PG DO - DevOps Capstone Project

Project -3

Tested By Kalaivani Ravishankar

Vani212008@gmail.com

https://github.com/vanishankar/Newrepo.git

Create a CI/CD Pipeline to convert the legacy development process to a DevOps process.

Introduction:

In this project I will apply the skills and knowledge which were developed throughout Post Graduate Program in DevOps

- **Working in AWS**
- **Docker**
- **Git, GitHub,**
- **Jenkins,**
- **Tomcat Apache and**
- **Maven**

Application:

The application I used sample java application. I used ec2 instance to create Docker server and Jenkins server

Tools used:

1. **EC2**
2. **Jenkins**
3. **Docker**
4. **Git**

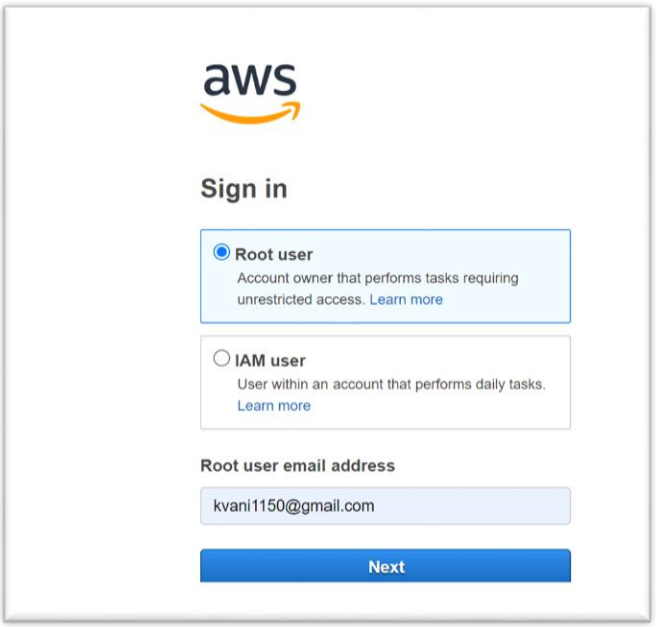
Steps to create ec2-instance:

Step 1: Login to the aws console: -

I use this link https://console.aws.amazon.com/ec2/ to login to aws console account.

Select Root User and I gave email address to login into aws console

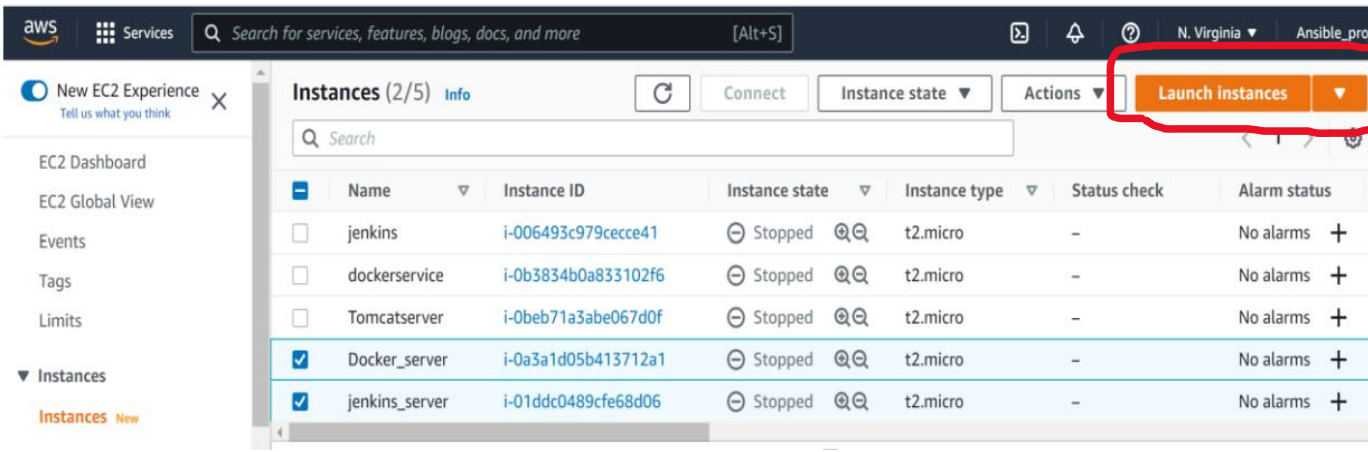
aws sign in page:



Step 2: Choose Launch Instance

First of all, click on 'Launch Instance' button shows in the below picture for launch/create new ec2 instance in aws:

