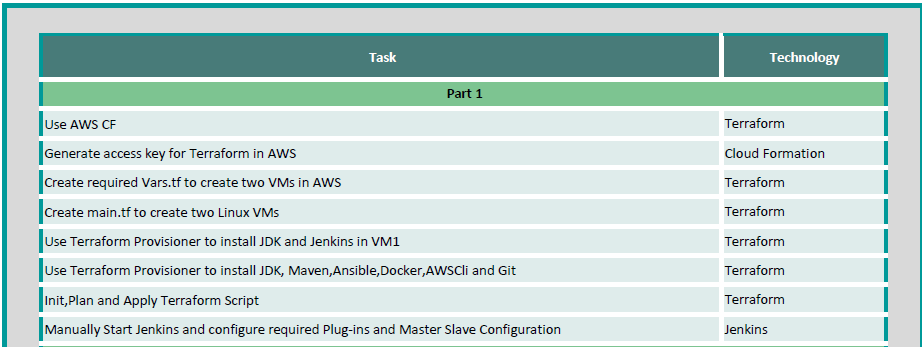
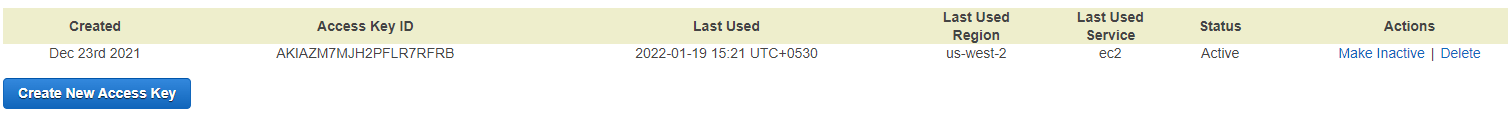
**Project 1**



Created access key

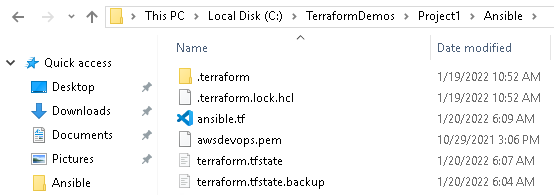


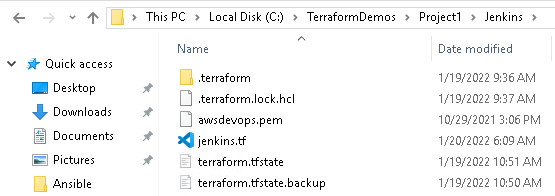
Created 2 terraform files to created 2 VM’s one for Jenkins and another for Ansible with provisioners.

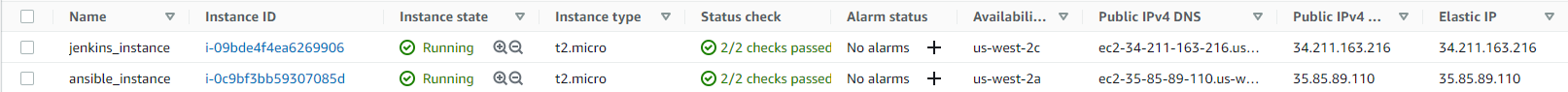


Created 2 folders Jenkins and ansible and placed the above files in their respective directories

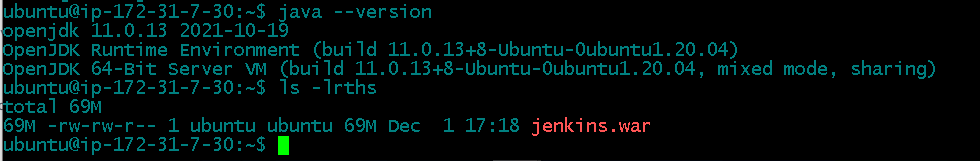
Ran terraform init, plan and apply on the respective folder to create 2 VM’s with name Jenkins\_instance and ansible\_instance with elastic public ip’s as shown below



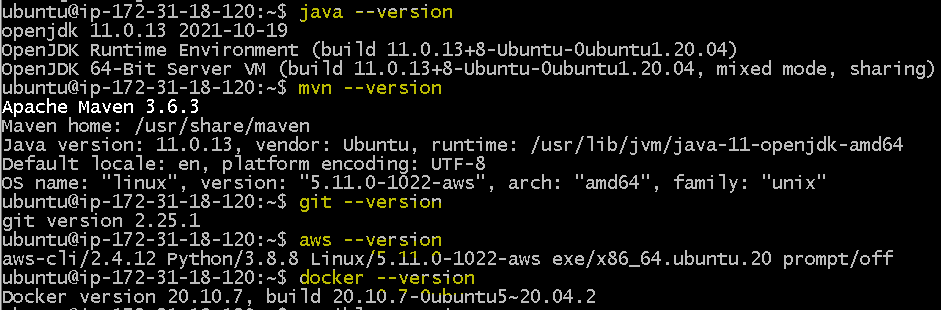


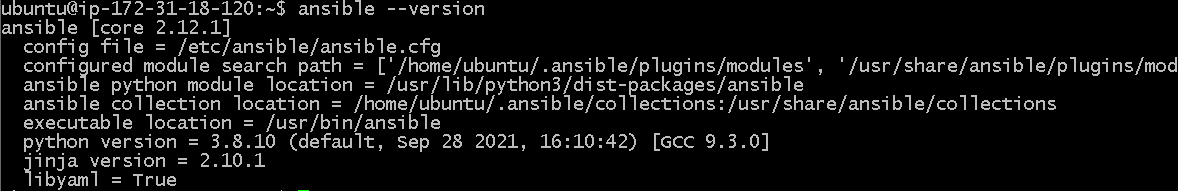


Login in to Jenkins’s server and verify java and Jenkins are installed.



Login to Ansible server and verify java, maven, ansible, docker, AWS CLI and GIT



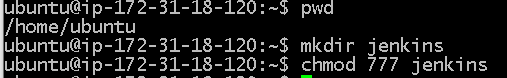


Now run aws configure to setup aws console authentication on Ansible server

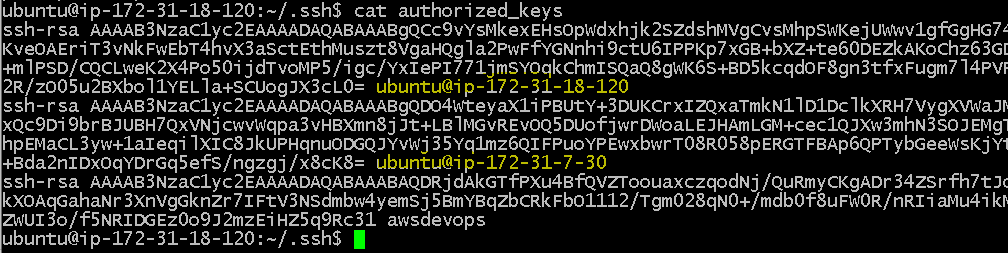
Now generate public and private keys using ssh-keygen command on both servers

Now on Jenkins and ansible switch to use root and add the “ubuntu ALL=(ALL:ALL) ALL” line to /etc/sudoers file on both servers

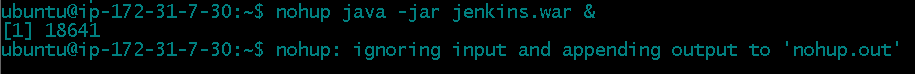
Now on ansible node server create Jenkins folder as shown below



Now on ansible node server add public key of ubuntu user from Jenkins server and ansible server to the file “/home/ubuntu/.ssh/authorized\_keys”



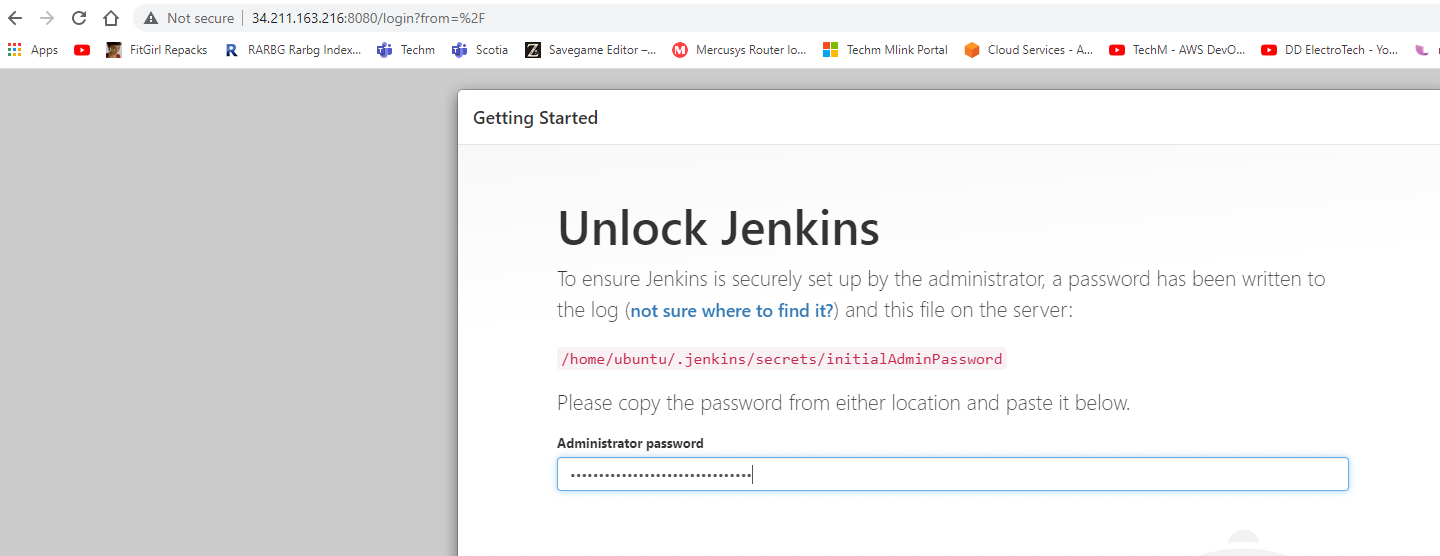
Now on Jenkins server run the below command to start Jenkins



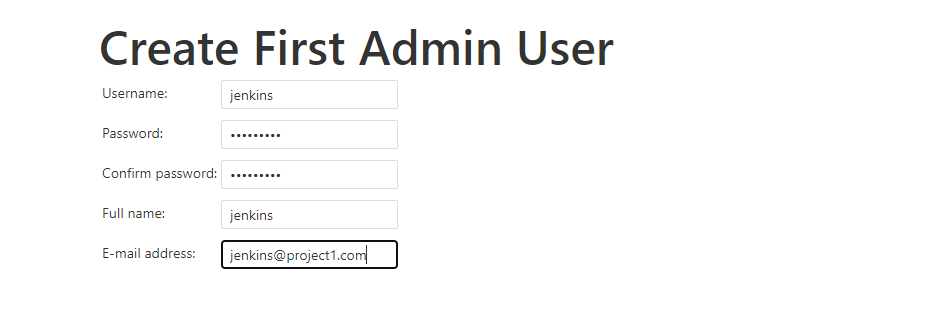
Get the admin password for initial set up from nohup file



bd35fba9684b4853aeb53632f645c029



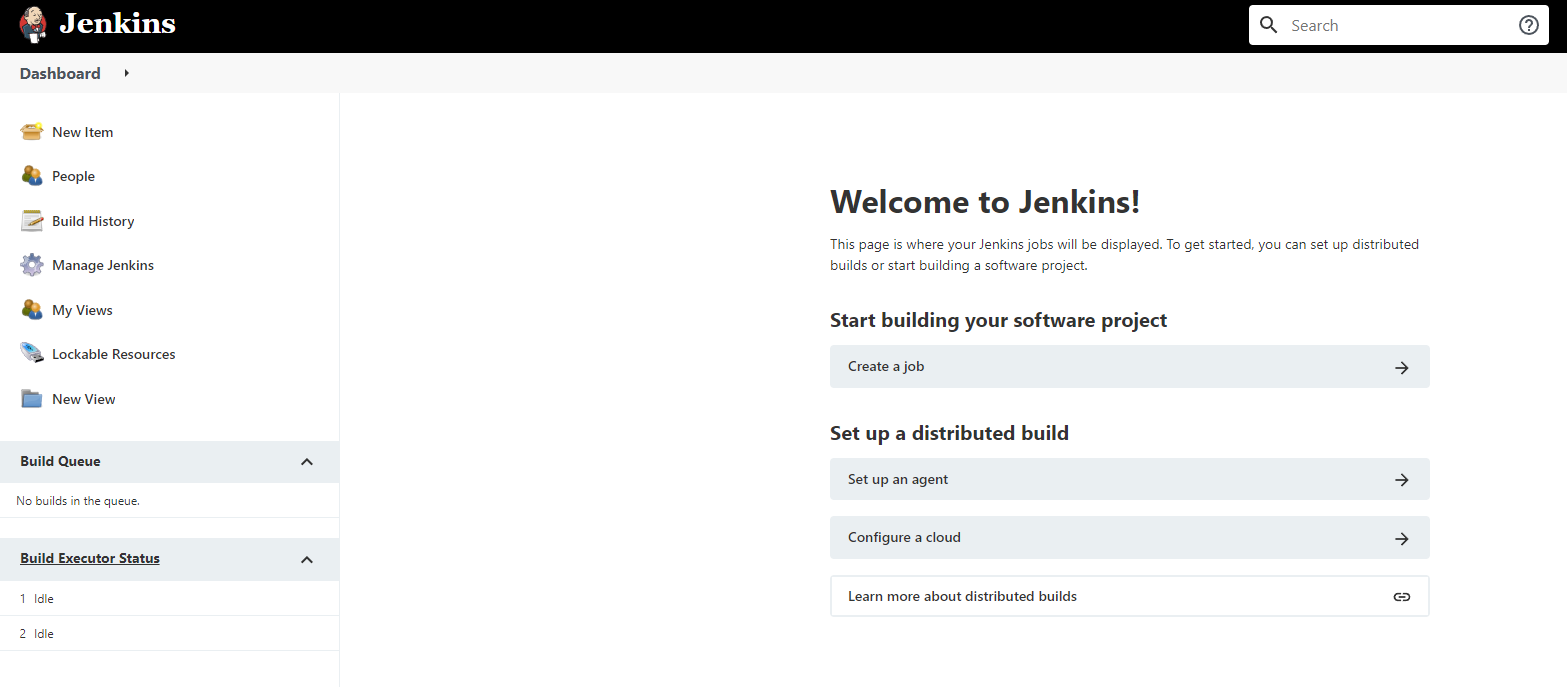
Install suggested plugins and in next screen create admin credentials and remember them,



