**Tomcat Application Deployment Using Puppet For Ubuntu Node**

**Muhammed Danish M S**

172-172, 5th floor Old Mahabalipuram Road

(Above Axis Bank-PTC Bus Stop)  
Thuraipakkam  
Chennai 600097

**🖂 muhammeddanishmsnk@gmail.com**

**✆+91 8157822517**

# Contents

* Overviews
* Flows
* Requirements

I. Jenkins

II. Puppet master

III. Ubuntu node

* Jenkins machine
* Jenkins Dashboard Prerequisites
* Result

# Overview

This article is about deploy of Tomcat web application in nodes using Puppet Master. Access source code from git repository and deploy to puppet nodes by master using Jenkins. Git is a source code repository manager. Source code is deployed to puppet master by using Jenkins. And that source code is copied to the given puppet nodes by writing puppet manifests. We have to write puppet modules for deploy of tomcat in Ubuntu and glowing a git file in tomcat.

Note:

Source code URL: https://github.com/muhammeddanish/java-app/

# Flow

**GIT => JENKINS => PUPPET MASTER => UBUNTU NODE**

# Requirments

Two centos machine and one Ubuntu. One Centos with 2Cpu, 3GB RAM with 50 GB disk files for puppet master. And other centos with 1CPU 1GB RAM and with 30GB disk space for Jenkins. The Ubuntu machine is with 1 CPU, 1 GB RAM and storage of 30 GB and is used as Ubuntu node.

Jenkins

Jenkins is an open-source continuous integration software tool written in the Java programming language for testing and reporting on isolated changes in a larger code base in real time. The software enables developers to find and solve defects in a code base rapidly and to automate testing of their builds.

Continuous integration has evolved since its conception. Originally, a daily build was the standard. Now, the usual rule is for each team member to submit work on a daily (or more frequent) basis and for a build to be conducted with each significant change. When used properly, continuous integration provides various benefits, such as constant feedback on the status of the software. Because CI detects deficiencies early on in development, defects are typically smaller, less complex and easier to resolve.

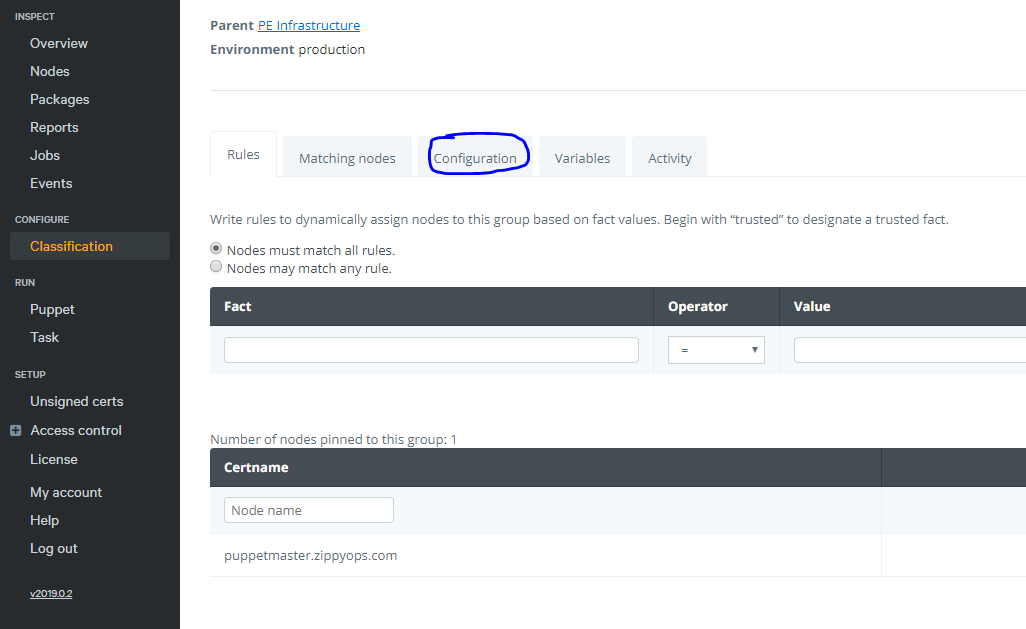
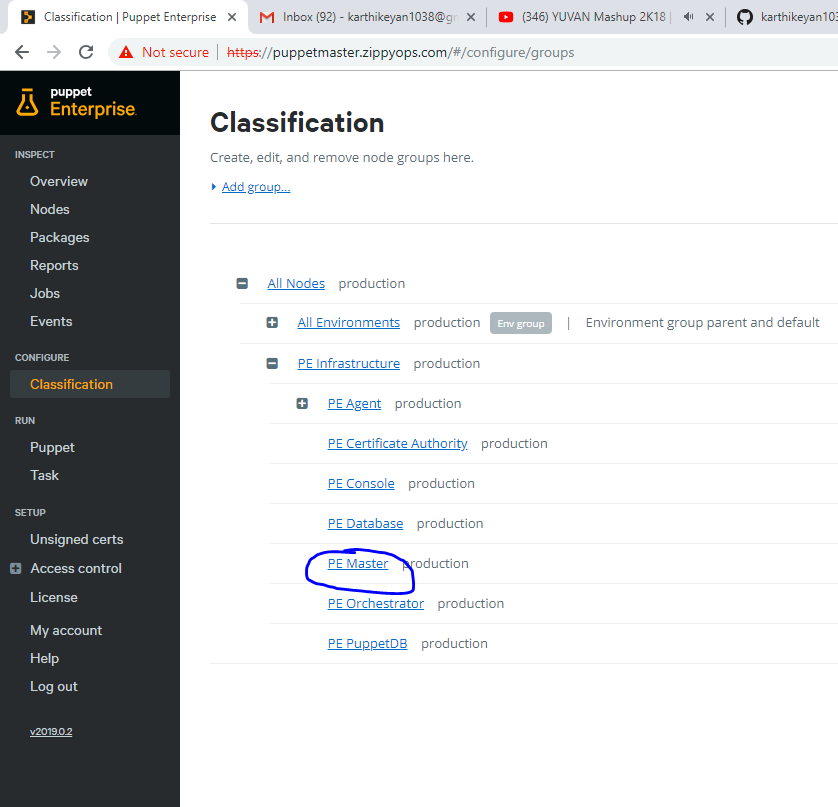
PUPPET:

Puppet is a configuration management tool that helps system administrators automate the provisioning, configuration, and management of a server infrastructure. Planning ahead and using config management tools like Puppet can cut down on time spent repeating basic tasks, and help ensure that your configurations are consistent and accurate across your infrastructure. Once you get the hang of managing your servers with Puppet and other automation tools, you will free up time which can be spent improving other aspects of your overall setup process. Puppet uses a client/server model where the managed servers, called Puppet agents, talk to and pull down configuration profiles from the Puppet master.

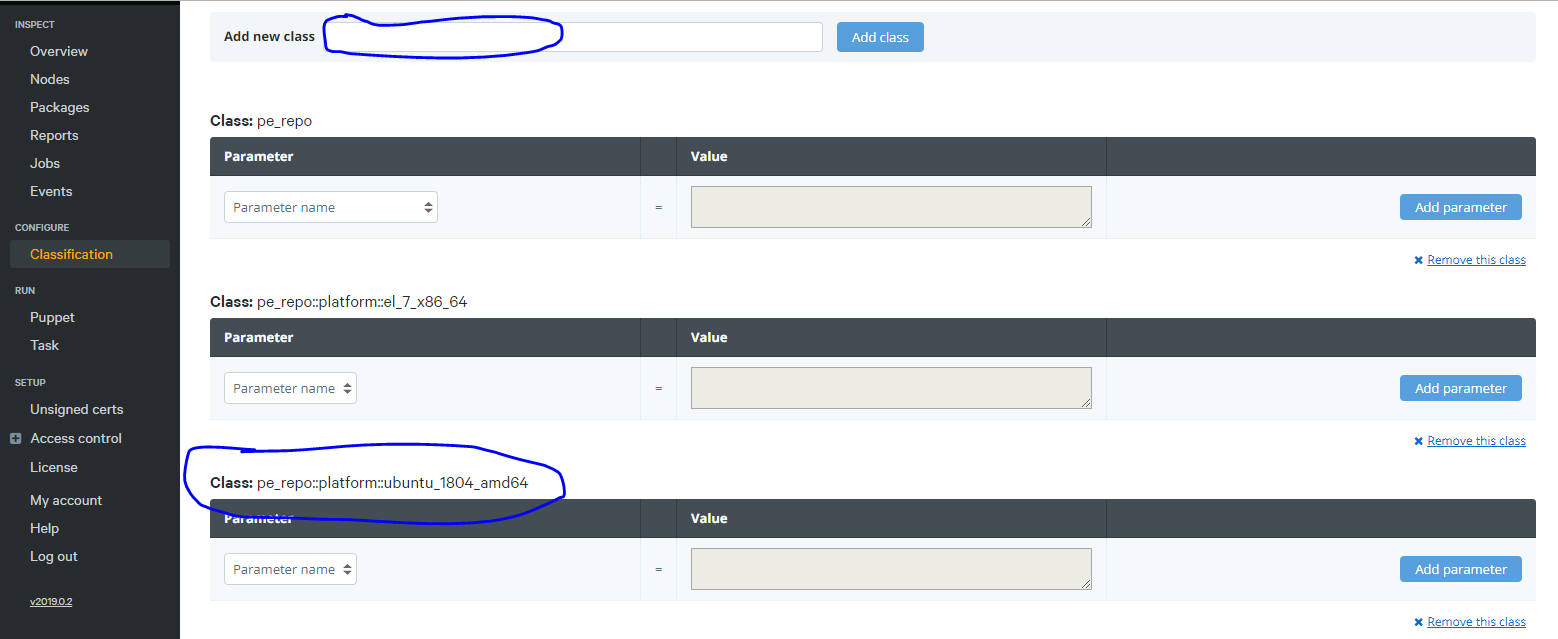
# Step-1 PUPPET MASTER

* Ensure that the machine is installed with the latest version puppet.
* After installing puppet run **puppet agent –t.**
* Enter host entries of master and node in **vi /etc/hosts.**
* Set time and date by using **timedatectl set-timezone Asia/Kolkata**
* Stop and disable firewalld by using the command: **Systemctl stop firewalld && systemctl disable firewalld**
* Disable Selinux by editing config file: **vi /etc/selinux/config => disabled**
* Run the console by giving ip address. I have used **192.168.1.70** for puppet master.
* Go to console and get to:

**CLASSIFICATION => PE INFRASTRUCTURE => PE MASTER**



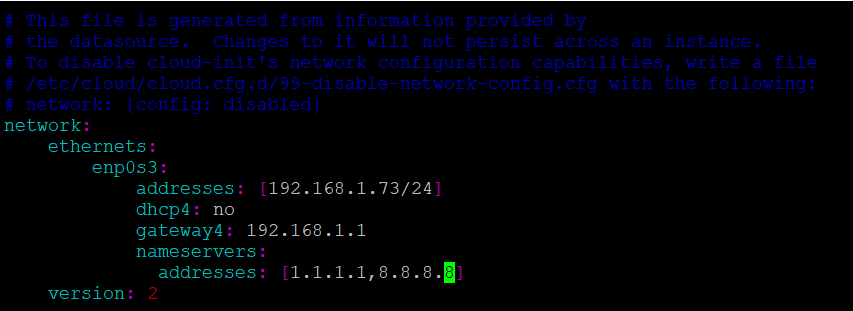
* Enter the **repo for Ubuntu 18.04** **pe\_repo:platform::Ubuntu\_1804\_amd64** so the puppet master will connect to Ubuntu node, otherwise it will not get connected. Then give commit.