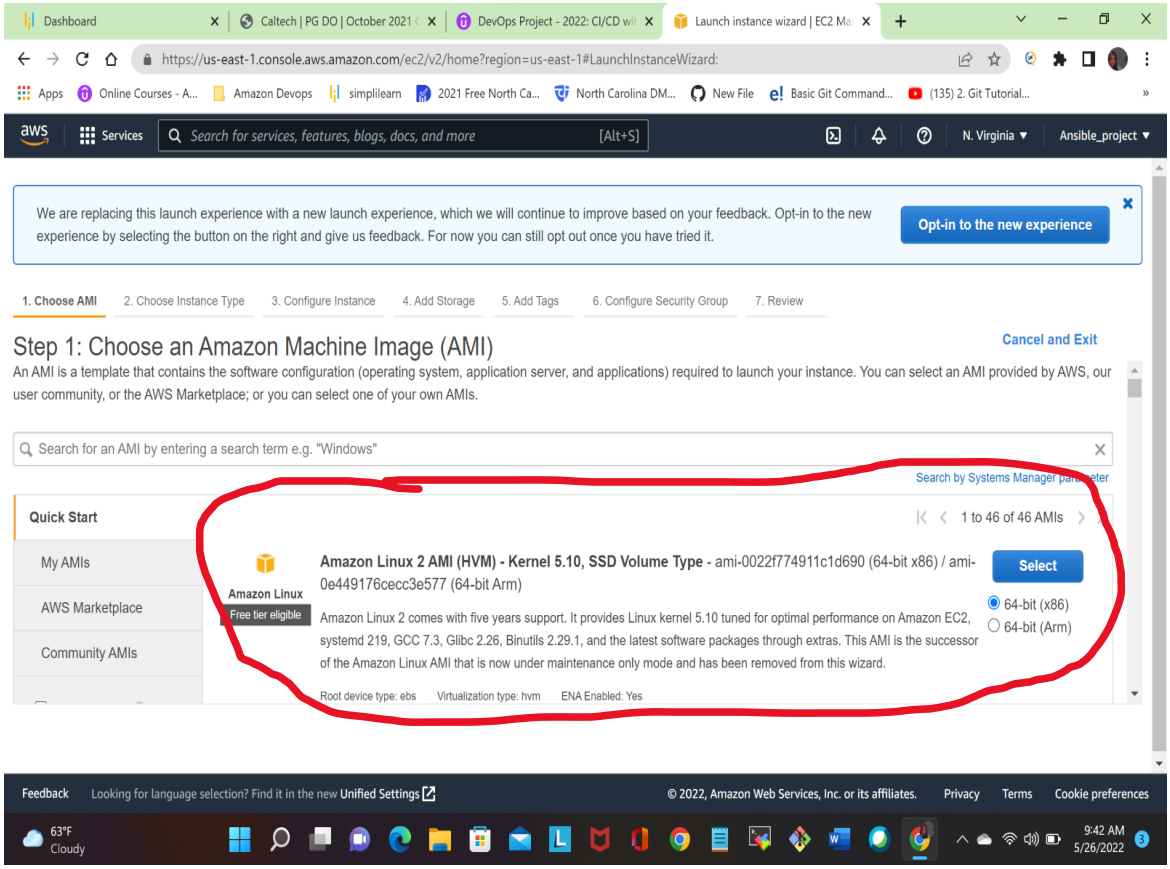
Launching ec2:



It will Lauch new instance

Step 3: choose AMI

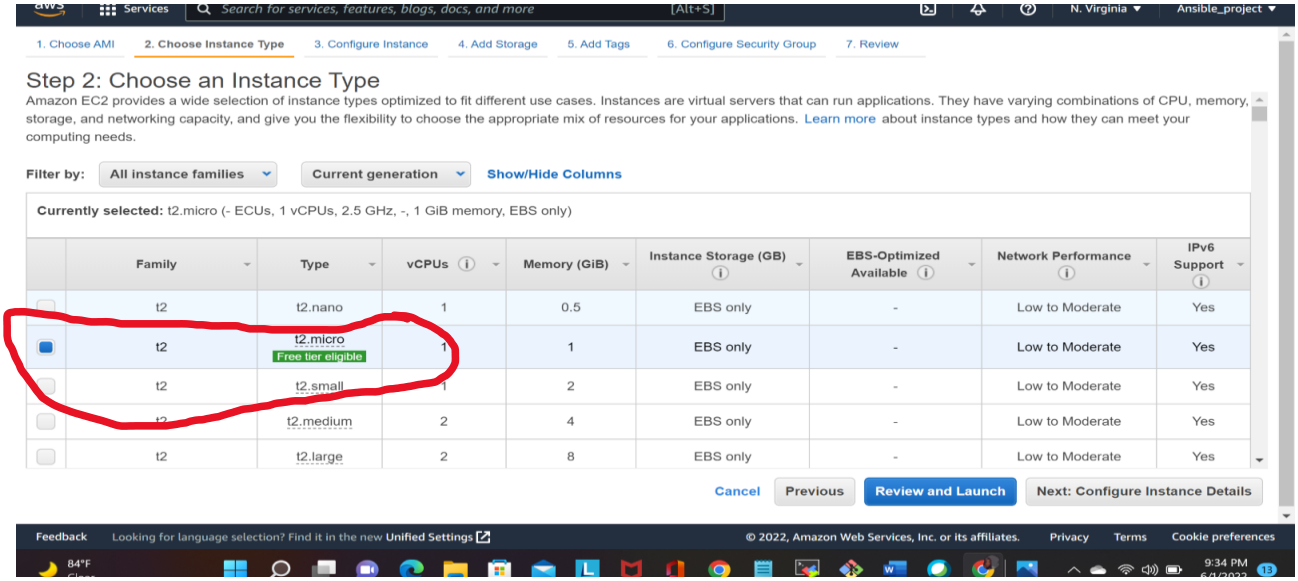
In this step, choose AMI (I chose Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type - ami-0022f774911c1d690 (64-bit x86) / ami-0e449176cecc3e577 (64-bit Arm)