Username : jenkins

Password : Twenty@20

Save and continue and u will get the below screen



Go to Manage Jenkins -> Manage nodes and clouds -> New node

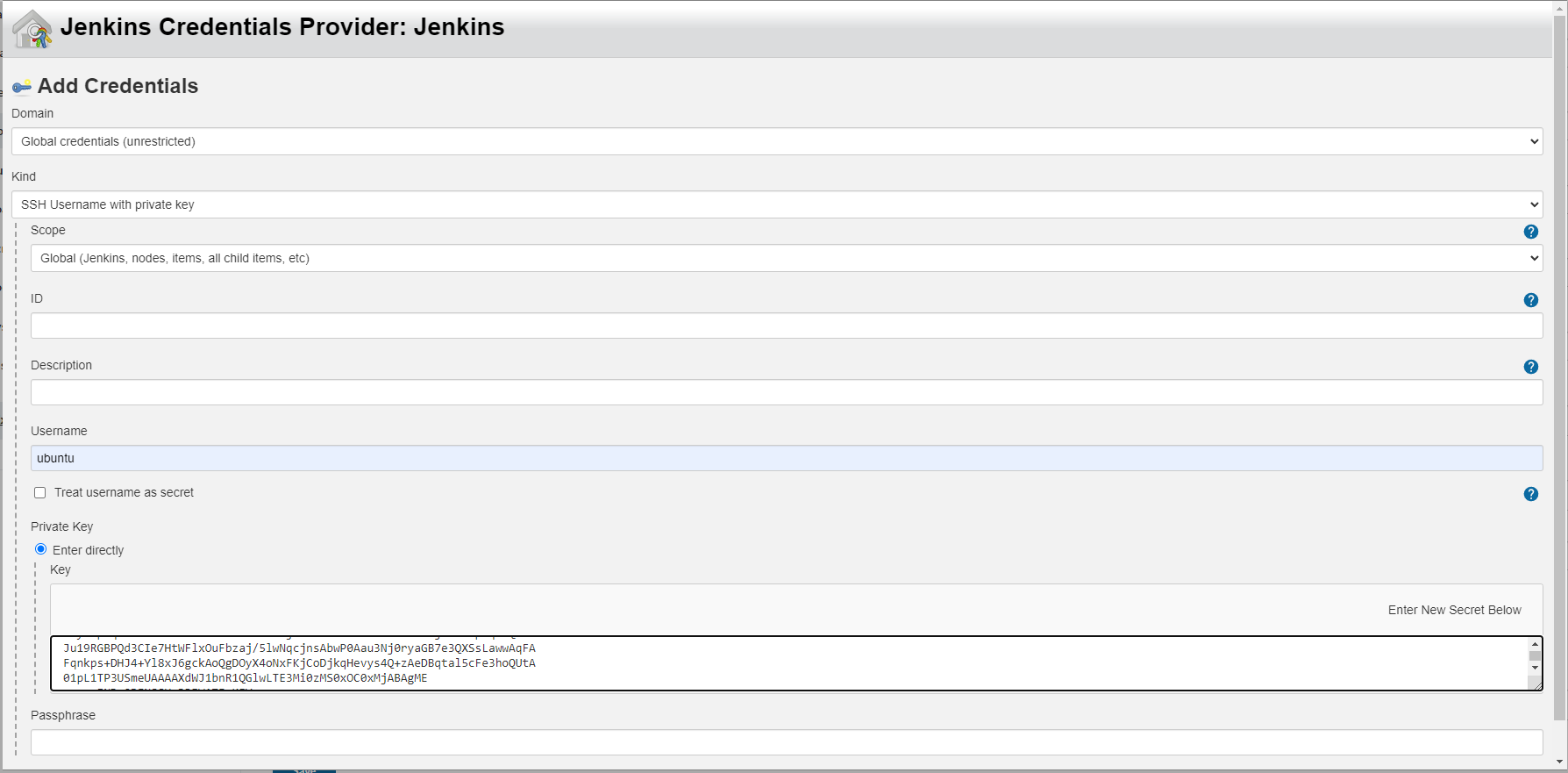
Select name and enable permanent.



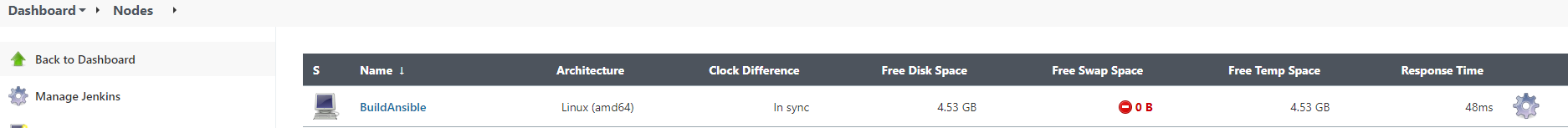
Enter Remote root directory, Labels and select Launch method as Launch agent via SSH.

Enter the private ip of ansible node as host.

Add credentials with kind as SSH username with private key and enter user name as ubuntu and enter private key of ubuntu user in Ansible node instance

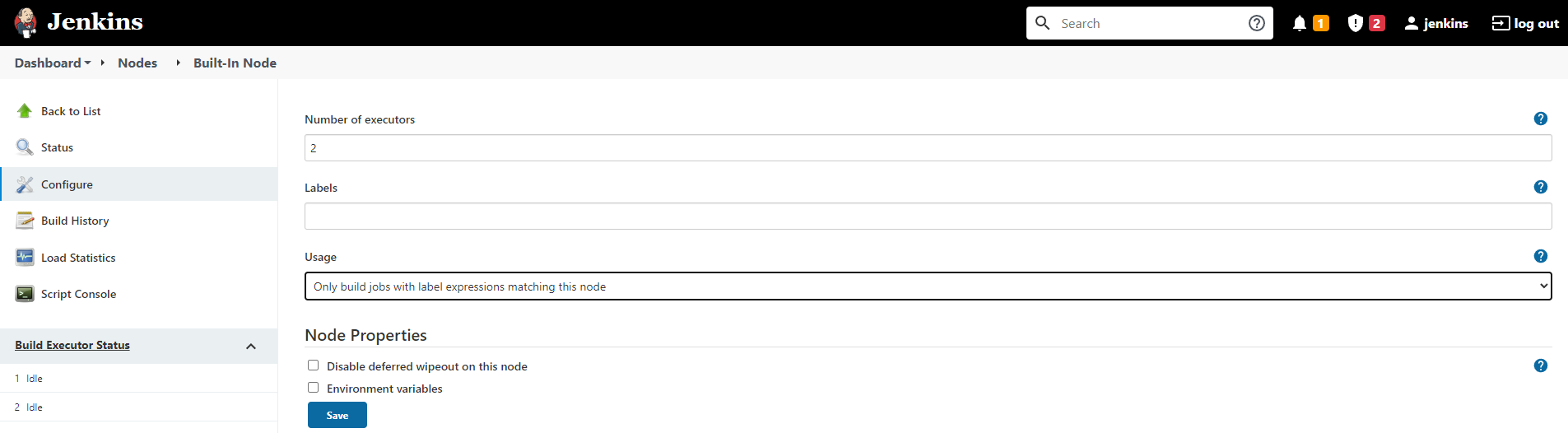


After the setup you can see the node is online as shown below





Make sure that build in node usage is changed to below.



From Manage Plugins screen install required plugins as below

Git Hub Integration

Maven Integration

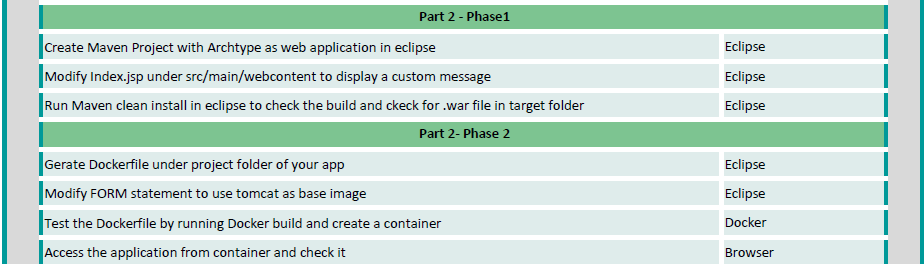
Docker

CloudBees Docker Build and Publish

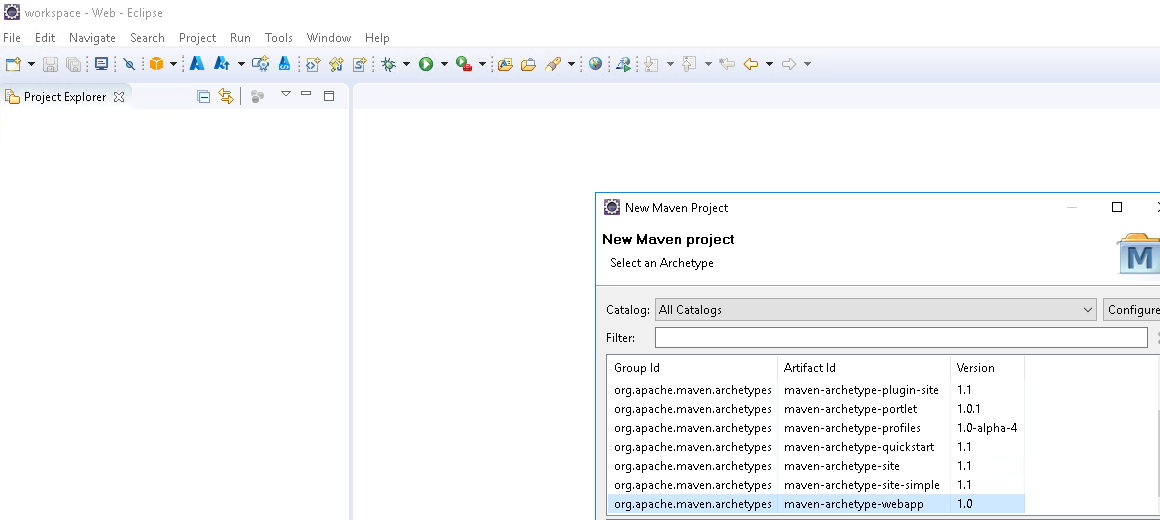
Ansible

Terraform

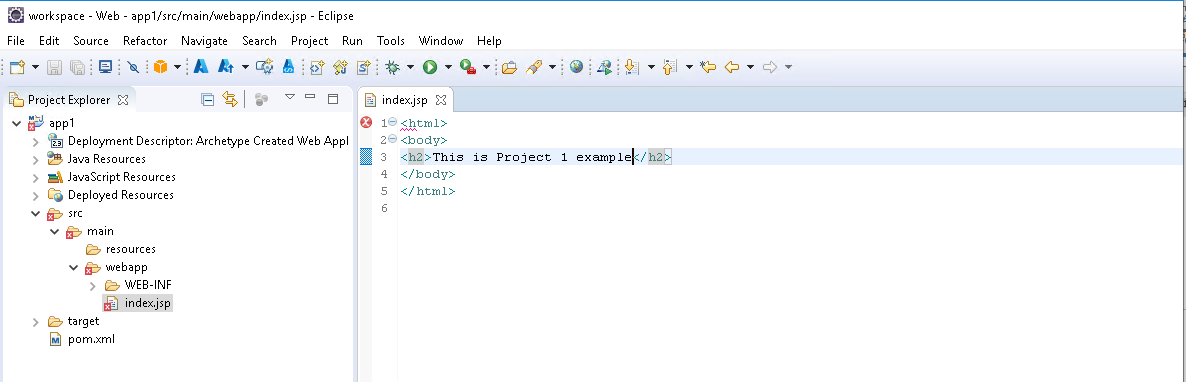
AWS plugins



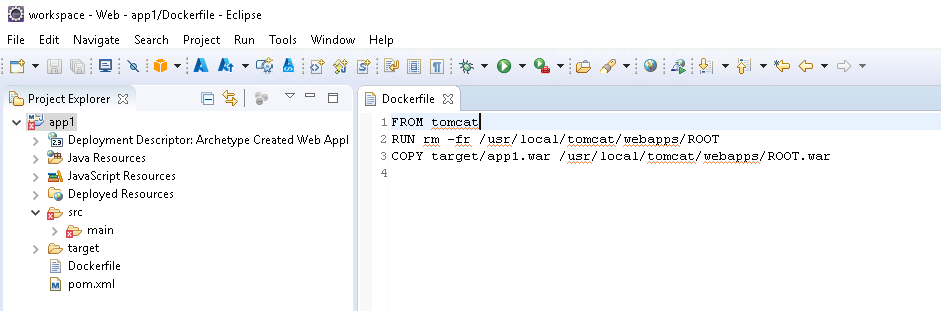
Using eclipse create new Maven project with archtype as web application.