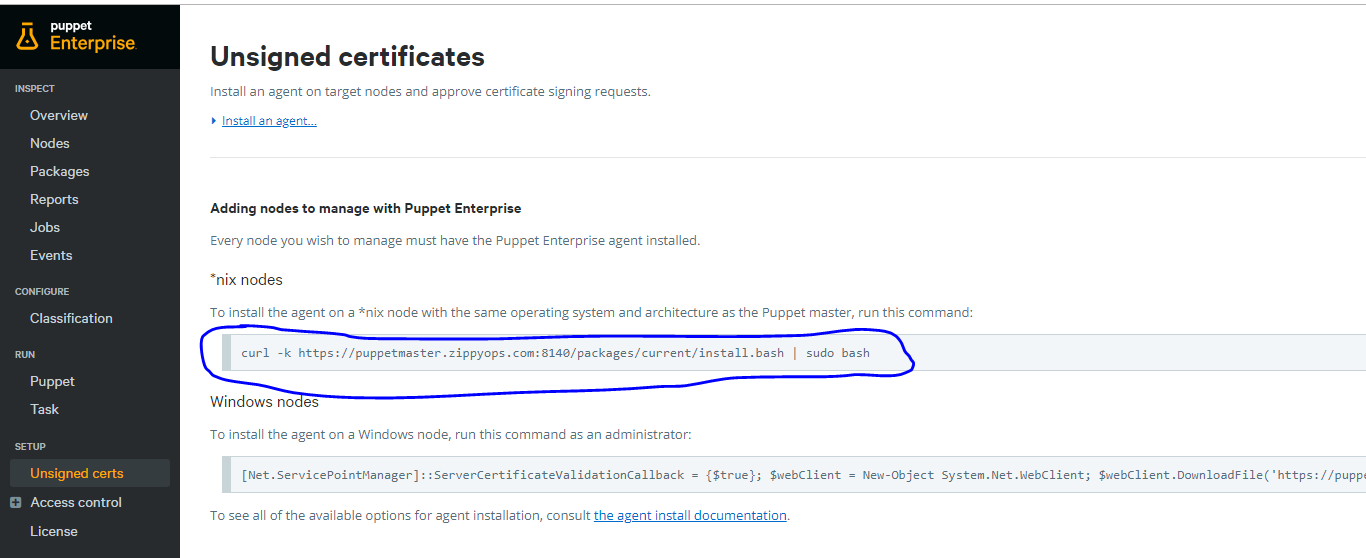


* Run **Puppet agent –t** in master after the repo added so it sucks all the configuration from the server puppet Ubuntu node.

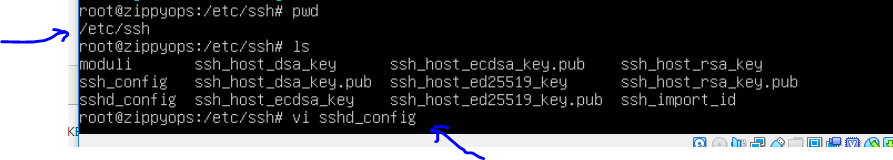
# Step-2 puppet ubuntu node

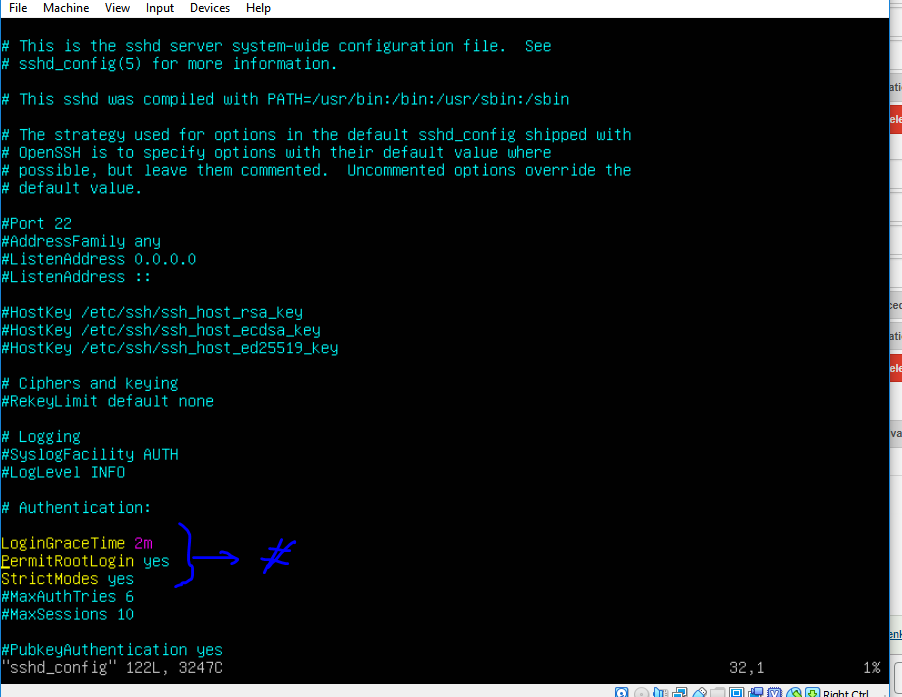
* Run the following commands in Ubuntu node:

1. **sudo apt-get update && apt-get upgrade**
2. **vi /etc/hosts =>** enter the master and node host entries.
3. **systemctl stop ufw && systemctl disable ufw**
4. Set static ip by editing **vi /etc/netplan/50-cloud-init.yaml**
5. **netplan apply**
6. Set time in Ubuntu node => **timedatectl set-timezone Asia/Kolkata**
7. Copy and paste the certificate to Ubuntu node( using the CURL statement) and now Ubuntu node is connected to master.



## ssh configuration in **ubuntu node** for jenkins connectivity





Note: This configuration is done to login as a root user for Ubuntu, basically it is logged in as main user.

# Step-3 puppet module creation

* In master node, go to **cd /etc/puppetlabs/code/environments/production/modules** and create a directory there, in name of the module we need to create:

**mkdir tomcat**

* Then move in to the tomcat directory by using cd command:

**cd tomcat**

* Create 2 more directories inside tomcat, named files and manifests:

**mkdir files manifests**

* Then go to manifests, **cd manifests**.
* Inside manifests create a file named init.pp:

**vi init.pp**

* Inside init.pp, add the following contents:

class tomcat {

package {'default-jdk':

ensure => present,

}

exec { 'useradd':

cwd => '/home/zippyops',

command => 'useradd -r -m -U -d /opt/tomcat -s /bin/false tomcat',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

exec { 'wget':

cwd => '/home/zippyops',

command => 'wget http://mirrors.estointernet.in/apache/tomcat/tomcat-9/v9.0.16/bin/apache-tomcat-9.0.16.tar.gz -P /tmp',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

exec { 'tar':

cwd => '/home/zippyops',

command => 'tar xf /tmp/apache-tomcat-9\*.tar.gz -C /opt/tomcat',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

exec { 'softlink':

cwd => '/home/zippyops',

command => 'ln -s /opt/tomcat/apache-tomcat-9.0.16 /opt/tomcat/latest',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

exec { 'change owner':

cwd => '/home/zippyops',

command => 'chown -RH tomcat: /opt/tomcat/latest',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

exec { 'change mode':

cwd => '/home/zippyops',

command => 'chmod +x /opt/tomcat/latest/bin/\*.sh',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

file { 'tomcat.service':

path => '/etc/systemd/system/tomcat.service',

ensure => file,

source => 'puppet:///modules/tomcat/tomcat.service',

}

exec { 'daemon-reload':

cwd => '/home/zippyops',

command => 'systemctl daemon-reload',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

service {'tomcat':

ensure => running,

enable => true,

}

file { 'tomcat-users.xml':

path => '/opt/tomcat/latest/conf/tomcat-users.xml',

ensure => file,

source => 'puppet:///modules/tomcat/tomcat-users.xml',

}

file { 'manager':

path => '/opt/tomcat/latest/webapps/manager/META-INF/context.xml',

ensure => file,

source => 'puppet:///modules/tomcat/context.xml',

}

file { 'host-manager':

path => '/opt/tomcat/latest/webapps/host-manager/META-INF/context.xml',

ensure => file,

source => 'puppet:///modules/tomcat/context.xml',

}

exec { 'tomcat-reload':

cwd => '/home/zippyops',

command => 'systemctl restart tomcat',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

file { 'javaapp':

path => '/opt/tomcat/latest/webapps/java-sample-app-1.0.0.war',

ensure => file,

source => 'puppet:///modules/tomcat/java-sample-app-1.0.0.war',

}

exec { 'tomcat-reload1':