I chose Amason Linux 2 Ami(hvm) and click select button

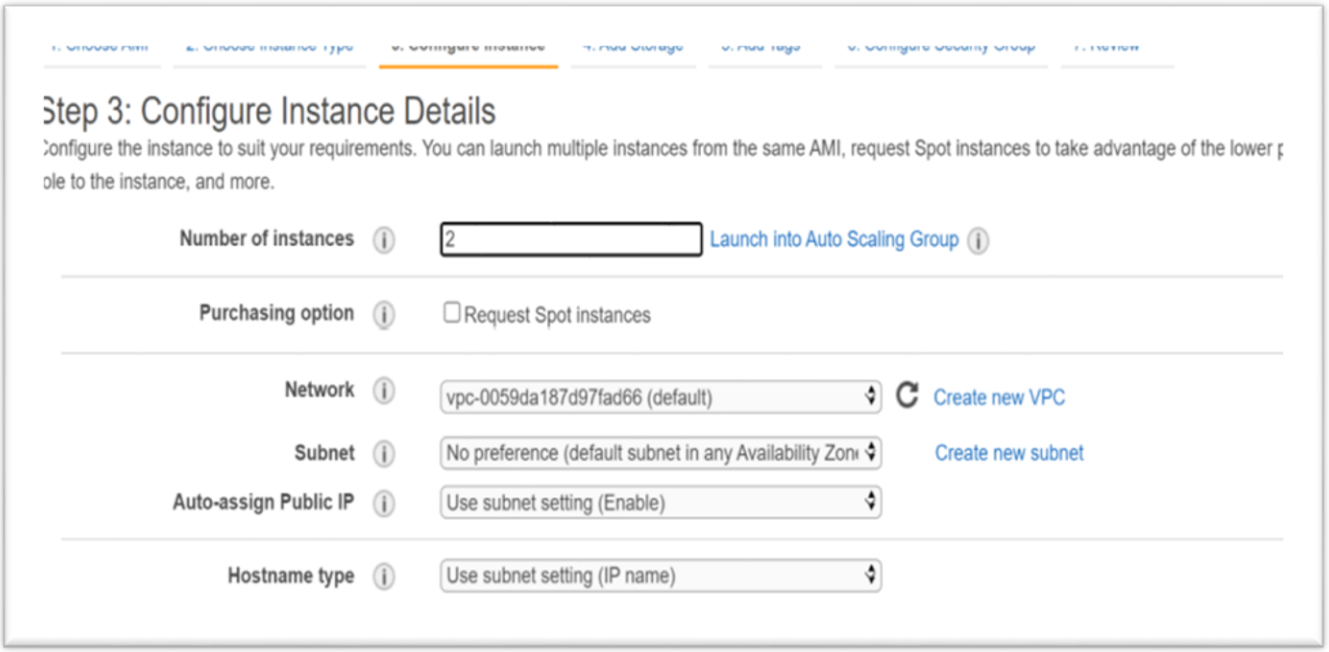
Step 4: Choose EC2 Instance Types

In this step, choose ec2 instance type shown in below picture:



I selected t2.micro and click configuration instance detail

Step 5: Configure Instance Detail:



In this step I selected number of instances is two and leave remaining as its

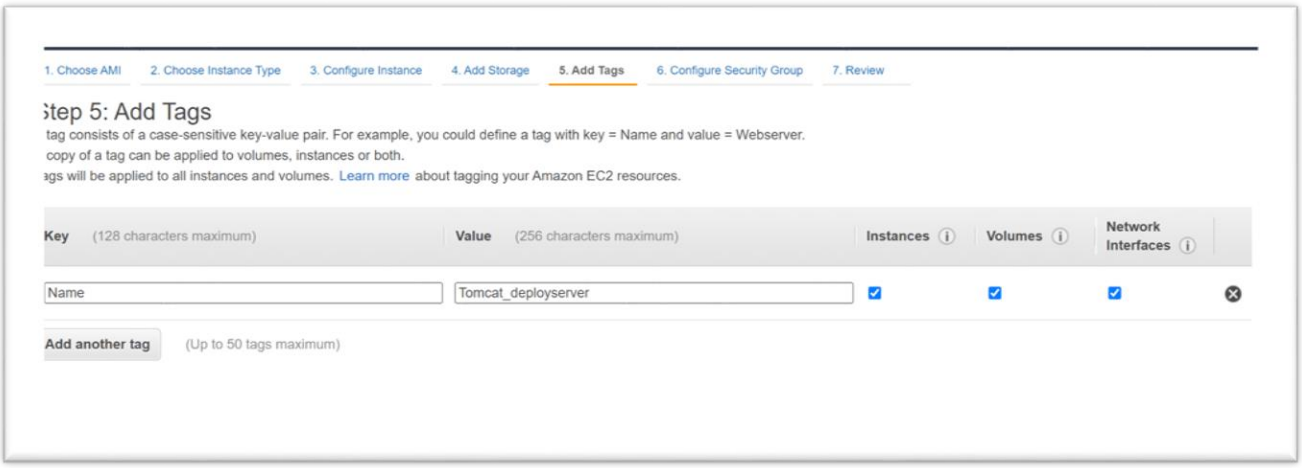
Step 6: Add Storage of Ec2 Instance

In this step, I chose storage. By default, ec2 t2-micro provide 8gb ssd. Shown in the below picture:

Step 7: Tag Instance of Ec2 Instance

In this step, I added tag of instance with a key-value pair

Key value pair shown in fig

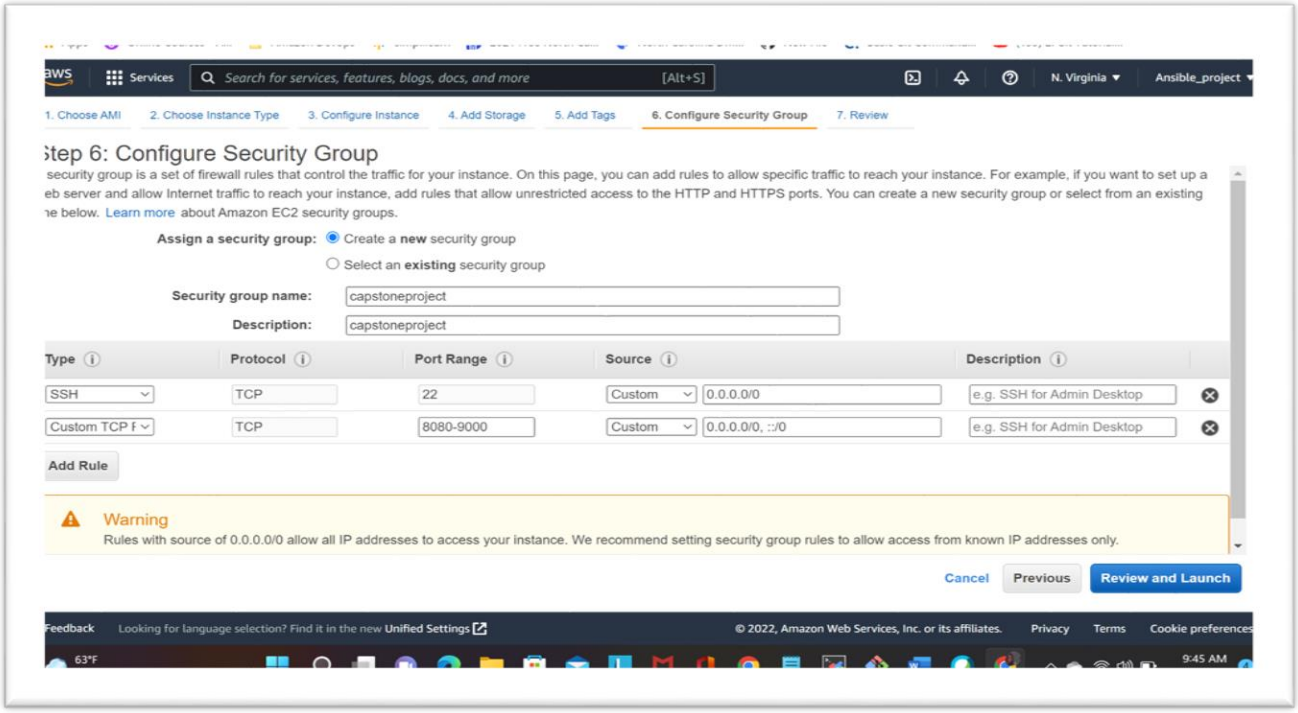


Key -----> name

Value --->Tomcat_deployserver

Step 8: Configure Security Groups

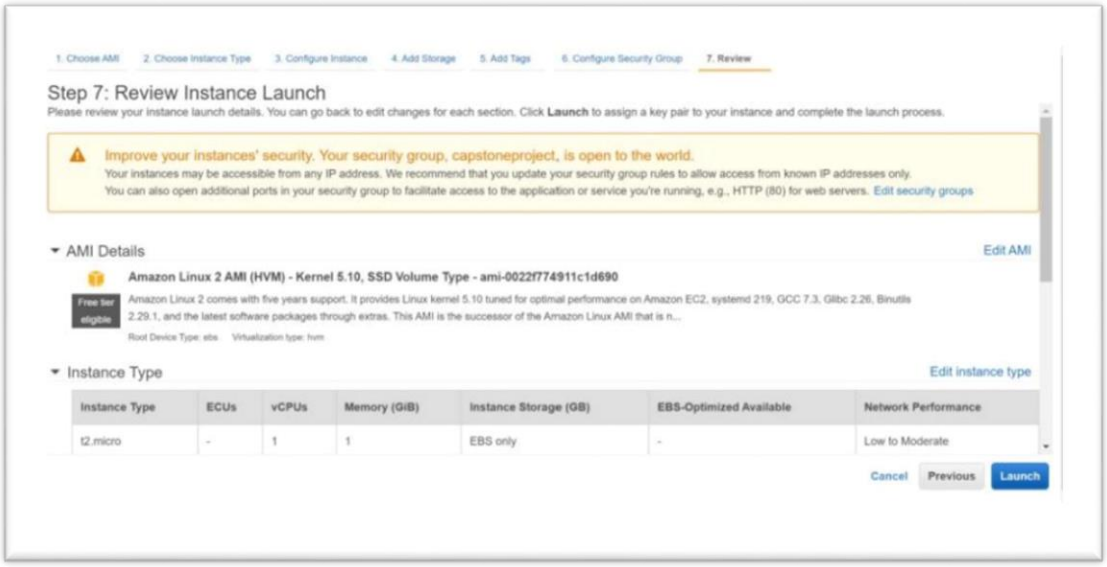
In this next step of configuring Security Groups