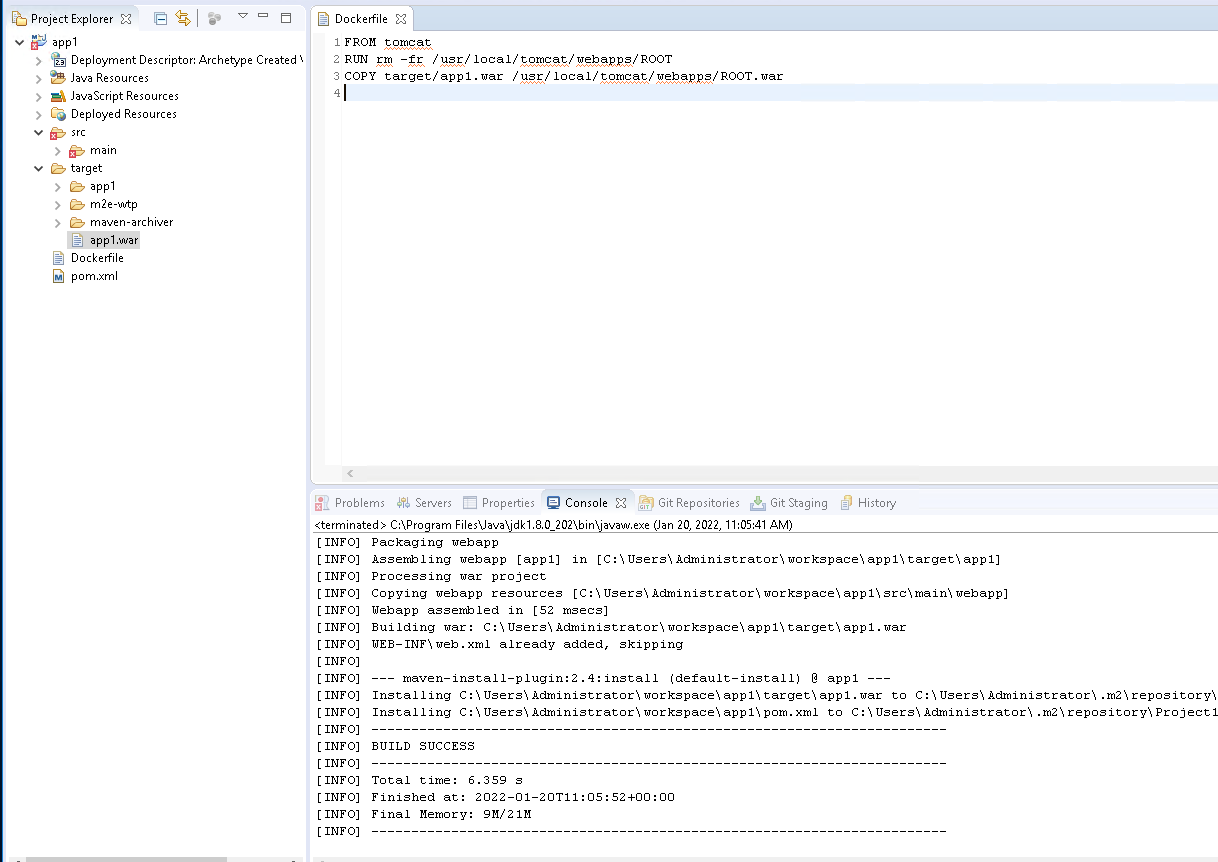
Modified the index.jsp as shown below and save the changes



Create the Dockerfile as shown below



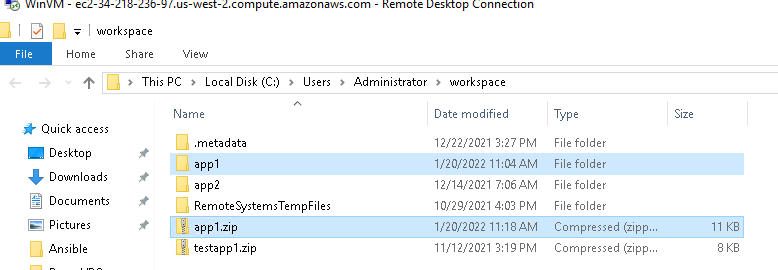
Right click on app1 project Runas -> Maven install to generate app1.war in target folder



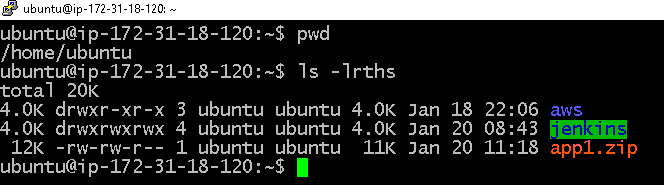
To Test the docker file

Now right click on app1 project -> show in -> system explorer

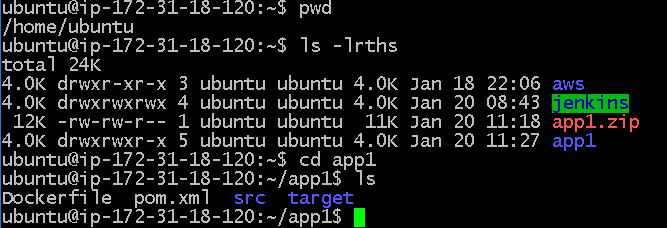
And compress the app1 folder to zip as shown below



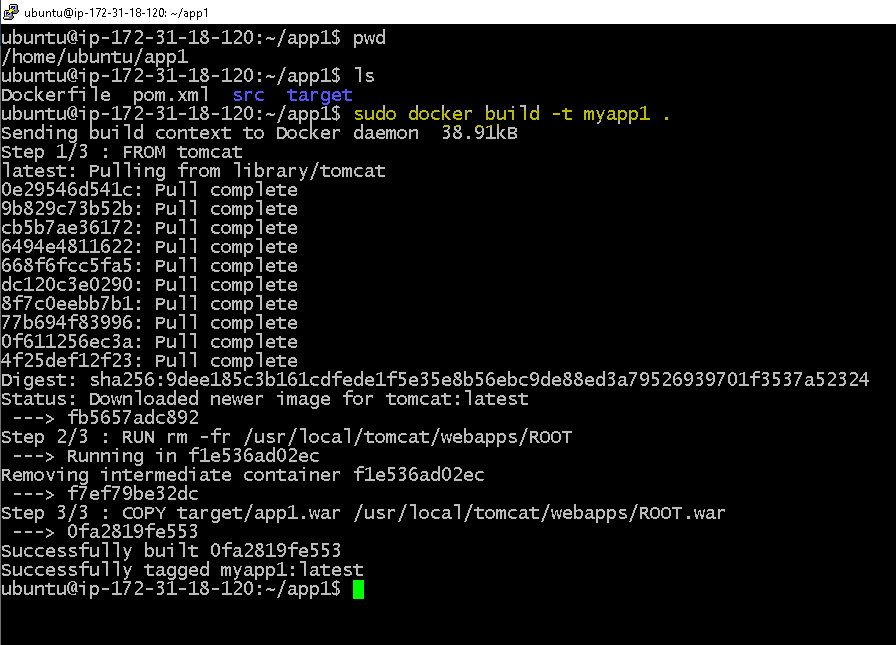
Now use winscp and copy app1.zip to ansible instance, as shown below



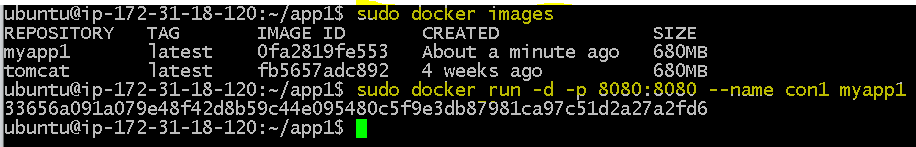
Unzip the app1.zip and open app1 folder to see docker and project files



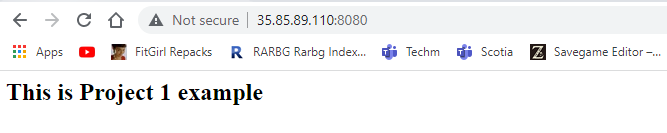
Now run sudo docker build -t myapp1 . to build docker image



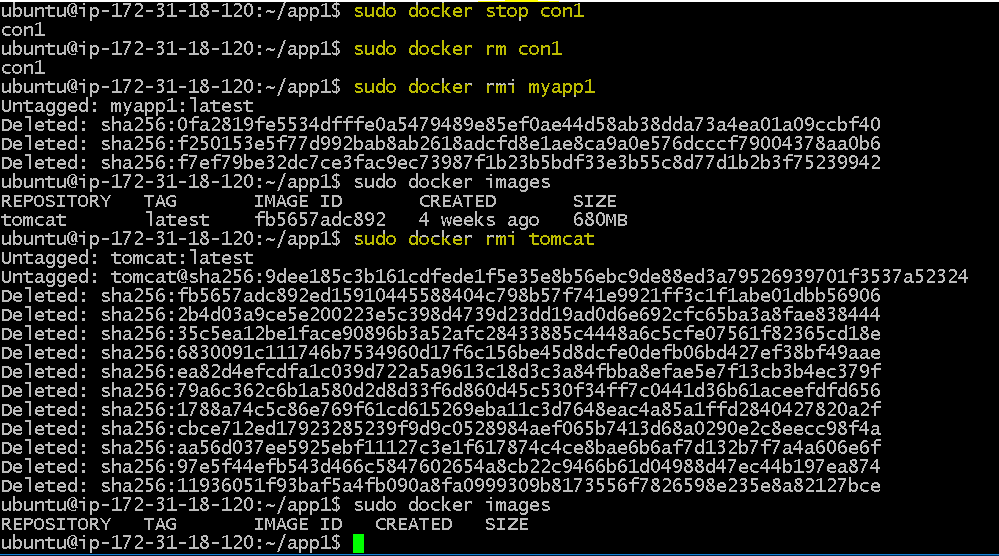
Create a container from myapp1 image as below.

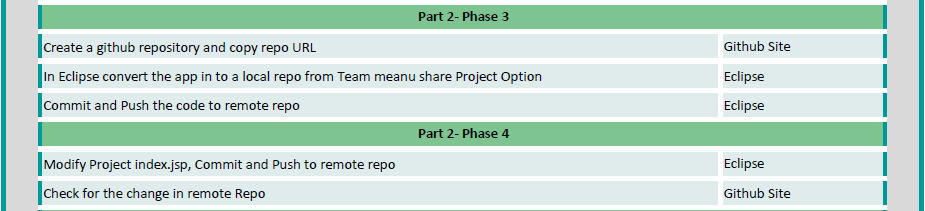


Now use ansible instance ip address with 8080 port to check the page.

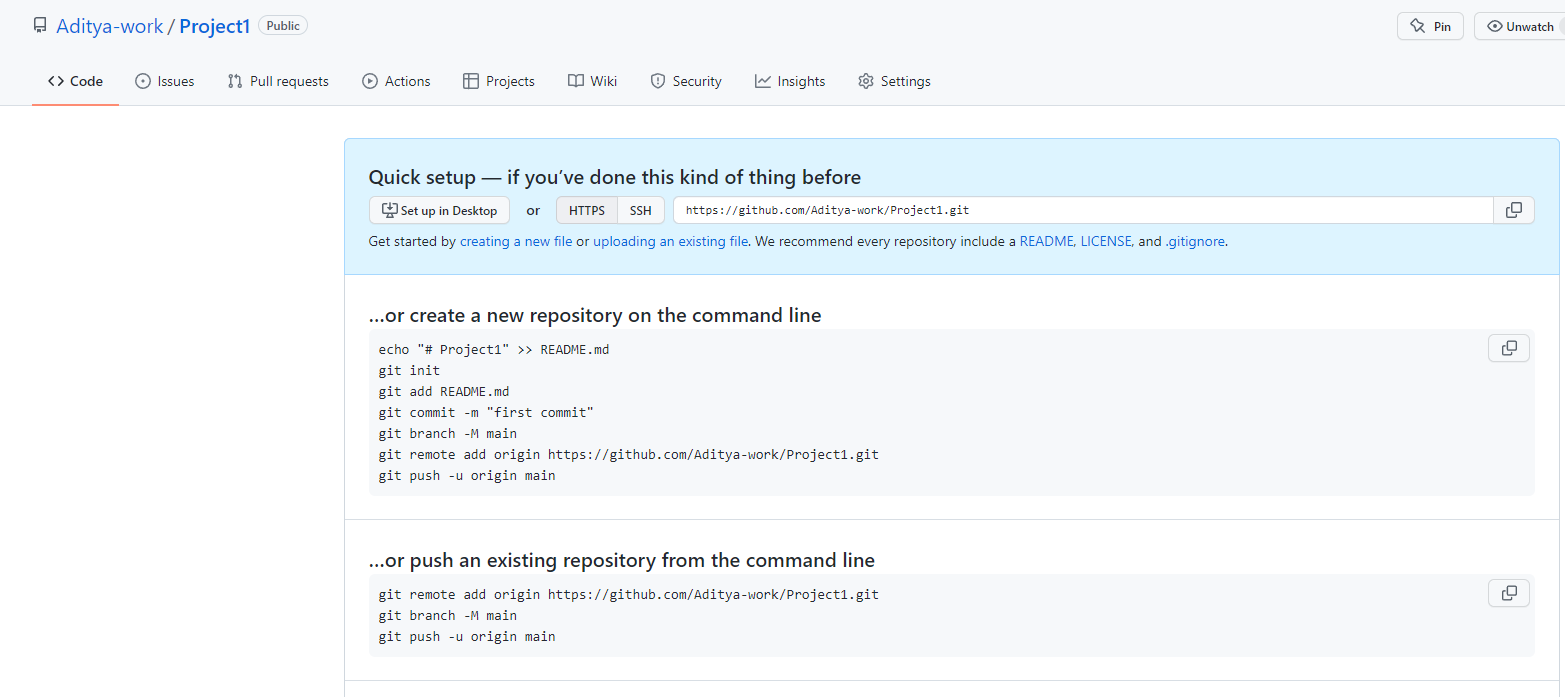


Once the test is successful, remove the container and image as below.