cwd => '/home/zippyops',

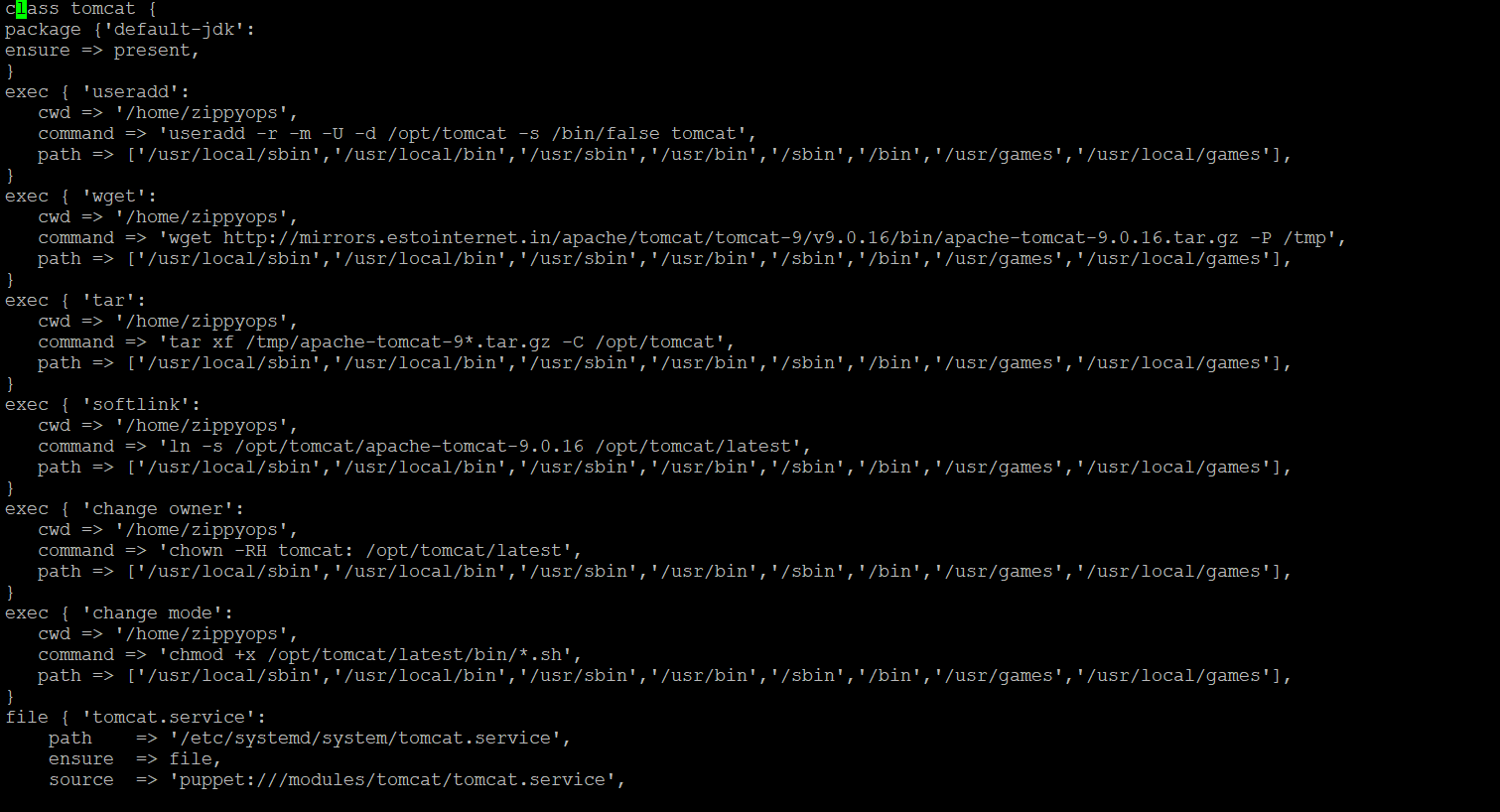
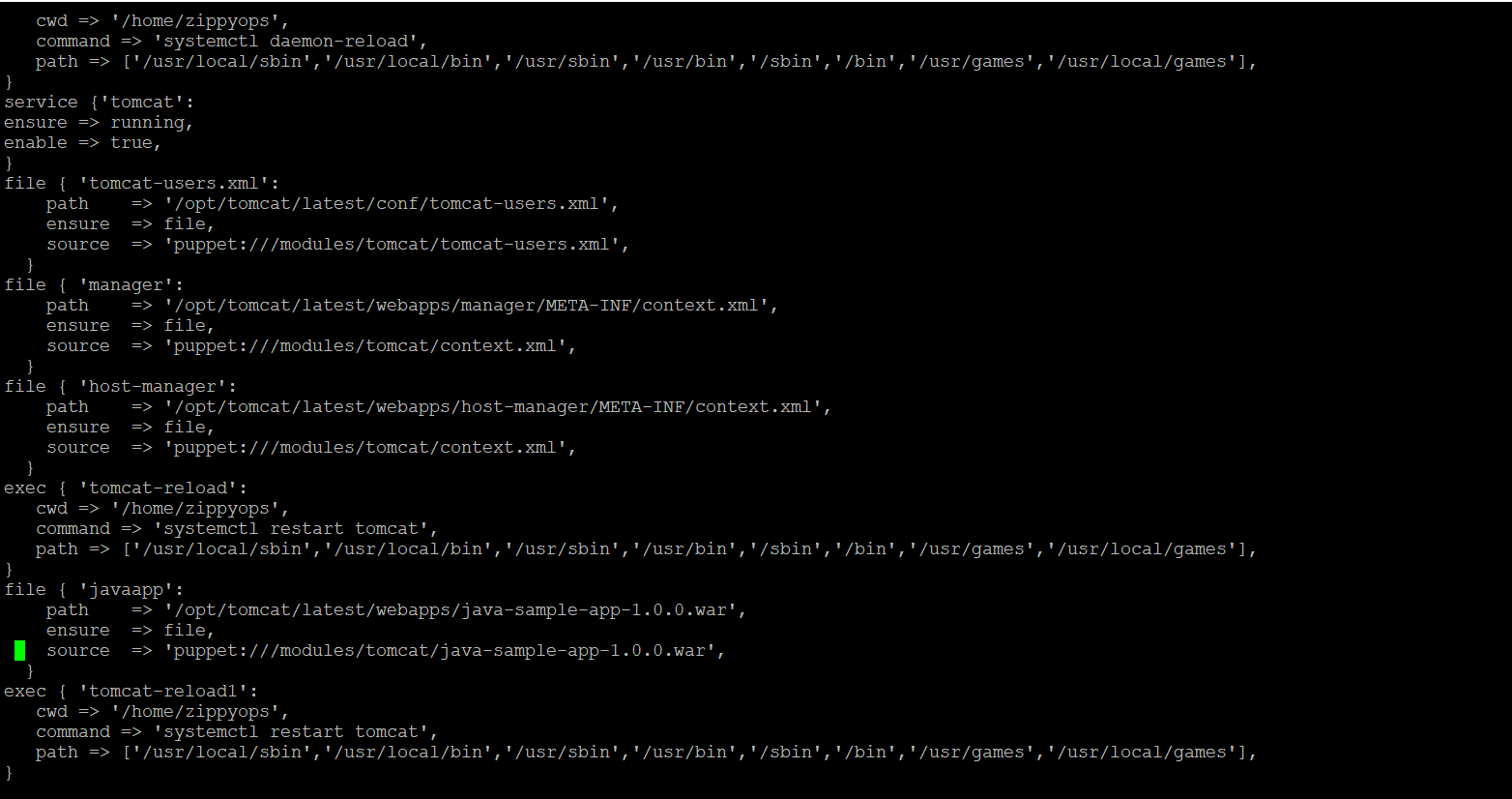
command => 'systemctl restart tomcat',

path => ['/usr/local/sbin','/usr/local/bin','/usr/sbin','/usr/bin','/sbin','/bin','/usr/games','/usr/local/games'],

}

}

Note: Use the same name for class and module.



* **cd ..** and then move to the directory file **by cd file**
* Inside file create 3 dependency files for init.pp:

1. Create a file named context.xml:

**vi context.xml**

And add the below contents to the file context.xml:

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

<!--

Licensed to the Apache Software Foundation (ASF) under one or more

contributor license agreements. See the NOTICE file distributed with

this work for additional information regarding copyright ownership.

The ASF licenses this file to You under the Apache License, Version 2.0

(the "License"); you may not use this file except in compliance with

the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

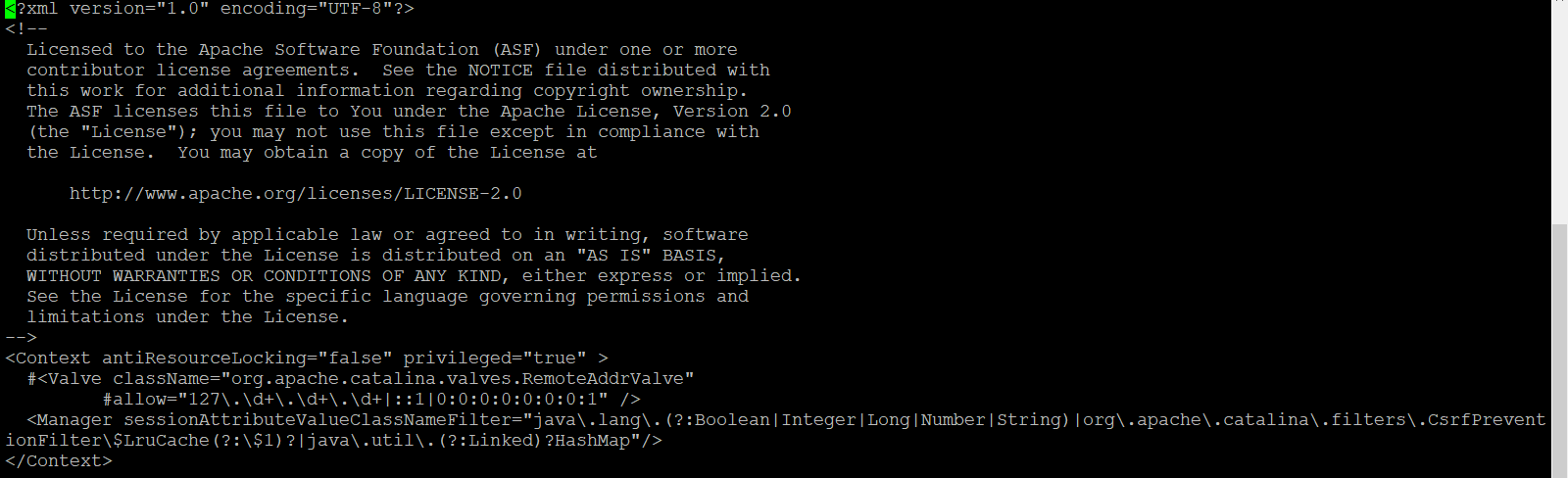
-->

<Context antiResourceLocking="false" privileged="true" >

#<Valve className="org.apache.catalina.valves.RemoteAddrValve"

#allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />

<Manager sessionAttributeValueClassNameFilter="java\.lang\.(?:Boolean|Integer|Long|Number|String)|org\.apache\.catalina\.filters\.CsrfPreventionFilter\$LruCache(?:\$1)?|java\.util\.(?:Linked)?HashMap"/>

