Kaushik Dey

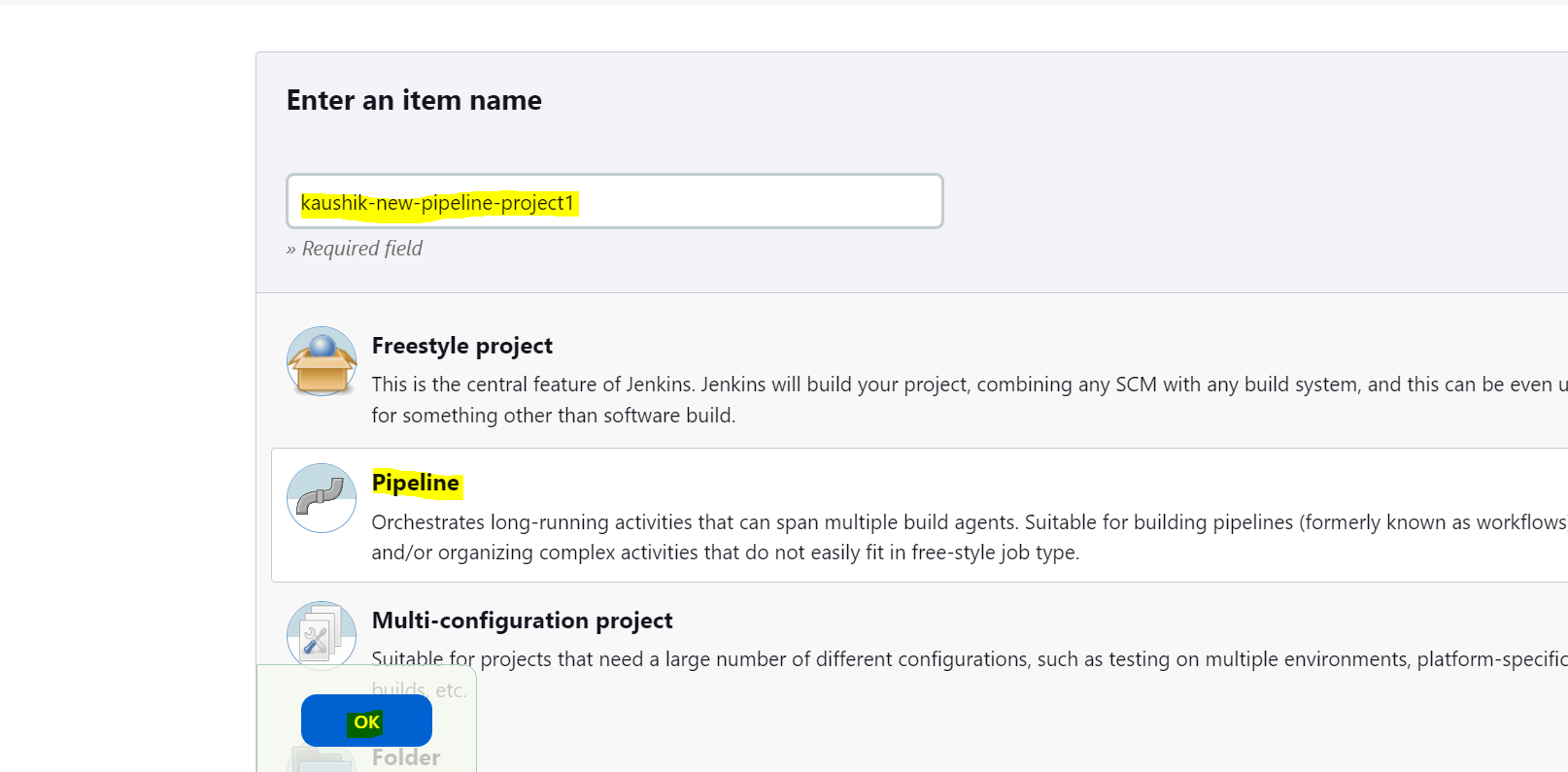
Kaushikdey1984@yahoo.com

jenkins pipeline scripts with auto deployment

Jenkins deployment

Step 1:

In this demo we are working with pipeline project, so first from the **new item** from Jenkins dashboard we got the following screen, enter your project name(**kaushik-new-pipeline-project1**) and choose **Pipeline** and click on **OK**.



Step 2:

Now we are going to create jobs and pipeline via script. Two types of pipeline are there . ( one is scripted and other is declarative pipeline). In this demo we use declarative pipeline with groovy script ). The screenshot is given below.

Declarative pipeline starting with the keyword pipeline & scripted pipeline starting with the keyword node.

All declarative pipelines start with the keyword **Pipeline**. Agent any (declare where we are going to execute the code). Currently agent any means any machine we can run the code.

Stage is subtask and its under stages. every stage name is unique. A set of tasks we are going to perform.

Here just using one predefined script (Hello World).

Graphical user interface, text, application

Description automatically generated

Step 3:

Here we are taking two stages ( stage (“Hello”) and other is stage (“kaushik”). Just add this script and save and Build Now , the following screenshot will be showing.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

We also can get its console output, the screen shot is given below.

Graphical user interface, text, application, email

Description automatically generated

Step 4 :

Just taking this groovy script and trying to understand the structure, the following groovy script will be there.

pipeline {

agent any

stages {

stage ('SCM checkout stage') {

steps {

echo 'Hello World'

}

}

stage('Build') {

steps {

echo "this is new stage creation"

}

}

stage('Deploy to QA') {

steps {

echo "this is new stage creation"

}

}

stage('Automated to QA test') {

steps {

echo "this is new stage creation"

}

}

stage('Deploy to UAT') {

steps {

echo "this is new stage creation"

}

}

}

}

The following screenshot is there.