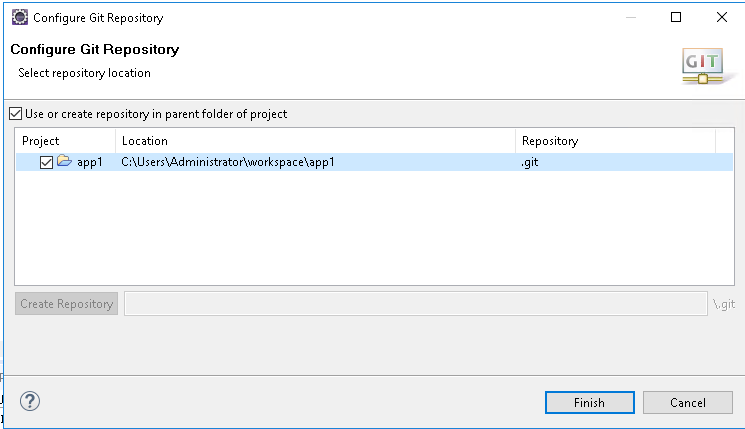




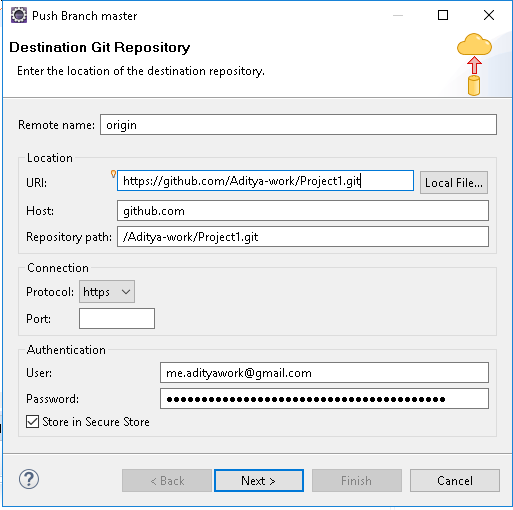
Created new git repo name Project1



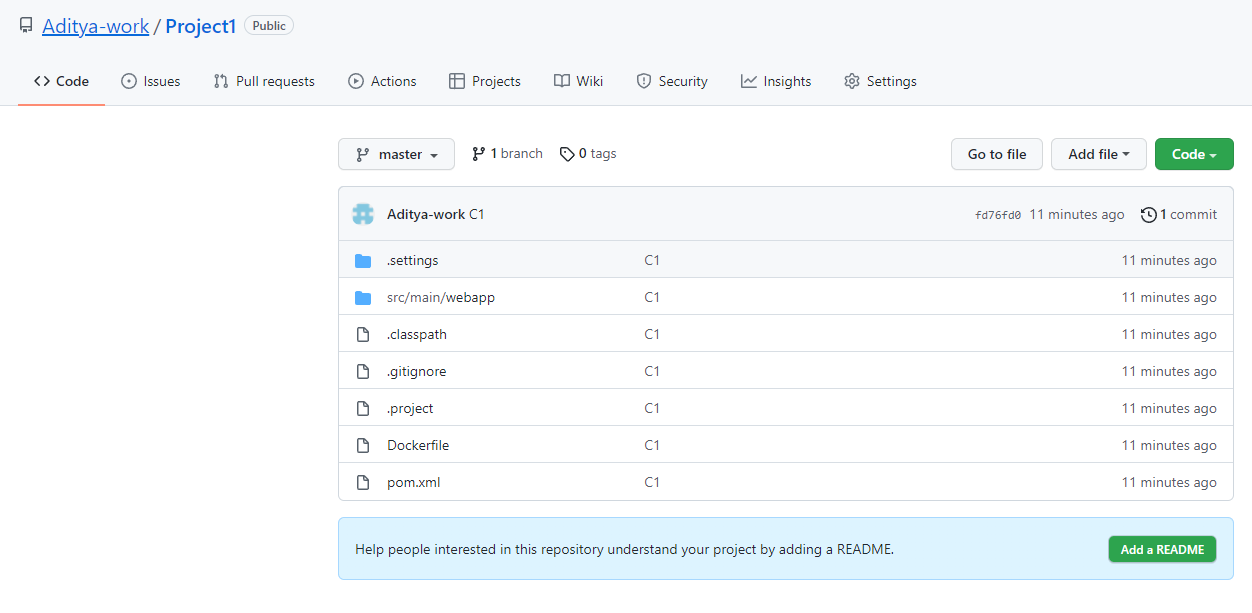
In eclipse right click on app1 -> Team -> Share Project and create local copy.



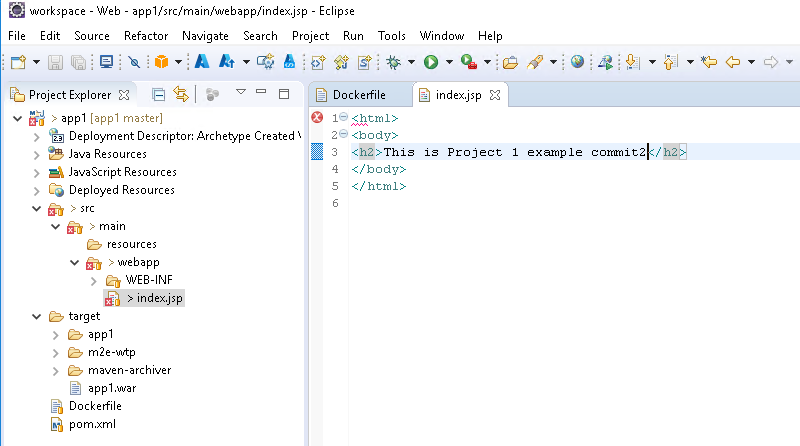
Now commit and push the changes.

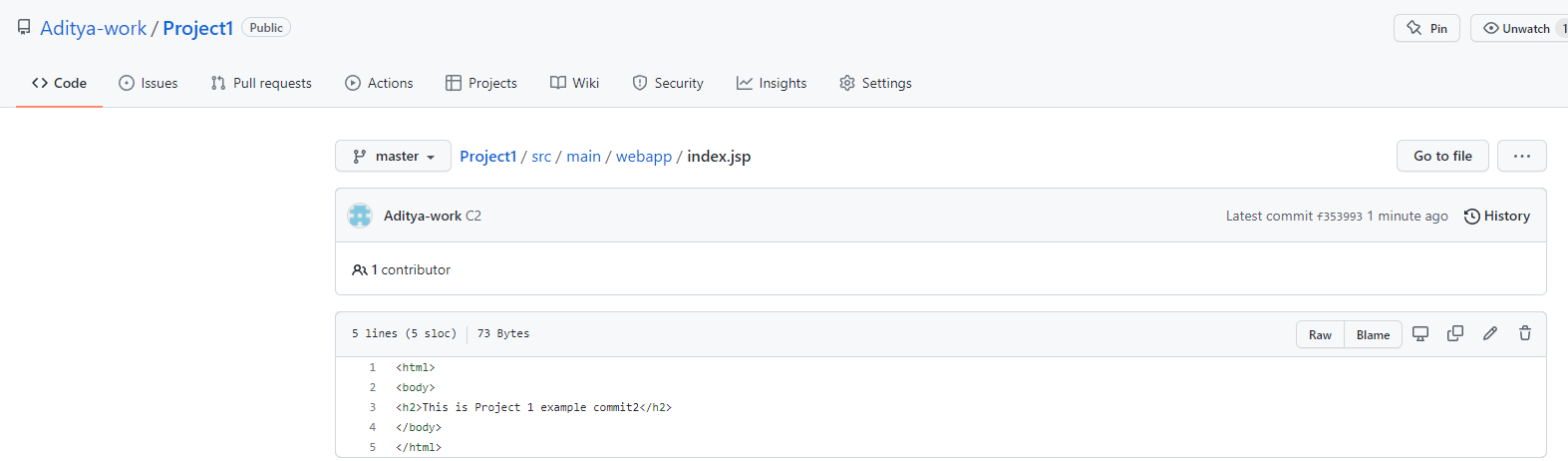


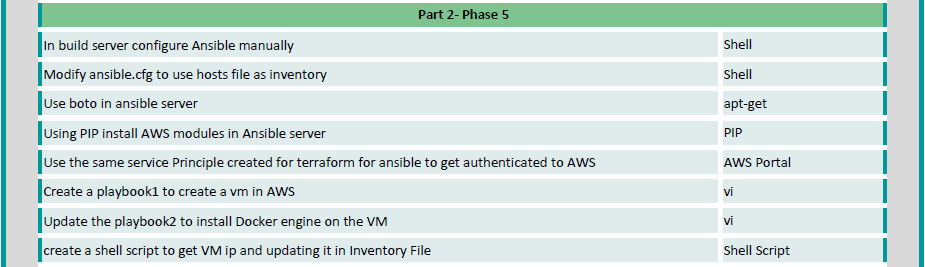
Verify the git hub Project1 repository to verify



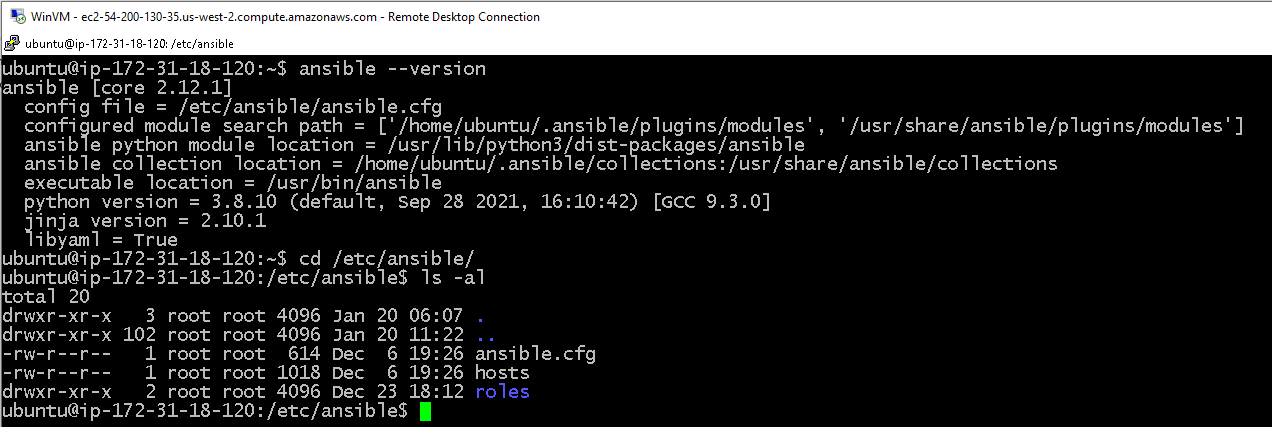
Modify the below index.jsp file and commit C2 and push and verify the git repository



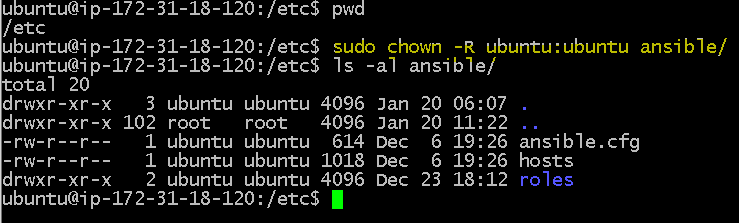




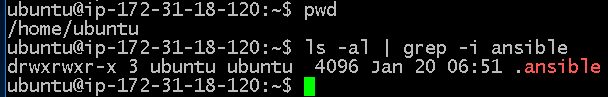
Ansible already installed using terraform earlier. You can verify using ansible –version command.



Change the owner ship of ansible folder from root to ubuntu as shown below.



Make sure that. ansible folder in home dir has ownership to ubuntu



In /etc/ansible/ansible.cfg enable inventory and set deprecation\_warnings as false



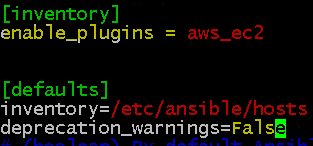


Now run the below commands

sudo apt install python3-pip

pip install boto3

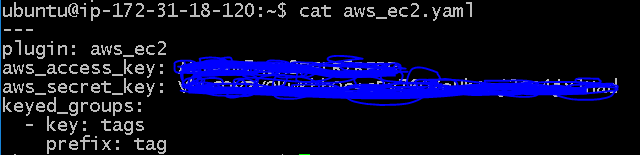
now open /etc/ansible/ansible.cfg file and add enable plugin line as shown below.

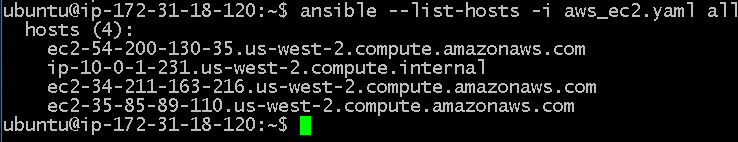


Now create the below aws\_ec2.yaml file with your AWS credentials and run

ansible --list-hosts -i aws\_ec2.yaml all

to check if it can list instances of EC2 or not





Copy you pem file to ansible master server as shown below and give permissions as 600

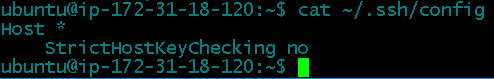


Now create playbook1 which creates EC2 instance and adds the EC2 instance private ip to hosts files



Now create ~/.ssh/config file with the below lines

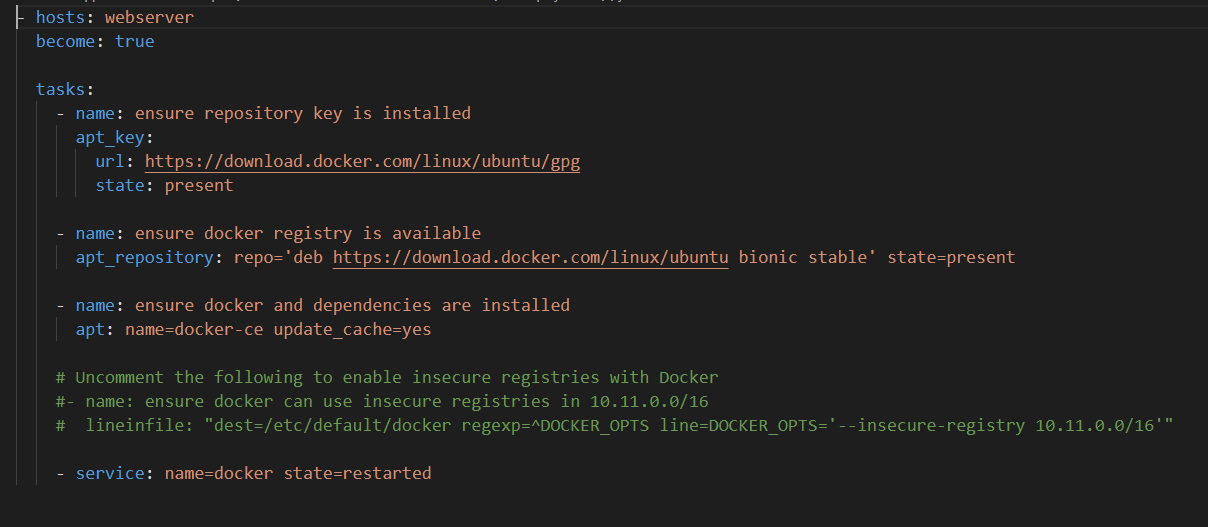
vi ~/.ssh/config



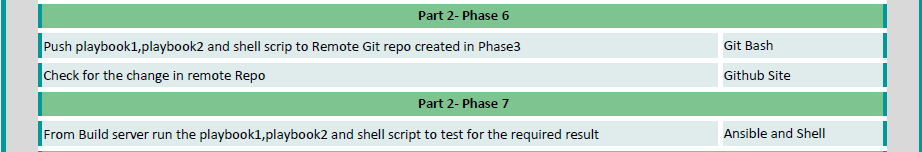
Give permissions to file as 400

sudo chmod 400 ~/.ssh/config

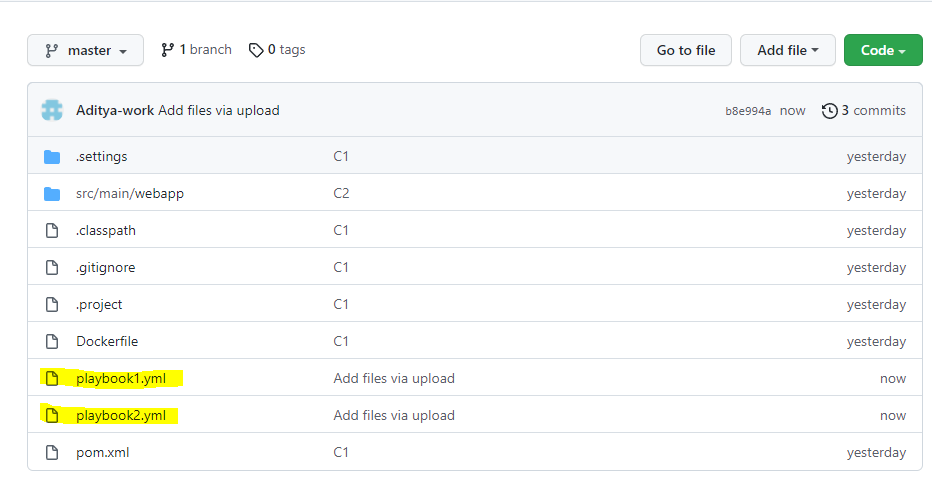
now create a playbook2.yml to install docker in EC2 instance created from playbook1.yml



