CI/CD Deployment Using Ansible CM Tool

Project 1

Post Graduate Program in DevOps

PG DO - Configuration Management with Chef, Puppet and Ansible

Vignesh Dharmaraj
vikidvg@gmail.com

Table of Contents

1. Introduction to the Project *2*

2. Installation of pre-requisites/tools *2*

2.1 Installing Git **2**

2.2 Creating GitHub Account **3**

2.3 Setting up Jenkins **3**

2.4 Install Ansible on Ubuntu **8**

2.5 Jenkins config for Ansible **12**

2.6 SSH Connectivity setup to Managed Nodes **13**

2.7 Test SSH Connectivity to Managed Nodes **15**

3. Execution of the project *15*

4. Testing of Project *21*

5. Project Results *22*

6. Conclusion *27*

1. Introduction to the Project

Objective: CI/CD Deployment Using Ansible CM Tool

Solution build should demonstrate below capabilities:

1. Configure Jenkins server as Ansible provisioning machine
2. Install Ansible plugins in Jenkins CI server
3. Prepare Ansible playbook to run Maven build on Jenkins CI server
4. Prepare Ansible playbook to execute deployment steps on the remote web container with restart of the web container post deployment

Project goal is to Automate Ansible integration with Jenkins CI server so that we can run and execute playbooks to deploy custom WAR files to a web container and then perform restart for the web container.

Tools required: Ansible, GitHub, Git, Linux (Ubuntu), Jenkins

2. Installation of pre-requisites/tools

In this section we can see how the required tools are installed to execute the project

Note: This project is implemented by the "PG DO - Certified Kubernetes Administrator (CKA) Training" Lab (Kubernetes New VM) Lab provided by SimpliLearn, so most of the tools may be already installed including the Linux (Ubuntu). Lab has three nodes with 4 GB RAM, 20 GB Hard Drive any Linux OS (Ubuntu).

2.1 Installing Git

Step 1: Verifying the Git installation

- Use the following command to check the version of Git:

  git --version

```
vikidvggmail@ip-172-31-29-62:~$ git --version
git version 2.7.4
```

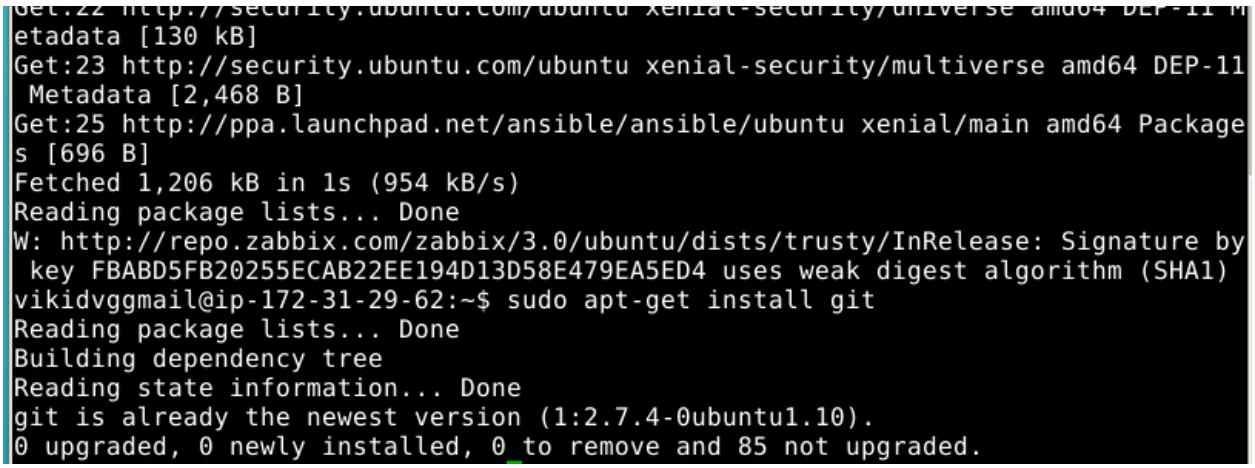
Note: Execute **Step 2** in case you don't get any results for **git --version** command.

Step 2: Installing the latest version of Git

- Execute the following commands on the terminal to install Git:

  sudo apt-get update

  sudo apt-get install git



2.2 Creating GitHub Account

Make sure you have a Github Account available. If not, please create one using the given link.
https://github.com/join?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home

2.3 Setting up Jenkins

Step 1: Downloading the Java Runtime Environment

1.1 Open the terminal.
1.2 Run ***sudo apt-get update*** to update the package lists.
1.3 Run ***sudo apt-get install openjdk-8-jdk*** to install the Java Runtime Environment.
1.4 Run ***java -version*** to verify the installation. It will print the JDK version as shown below:

```
vikidvggmail@ip-172-31-29-62:~$ java -version
openjdk version "1.8.0_282"
OpenJDK Runtime Environment (build 1.8.0_282-8u282-b08-0ubuntu1~16.04-b08)
OpenJDK 64-Bit Server VM (build 25.282-b08, mixed mode)
```

Step 2: Downloading and installing the Jenkins app

1.1 Open the terminal.

1.2 Run ***wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -*** to install Jenkins.

```
susmitaadhyapak@susmitaadhyapak:~$ wget -q -O - https://pkg.jenkins.io/debian/je
nkins.io.key | sudo apt-key add -
```

2.3 Run ***sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'*** command.

```
susmitaadhyapak@susmitaadhyapak:~$  sudo sh -c 'echo deb http://pkg.jenkins.io/d
ebian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

2.4 Run ***sudo apt-get update***

2.5 Run ***sudo apt-get install jenkins*** to install Jenkins.

```
susmitaadhyapak@susmitaadhyapak:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-lato javascript-common jruby libbytelist-java libhawtjni-runtime-java
  libheadius-options-java libinvokebinder-java libjansi-java
  libjansi-native-java libjcodings-java libjffi-java libjffi-jni
  libjnr-constants-java libjnr-enxio-java libjnr-ffi-java libjnr-netdb-java
  libjnr-posix-java libjnr-unixsocket-java libjnr-x86asm-java
  libjoda-time-java libjruby-joni-java libjs-jquery libjzlib-java liblua5.2-0
  libreadline7 libssl1.0.2 libtcl8.6 libunsafe-mock-java libyaml-snake-java
  libyecht-java nailgun rake vim-gui-common
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  daemon
The following NEW packages will be installed:
  daemon jenkins
0 upgraded, 2 newly installed, 0 to remove and 306 not upgraded.
Need to get 70.6 MB of archives.
After this operation, 71.2 MB of additional disk space will be used.
```

2.6 Run ***sudo service jenkins status*** to check the status of the installation. Once you verify the status as active, you can press ***Ctrl+z*** to exit from the process.

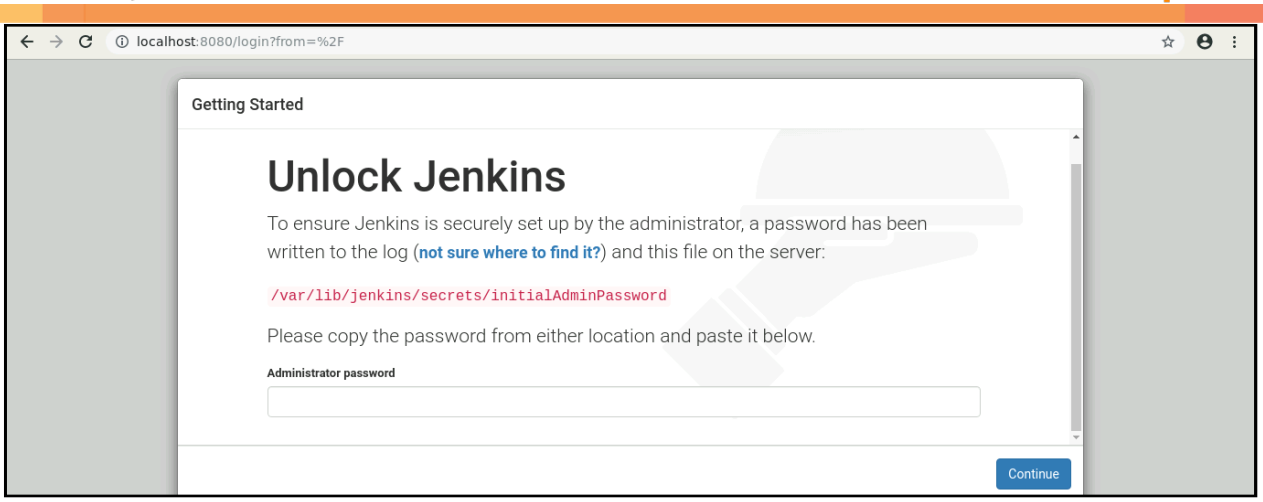
```
vikidvggmail@ip-172-31-29-62:~$ sudo service jenkins status
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Thu 2021-07-22 09:59:50 UTC; 30min ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1563 ExecStart=/etc/init.d/jenkins start (code=exited, status=0/SUCCE
    Tasks: 0
   Memory: 0B
      CPU: 0

Jul 22 09:59:48 ip-172-31-29-62 systemd[1]: Starting LSB: Start Jenkins at boot
Jul 22 09:59:49 ip-172-31-29-62 jenkins[1563]: Correct java version found
Jul 22 09:59:49 ip-172-31-29-62 jenkins[1563]:  * Starting Jenkins Automation Se
Jul 22 09:59:49 ip-172-31-29-62 su[1745]: Successful su for jenkins by root
Jul 22 09:59:49 ip-172-31-29-62 su[1745]: + ??? root:jenkins
Jul 22 09:59:49 ip-172-31-29-62 su[1745]: pam_unix(su:session): session opened f
Jul 22 09:59:50 ip-172-31-29-62 jenkins[1563]:    ...done.
Jul 22 09:59:50 ip-172-31-29-62 systemd[1]: Started LSB: Start Jenkins at boot t
lines 1-17/17 (END)
```

2.7 Run the following commands to start Jenkins.

sudo systemctl start jenkins
sudo systemctl status jenkins

```
susmitaadhyapak@susmitaadhyapak:~$ sudo systemctl start jenkins
susmitaadhyapak@susmitaadhyapak:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
   Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)
   Active: active (exited) since Tue 2021-03-23 08:02:03 UTC; 4min 2s ago
     Docs: man:systemd-sysv-generator(8)