Graphical user interface, application

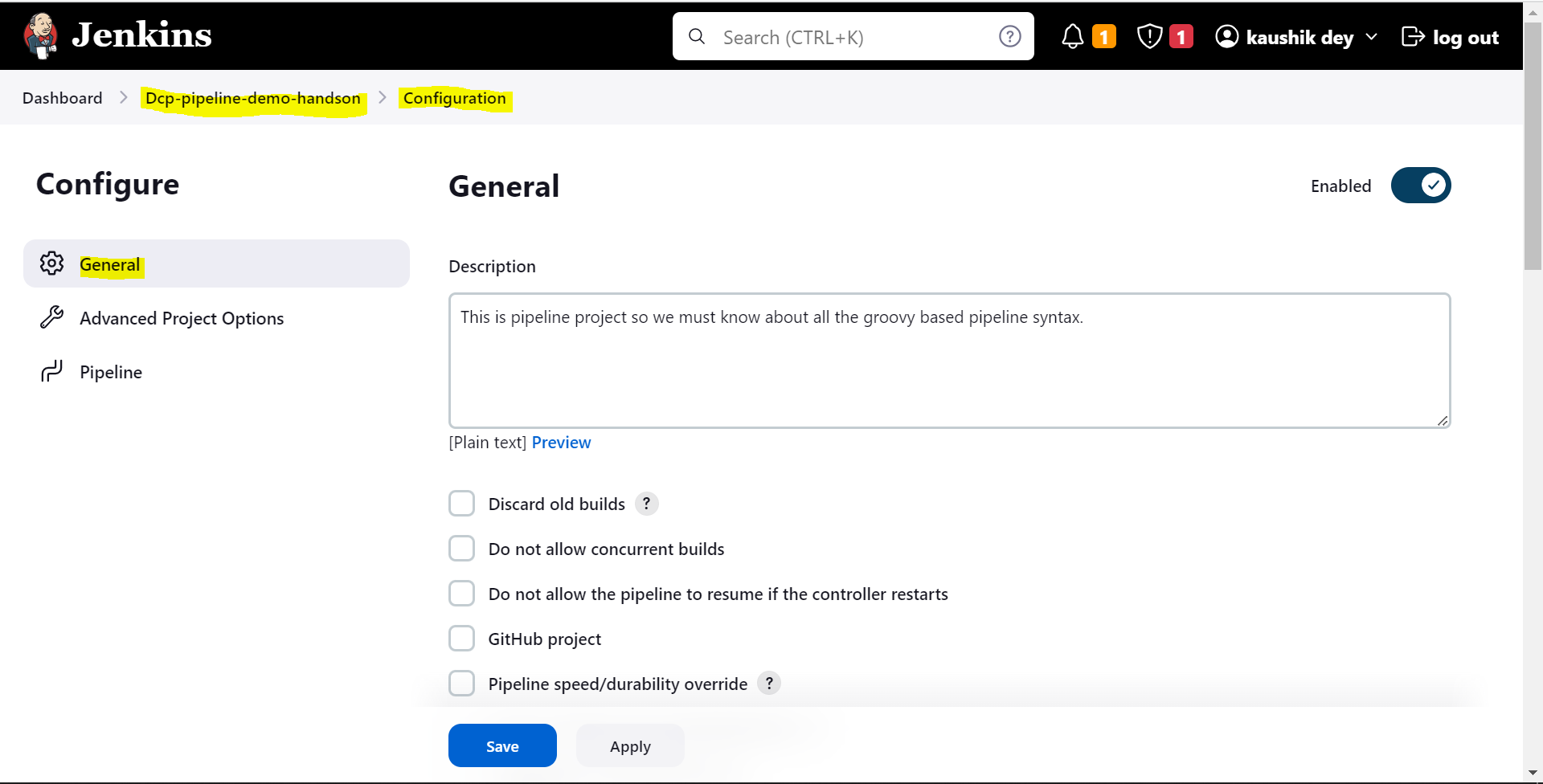
Description automatically generated

Step 5 : Now in this case we are actually going to create one pipeline with GitHub code repository.

**Our master and Slave connection demo is done and try this demo if master and slave instances are connected.**

Build a Pipeline project using Groovy Script.

Step 5.1 : Create a Pipeline Project by navigating to Jenkins -> New Item



For generating pipeline we should consider the following steps



But we can change this pipeline script . this is generated from the Hello World Template but we can edit and usable for our format.

One more thing that we can also create our own script via **pipeline syntax generator.** The screenshot is given below.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Step 5.2 :

Now in this declarative pipeline we are going to build maven project. The code syntax is given below.

In this pipeline we just want to build SCM checkout and Build block from the stage.

pipeline {

agent any

stages {

stage('SCM Checkout') {

steps {

echo 'Perform SCM Checkout'

**git 'https://github.com/LoksaiETA/Java-mvn-app2.git'**

}

}

stage('Build') {

steps {

echo 'Perform Maven build'

**sh "mvn -Dmaven.test.failure.ignore=true clean package"**

}

}

stage('Deploy to QA') {

steps {

echo 'Deploy'

}

}

stage('Automated QA test') {

steps {

echo 'Automated'

}

}

stage('Deploy to UAT') {

steps {

echo 'Automated'

}

}

stage('Deploy to PROD') {

steps {

echo 'Automated'

}

}

}

}

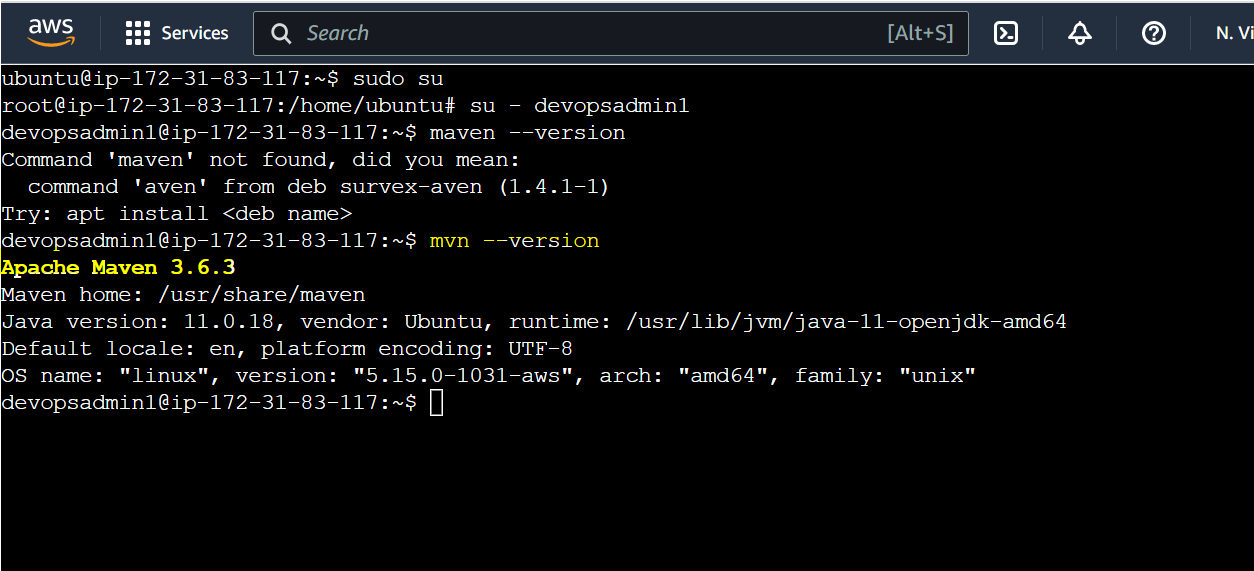


**Now if we want to run this job in master instance then agent any but if we want to run this code in slave machine then we must mention its label, the code syntax is given below.**