Playbook1.yml already has lines to copy private ip of newly created EC2 instance to inventory file



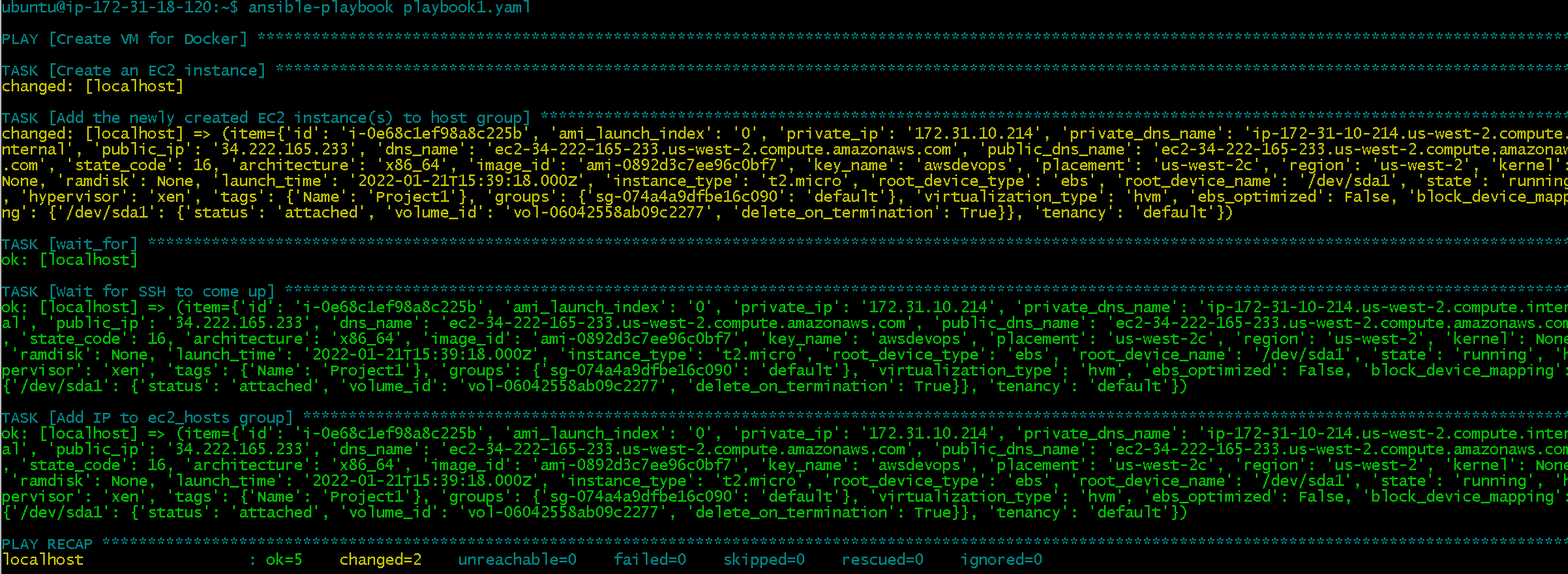
Uploaded both playbooks in github Project1 repository



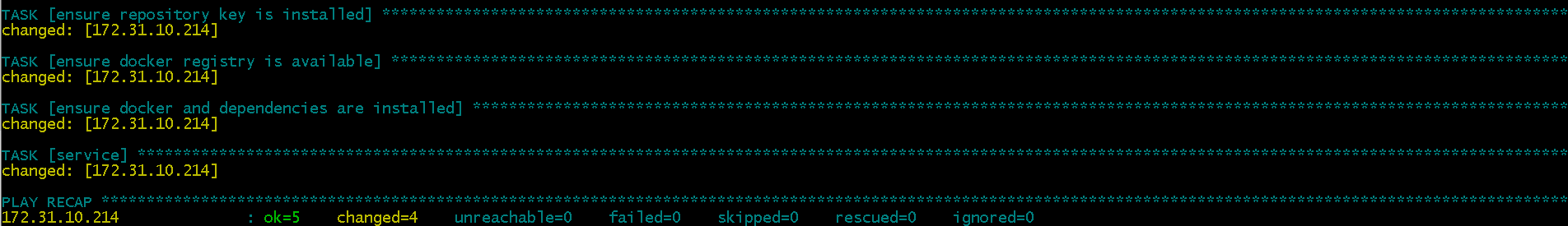
Manually ran playbook1 and playbook2 and the test was successful.

Playbook1 created new EC2 instance and added new EC2 private key to hosts file in Ansible Built server

Playbook2 successfully installed Docker on newly created EC2 instance









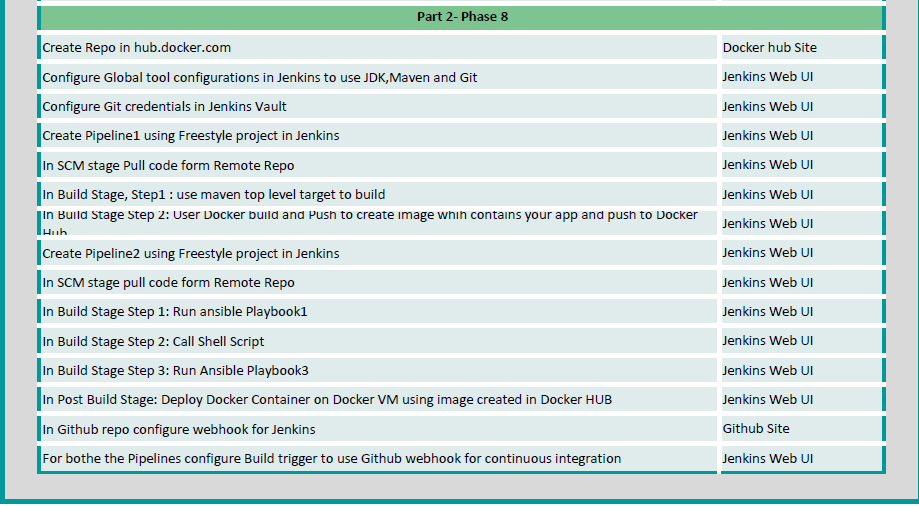
Now Create playbook3.yml which runs on new instance to stop, remove existing container and image and will create the container with latest image.



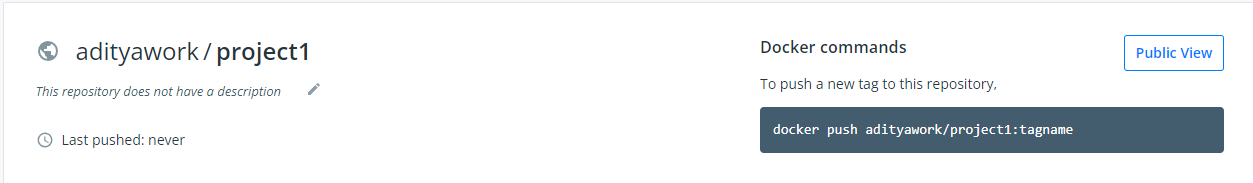
Now create a shell script which will validate if Docker EC2 instance Project1 exists or not.

If exists, it will call playbook3 which will simply use same EC2 instanace to deploy latest docker image, container.

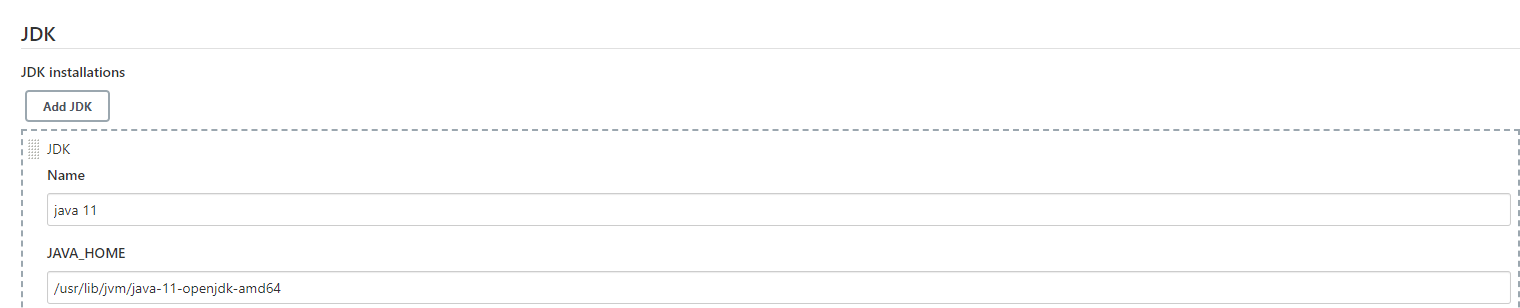
If doesn’t exist, it will call playbook 1 and 2 to create new EC2 instance with name Project1 and will deploy the latest docker image

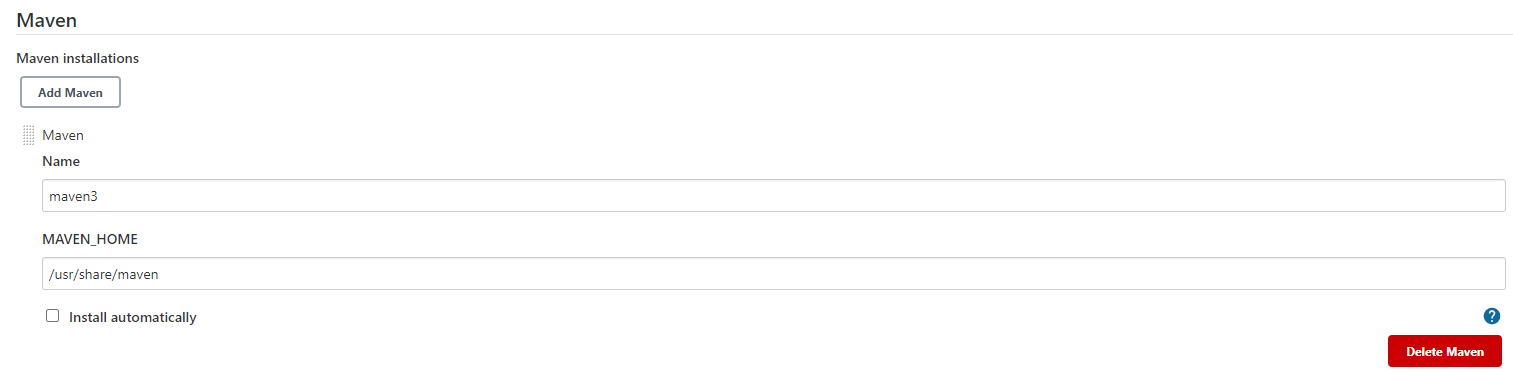


Create new repo in Docker as project1



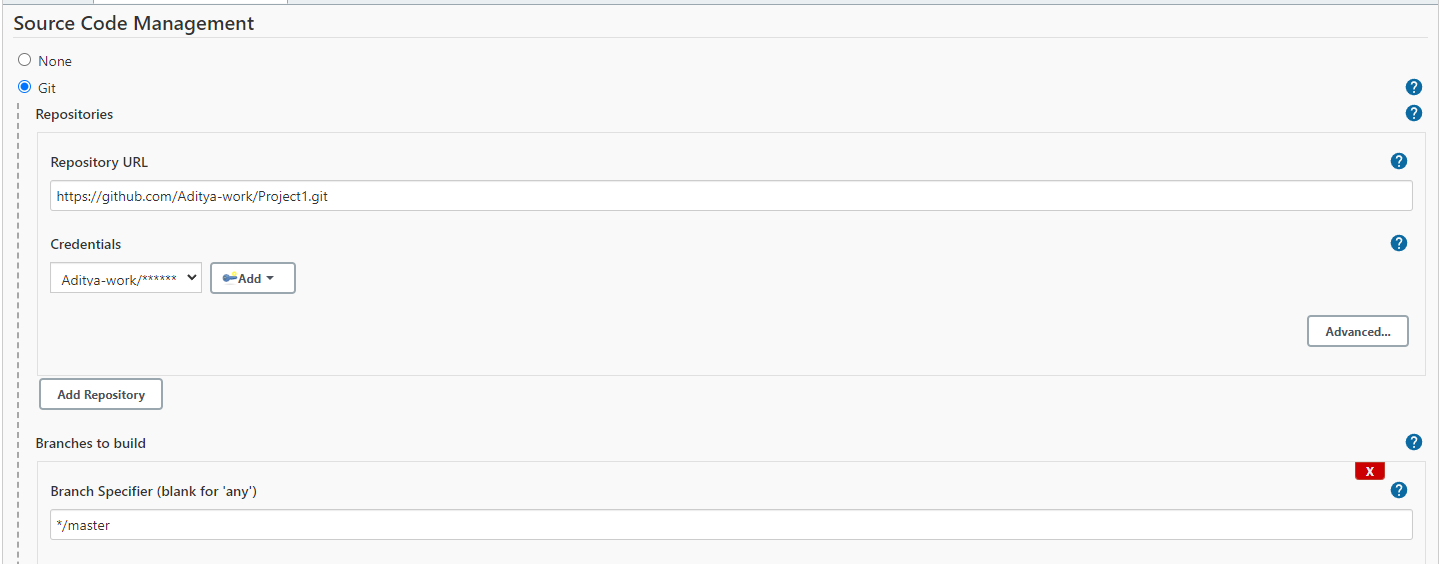
Add JDK, Maven and GIT(automatically detected) in Jenkins global configuration