I created security group called capstone project and Defining protocols which I want enabled on my instance

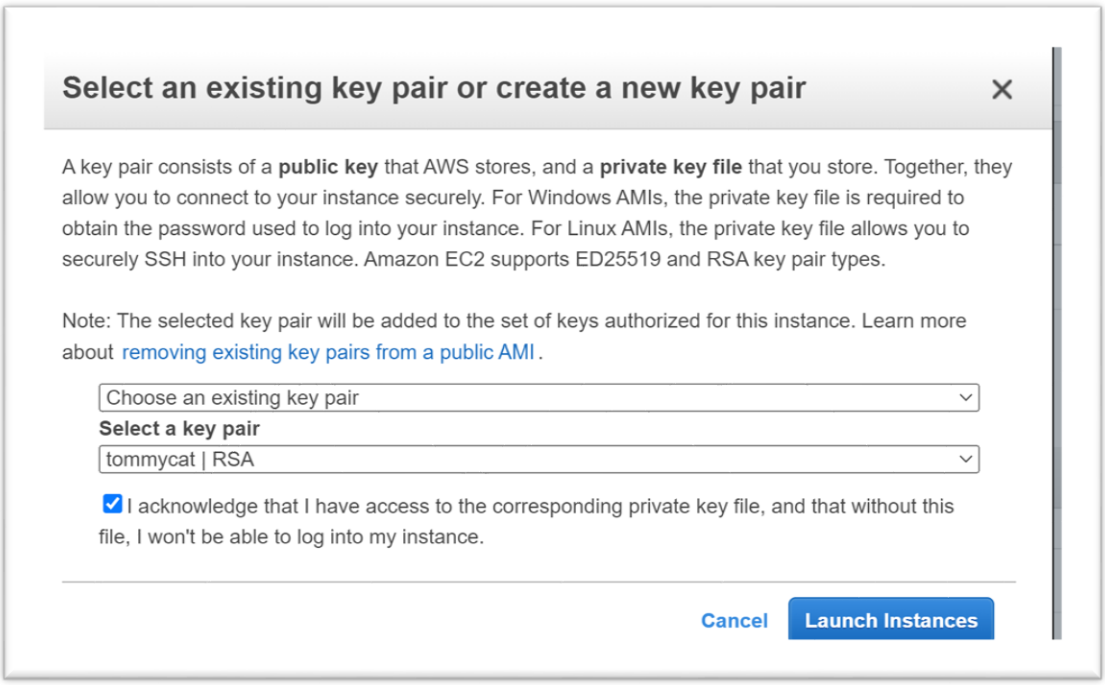
Step 9: Once, the firewall rules are set- Review and launch

Shown in below fig



After clicking launch button it will take into creating keypair page

Step 10: Create Key-Pair for Instance Access: Creating a keypair for ec2 instance, (I used existing keypair) create new key pair and add the name of this key. Then download and save it in your secured folder. Shown in below fig.



And finally click launch instance. It will launch 2 instances

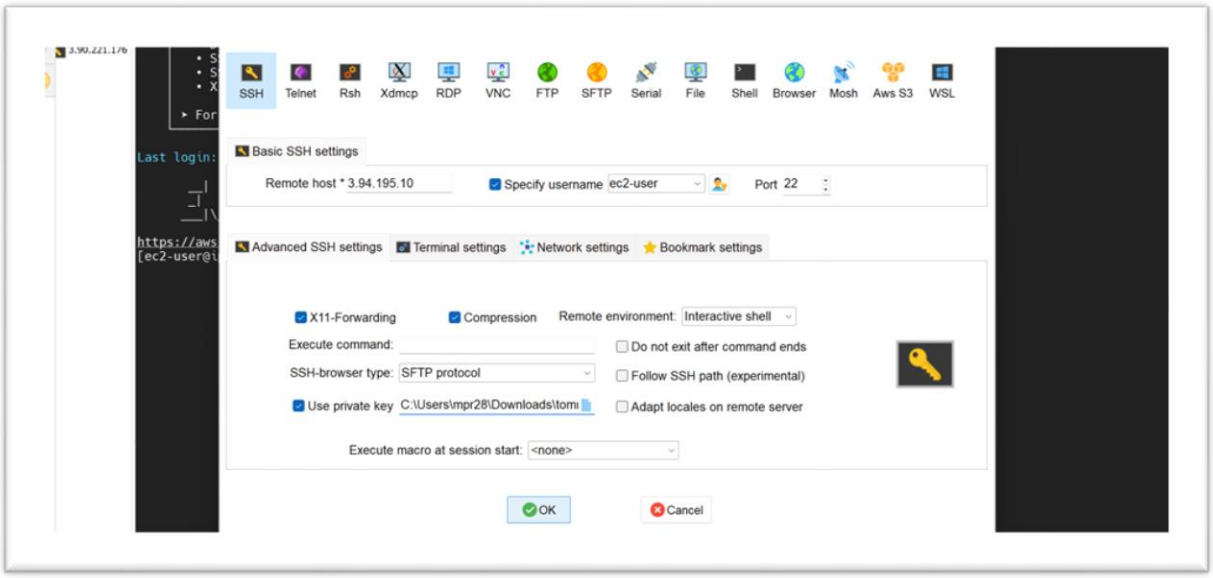
Next, we are going to connect our using mobaxterm to connect ec2 instance using pem file