Code : agent { label 'slave\_m1'}

Step 5.3 :

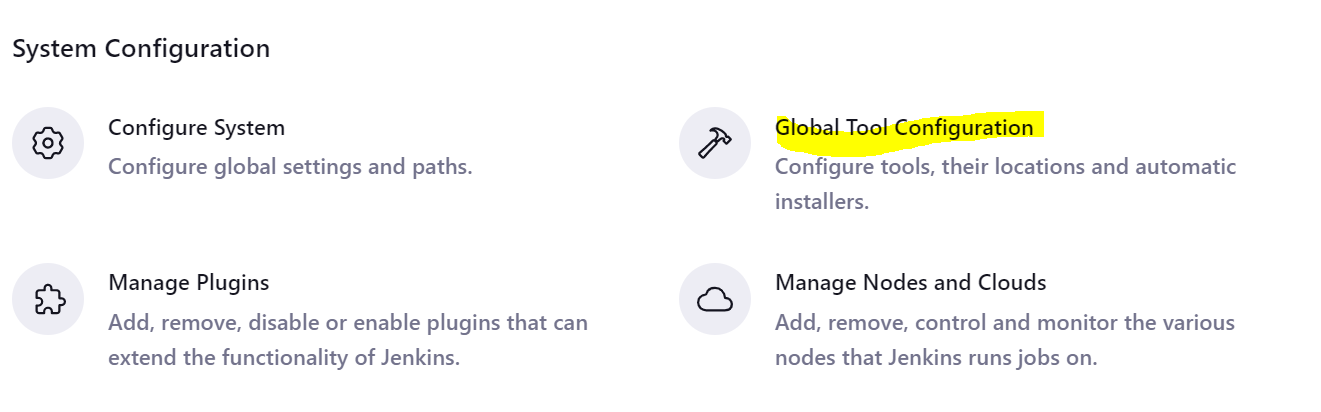
Mvn tool is installed in slave instance. Just check out it is installed or not. The screenshot is given below.



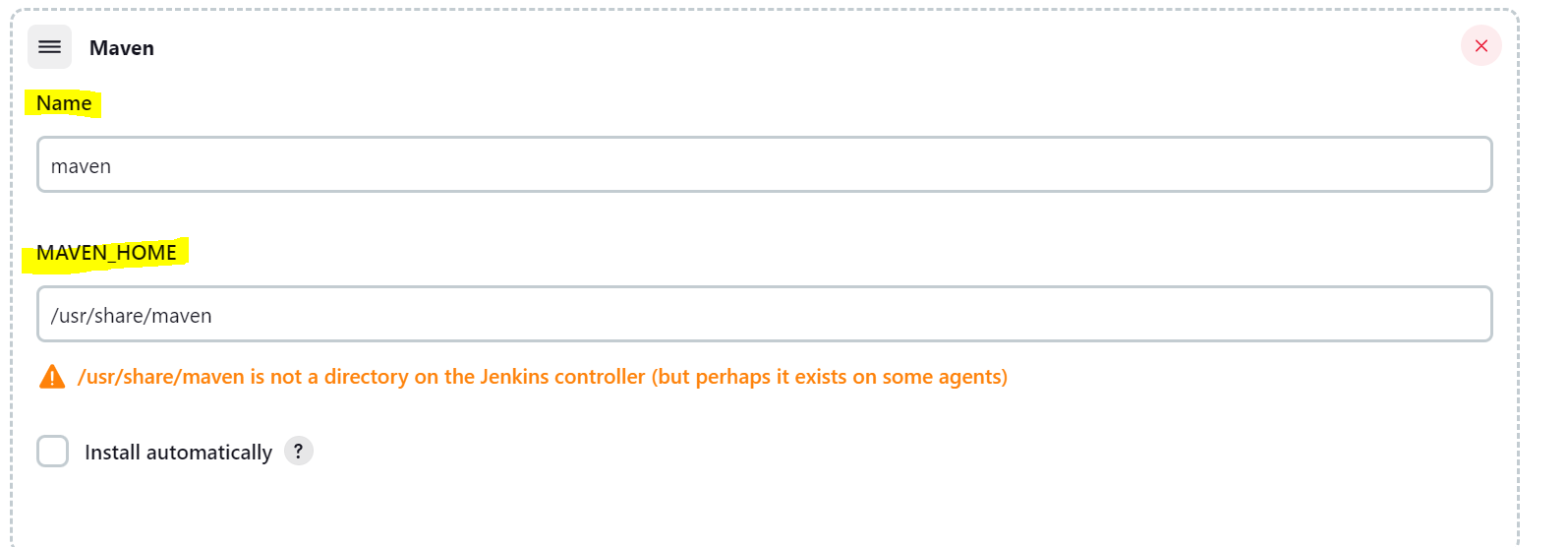
Step 5.4:

So, in global tool configuration we must add maven, so globally master and slave instance can work together with Mvn build configuration.

The screenshot is given below.



Inside global Tool configuration we must search mvn block with following screenshot.



And add slave Instance (ubuntu@ip-172-31-83-117: ~$ sudo su) Maven Home directory and it automatically detects.

/usr/share/maven is not a directory on the Jenkins controller (but perhaps it exists on some agents) this path is coming from any slave instance.