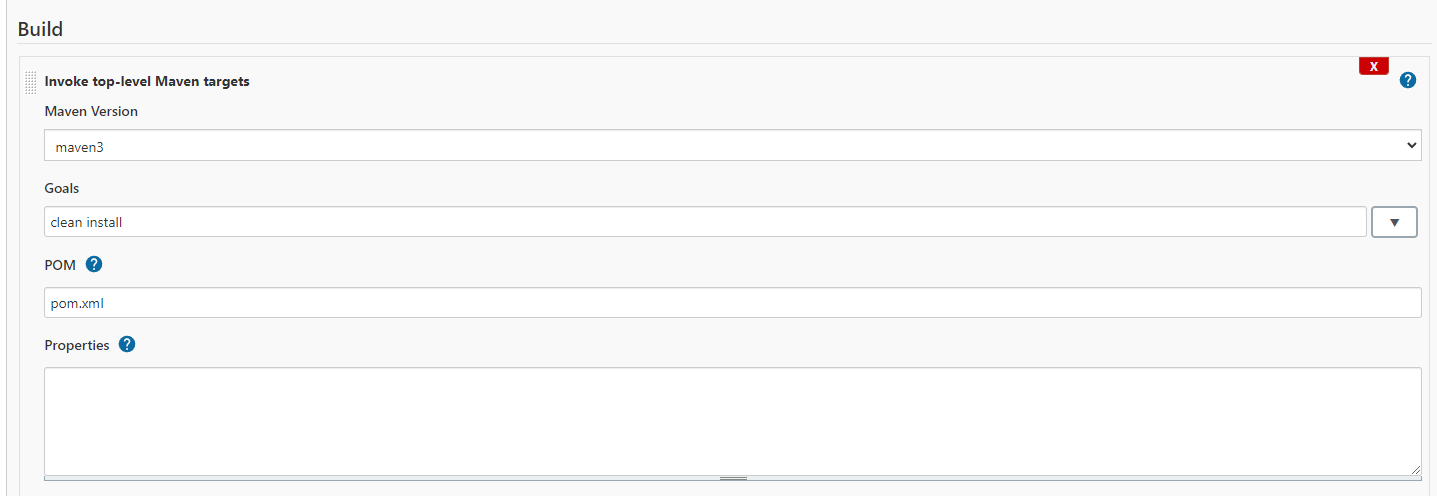


Created new pipeline with name Build

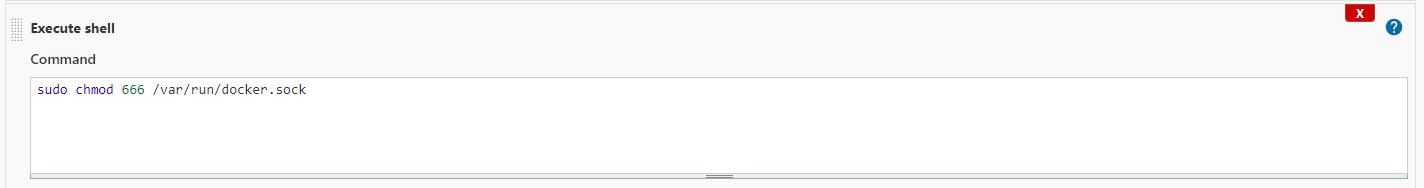
And select SCM as git and added credentials



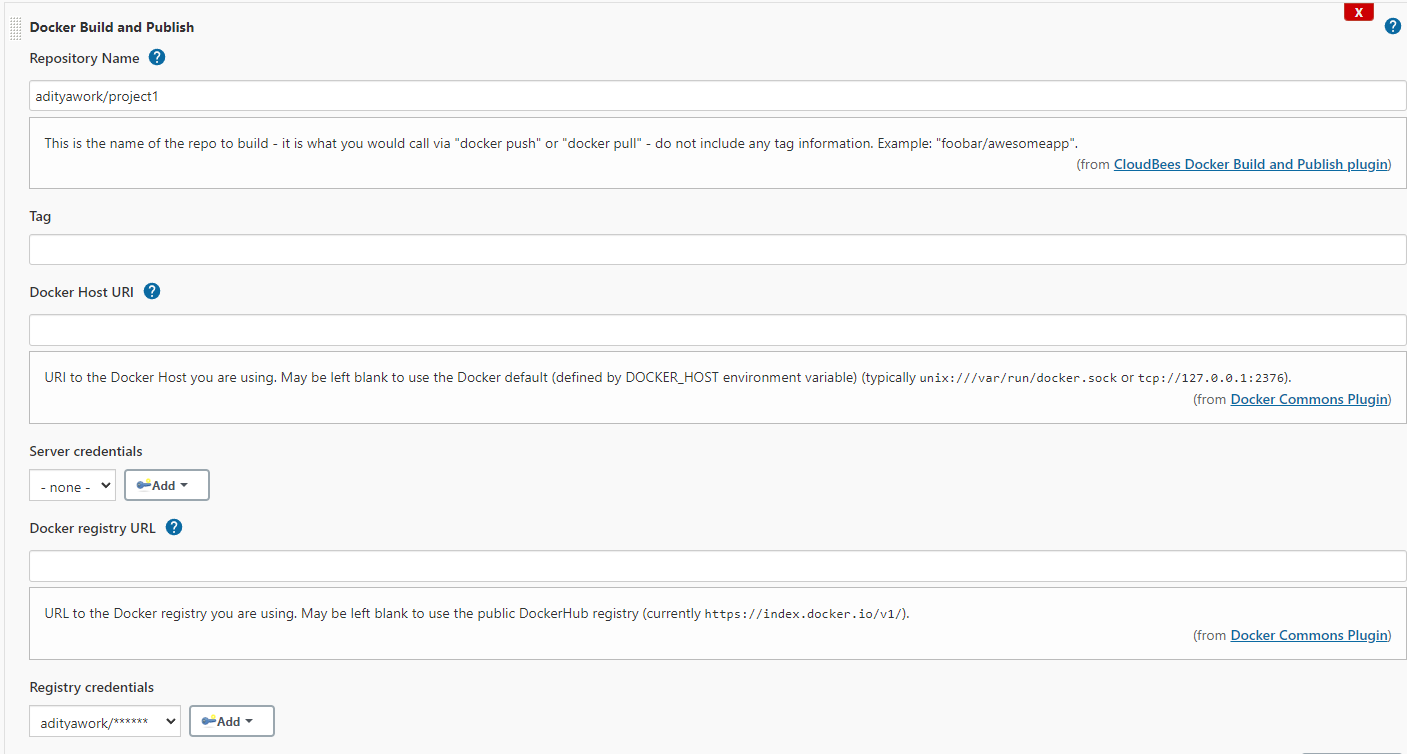
Select build as top level maven targets and fill out the details as below



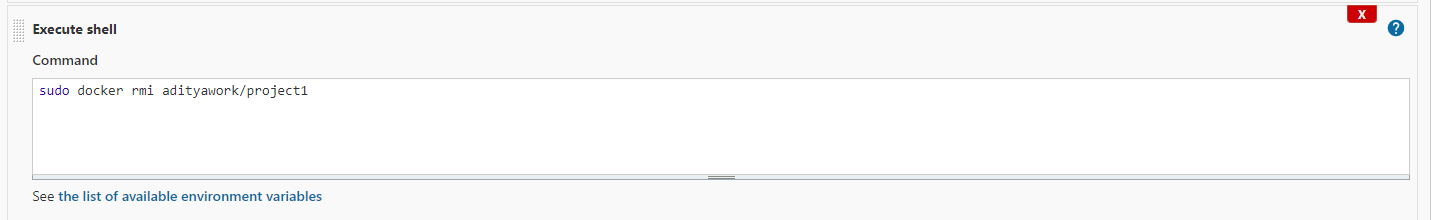
Add execute shell build step as below as the permission for group keeps on changing after server reboot



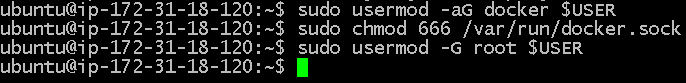
Add one more build step and select Docker Build and Publish



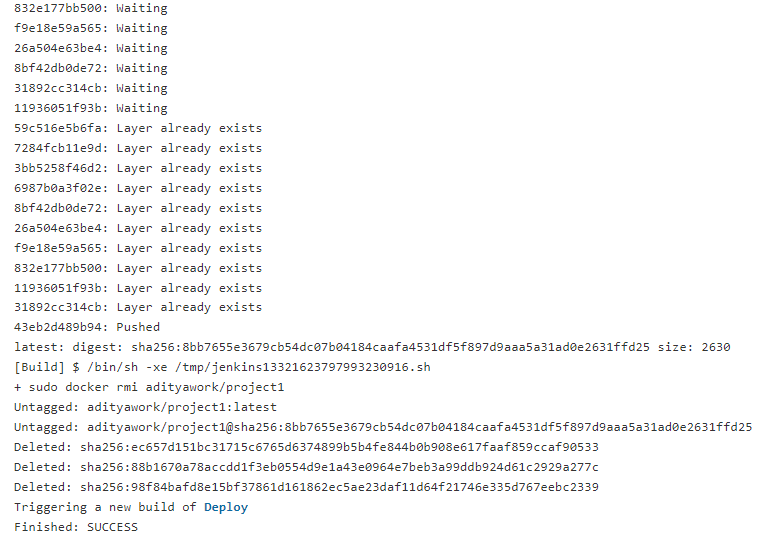
After pushing the image we don’t need it running on Ansible server, hence add one more build step to remove the image from Ansible instance.



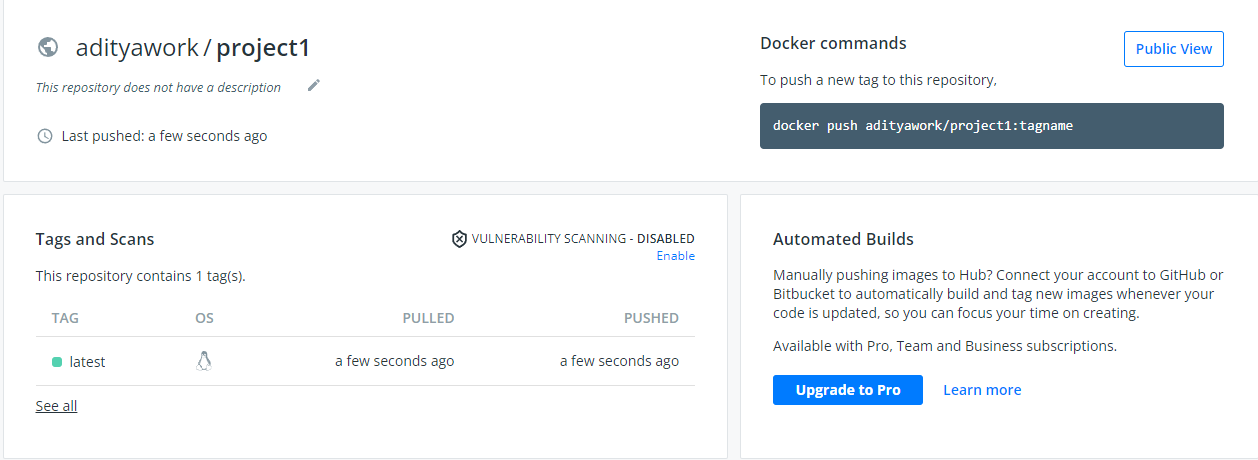
On Ansible server run the below commands one more time.



Now run the Build pipeline and check the status



Build was successful and the image is pushed to Docker repo successfully



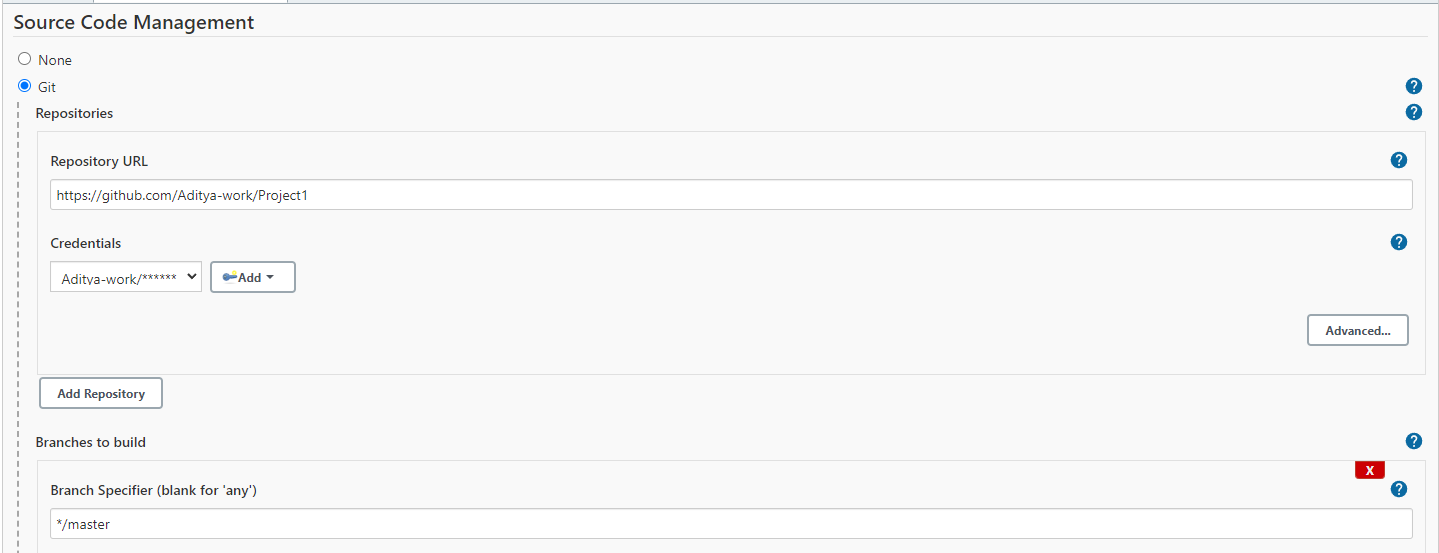
Now update playbook2.yml in the git hub repo with the below content.