Use pem file in MobaXterm to connect ec2 machine in AWS: -

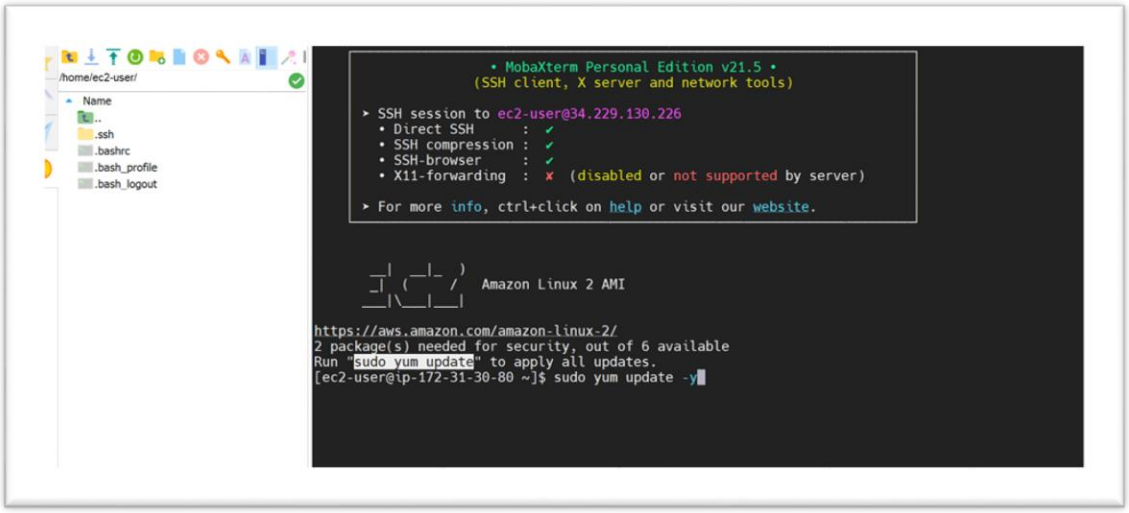
I used MobaXterm to connect ec2 machine. I downloaded mobaxterm .go to the session settings to create a new session, fill up the remote host and username.
And click advance setting and select use private key, select your private key.

It will help to connect to ec2-instance. As shown in the below fig:10

Configuring mobaxterm:-



After clicking ok it taking into page shown in below fig

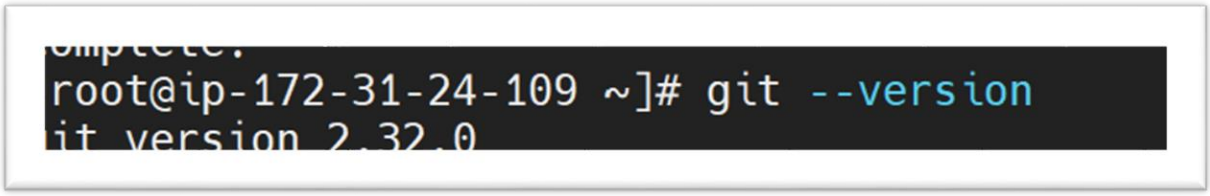


I created two ec2 instance machine (Jenkins and Docker server)

Install Git on Docker_ server and Jenkins server

Using following command

yum install git –y

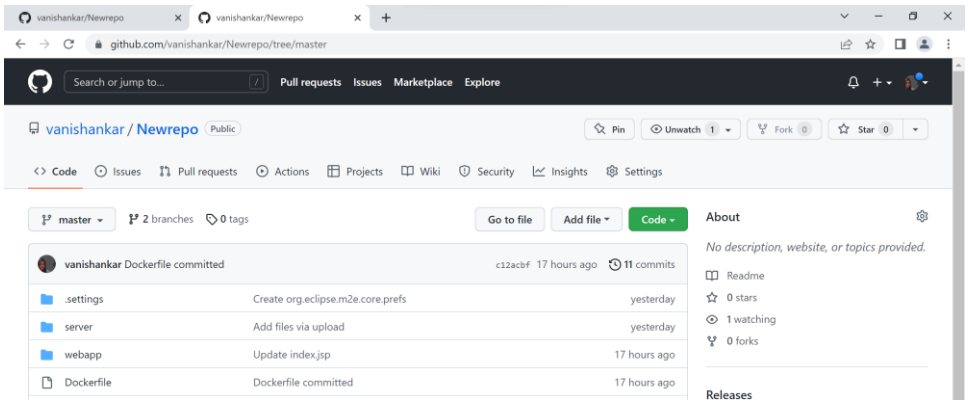


Git will be successfully installed in both server

GitHub Repo

• Create a GitHub project and check in the code of Aetna project. Also, we will be checking in the Docker file.