</Context>

1. Second file: **vi tomcat.service**

[Unit]

Description=Tomcat 9 servlet container

After=network.target

[Service]

Type=forking

User=tomcat

Group=tomcat

Environment="JAVA\_HOME=/usr/lib/jvm/default-java"

Environment="JAVA\_OPTS=-Djava.security.egd=file:///dev/urandom -Djava.awt.headless=true"

Environment="CATALINA\_BASE=/opt/tomcat/latest"

Environment="CATALINA\_HOME=/opt/tomcat/latest"

Environment="CATALINA\_PID=/opt/tomcat/latest/temp/tomcat.pid"

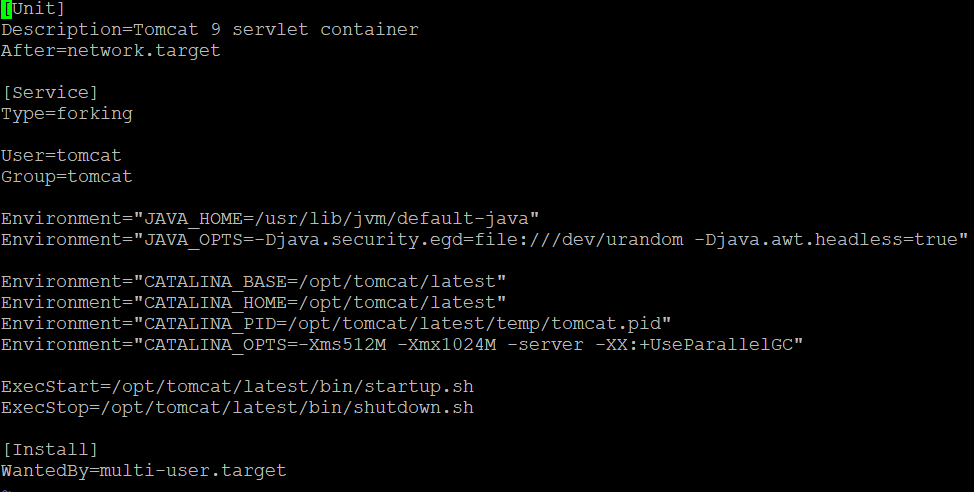
Environment="CATALINA\_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC"

ExecStart=/opt/tomcat/latest/bin/startup.sh

ExecStop=/opt/tomcat/latest/bin/shutdown.sh

[Install]

WantedBy=multi-user.target

1. third file : **vi tomcat-users.xml**

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

<!--

Licensed to the Apache Software Foundation (ASF) under one or more

contributor license agreements. See the NOTICE file distributed with

this work for additional information regarding copyright ownership.

The ASF licenses this file to You under the Apache License, Version 2.0

(the "License"); you may not use this file except in compliance with

the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License.

-->

<tomcat-users xmlns="http://tomcat.apache.org/xml"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"

version="1.0">

<!--

NOTE: By default, no user is included in the "manager-gui" role required

to operate the "/manager/html" web application. If you wish to use this app,

you must define such a user - the username and password are arbitrary. It is

strongly recommended that you do NOT use one of the users in the commented out

section below since they are intended for use with the examples web

application.

-->

<!--

NOTE: The sample user and role entries below are intended for use with the

examples web application. They are wrapped in a comment and thus are ignored

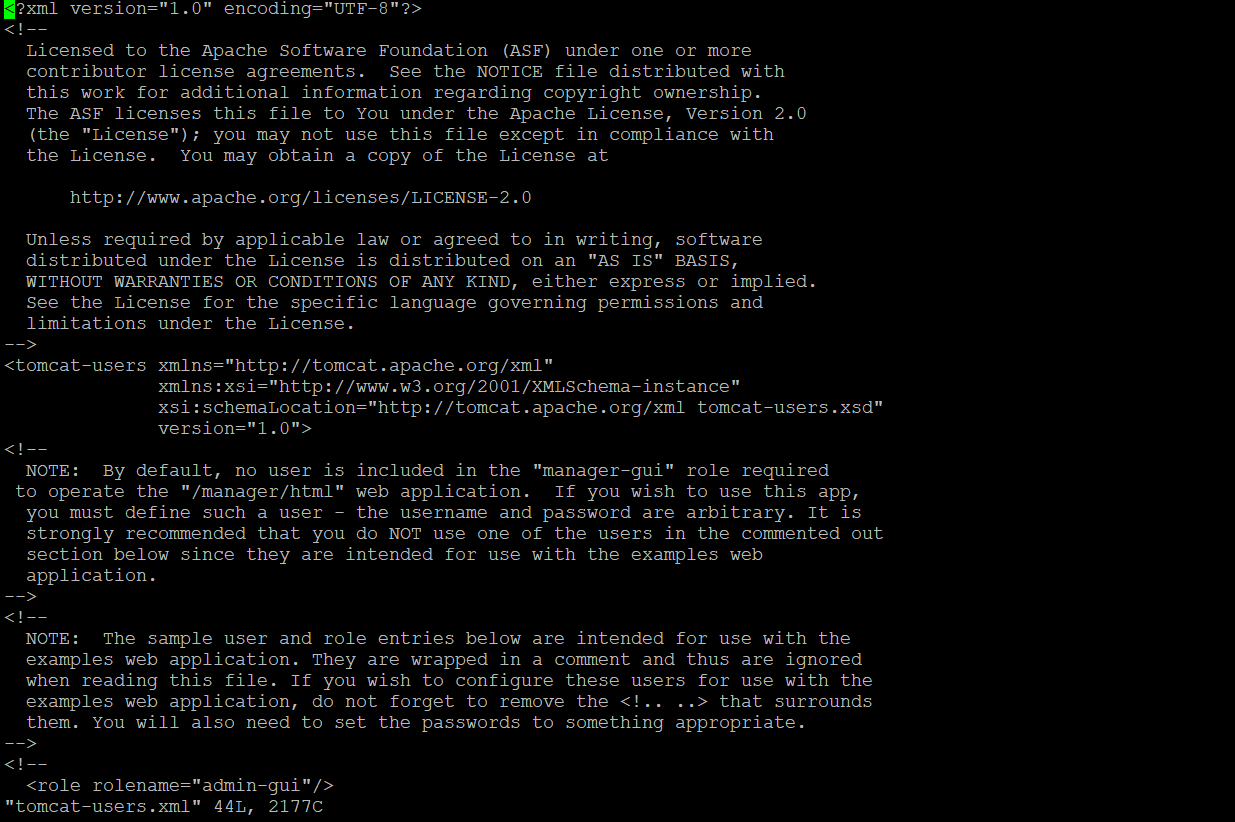
when reading this file. If you wish to configure these users for use with the

examples web application, do not forget to remove the <!.. ..> that surrounds

them. You will also need to set the passwords to something appropriate.

-->

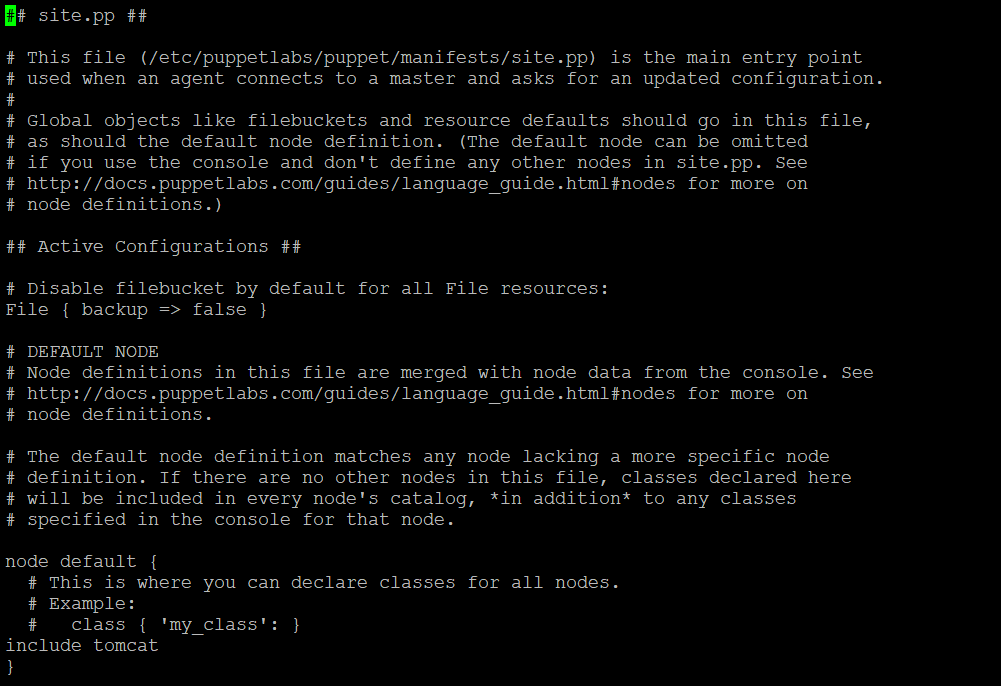
<!--

 <role rolename="admin-gui"/>

* Next move to site.pp file of puppet master**:**

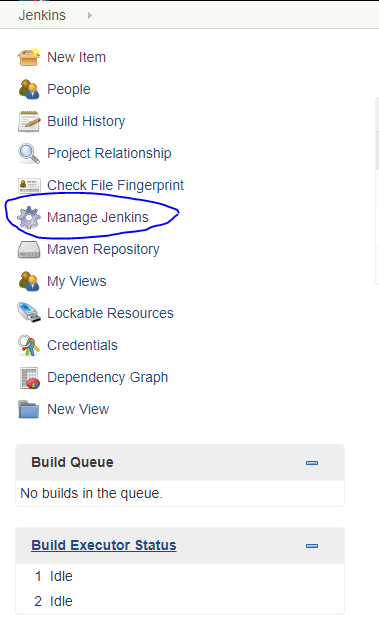
**vi /etc/puppetlabs/code/environments/production/manifests/site.pp**