Mar 23 08:02:01 susmitaadhyapak systemd[1]: Starting LSB: Start Jenkins at boot
Mar 23 08:02:01 susmitaadhyapak jenkins[5672]: Correct java version found
Mar 23 08:02:01 susmitaadhyapak jenkins[5672]:  * Starting Jenkins Automation Se
Mar 23 08:02:01 susmitaadhyapak su[5737]: Successful su for jenkins by root
Mar 23 08:02:01 susmitaadhyapak su[5737]: + ??? root:jenkins
Mar 23 08:02:01 susmitaadhyapak su[5737]: pam_unix(su:session): session opened f
Mar 23 08:02:03 susmitaadhyapak jenkins[5672]:    ...done.
Mar 23 08:02:03 susmitaadhyapak systemd[1]: Started LSB: Start Jenkins at boot t
Mar 23 08:05:53 susmitaadhyapak systemd[1]: Started LSB: Start Jenkins at boot t
lines 1-14/14 (END)
```

2.8 Open **localhost:8080** in the browser, and you will need to enter the initial password.



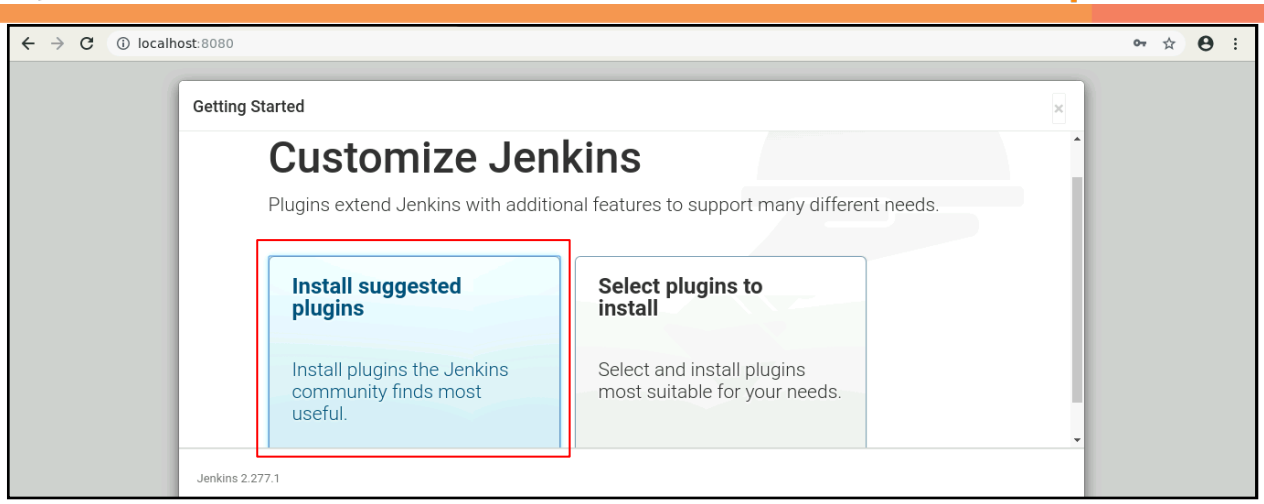
2.9 In your terminal run the following command:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

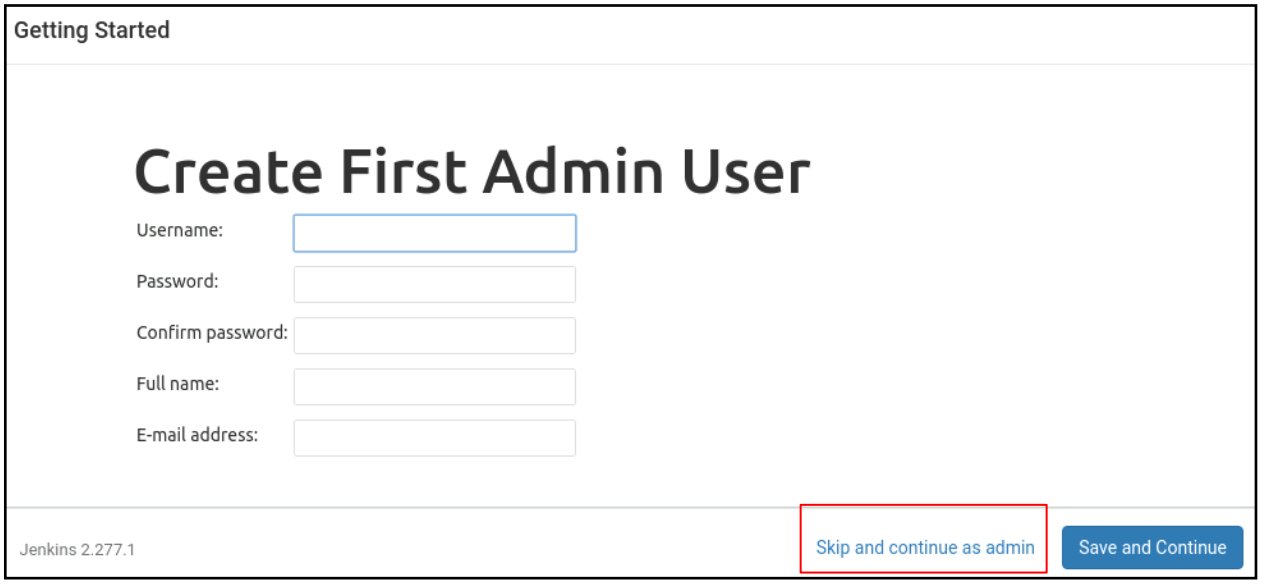
```
susmitaadhyapak@susmitaadhyapak:~$ sudo cat /var/lib/jenkins/secrets/initialAdmi
nPassword
876821d4689a453c87c48116a59a001b
susmitaadhyapak@susmitaadhyapak:~$
```

2.10 Copy this password and paste it on your Jenkins page in the browser.

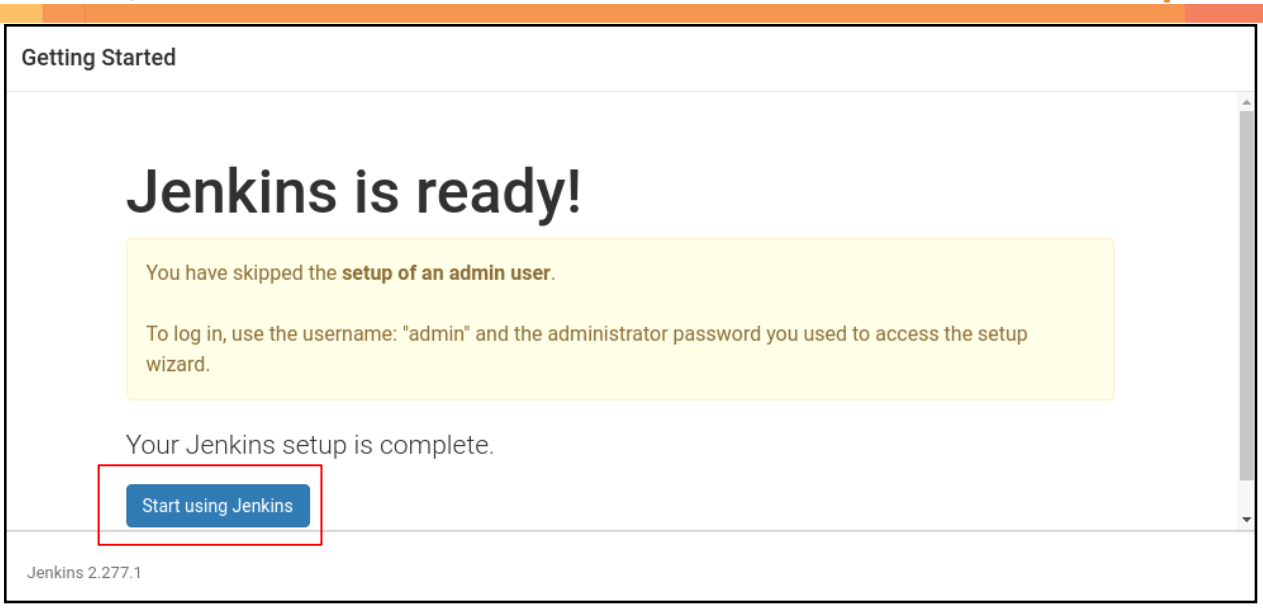
2.11 Now, click on I**nstall the suggested plugins**.



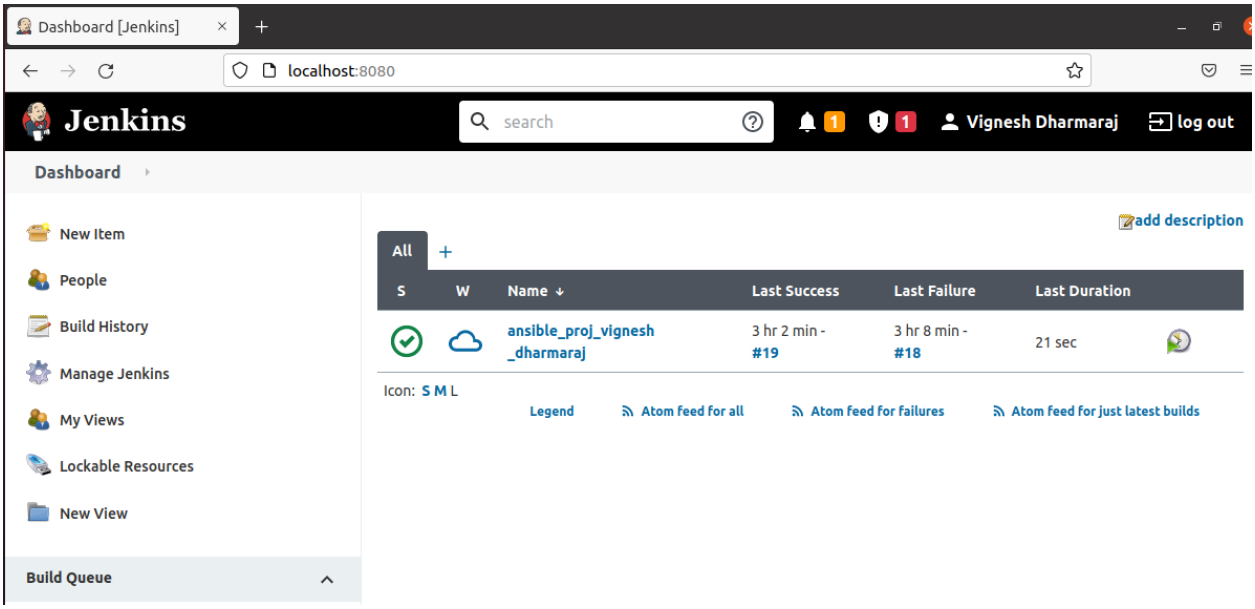