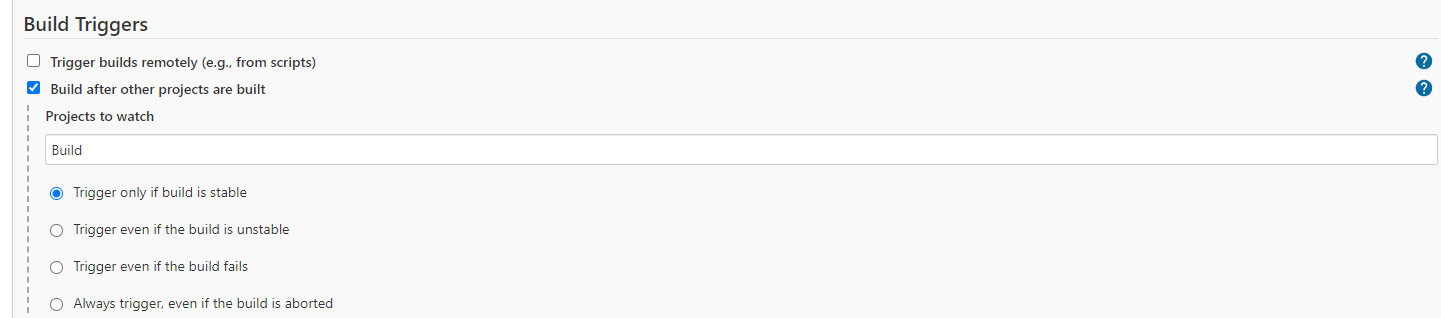
The above playbook contains instructions to install docker, supporting packages and also will pull the image from Dcoker repo and will create a container in new EC2 instance

Now Create new pipeline with name Deploy

Configure SCM



Select Build trigger to start after Deploy Pipeline automatically after Build pipeline is done successfully



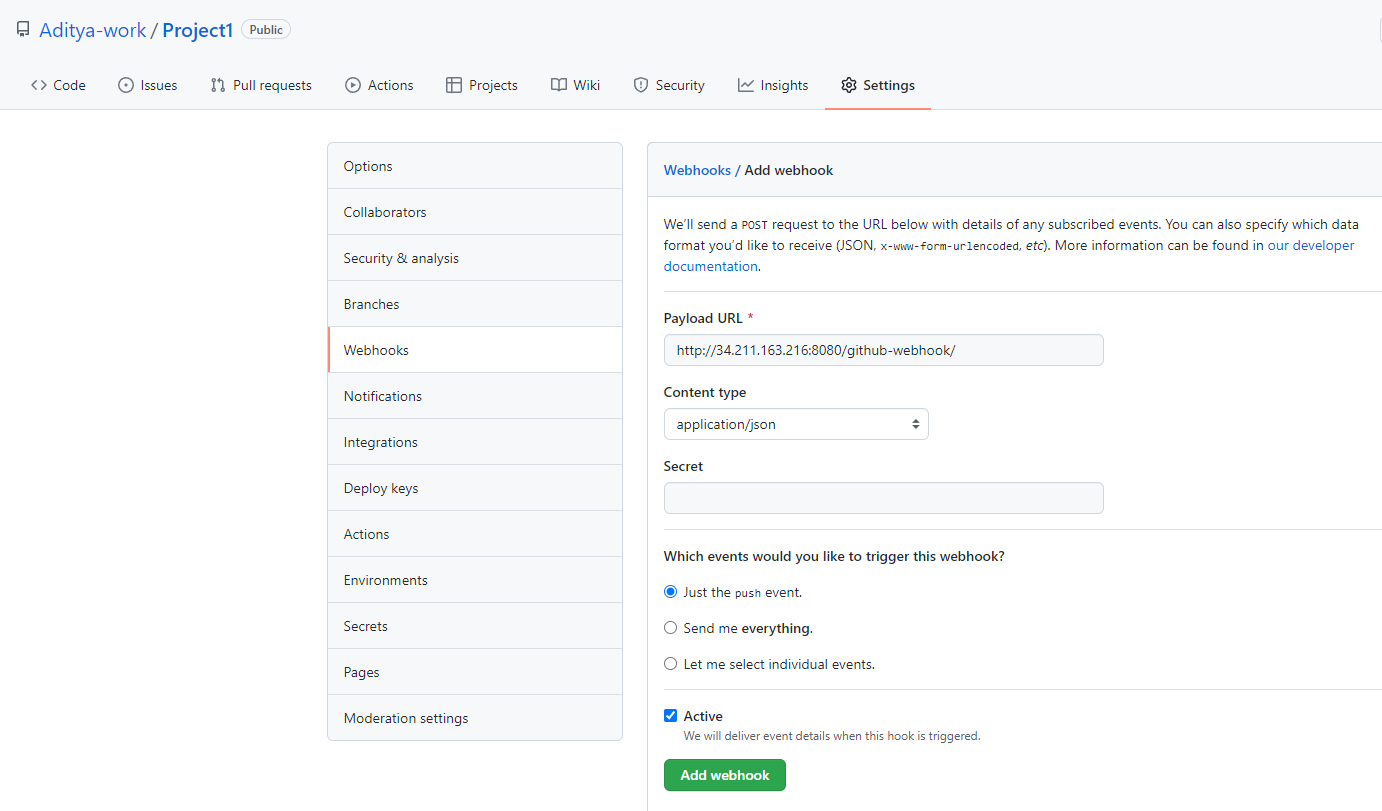
Create a build step to change permissions and invoke the shell script



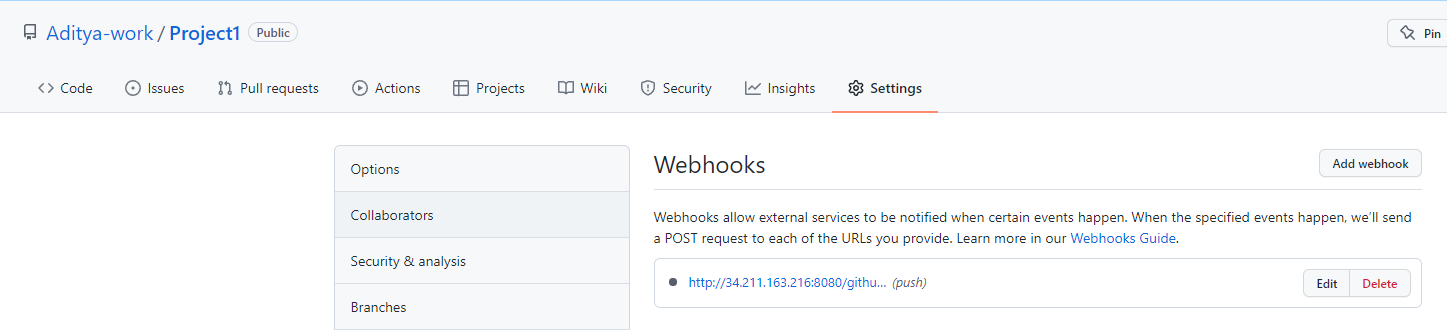
Now go to Github Project1 repo setting and configure the webhooks as shown below.

Payload URL is Jenkins URL/github-webhook/

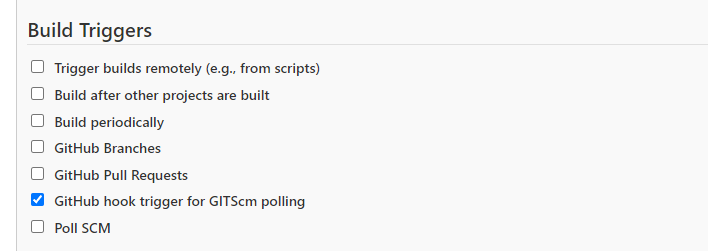
NOTE:- Here elastic ip for Jenkins server will be useful. If we use dynamic public ip the URL will keep on changing causing webhook issues



Now add the webhook



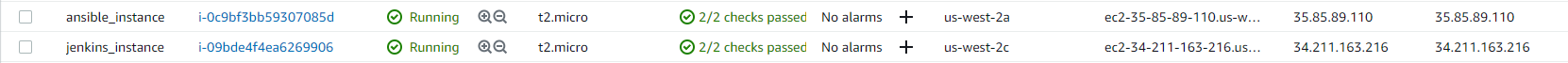
Now go to Build Pipeline and enable GitHub hook trigger



