. It helps to avoid a lot of runtime issues with the build script.

++++++++++++++++++++++++++++++
Our First Declarative Pipeline
++++++++++++++++++++++++++++++

```
pipeline {
    agent any
    stages {
        stage('Welcome Step') {
            steps {
                echo 'Welcome to Jenkins Scripting'
            }
        }
    }
}
```

pipeline : Entire Declarative pipeline script should be written inside the pipeline block. Itâ€™s a mandatory block.

agent : Specify where the Jenkins build job should run. agent can be at pipeline level or stage level. Itâ€™s mandatory to define an agent.

stages : stages block constitutes different executable stage blocks. At least one stage block is mandatory inside stages block.

stage : stage block contains the actual execution steps. Stage block has to be defined within stages block. Itâ€™s mandatory to have at least one stage block inside the stage block. Also its mandatory to name each stage block & this name will be shown in the Stage View after we run the job.

steps : steps block contains the actual build step. Itâ€™s mandatory to have at least one step block inside a stage block.

Depending on the Agentâ€™s operating system (where Jenkins job runs), we can use shell, bat, etc., inside the steps command.

++
Build Pipeline Script
++

```
pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
        stage('GetCode'){
            steps{
                git branch: 'main',
                url: 'https://github.com/ashokitschool/maven_web_app_jenkins_pipeline.git'
            }
         }
        stage('Build'){
            steps{
                sh 'mvn clean package'
            }
        }
```

```
    }
}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Build Pipeline + Sonar Qube Server - Script
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
       stage('GetCode'){
            steps{
                git 'https://github.com/ashokitschool/maven-web-app.git'
            }
         }
       stage('Build'){
            steps{
                sh 'mvn clean package'
            }
         }
        stage('SonarQube analysis') {
            steps{
                withSonarQubeEnv('Sonar-Server-7.8') {
                    sh "mvn sonar:sonar"
                }
            }
        }

    }
}

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
JENKINS PIPELINE ( JENKINS + MAVEN + GIT HUB + SONAR + TOMCAT )
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Note: Install ssh-agent plugin and generate code using pipeline syntax

pipeline{
    agent any
    environment {
        PATH = "$PATH:/opt/apache-maven-3.6.3/bin"
    }
    stages{
       stage('GetCode'){
            steps{
                git 'https://github.com/ashokitschool/maven-web-app.git'
            }
         }
       stage('Build'){
            steps{
                sh 'mvn clean package'
            }
         }
        stage('SonarQube Analysis') {
            steps{
                withSonarQubeEnv('Sonar-Server-7.8') {
                    sh "mvn sonar:sonar"
                }
            }
        }
                stage('Code deploy') {
            steps{
                                sshagent(['Tomcat-Server-Agent']) {
                                  sh 'scp -o StrictHostKeyChecking=no target/01-maven-web-app.war
```

```
ec2-user@13.235.68.29:/home/ec2-user/apache-tomcat-9.0.63/webapps'
                                    }
            }
        }
    }
}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Email Notifications In Jenkins
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

-> We can configure Email notifications in Jenkins

-> With this option we can send email notification to team members after jenkins job execution
completed

-> We need to configure SMTP properties to send emails

-> Go To Manage Jenkins
-> Add Email Extension Server
-> We will add company provided SMTP server details

Note: For practise we can use GMAIL SMTP Properties

-> Once SMTP properties added then we can configure email notification as 'Post Build Action' in
Jenkins job

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

pipeline {
    agent any
    stages {
        stage('Hello') {
            steps {
                echo "Hello world"
                    }
            }
        }
    post{
        always{
            mail to: "ashokitschool@gmail.com",
            subject: "Test Email",
            body: "Test"
        }
    }
}
-------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------
```