2.12 You can either create an admin user or skip and continue as admin. Select **Skip and continue as admin**.



2.13 In the Instance configuration page, click on the **Start using Jenkins** button.



2.14 Now, you can work with Jenkins as shown in the screenshot below.



2.4 Install Ansible on Ubuntu

- Use the below commands on Ubuntu system to install ansible software:

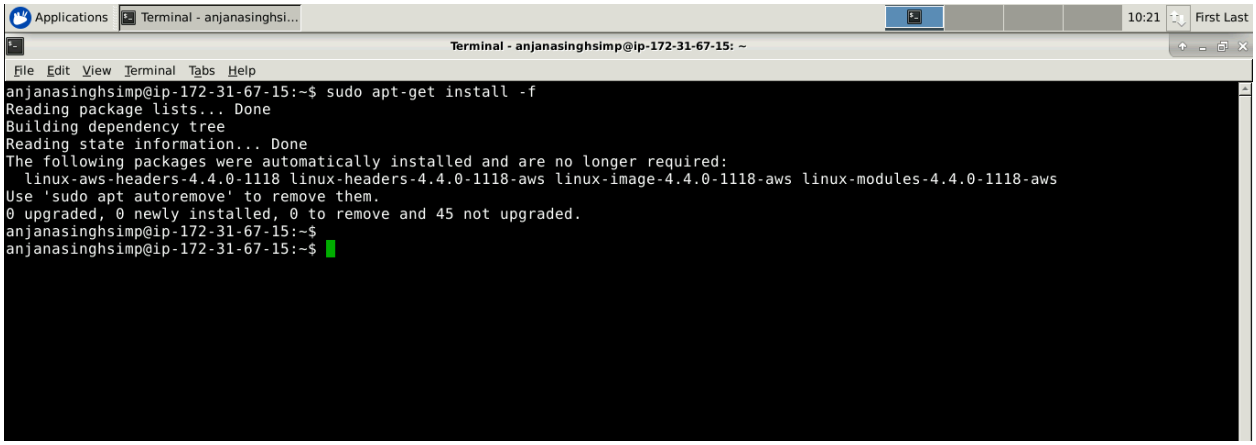
sudo apt-get install -f

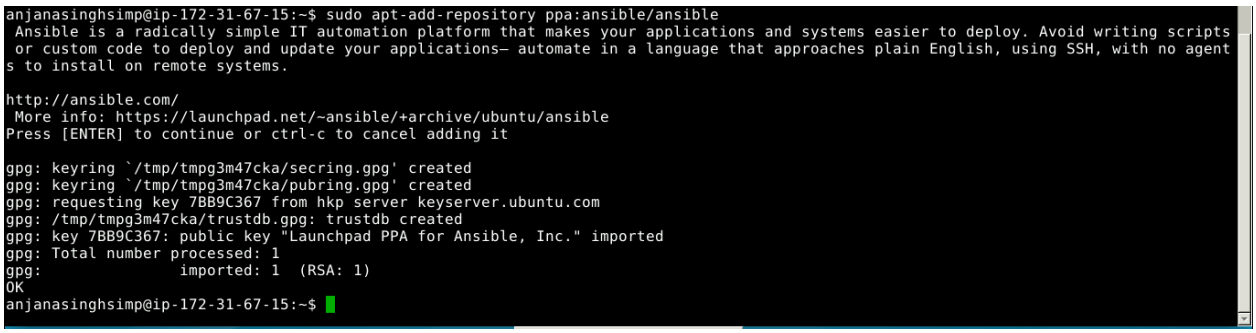
sudo apt-get install software-properties-common

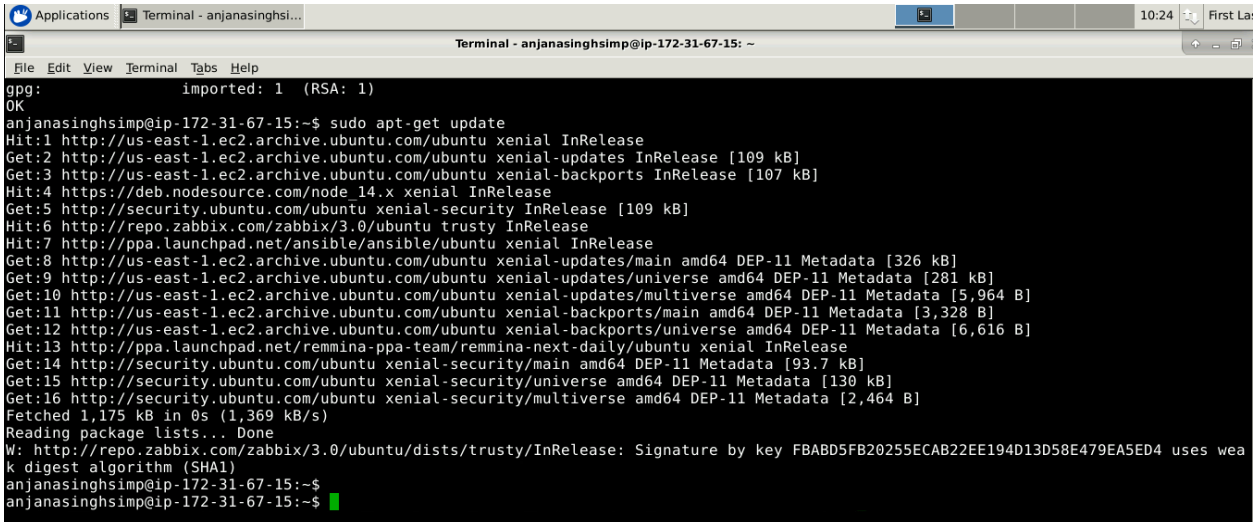
sudo apt-add-repository ppa:ansible/ansible

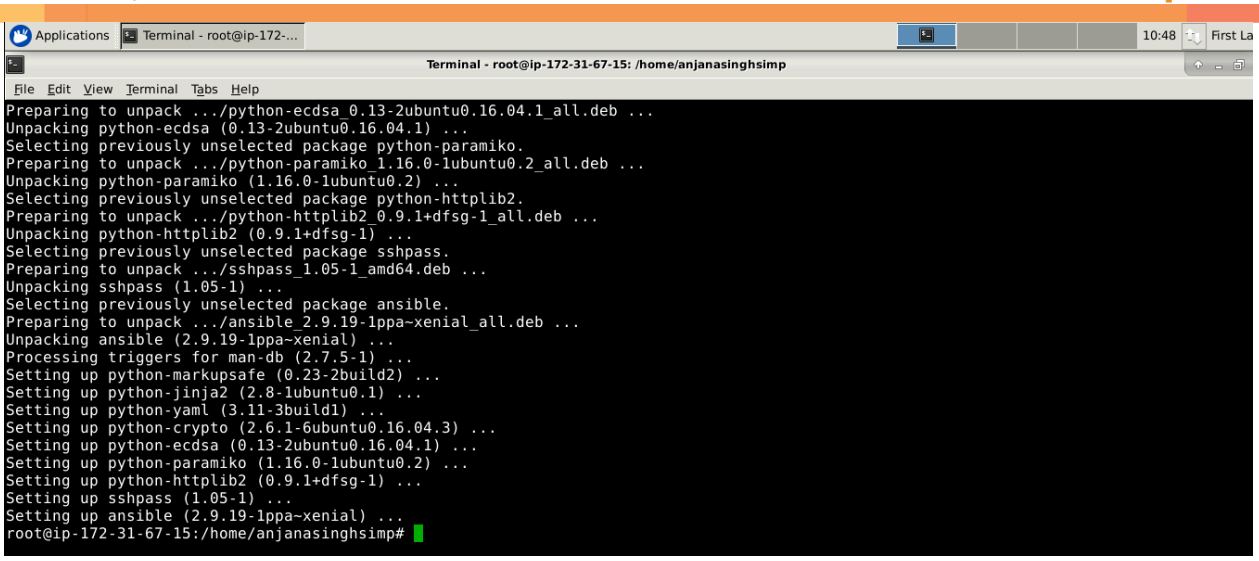
sudo apt-get update

sudo apt-get install ansible







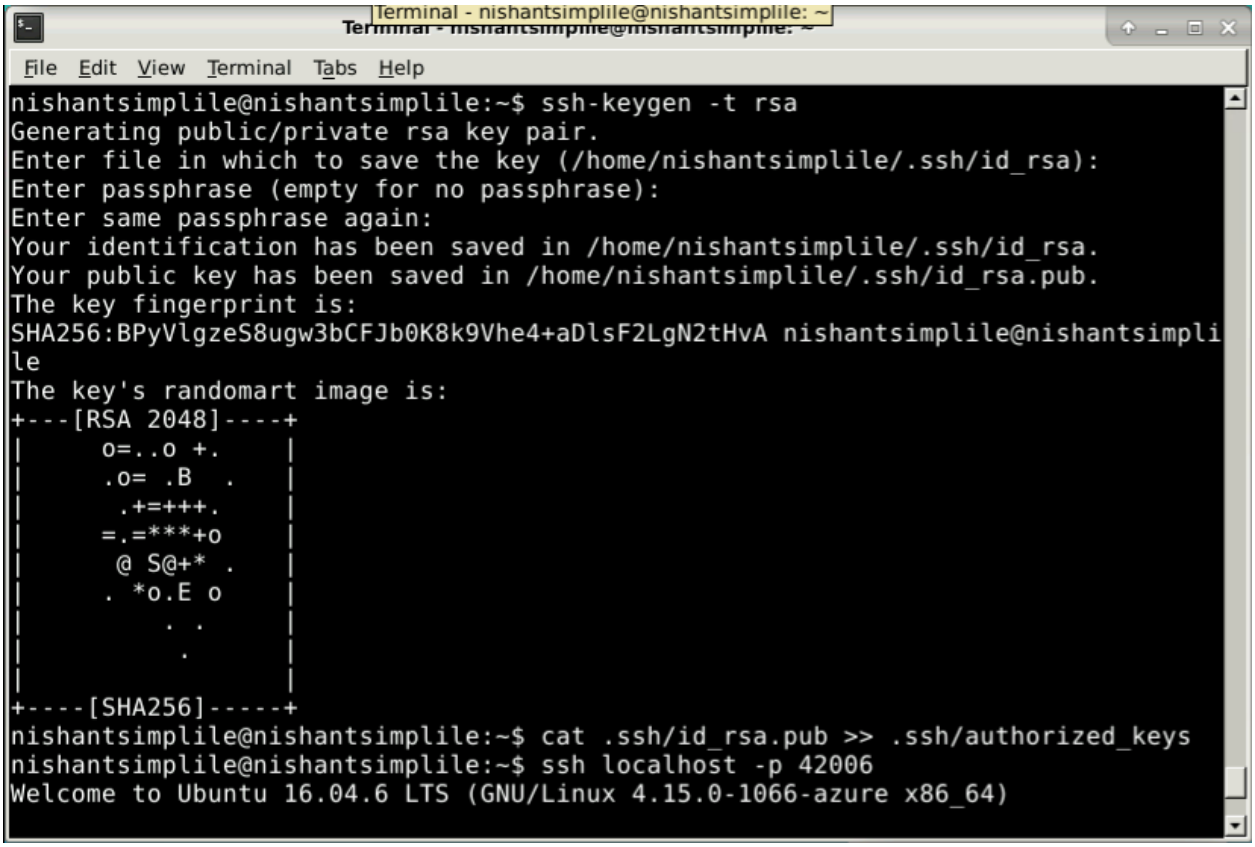


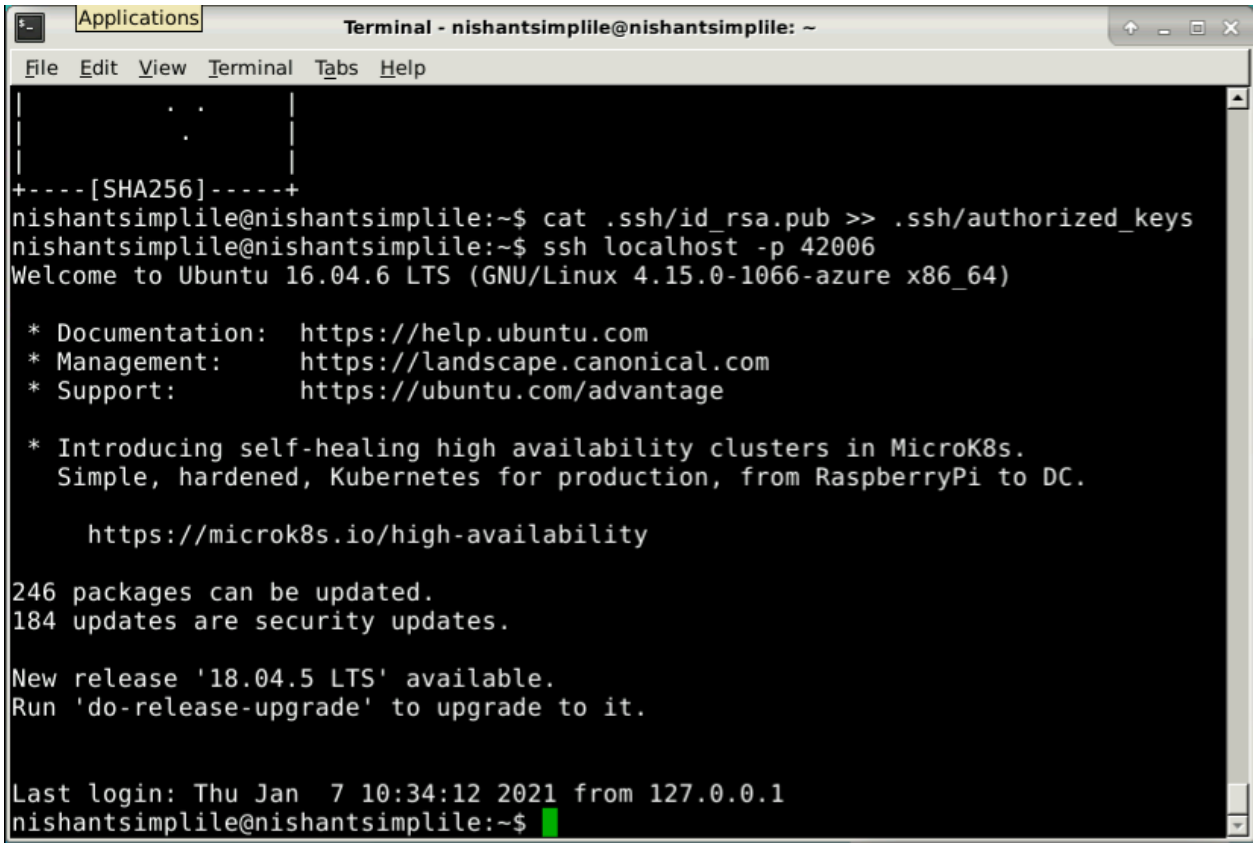
- Establish SSH key pair in linux system to have SSH connectivity with localhost using the following commands:

ssh-keygen -t rsa

cat .ssh/id_rsa.pub >> .ssh/authorized_keys

ssh localhost -p 42006





- Now, add the host localhost in the ansible host file /etc/ansible/hosts.

sudo vi /etc/ansible/hosts

- When the file opens, add the below two lines of the code at the end of the file:

[webservers]

localhost:42006

Step 2: Establish connectivity between Ansible controller and node machine

- Execute the below command to validate host inventory file:

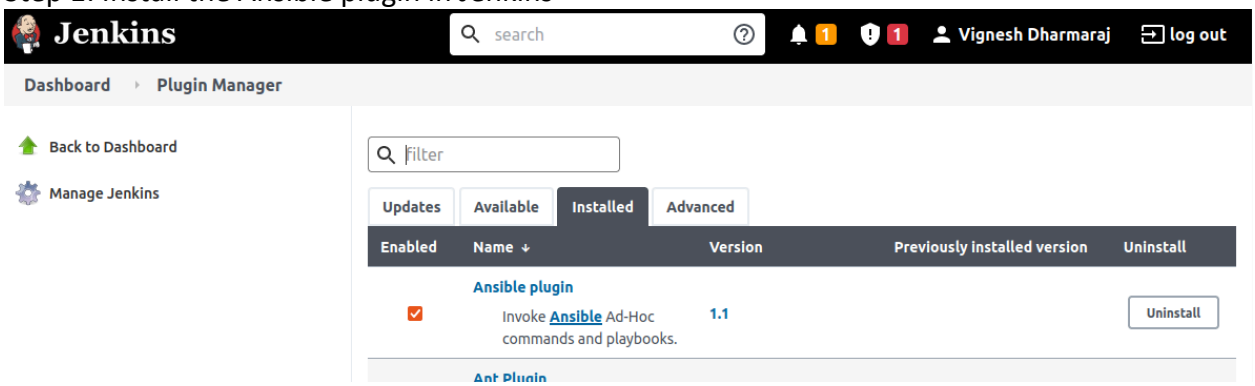
ansible -m ping webservers