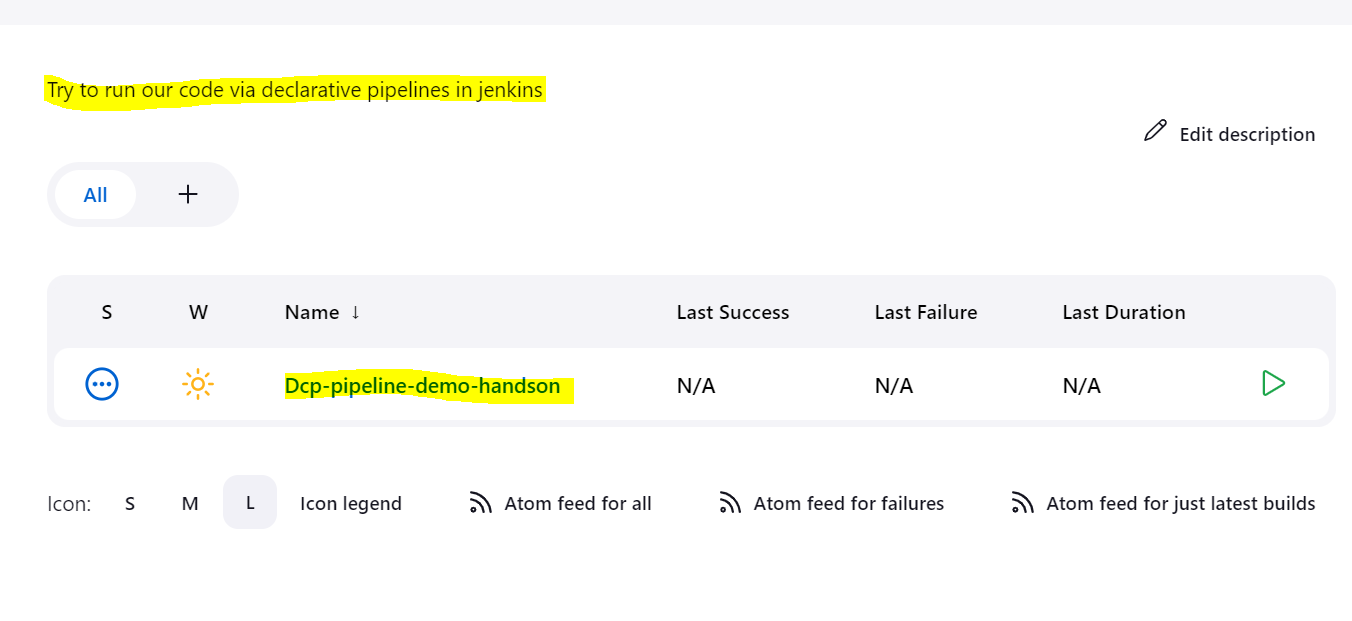
And add code for global tool configuration in our pipeline code. The code is given below.

tools { maven "maven" }

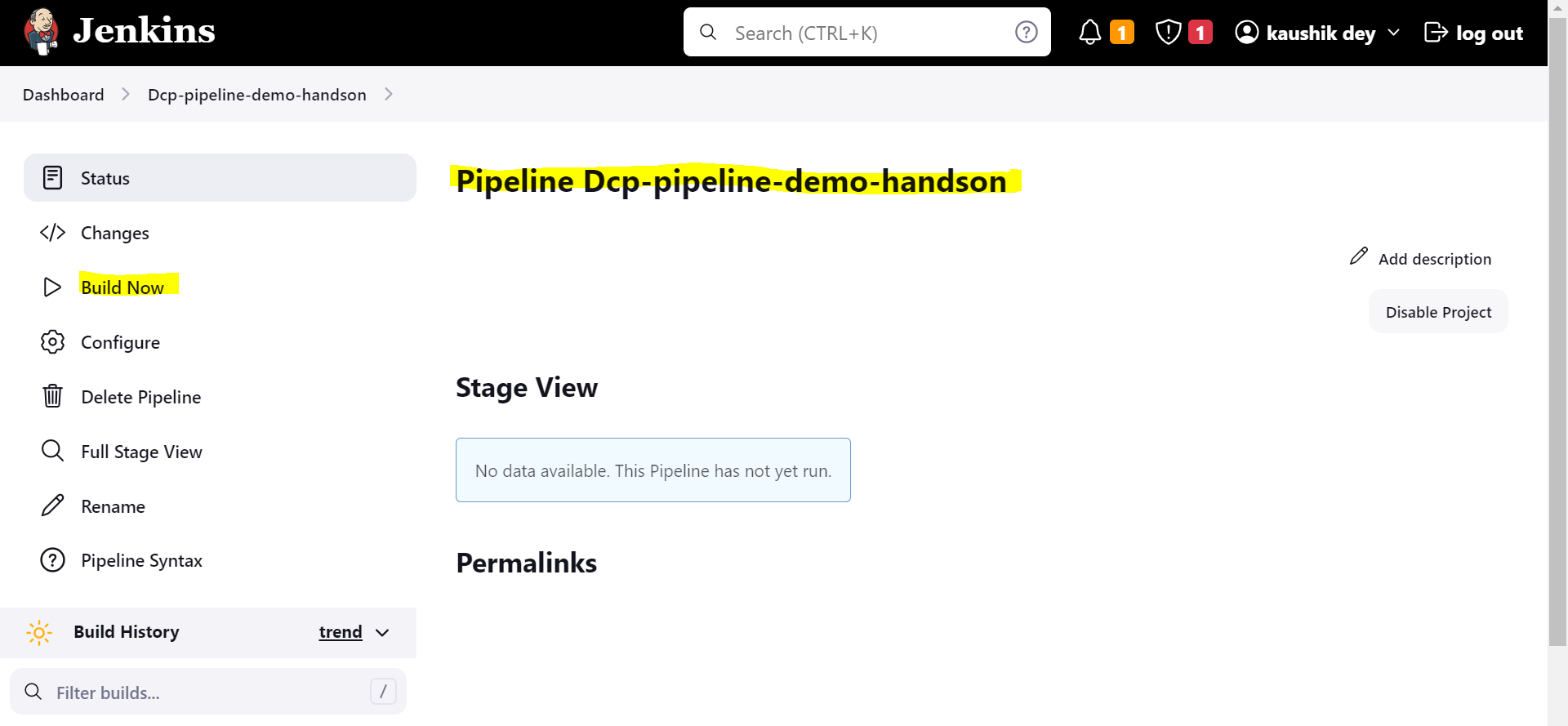
[Which name you are set in your Maven Name inside global tooling ] please remember that declarative code is also checking internally its syntaxes. If you miss anything {, } it showing error.

Step 5.5:

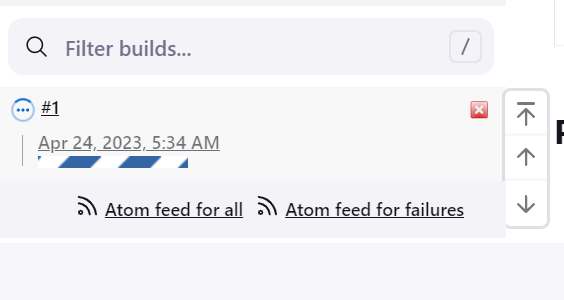
Now we have to run this code with the following screenshot.