• The link to the public GitHub repo is
https://github.com/vanishankar/Newrepo.git

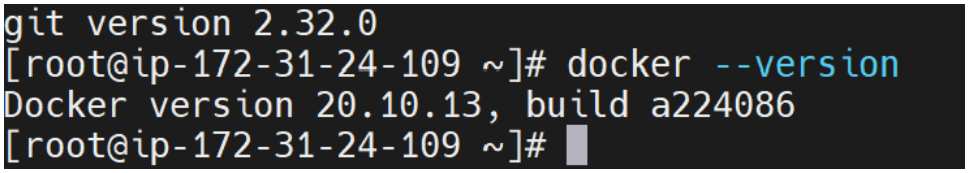


Installing Docker:

Command to install docker

Yum install docker.

The following pic shows docker version



Command to start docker and check the status:

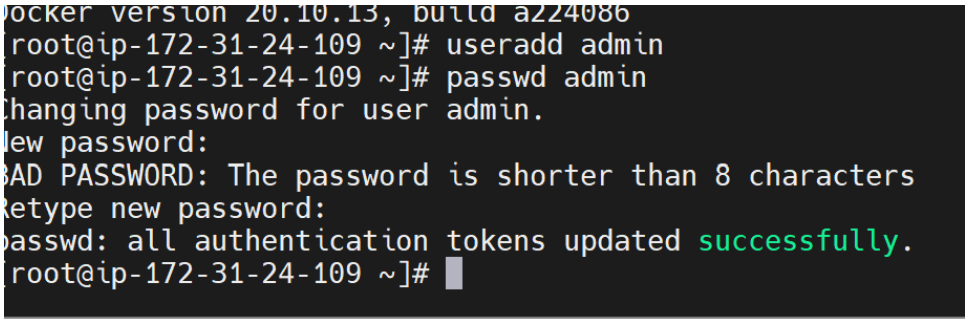
service docker start
service docker status

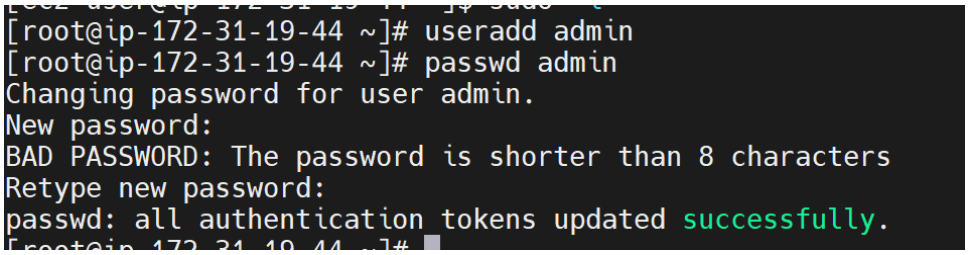
Creating user:

I created user called admin in both the server using following commands

useradd admin

passwd admin (Shown in below fig)





Add a user to docker group to manage docker: -

I added a user in docker group using following command

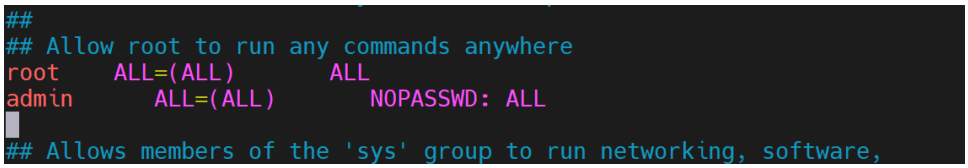
usermod -aG docker admin

Adding Users to Sudoers File:

Next step is to add admin user into sudoers file using following commands

sudo visudo

Shown in the following fig

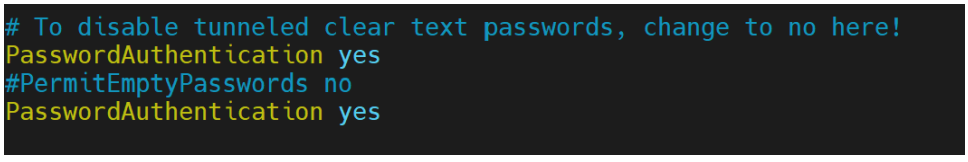


The place where you see `root ALL=(ALL:ALL) ALL`, that's where we'll be amending our user name. And next use following command to configure

sshd_config file to enable password less authentication, show in the following

Enable Password Authentication in below File Path

vi /etc/ssh/sshd_config. uncomment the password authentication and change **no** to yes shown **in following fig**



Reload Service with the command

service sshd reload

```
[root@ip-172-31-24-109 ~]# visudo
[root@ip-172-31-24-109 ~]# vi /etc/ssh/sshd_config
[root@ip-172-31-24-109 ~]# service sshd restart
Redirecting to /bin/systemctl restart sshd.service
[root@ip-172-31-24-109 ~]#
```

And I followed same steps in Jenkins server to add user and enable password less authentication.