```
root@docker:~# ansible -m ping webservers
localhost | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

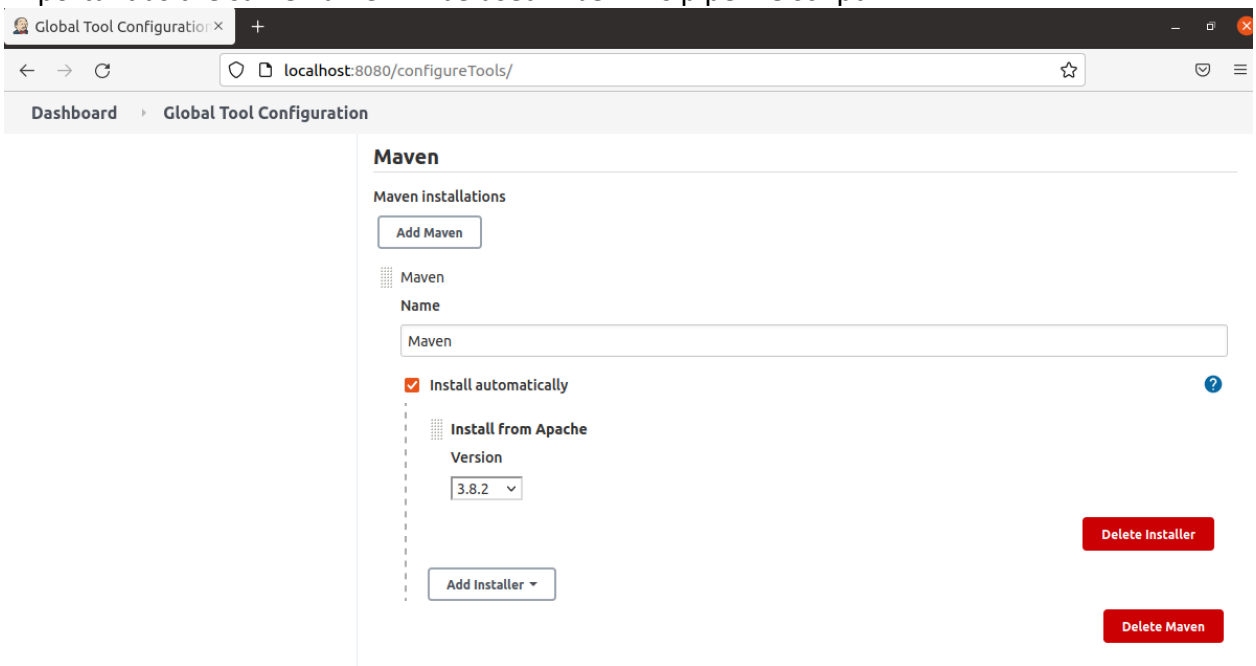
2.5 Jenkins config for Ansible

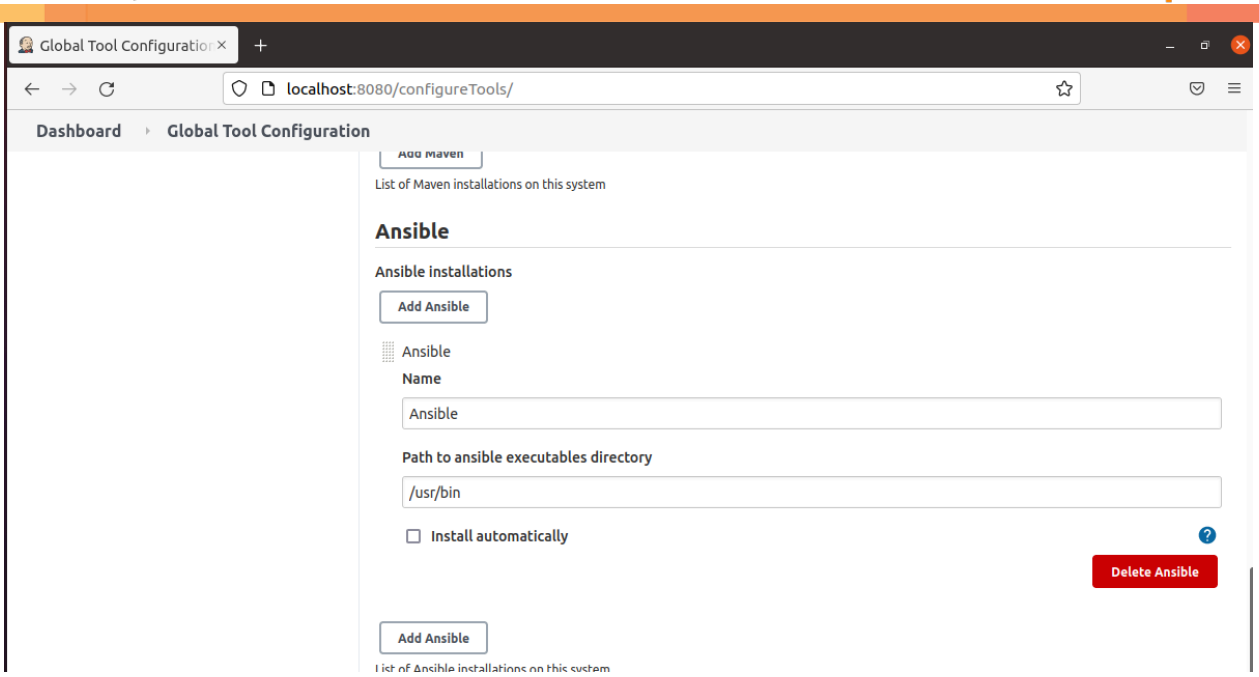
Jenkins and Ansible is already installed in the Ansible controller (master node), further Jenkins configuration has to be done as below:

Step 1: Install the Ansible plugin in Jenkins



Step 2: Global tools config to be done for Ansible and Maven as below, the 'Name' configured is important as the same name will be used in Jenkins pipeline script.

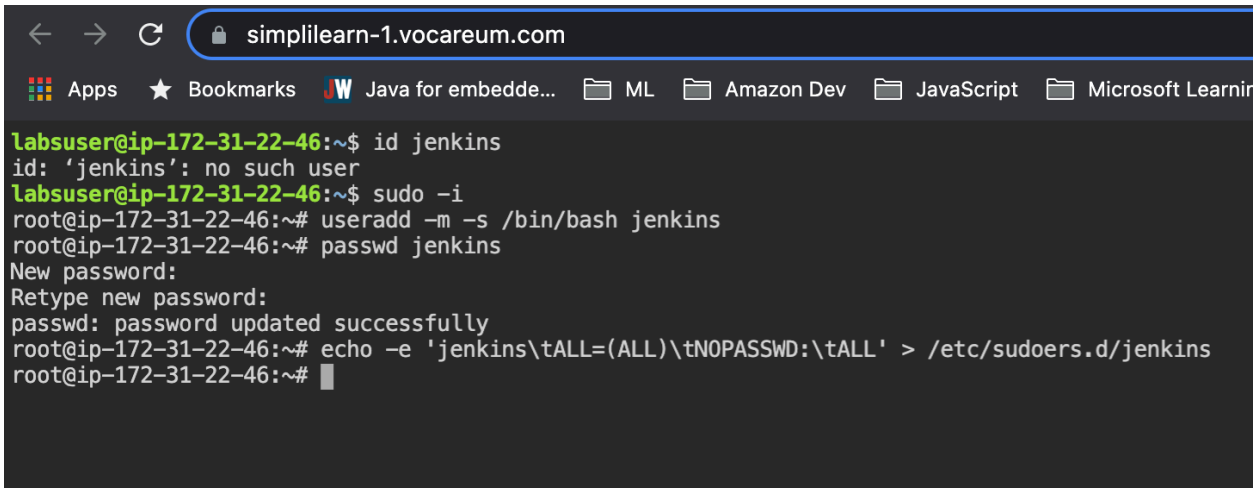




2.6 SSH Connectivity setup to Managed Nodes

SSH connectivity to be setup between the Ansible provisioning node and the managed nodes. As the Ansible provisioning is done through Jenkins, a user called 'Jenkins' need to be created in all the managed nodes and SSH connectivity has to be done with that user.