Then add class name tomcat to the site.pp file

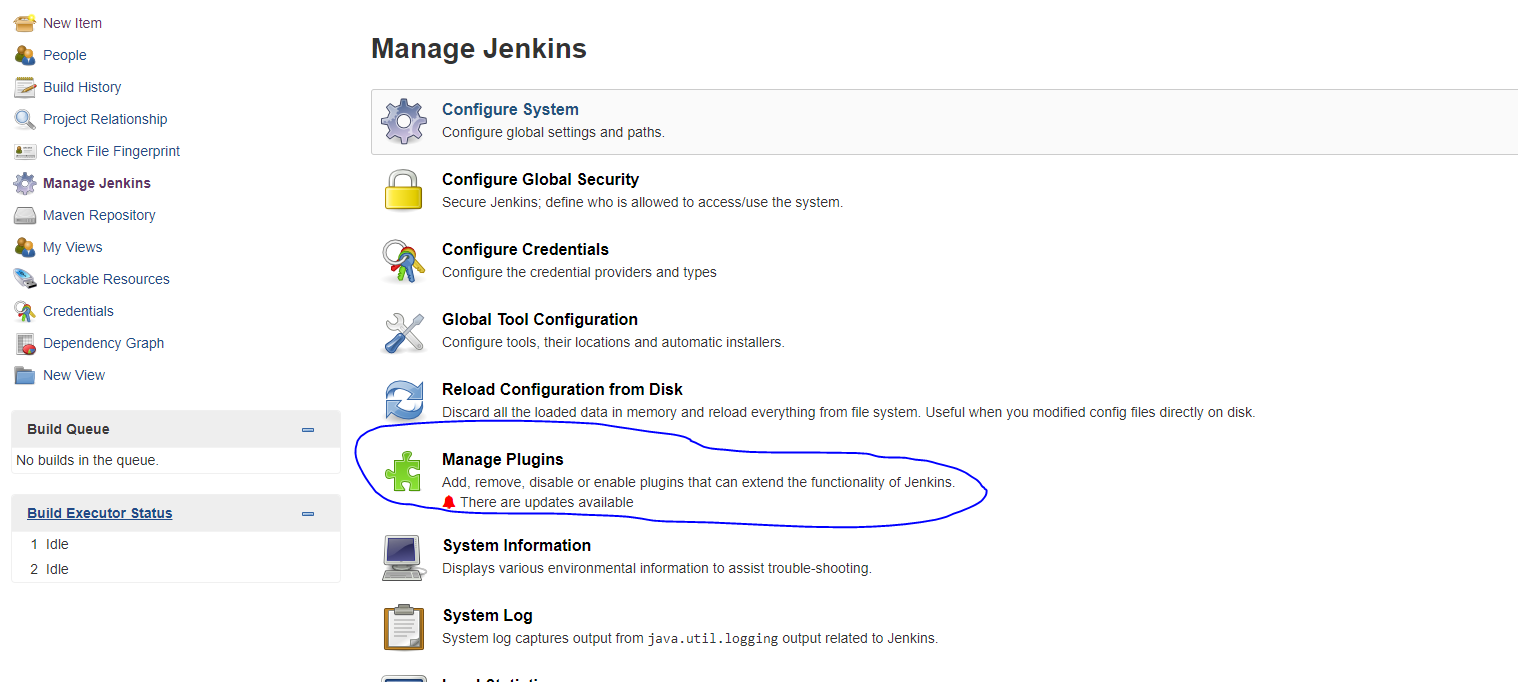


# Step-2 Jenkins Dashboard Prerequisites

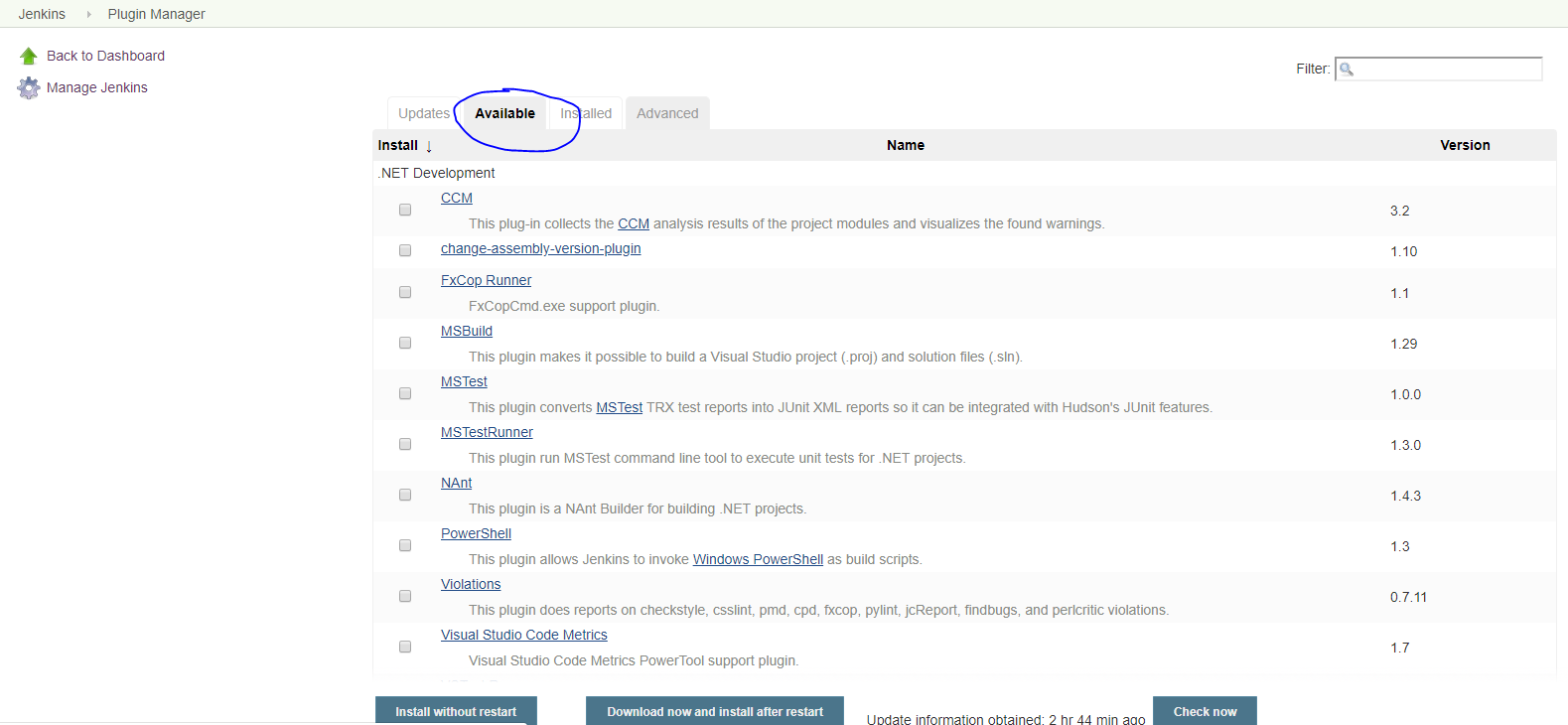
* Open the Jenkins Dashboard and click the manage Jenkins and set the environment for the project building process.

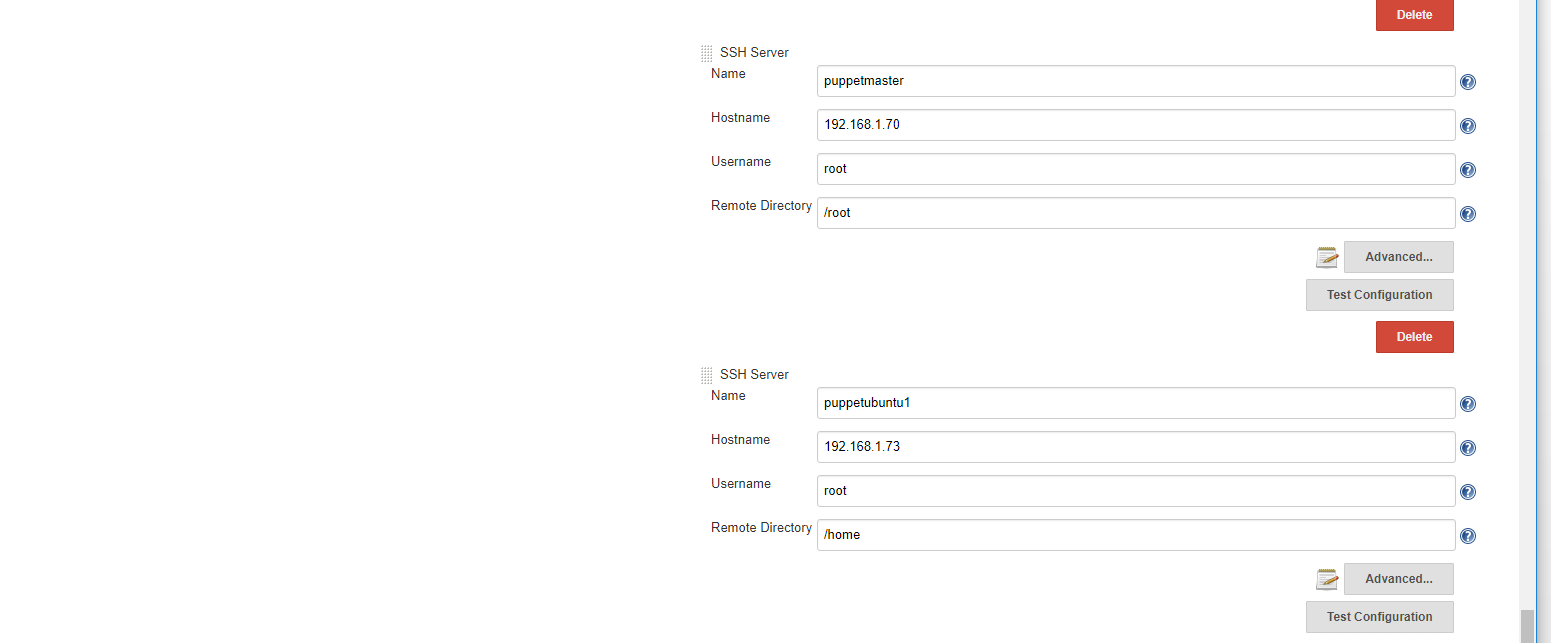


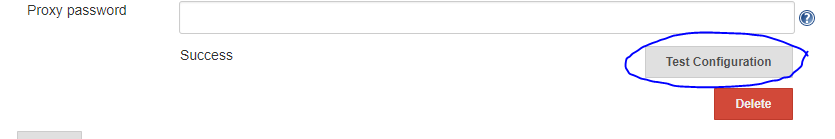
* First click the manage plugins and add plugins required for the task we are going to do.