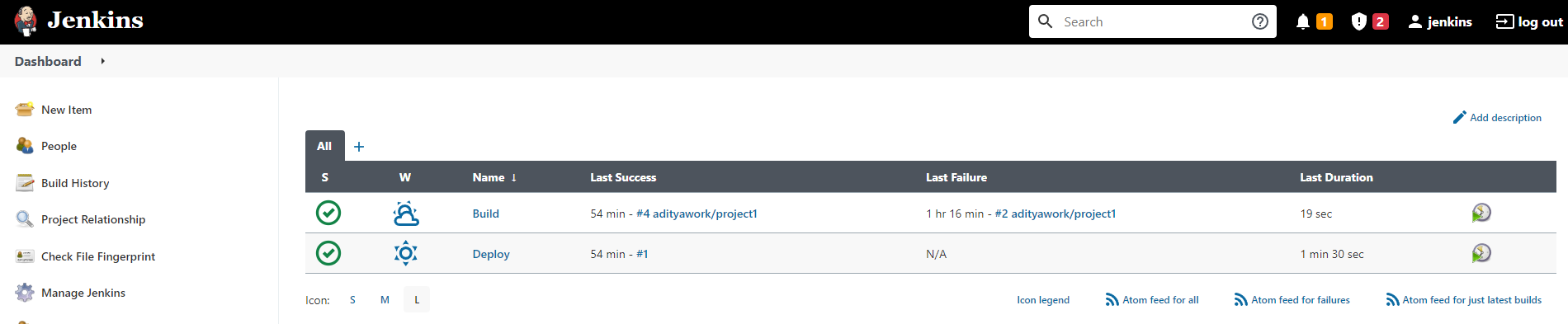
**TESTING**

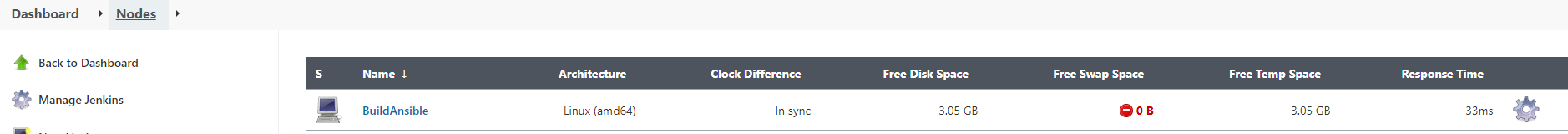
**First Time test**

Jenkins and Ansible should be running

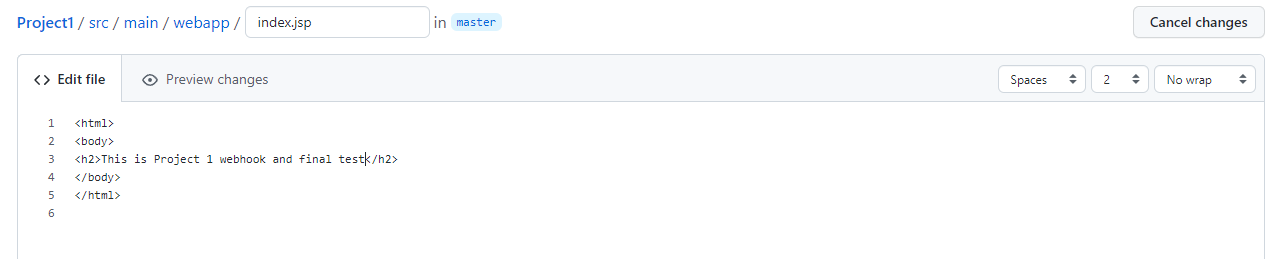


Start Jenkins server, check the node and open pipelines

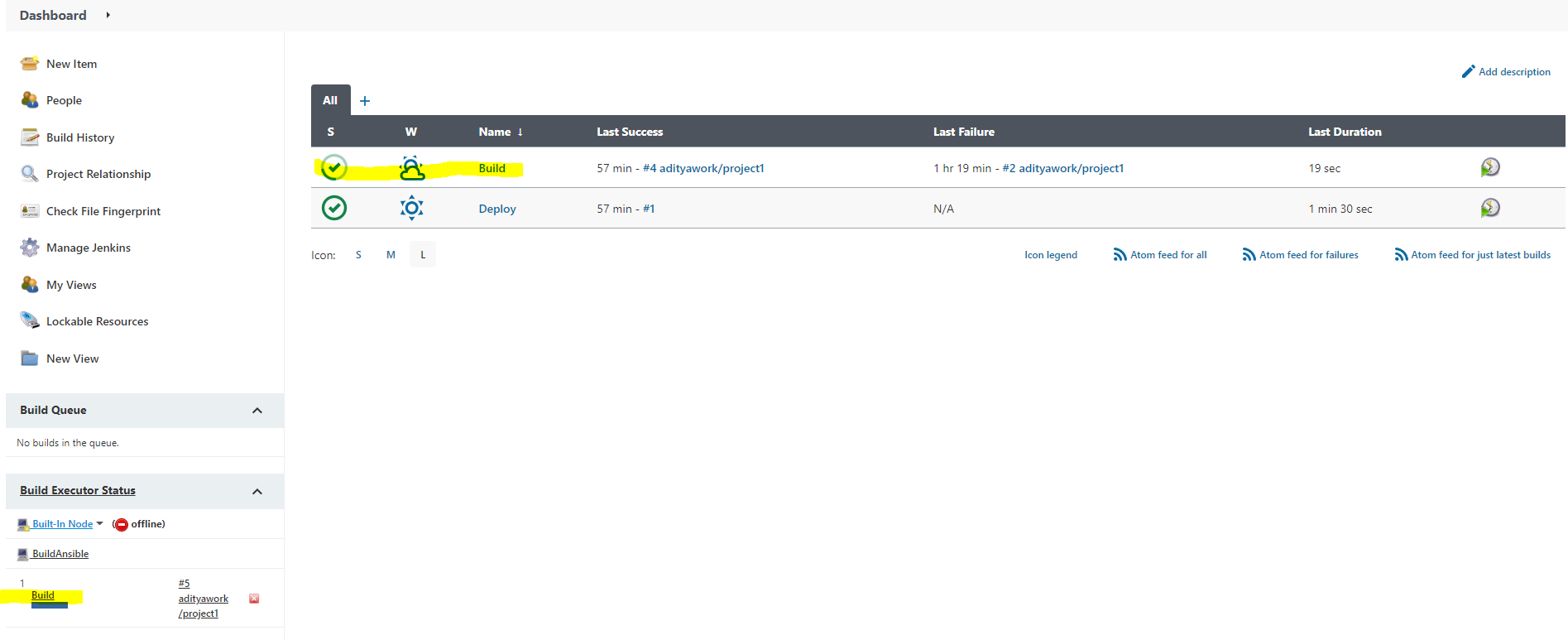


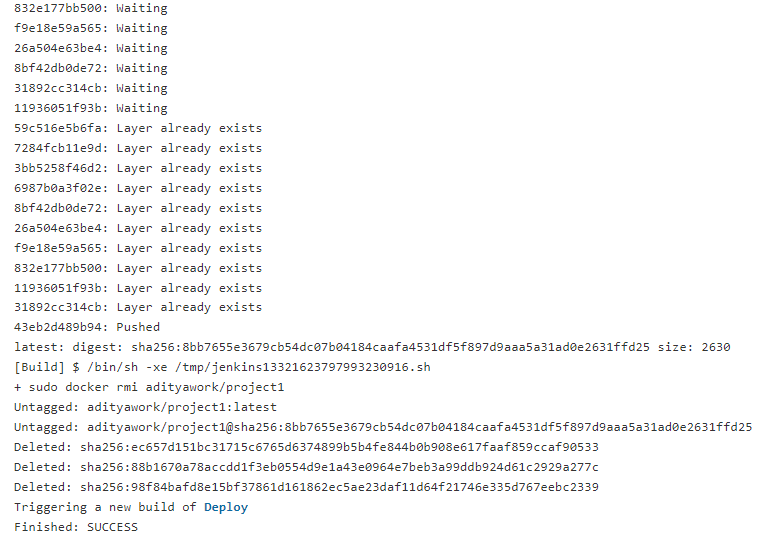


Now make a change in GIT repository to index.jsp and commit the changes

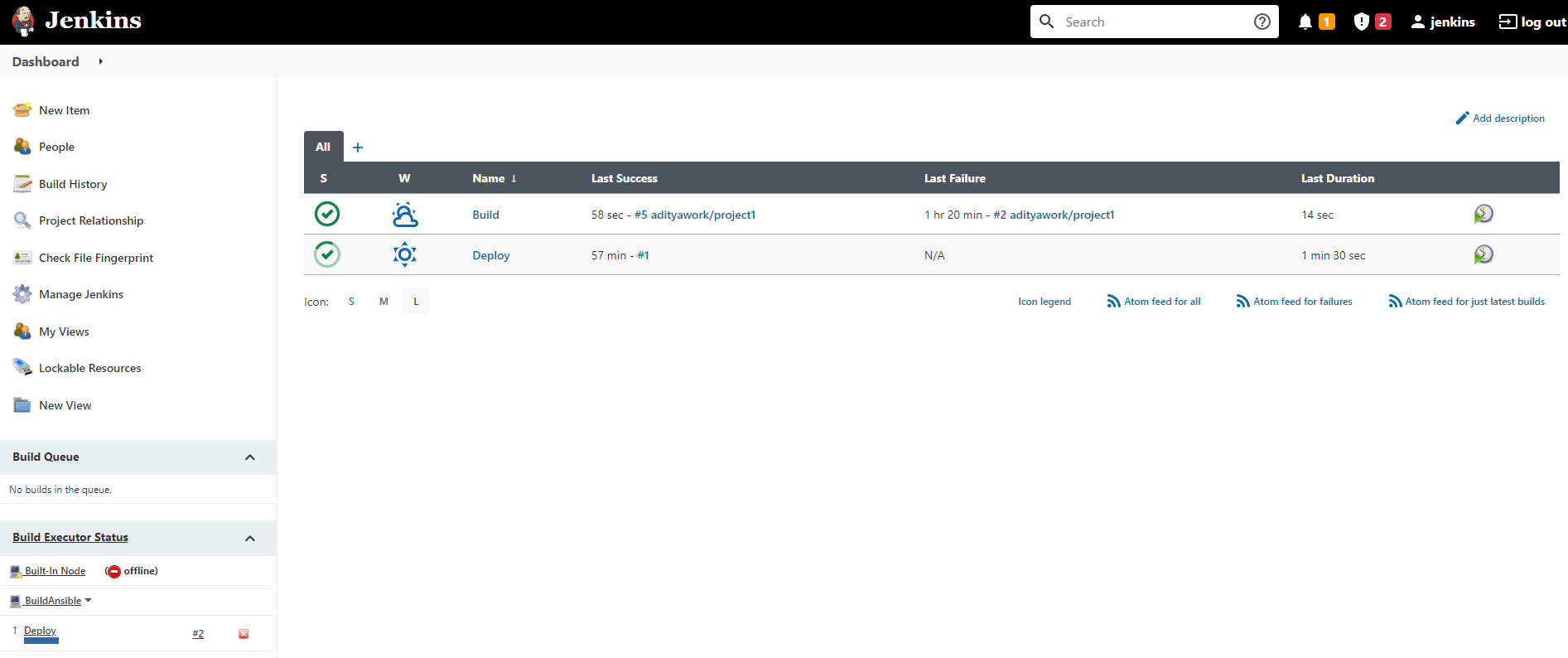


Once the commit is done, Build pipeline is triggered automatically on jenkins

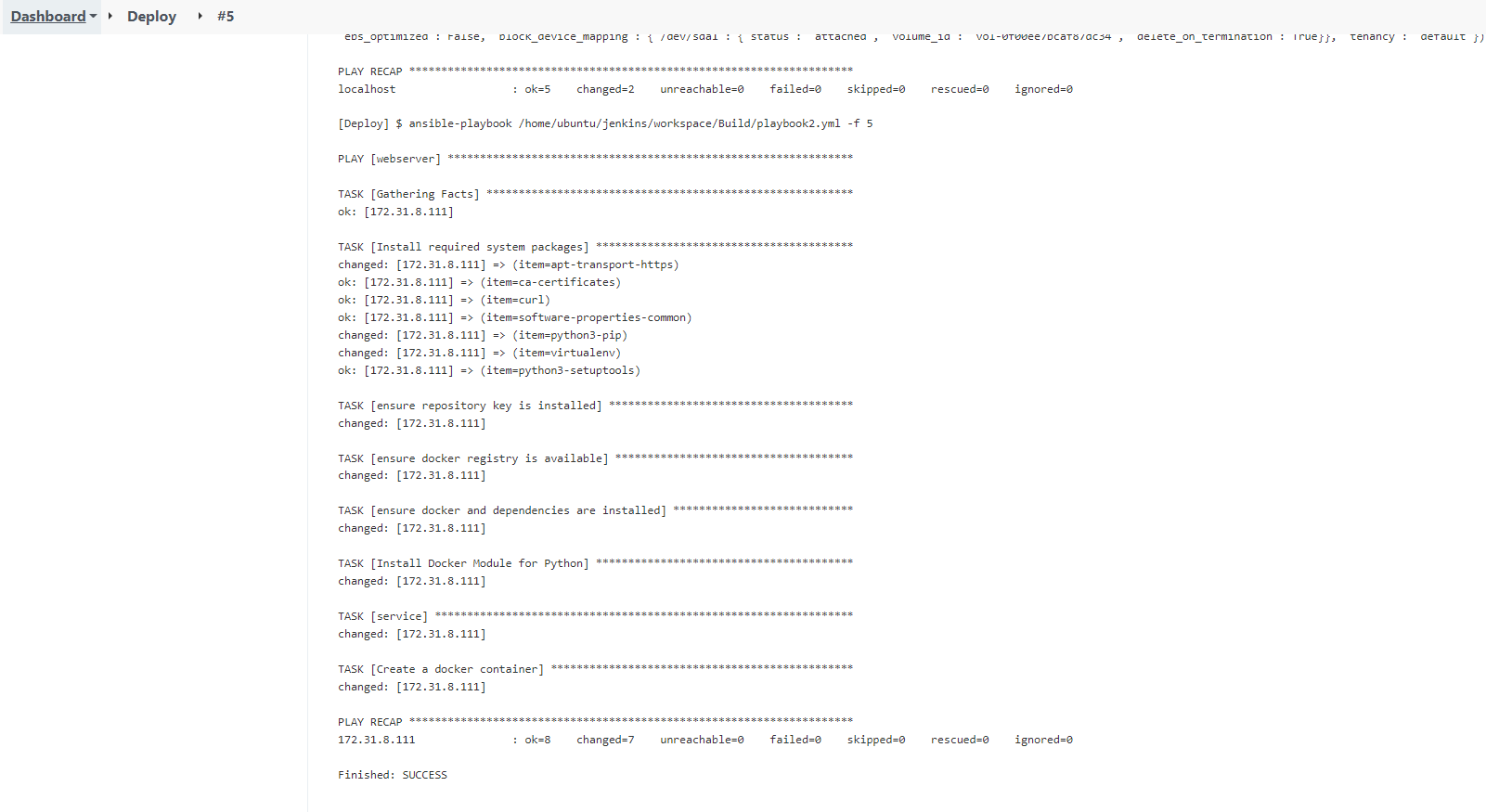




Build pipeline triggers Deploy pipeline as shown below



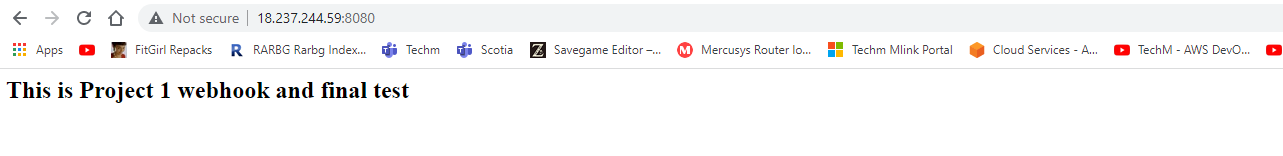
Deploy pipeline ran successfully



EC2 instance created

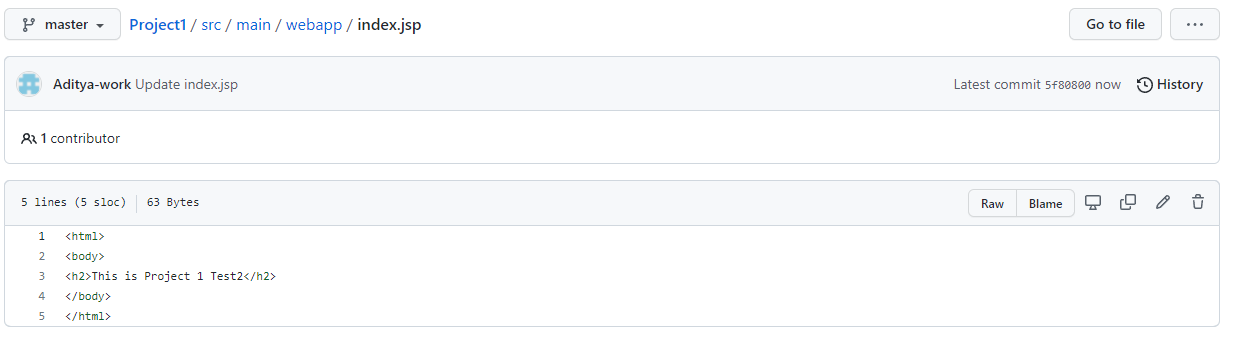


Able to open the page using public ip.



**Second Time Test**

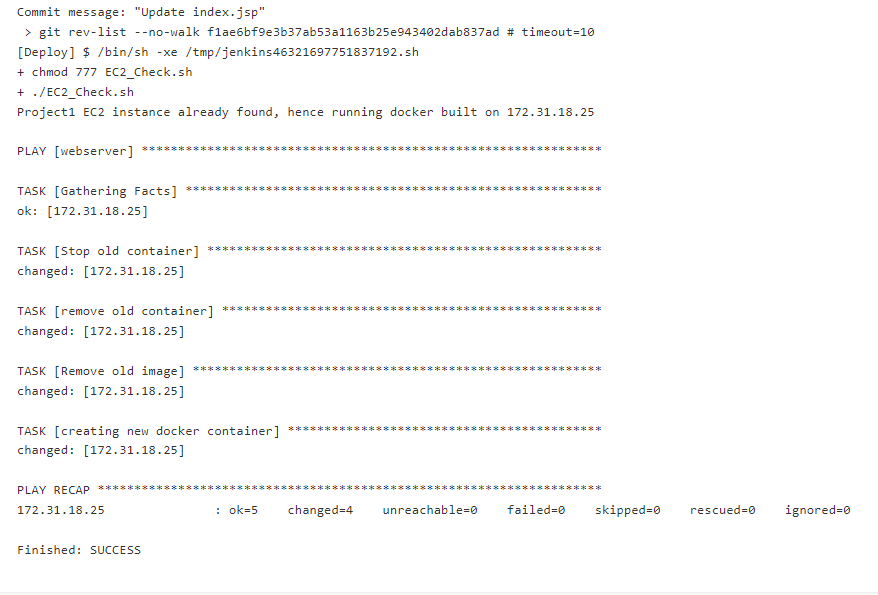
Edit Index.jsp and commit changes



Build is successful



Deploy pipeline calls EC2\_Check.sh script and it finds Project1 EC2 instance is already running, hence instead creating new EC2 instance it uses same Project1 server from Test1 and will deploy the image.



Webpage is updated and the test is successful.