Step 1: Create user 'Jenkins' in the master node (already created as part of Jenkins installation) & managed nodes and do the necessary config for the user to execute all the tasks with elevated user privilege



Step 2: Ensure SSHD config file (/etc/ssh/sshd_config) has below parameters enabled
PasswordAuthentication yes
PermitRootLogin yes

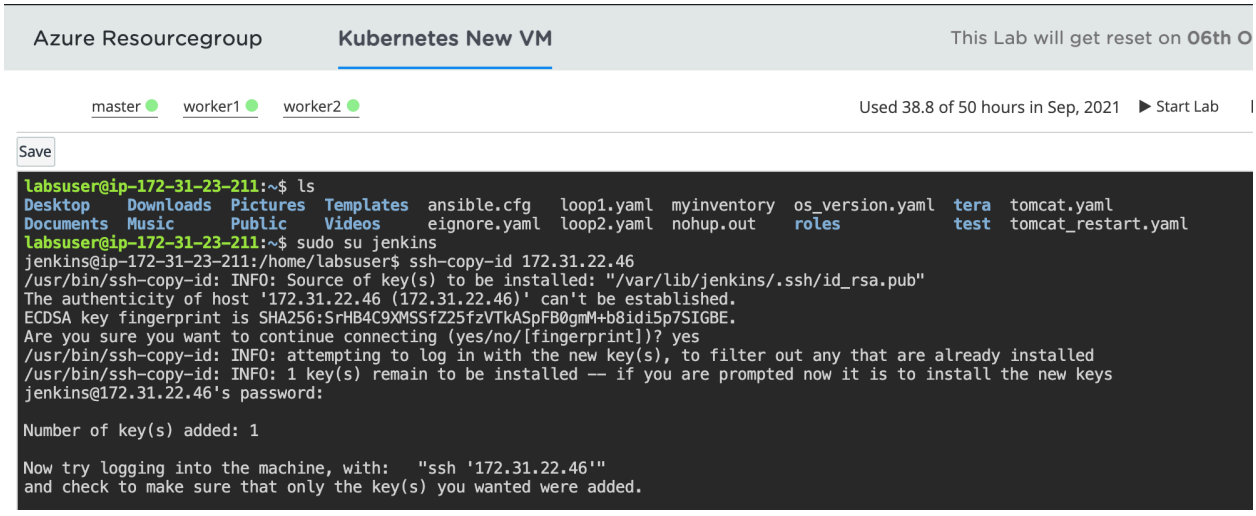
After modifying the SSHD config file restart SSHD service with below command:
sudo systemctl restart sshd

note the ip of the managed node by using the below command:
ip a

```
root@ip-172-31-22-46:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 02:b0:cc:75:c3:93 brd ff:ff:ff:ff:ff:ff
    inet 172.31.22.46/20 brd 172.31.31.255 scope global dynamic ens5
       valid_lft 2238sec preferred_lft 2238sec
    inet6 fe80::b0:ccff:fe75:c393/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:a4:c5:ac:1c brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
root@ip-172-31-22-46:~#
```

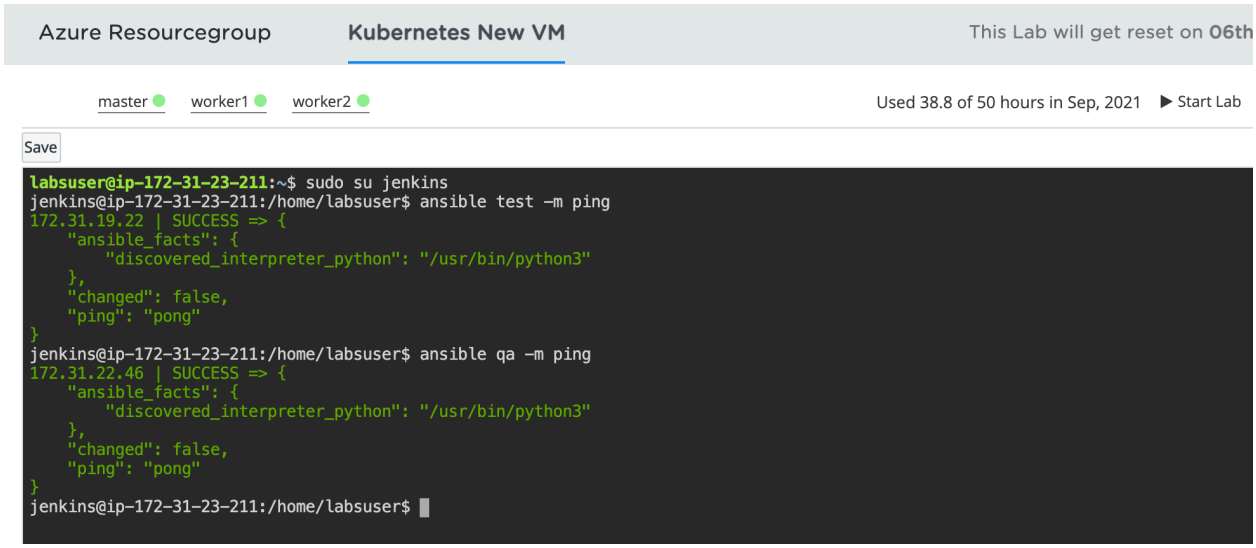
Ens5: inet has the ip address

Step 3: From the master node ssh-copy-id to the managed node ip to establish the ssh connectivity



2.7 Test SSH Connectivity to Managed Nodes

After all the above tools installation and configuration steps are followed, Now the SSh connectivity is established between the ansible controller machine and the managed nodes with Jenkins user, tested as shown below:



Connectivity has been setup with 2 managed nodes and the ansible provisioning required by the project will be deployed to both the nodes below:

Ansible managed nodes / machines
172.31.19.22
172.31.22.46

3. Execution of the project

All the necessary code is created in the below GitHub repository, this repo can be cloned to execute the Jenkins pipeline
https://github.com/vdharmaraj/PGDO_CM_Project_1

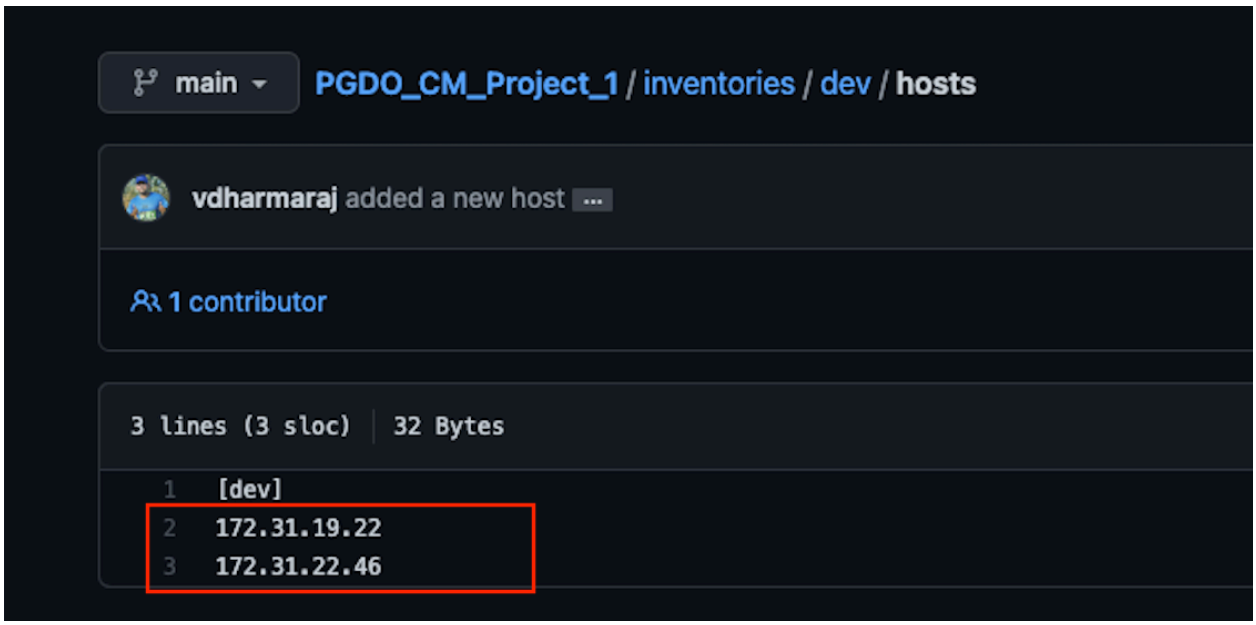
Below steps can be followed by anyone to execute the project:

Step 1:

with $git clone https://github.com/vdharmaraj/PGDO_CM_Project_1.git command you can clone the project into your machine and the changes to be made according to your setup is explained in further steps. You can also fork/clone the project from github, as it is a public repository.

Step 2:

In the inventory file of the playbook change your managed host according to your need, after testing the connectivity to your host



Step 4:

In the Jenkinsfile, checkout stage related git clone url and branch has to be changed according to your github links.

This stage in Jenkinsfile is just used for displaying the Git repository fetch.

main ▾ PGDO_CM_Project_1 / Jenkinsfile

vdharmaraj jenkins file initial commit

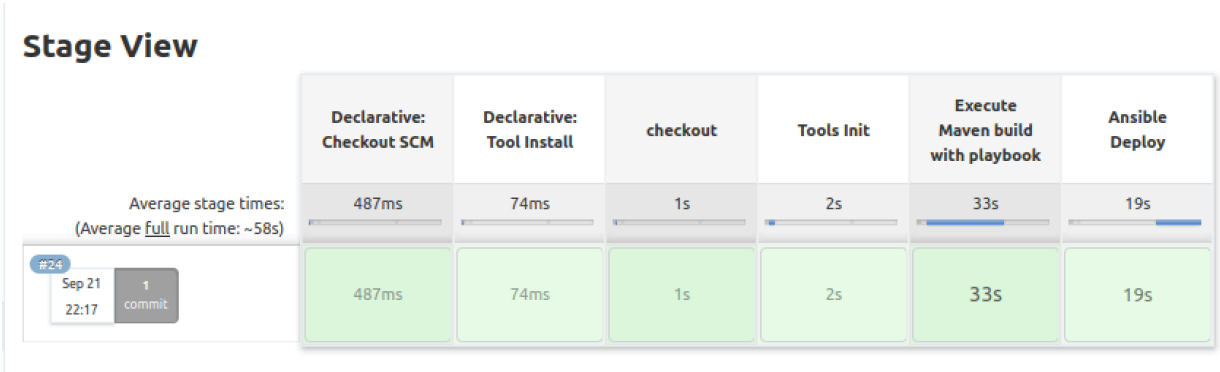
1 contributor

56 lines (35 sloc) | 1.08 KB

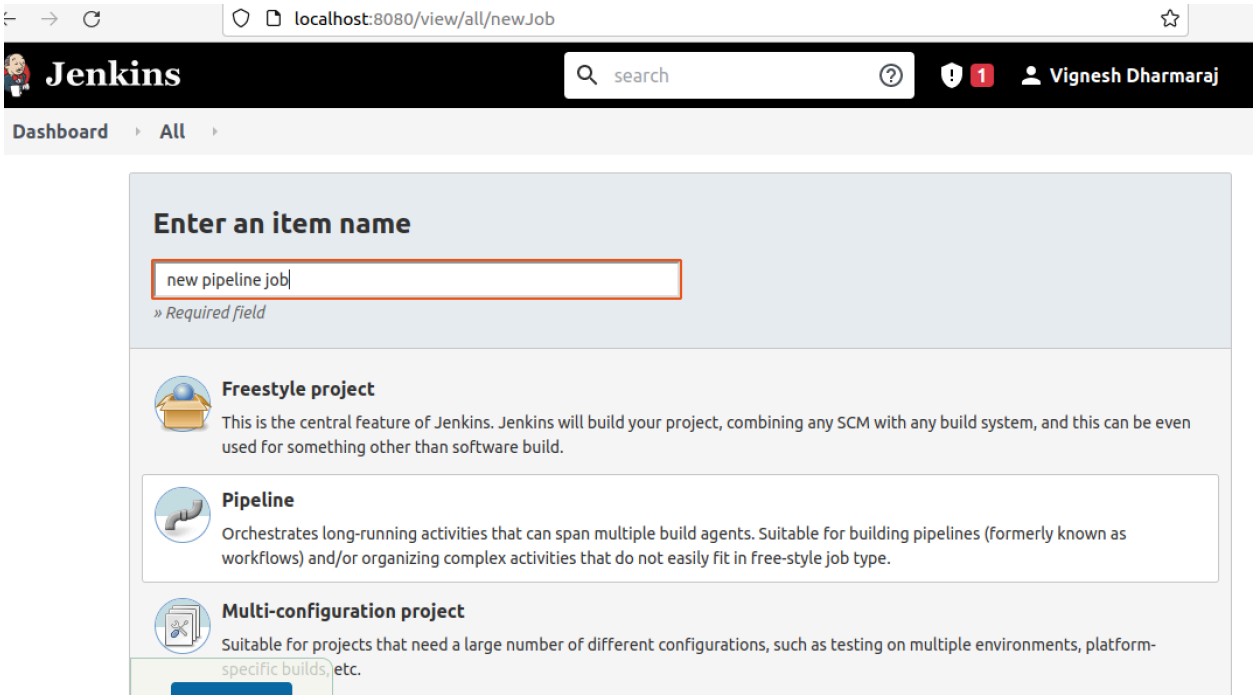
```
pipeline {
    agent any

    tools
    {
       maven "Maven"
    }

    stages {
      stage('checkout') {
           steps {

                git branch: 'main', url: 'https://github.com/vdharmaraj/PGDO_CM_Project_1.git'

          }
        }
```