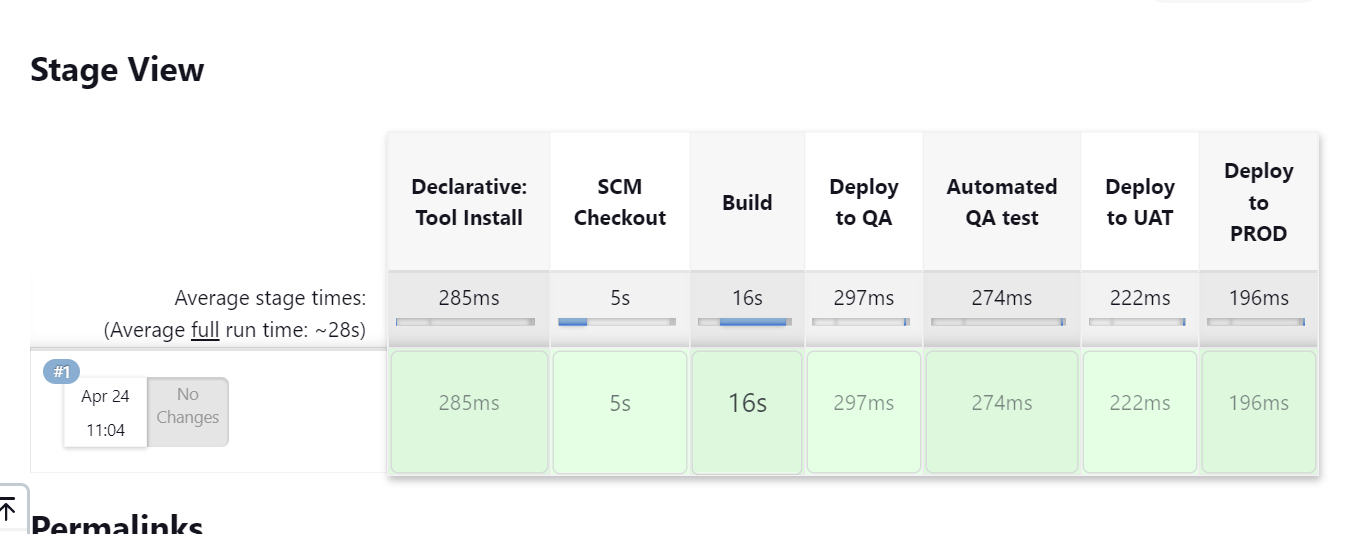
Build Now screen shot is given below. Click on Build Now button.



You can see that job is running with the following screenshots.



Now I hope that our job is succeeded. The Following screenshot is there.



See also the console output just simply click in the job number. The screenshot is given below.

Graphical user interface, text, application, chat or text message

Description automatically generated

And the console output screenshot is also given . Graphical user interface, text, application

Description automatically generated

Step 5.6:

Now check our slave Instances also that workspace is created or not there. The following screenshot is given below.

Now in our slave machine we can see that workspace folder is created.

devopsadmin1@ip-172-31-83-117:~$ ls

authorized\_keys caches remoting remoting.jar workspace

devopsadmin1@ip-172-31-83-117:~$ ls -ltr

total 1352

-rw-rw-r-- 1 devopsadmin1 devopsadmin1 0 Apr 24 03:02 authorized\_keys

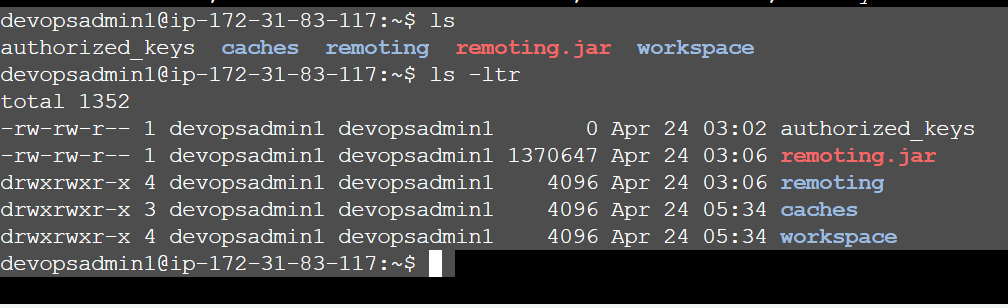
-rw-rw-r-- 1 devopsadmin1 devopsadmin1 1370647 Apr 24 03:06 remoting.jar

drwxrwxr-x 4 devopsadmin1 devopsadmin1 4096 Apr 24 03:06 remoting

drwxrwxr-x 3 devopsadmin1 devopsadmin1 4096 Apr 24 05:34 caches

drwxrwxr-x 4 devopsadmin1 devopsadmin1 4096 Apr 24 05:34 workspace

devopsadmin1@ip-172-31-83-117:~$



So, we can say that our job is successfully run in slave machine.

devopsadmin1@ip-172-31-83-117:~/workspace$ ls

Dcp-pipeline-demo-handson Dcp-pipeline-demo-handson@tmp

devopsadmin1@ip-172-31-83-117:~/workspace$ ll

total 16

drwxrwxr-x 4 devopsadmin1 devopsadmin1 4096 Apr 24 05:34 ./

drwxr-x--- 8 devopsadmin1 devopsadmin1 4096 Apr 24 05:34 ../

drwxrwxr-x 7 devopsadmin1 devopsadmin1 4096 Apr 24 05:34 Dcp-pipeline-demo-handson/

drwxrwxr-x 2 devopsadmin1 devopsadmin1 4096 Apr 24 05:34 'Dcp-pipeline-demo-handson@tmp'/

devopsadmin1@ip-172-31-83-117:~/workspace$

Text

Description automatically generated with medium confidence

We also see that in target folder the testing file is also created, the code snippets are given below.