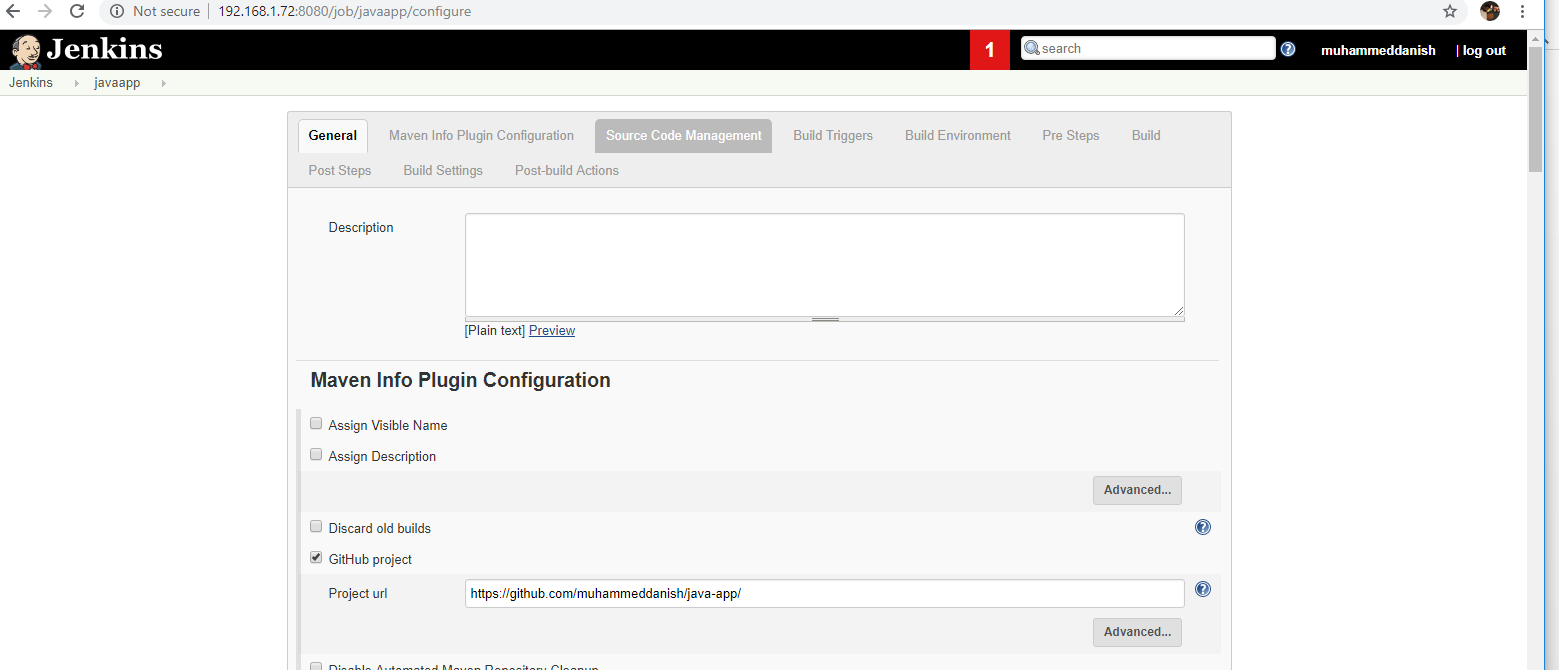


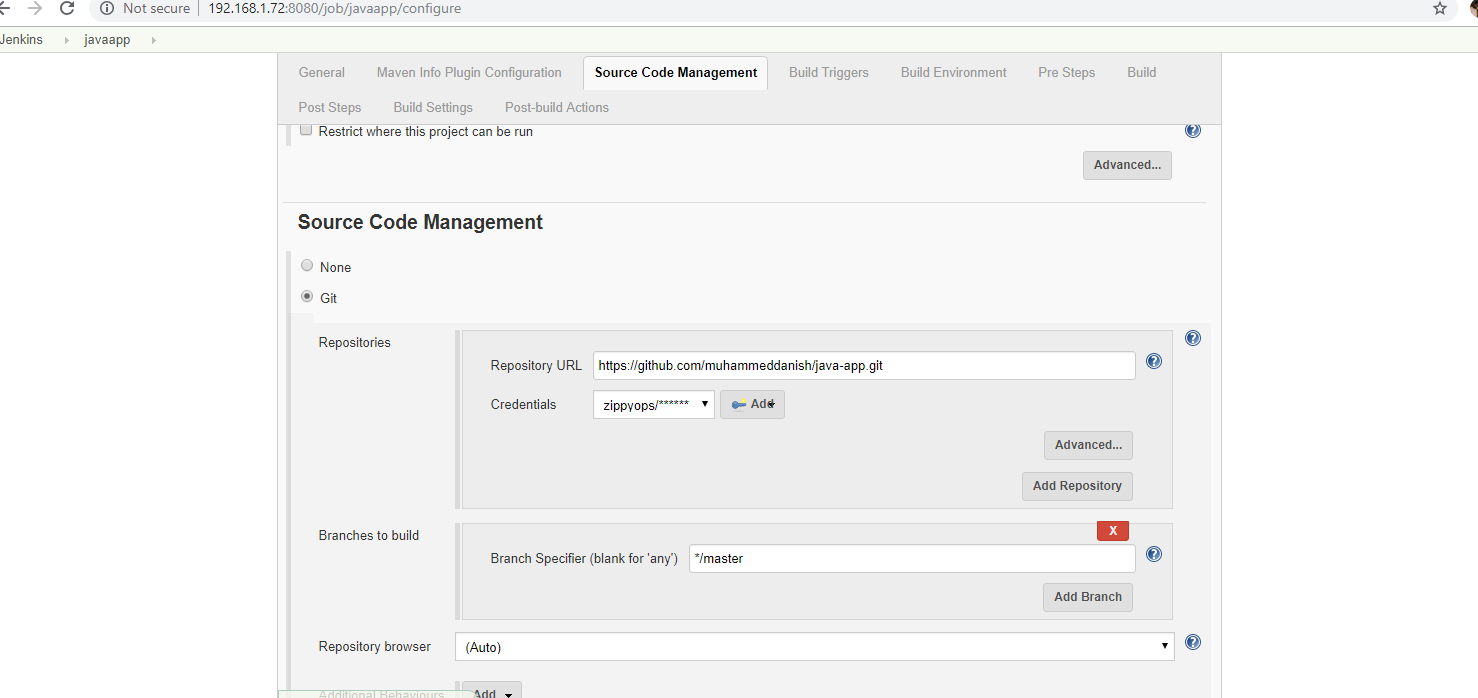
* Click the available plugins to install the required ones.



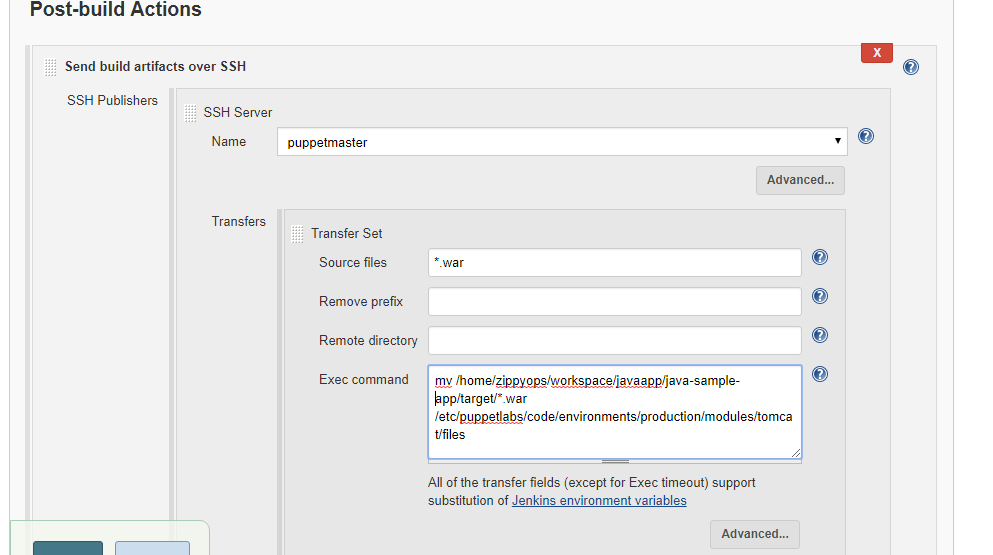
* Search the publish over ssh plugins that you want for the task.
* And select all the plugins you want.
* After selecting, click download now and install without restart.
* SSH plugins is used to connect the node by ssh connectivity, to run the execute command in the local machine.
* Then search for maven plugins and add the plugins.
* The maven is used to build the source file. That is to convert the source file to .war file.
* After finishing the installation, then click the configure system which is in the Manage Jenkins.
* Go to Configure management.
* Then click advance and insert username, host address and password of puppet master to connect puppet master to Jenkins via ssh.
* After completing the details, click the test Configuration to check the connectivity. If success arise its ok, if any Error occurs check details you have mentioned above are correct .

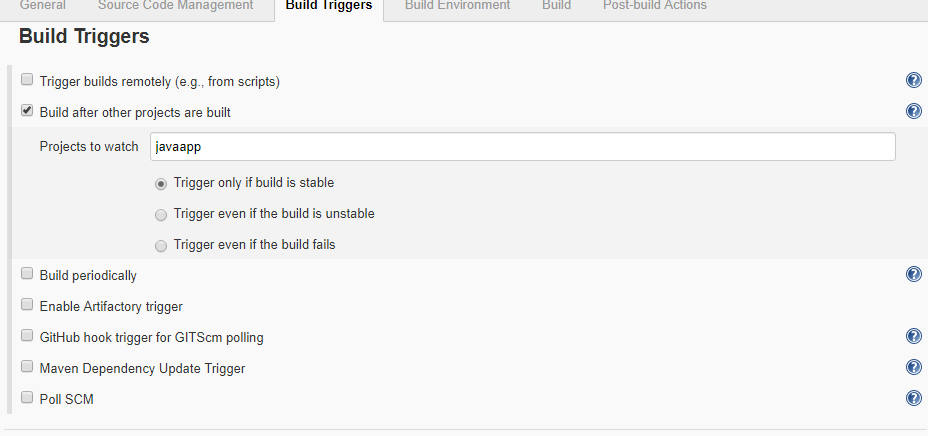
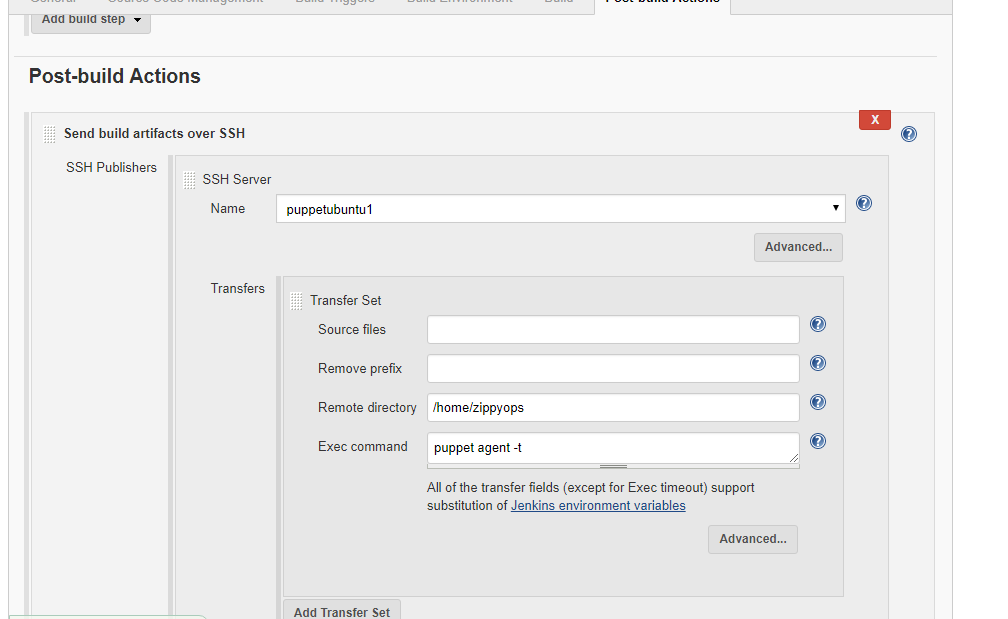