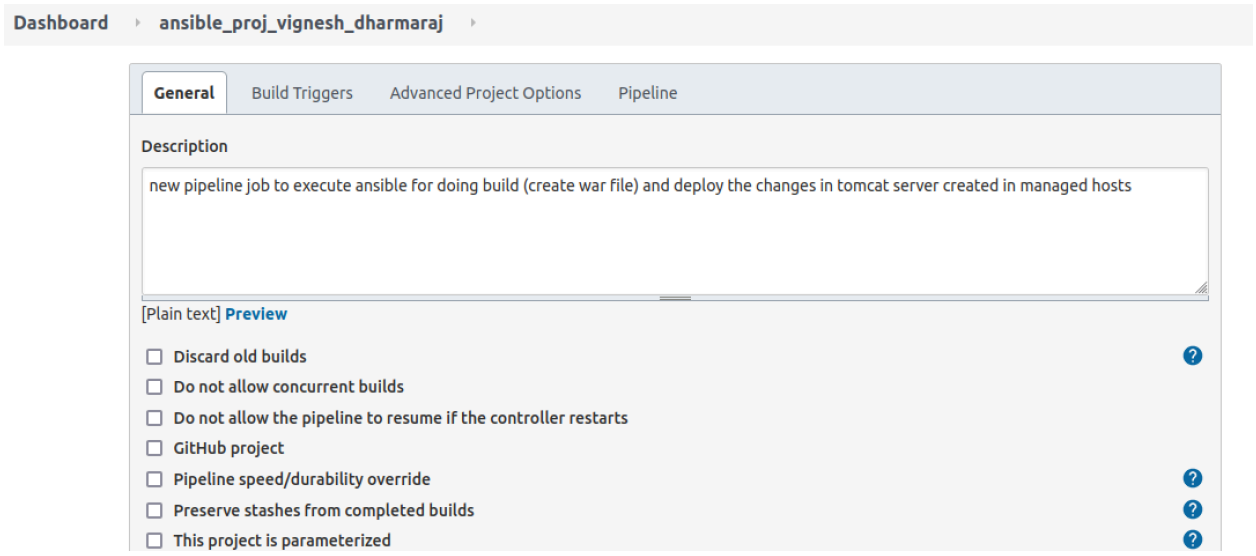
Step 5:

Create new pipeline job in Jenkins to poll the SCM (Git) and execute the Jenkins pipeline file with below stages:

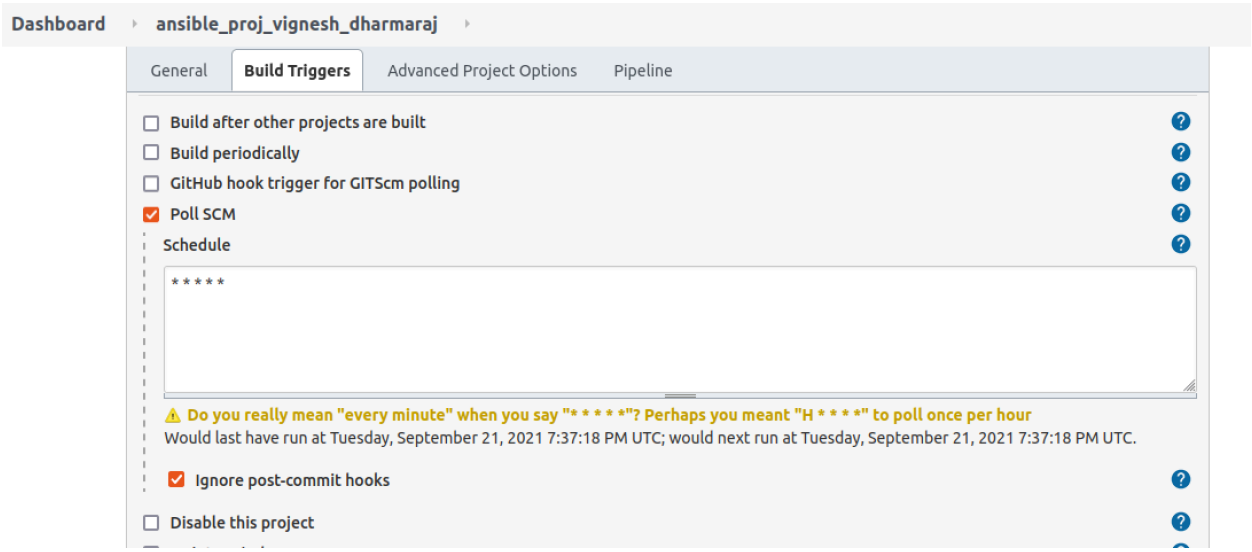


Jenkins pipeline job is created with below configuration:

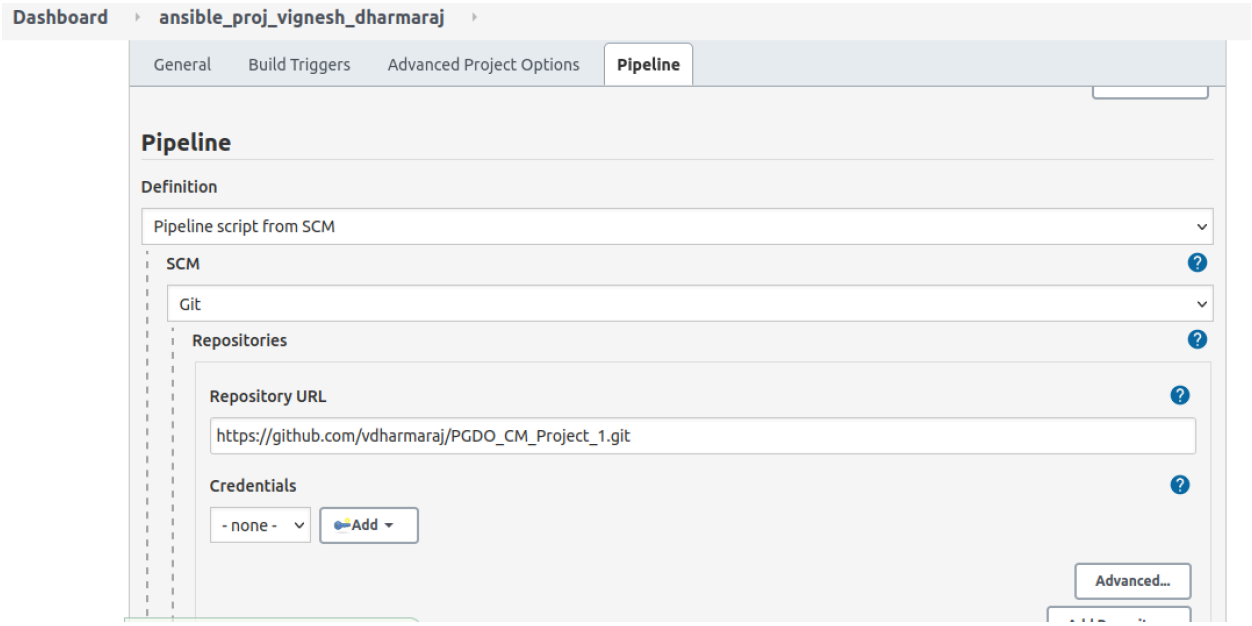




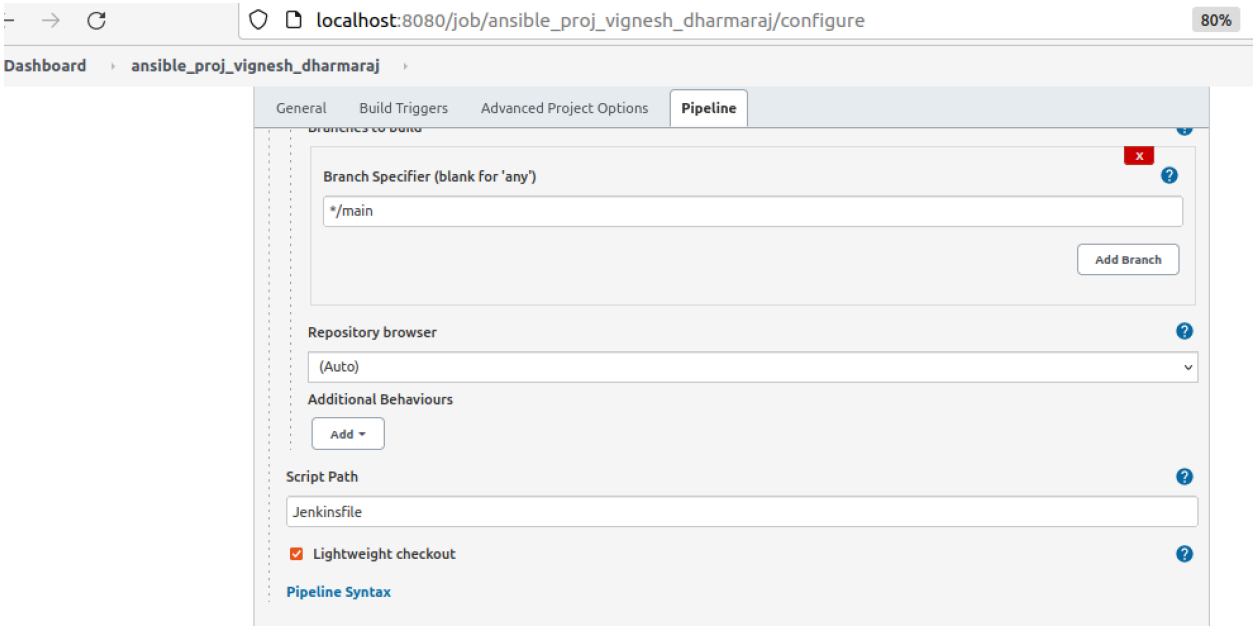
To poll the SCM for new commits every minute



Repository URL is configured in the below step which will be used to poll the SCM for any commits in the code repository.