devopsadmin1@ip-172-31-83-117:~/workspace$ ls

Dcp-pipeline-demo-handson Dcp-pipeline-demo-handson@tmp

devopsadmin1@ip-172-31-83-117:~/workspace$ cd Dcp-pipeline-demo-handson

devopsadmin1@ip-172-31-83-117:~/workspace/Dcp-pipeline-demo-handson$ ls

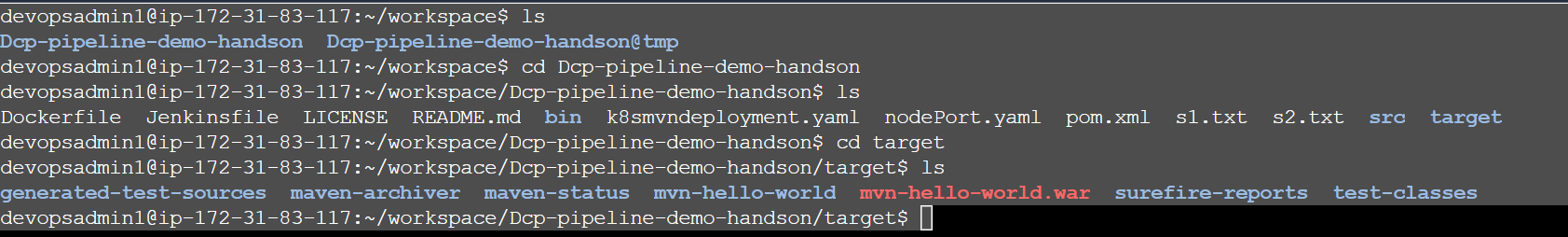
Dockerfile Jenkinsfile LICENSE README.md bin k8smvndeployment.yaml nodePort.yaml pom.xml s1.txt s2.txt src target

devopsadmin1@ip-172-31-83-117:~/workspace/Dcp-pipeline-demo-handson$ cd target

devopsadmin1@ip-172-31-83-117:~/workspace/Dcp-pipeline-demo-handson/target$ ls

generated-test-sources maven-archiver maven-status mvn-hello-world **mvn-hello-world.war** surefire-reports test-classes

devopsadmin1@ip-172-31-83-117:~/workspace/Dcp-pipeline-demo-handson/target$



Step 5.7:

Our pom.xml files looks like.

devopsadmin1@ip-172-31-83-117:~/workspace/Dcp-pipeline-demo-handson$ cat pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.dev3l.hello\_world</groupId>

<artifactId>mvn-hello-world</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<name>mvn-hello-world Maven Webapp</name>

<url>http://maven.apache.org</url>

<dependencies>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

</dependency>

<!-- junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<!-- log4j -->

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.5</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.5</version>

</dependency>

</dependencies>

<build>

<finalName>mvn-hello-world</finalName>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<!-- Java EE 6 doesn't require web.xml, Maven needs to catch up! -->

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>2.4</version>

<configuration>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</build>

</project>

devopsadmin1@ip-172-31-83-117:~/workspace/Dcp-pipeline-demo-handson$

Test our pipe Line using QA server ( SLAVE Instance).

Step 6 :

How to configure QA server.

Install Tomcat run in port 8080. To install tomcat pre-requisite is JDK.

Connecting the tomcat server with Jenkins Master.

How to Config. QA Server

Install Tomcat run in port 8080

pre-req. is jdk

create user for tomcat server

create ssh keys

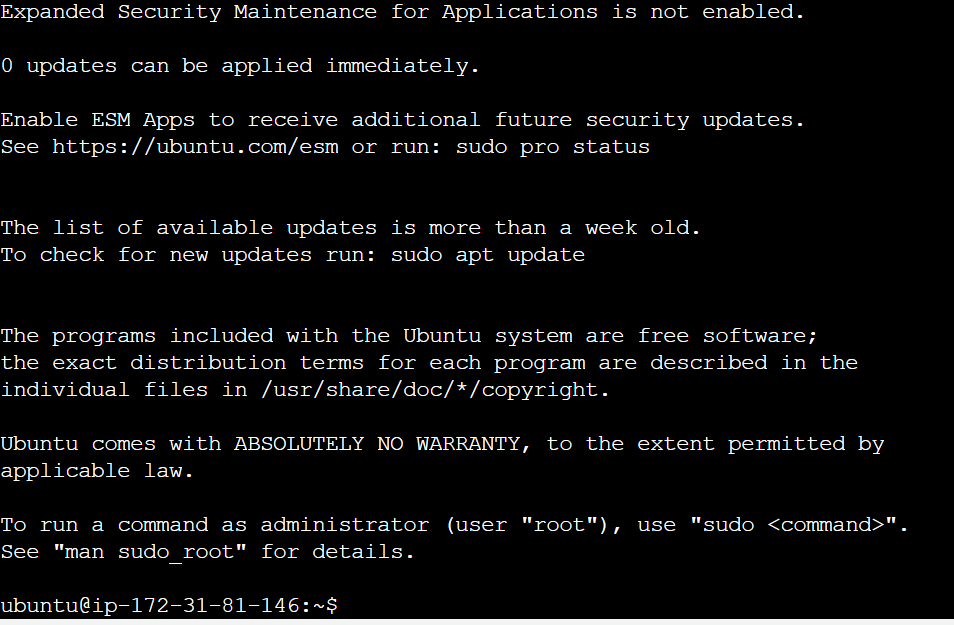
Jenkins\_Master

Jenkins\_Slave

QA\_Server

Step 7:

Our QA server is ready, the screen shot is given below.