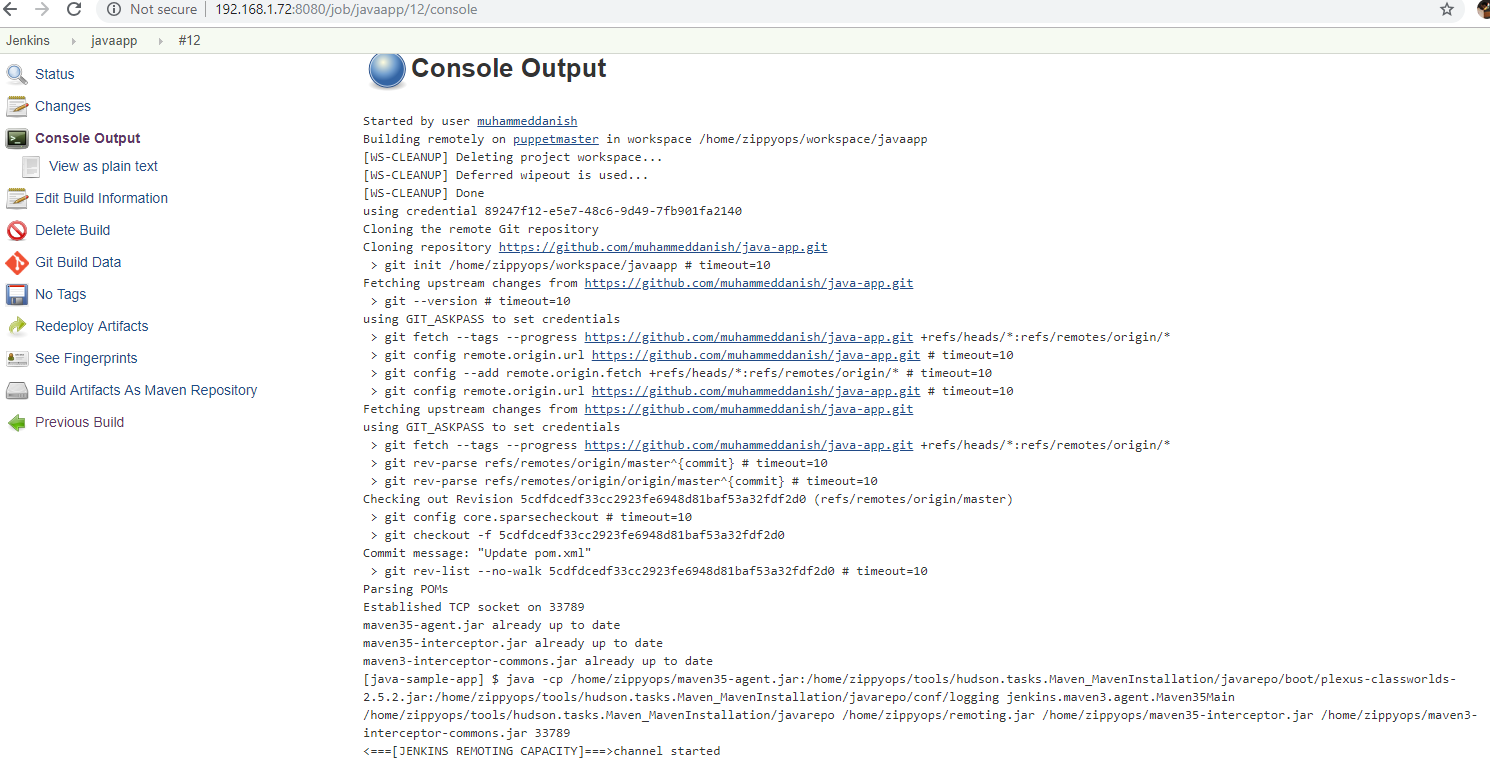
* To test the configuration click Test configuration.
* Click save to save ssh configuration.
* Now create the Jenkins maven project. I created the upstream project with the name javaapp.
* Go inside the project. Then click the configure to set the configuration for the project.
* Inside the configure give the GitHub URL and the Repository URL of your source code that you going to build and in the Credentials add the git username and password.

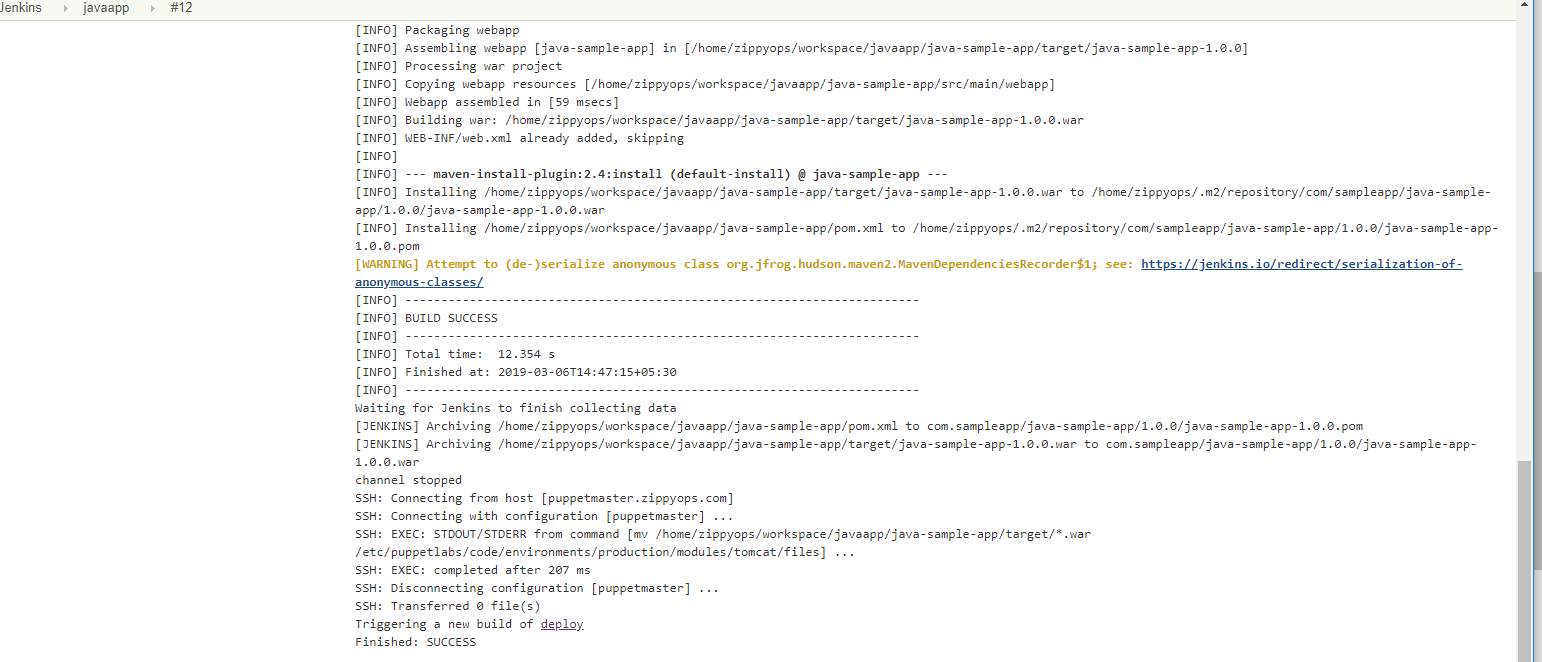


* Give the move command in post build section to move the .war file from the workspace to the tomcat module, so we can glow the file in tomcat. Then save the project.