Jenkinsfile reference from the repository is given in the 'Script Path' as below:

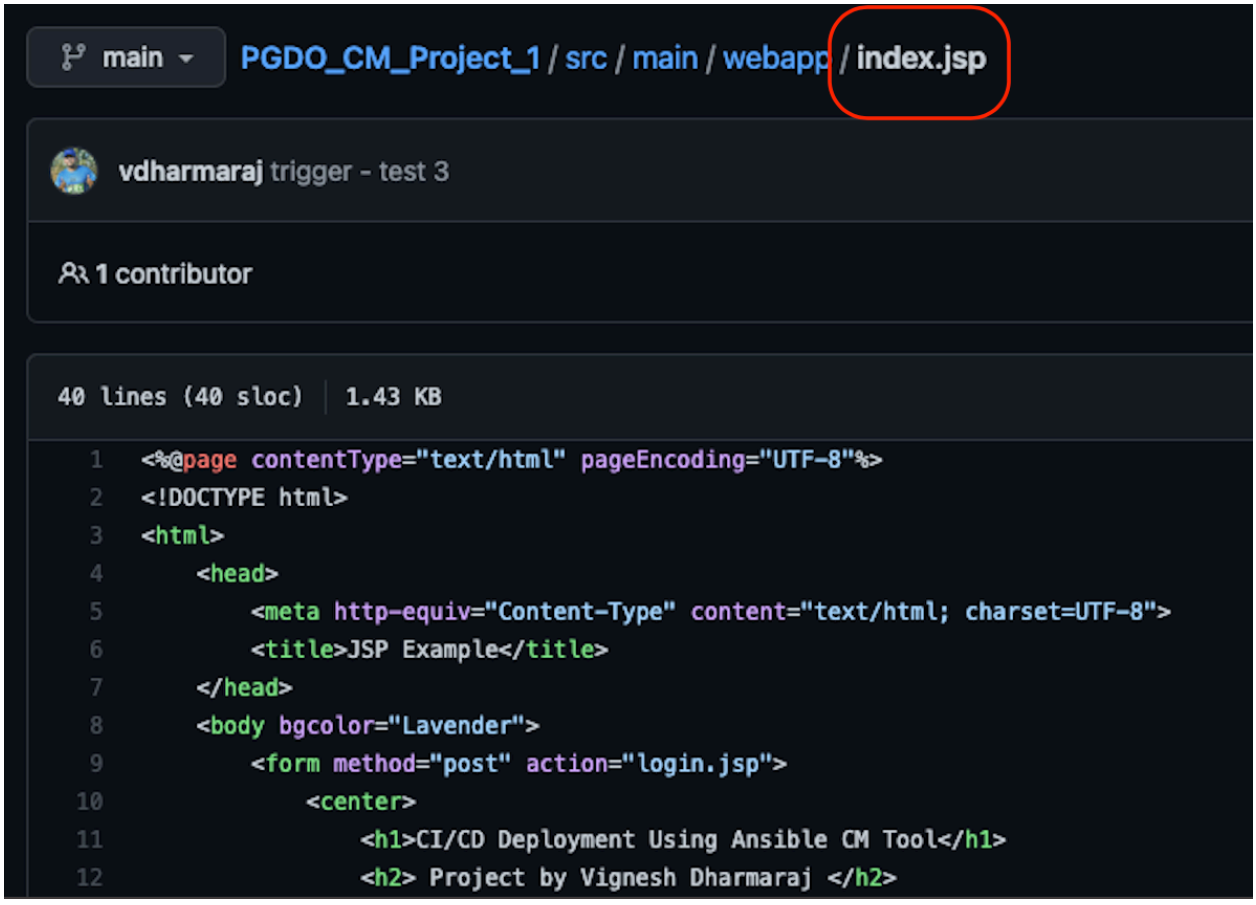


Save and apply will save the Jenkins pipeline job and will be scheduled to poll the SCM (git) every minute for changes committed to the project repository.

This step completes all the project configurations required.

4. Testing of Project

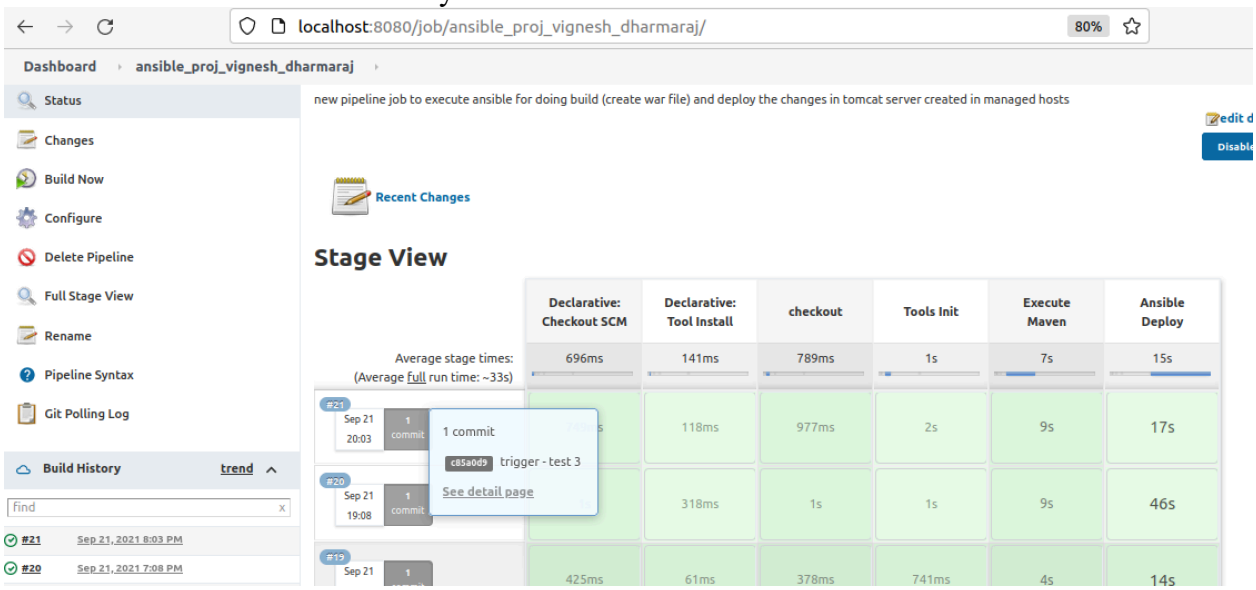
In our Repository the index.jsp file, which is part of java web App source code, this servlet file can be modified and the changes can be committed to the repository which will trigger the Jenkins Poll SCM job and the CI/CD process will be triggered to deploy the new application changes in the managed hosts with the help of ansible

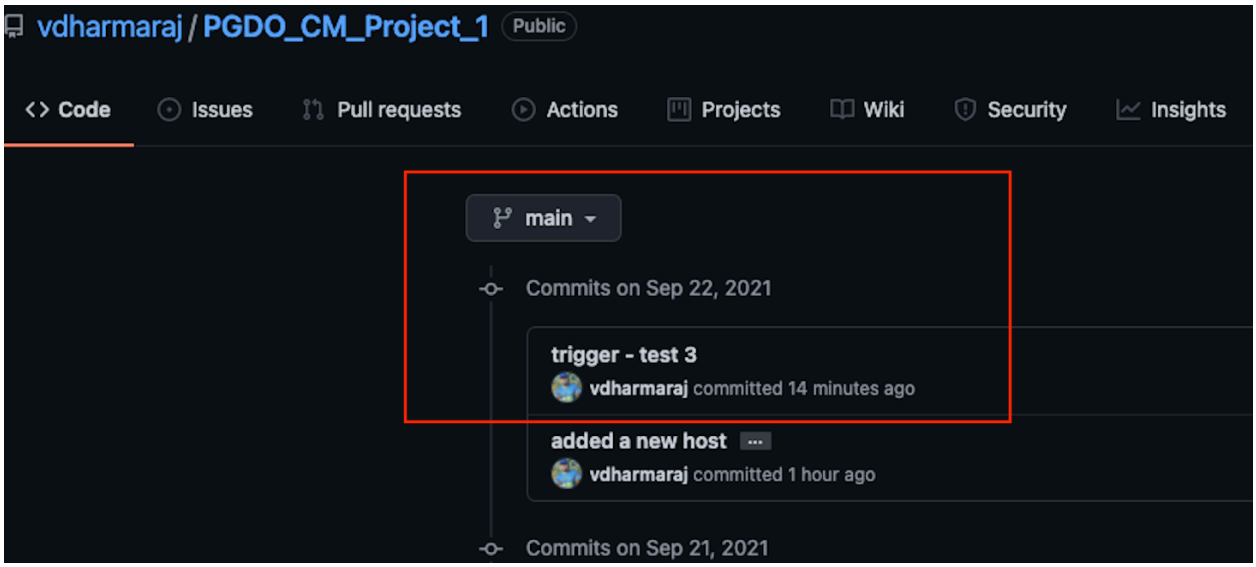


5. Project Results

Result 1:

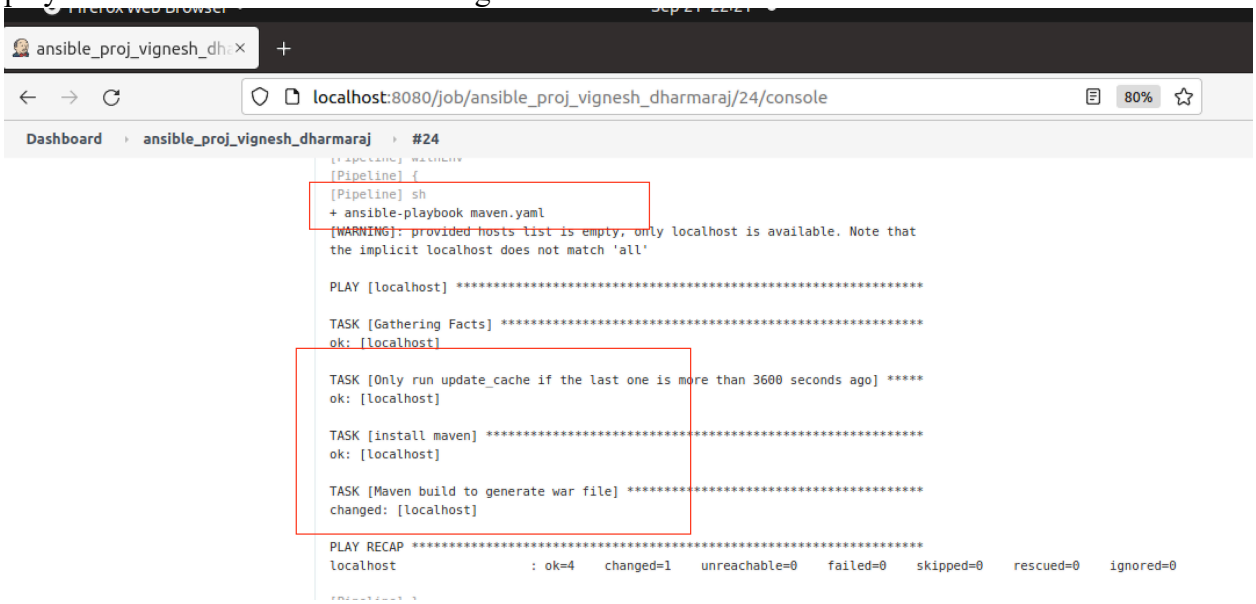
Jenkins job will be triggered automatically corresponding to the repository commit version, this can be seen in the Job build history