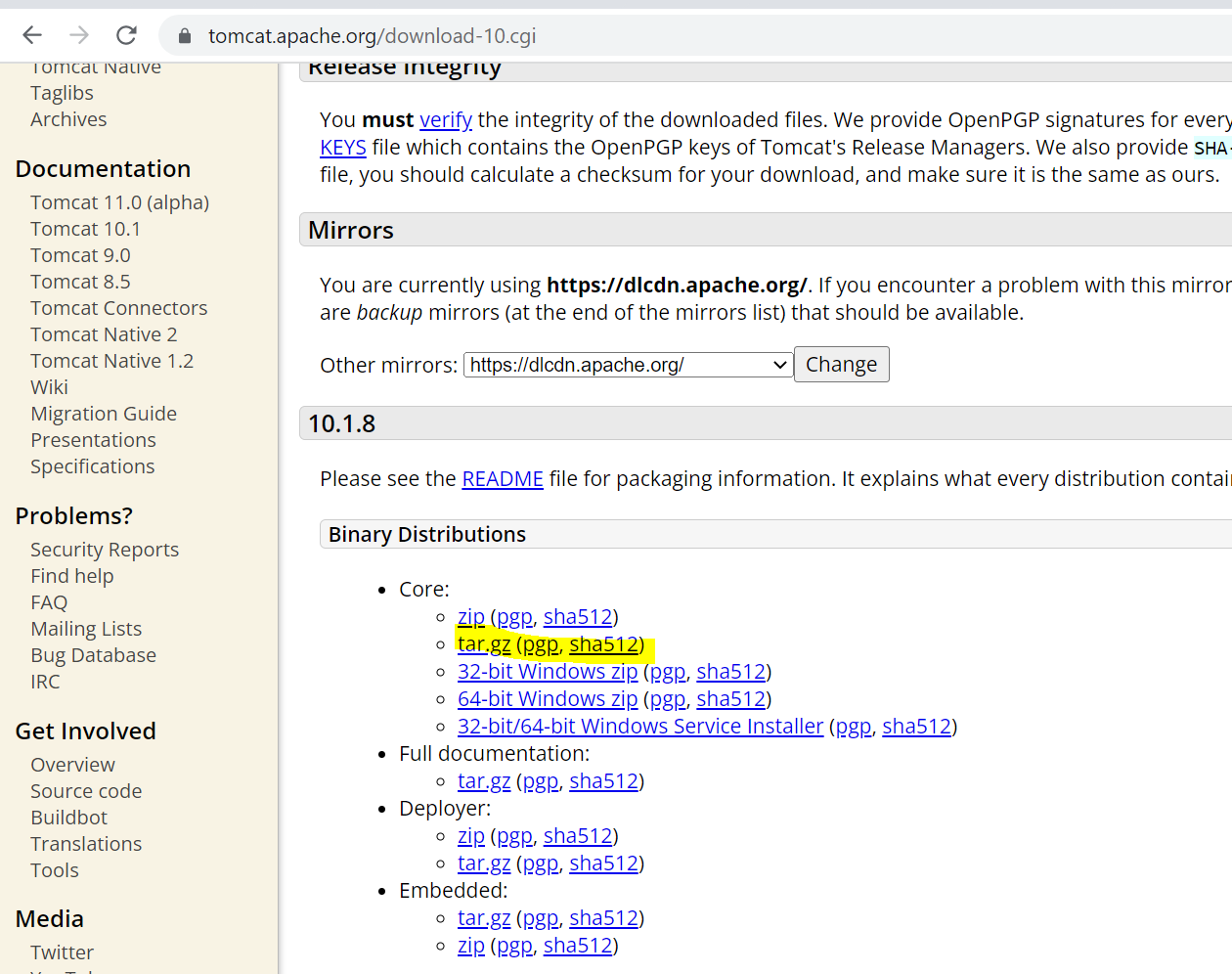
Now we have to install JDK and Tomcat in my QA server.

* sudo apt update
* sudo apt install openjdk-11-jre -y
* java –version

#Install tomcat in Amazon Linux Instance: go to the official website for tomcat installation.

<https://tomcat.apache.org/download-10.cgi>

apache-tomcat-10.1.8.tar.gz => downloaded from



* cd /opt
* wget https://dlcdn.apache.org/tomcat/tomcat-10/v10.1.8/bin/apache-tomcat-10.1.8.tar.gz
* ls
* tar -xvzf /opt/apache-tomcat-10.1.8.tar.gz
* ls
* mv apache-tomcat-10.1.8 tomcat
* ls
* cd tomcat/
* ls
* cd bin
* ls
* ./startup.sh
* history

Now got the message from QA server console ,

Using CATALINA\_BASE: /opt/tomcat

Using CATALINA\_HOME: /opt/tomcat

Using CATALINA\_TMPDIR: /opt/tomcat/temp

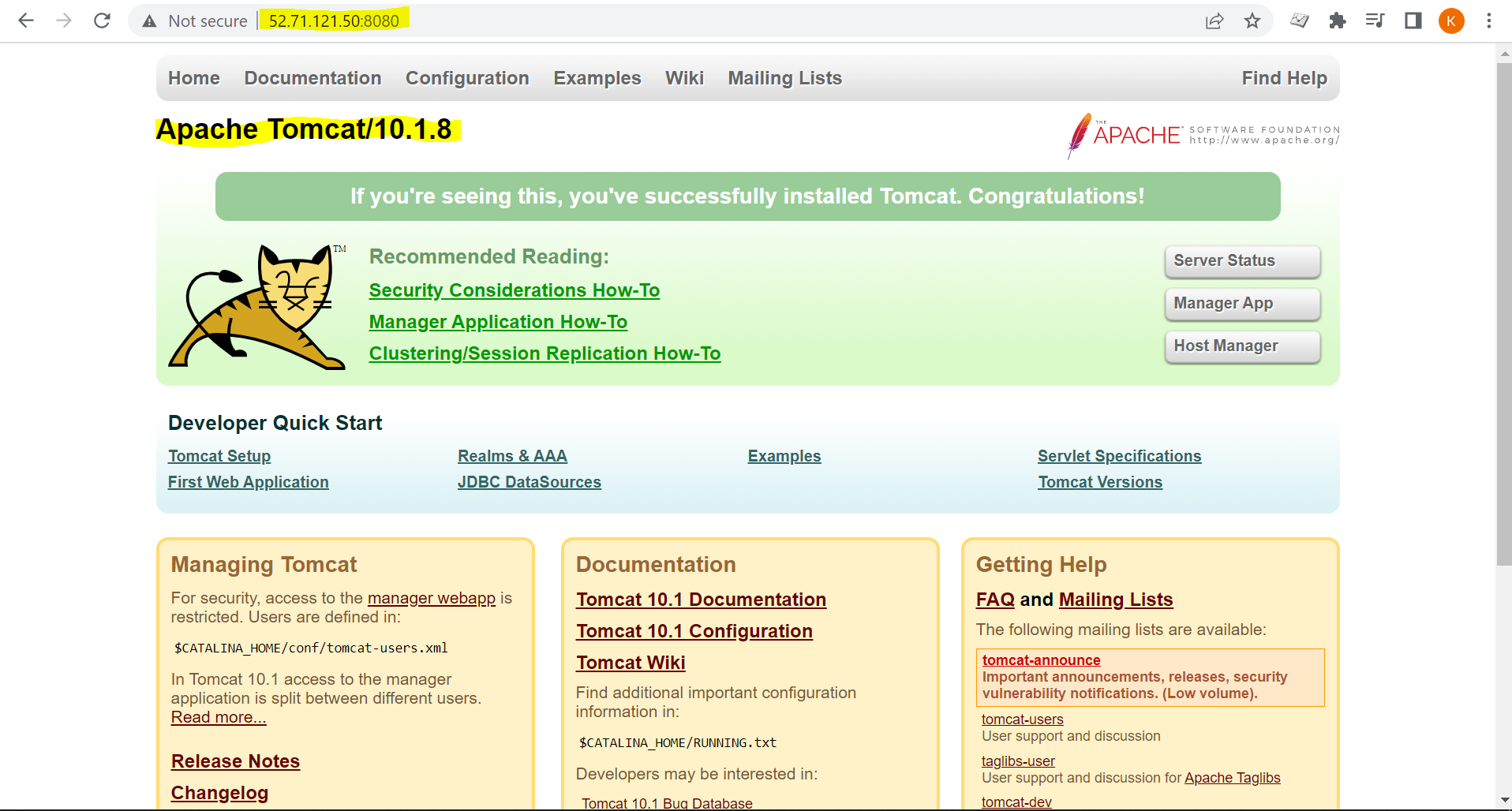
Using JRE\_HOME: /usr

Using CLASSPATH: /opt/tomcat/bin/bootstrap.jar:/opt/tomcat/bin/tomcat-juli.jar

Using CATALINA\_OPTS:

Tomcat started.