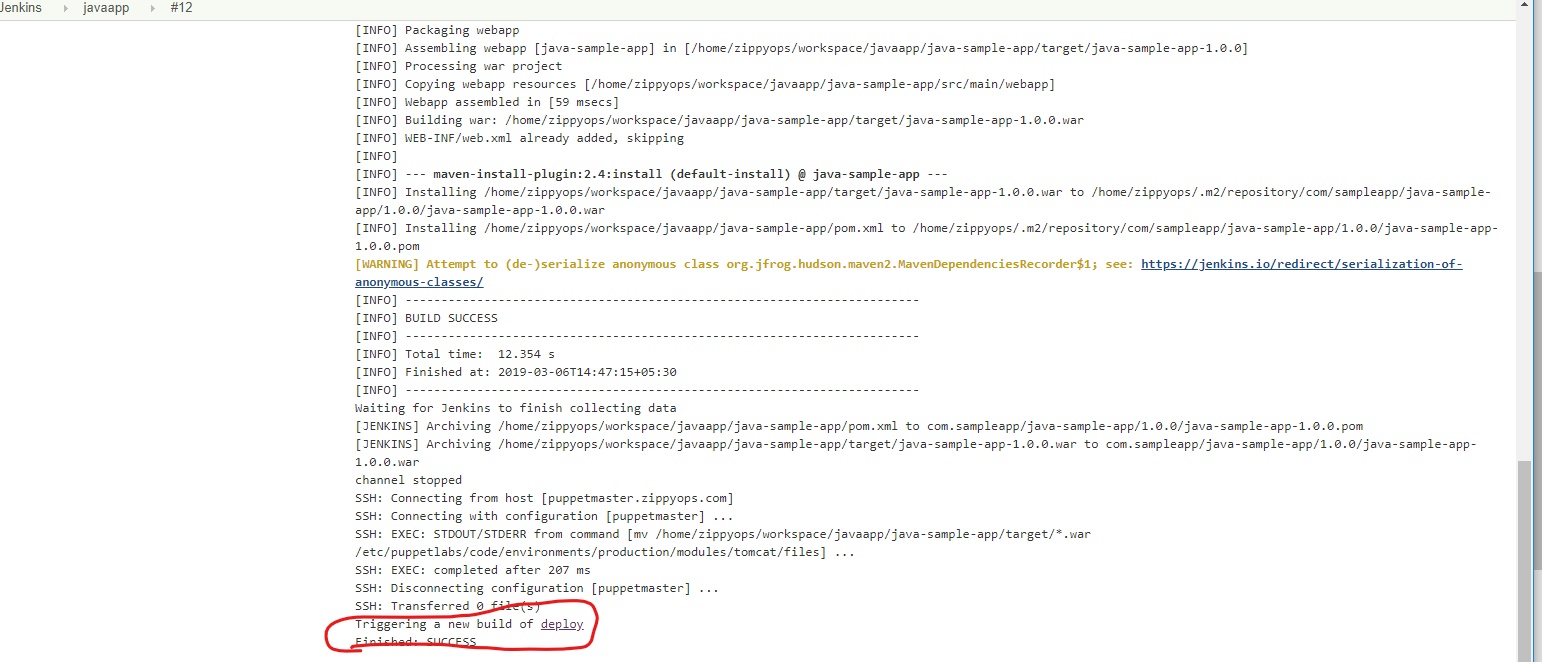


* Now create the downstream project to run the puppet agent –t command in node. In my case, had created a freestyle project with the name deploy.
* In deploy project, add the upstream project as the project to watch. That is, the downstream project will start to build once the upstream project completes its build.
* Now add puppet agent –t as the post build command to execute over ssh in node. 
* Save the downstream project and then again go to the upstream project. From upstream project build the project.
* The console output for the upstream project

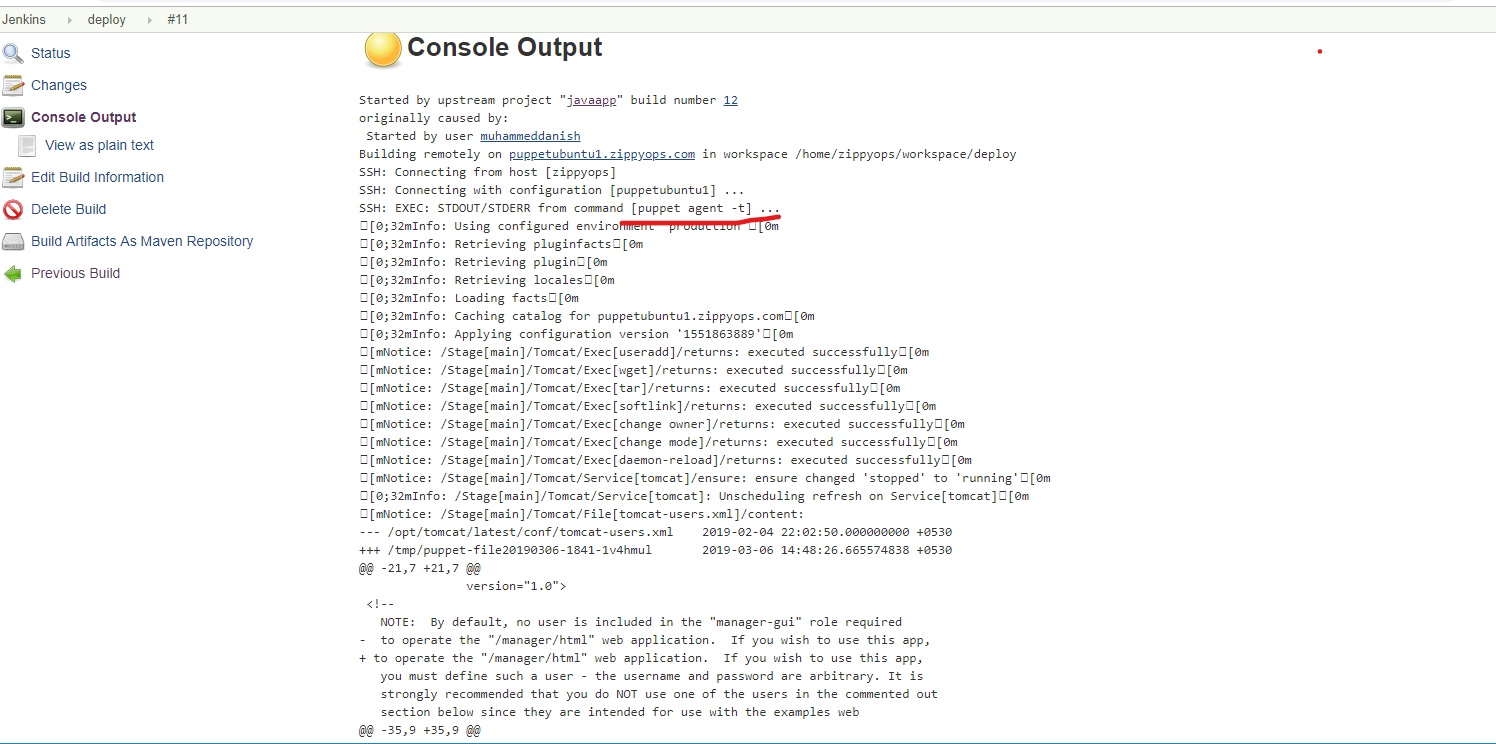


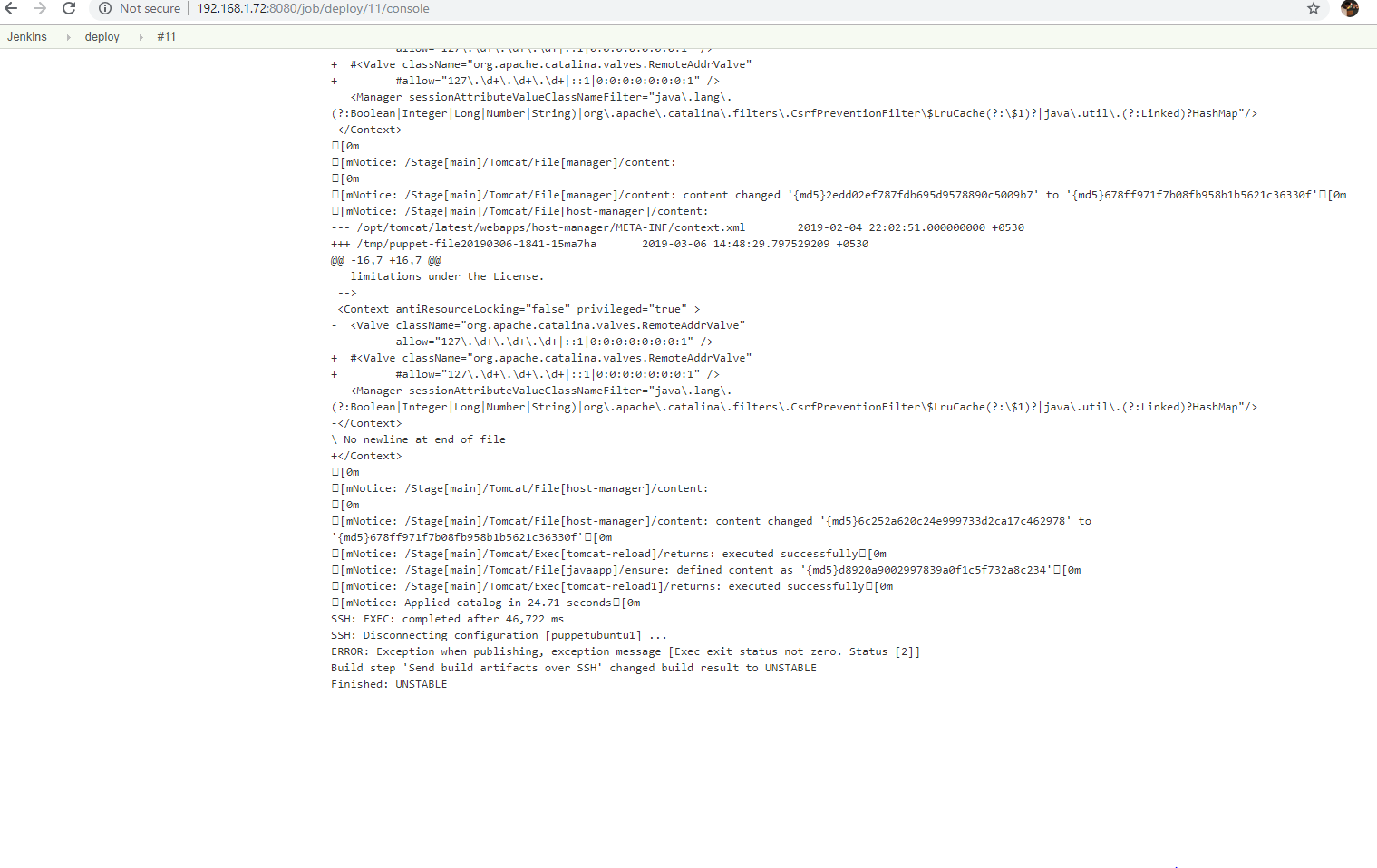


* The build for downstream project will be executed automatically once the build for upstream is success.



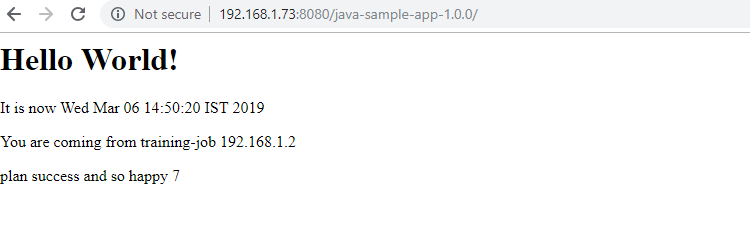
* The console output for the downstream project





# result

* Enter the ip of Ubuntu node with the .war file name in browser to view the output. In my case:

 <http://192.168.1.73:8080/java-sample-app-1.0.0>