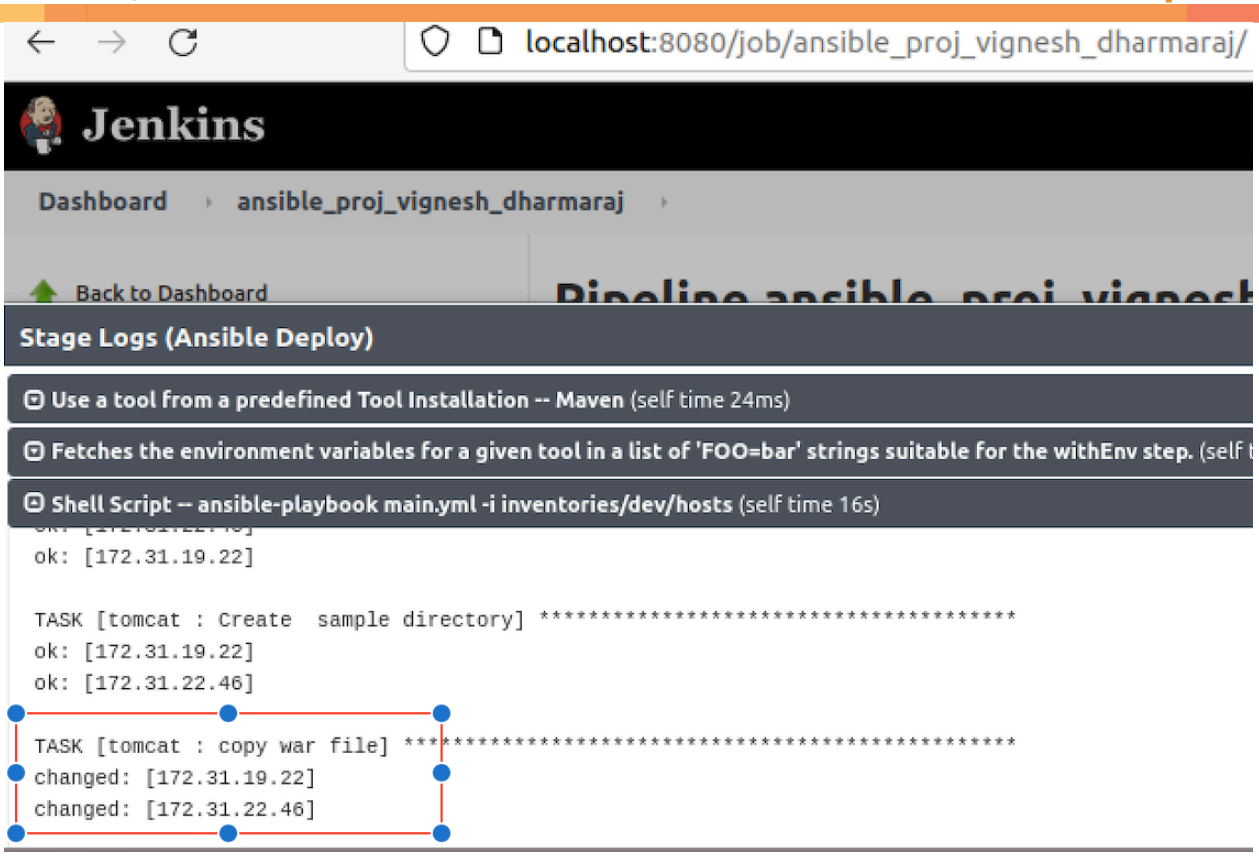
Result 2:

Jenkins job console output is verified to see of the mvn packaging/build is created by a ansible playbook in the Jenkins CI server to generate new war file



Result 3:

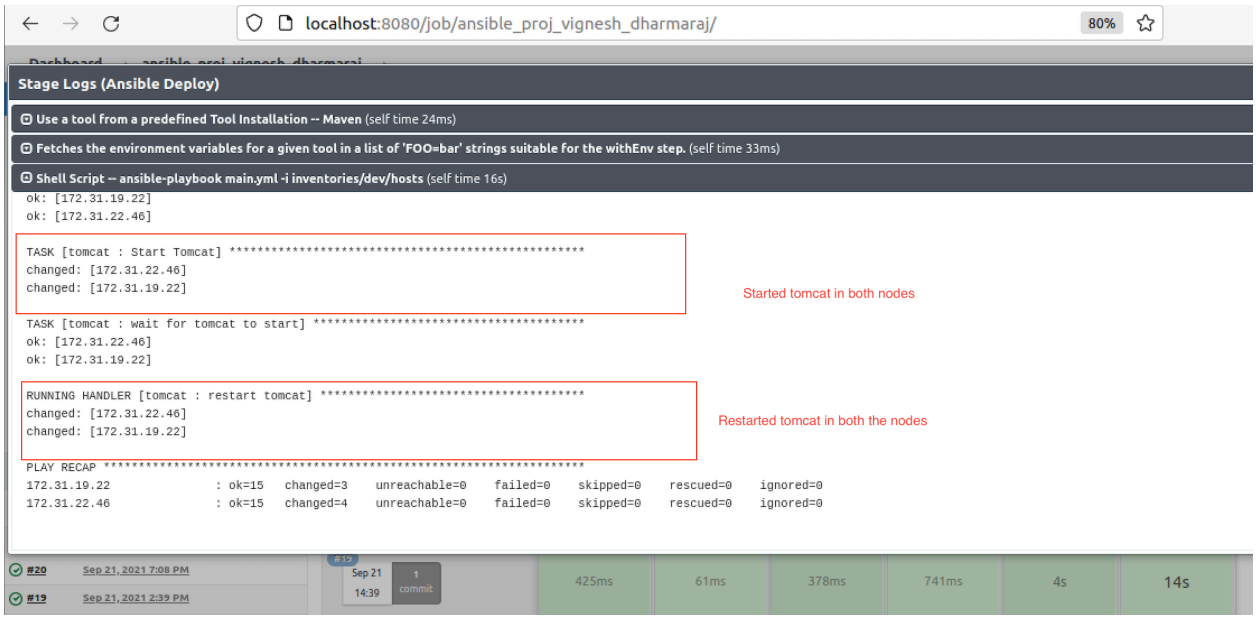
Jenkins job console output is verified to check if new war file is kept in tomcat server folder and then the tomcat is started and restarted in both the managed nodes


localhost:8080/job/ansible_proj_vignesh_dharmaraj/
Jenkins
Dashboard
ansible_proj_vignesh_dharmaraj
Back to Dashboard
Stage Logs (Ansible Deploy)
Use a tool from a predefined Tool Installation -- Maven (self time 24ms)
Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.
Shell Script -- ansible-playbook main.yml -i inventories/dev/hosts (self time 16s)
ok: [172.31.19.22]
TASK [tomcat : Create sample directory] ************************************
ok: [172.31.19.22]
ok: [172.31.22.46]
TASK [tomcat : copy war file] ***
changed: [172.31.19.22]
changed: [172.31.22.46]

github.com/vdharmaraj/PGDO_CM_Project_1/blob/main/roles/tomcat/tasks/main.yml

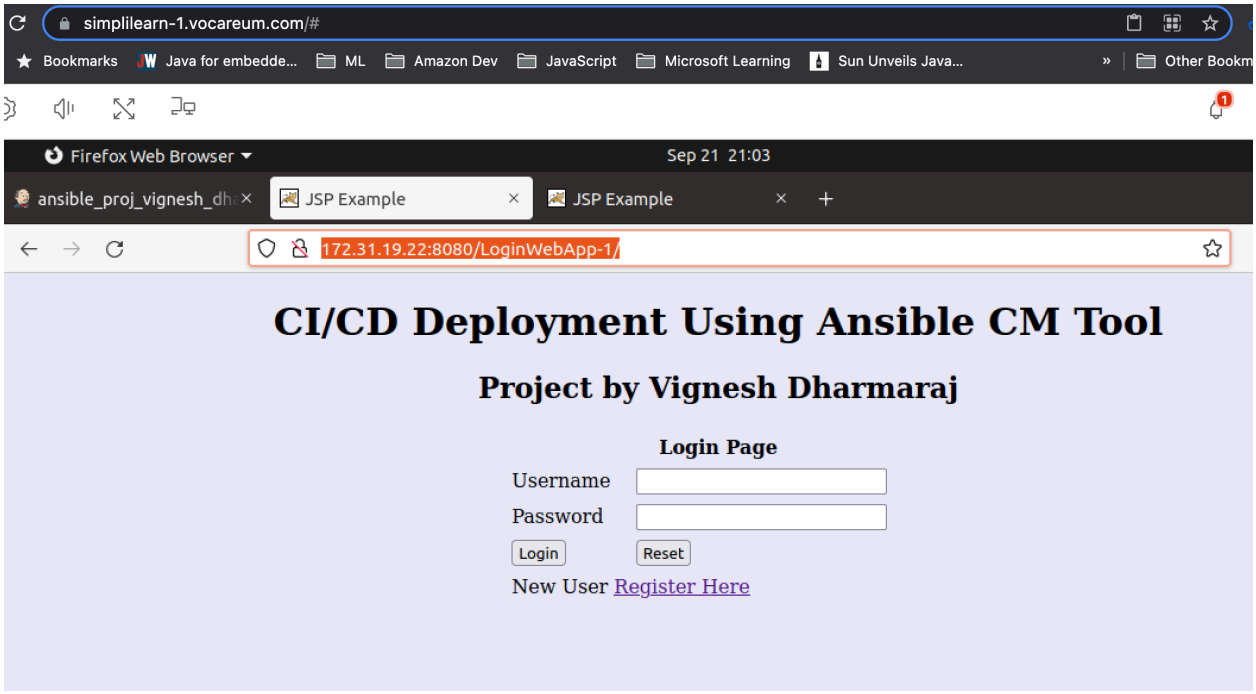
Bookmarks | Java for embedde... | ML | Amazon Dev | JavaScript | Microsoft Learning | Sun Unveils Java... | TV Series, Anime...

```
23
24  - name: Symlink install directory
25    file: src=/opt/apache-tomcat-7.0.61 path=/usr/share/tomcat state=link
26
27  - name: Change ownership of Tomcat installation
28    file: path=/usr/share/tomcat/ owner=tomcat group=tomcat state=directory recurse=yes
29
30  - name: Configure Tomcat server
31    template: src=server.xml dest=/usr/share/tomcat/conf/
32
33
34  - name: Create  sample directory
35    file:
36      path: "/opt/apache-tomcat-7.0.61/webapps/samples"
37      state: directory
38      mode: 0777
39    become: true
40
41
42  - name: copy war file
43    copy: src=./target/LoginWebApp-1.war dest=/opt/apache-tomcat-7.0.61/webapps/
44
45
46    notify: restart tomcat
47
48  - name: Install Tomcat init script
49    copy: src=tomcat-initscript.sh dest=/etc/init.d/tomcat.sh mode=0755
50
51  - name: Start Tomcat
52    shell: /etc/init.d/tomcat.sh start
53
54  - name: wait for tomcat to start
55    wait_for: port={{http_port}}
```



Result 4:

Deployed application with the changes can be tested by accessing the URL http://<node_ip>:8080/LoginWebApp-1/



6. Conclusion

All the below objectives is achieved and verified.

1. Configure Jenkins server as Ansible provisioning machine
2. Install Ansible plugins in Jenkins CI server
3. Prepare Ansible playbook to run Maven build on Jenkins CI server
4. Prepare Ansible playbook to execute deployment steps on the remote web container with restart of the web container post deployment

We have automated the WAR file deployment using Ansible and Jenkins.
CI server Jenkins had ansible playbooks to do the maven build and the deployment to remote hosts including install/start/restart of the tomcat server is also done by ansible playbook & ansible role.

Finally we had seen all the evidences to Automate Ansible integration with Jenkins CI server so that we can run and execute playbooks to deploy custom WAR files to a web container and then perform restart for the web container.