Now if we want to see tomcat page, the screen shot is given below.



Use the following command in order to create public key and secret key and those can be used for creating testing QA environment.

sudo apt update

sudo apt install openjdk-11-jre

java -version

sudo apt install maven -y

Adding a New User for Slave (QA server):

useradd -m -s /bin/bash devopsadmin

su - devopsadmin

ssh-keygen -t rsa -b 4096 -m PEM

ls ~/.ssh

#You should see following two files:

#id\_rsa - private key

#id\_rsa.pub - public

#cat id\_rsa & copy the private key and paste it into jenkins node config. enter private key directly field. Then,

cat id\_rsa.pub > authorized\_keys

chown -R devopsadmin /home/devopsadmin/.ssh

chmod 600 /home/devopsadmin/.ssh/authorized\_keys

chmod 700 /home/devopsadmin/.ssh

Screen shot is given below.

Graphical user interface, text

Description automatically generated

Step 8 :

Deployment: copy the artifacts from build server to any target QA\_servers

\*war. in slave Machine

Please note artifcats are created in workspace.

Slave: target/\*.war ==> /opt/tomcat1/webapps/\*.war

Only war file can move but not the other files.

Please note when you copy any artifacts it will be copied from this path.

root@ip-172-31-92-59:/opt/tomcat1/webapps#

To make devops admin as a owner we have to add following commands.

root@ip-172-31-92-59:/opt/tomcat1/webapps# chown -R devopsadmin /opt/tomcat1

-R means recursive. What files & folder are present we can provide admin credentials.

from the following code we can also see that root and devopsadmin in the same label.

root@ip-172-31-92-59:/opt/tomcat1/webapps# ll

total 28

drwxr-x--- 7 devopsadmin root 4096 Apr 14 19:40 ./

drwxr-xr-x 9 devopsadmin root 4096 Apr 24 11:49 ../

drwxr-x--- 3 devopsadmin root 4096 Apr 24 11:49 ROOT/

drwxr-x--- 16 devopsadmin root 4096 Apr 24 11:49 docs/

drwxr-x--- 7 devopsadmin root 4096 Apr 24 11:49 examples/

drwxr-x--- 6 devopsadmin root 4096 Apr 24 11:49 host-manager/

drwxr-x--- 6 devopsadmin root 4096 Apr 24 11:49 manager/

root@ip-172-31-92-59:/opt/tomcat1/webapps#

Step 9 : Now we must install plugins from jenkins portal.

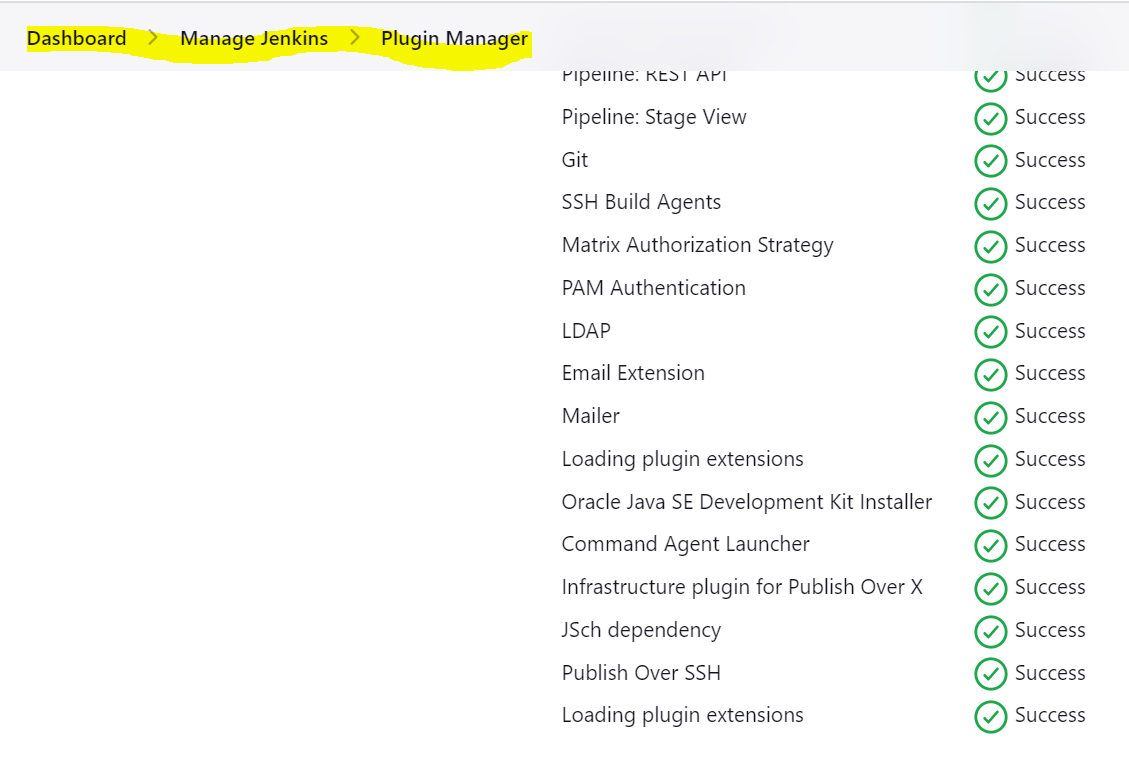
The plugin name is: Publish Over SSH.

Install this plugin in jenkins portal. The screenshot is given below. Now we must install this plugin. Publish Over SSH is used for Config. Target server in Jenkins

Graphical user interface, text, application, chat or text message

Description automatically generated

Plugins installation is successfully , the screenshot is given below.



Now restart the server and it will take some time for getting updated. Just enable the checkmark on

A picture containing graphical user interface

Description automatically generated

Now we have to configure the system for Publish over SSH. The following steps we have to perform

* Manage Jenkins -> Configure System

Now we can see that these plugins are available under configure system and the screenshot is given below.

Graphical user interface, text, application, email

Description automatically generated

Step 10 :

Pending Tomorrow will start from here.