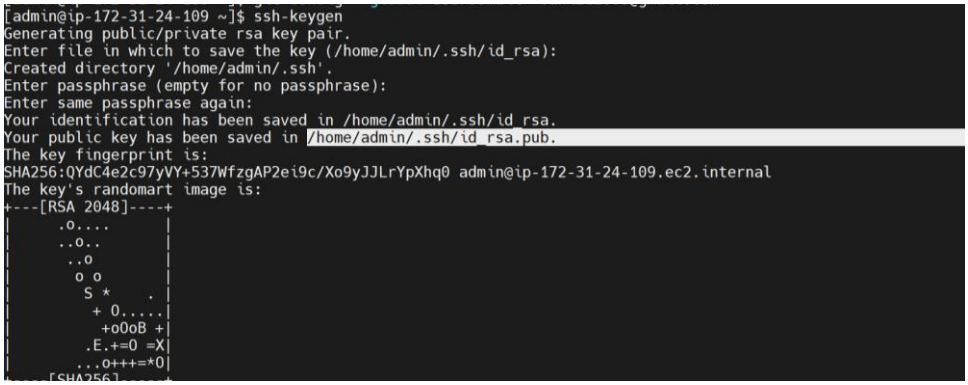
Next steps are to create a connection between docker machine and Jenkins by downloading sshkey ,using following command .login as an admin using following command

sudo su – admin

And generate sshkey using command ssh-keygen

After generating I copied the public key to Jenkins server using following command ssh-copy-id

admin@172-31-24-109 (ssh-copy-id user@<target-server> - Authorized_keys)



And next step is to configured Jenkins server to install Jenkins, Maven, java

JAVA

http://openjdk.java.net/install/

Install Java in Linux

yum install `java-1.8.0-openjdk-devel`

Java is installed in usr/lib directory

Set Java Home Path in Bash profile

JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.amzn2.0.2.x86_64

PATH=$PATH:$HOME/bin:$JAVA_HOME:

Install maven

Download maven packages https://maven.apache.org/download.cgi onto Jenkins server. In this case, I am using /opt/maven as my installation directory.

I used wget command to download maven (wget is used to download files from an HTTP/HTTPS or FTP server)

wget https://dlcdn.apache.org/maven/maven-3/3.8.5/binaries/apache-maven-3.8.5-bin.tar.gz

Next step is to extract the archive using following command

tar xzvf apache-maven-3.8.5-bin.tar.gz

Maven home path in bash profile

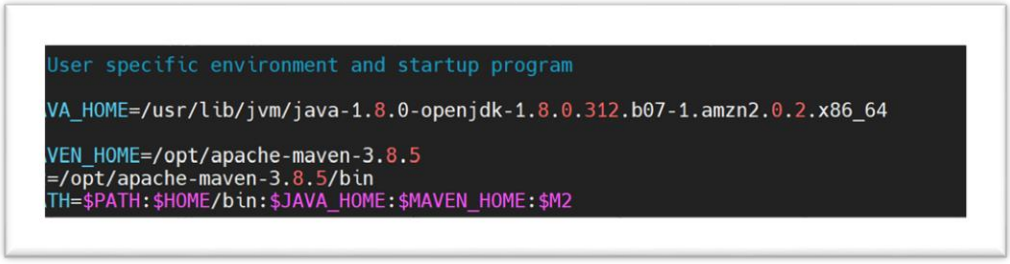
JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.amzn2.0.2.x86_64

MAVEN_HOME=/opt/apache-maven-3.8.5

M2=/opt/apache-maven-3.8.5/bin

PATH=$PATH:$HOME/bin:$JAVA_HOME:$MAVEN_HOME:$M2

The following fig shows maven and java path in .bash_profile



Install Jenkins:-

Install Jenkins using the rpm or by setting up the Repo

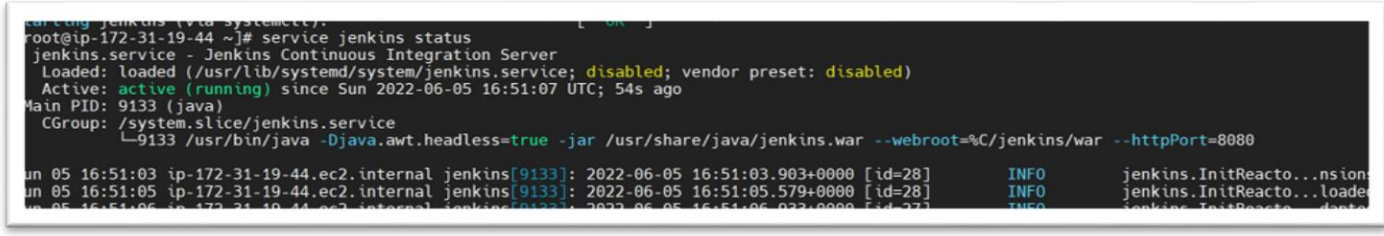
I downloaded the latest Version of Jenkins form https://pkg.jenkins.io/redhat-stable/ and install

sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
amazon-linux-extras install epel

After downloading Start the Jenkins Services using following command

Service jenkins start

(Shown in following Fig.)



Accessing Jenkins

By default, jenkins runs at port 8080, You can access jenkins at

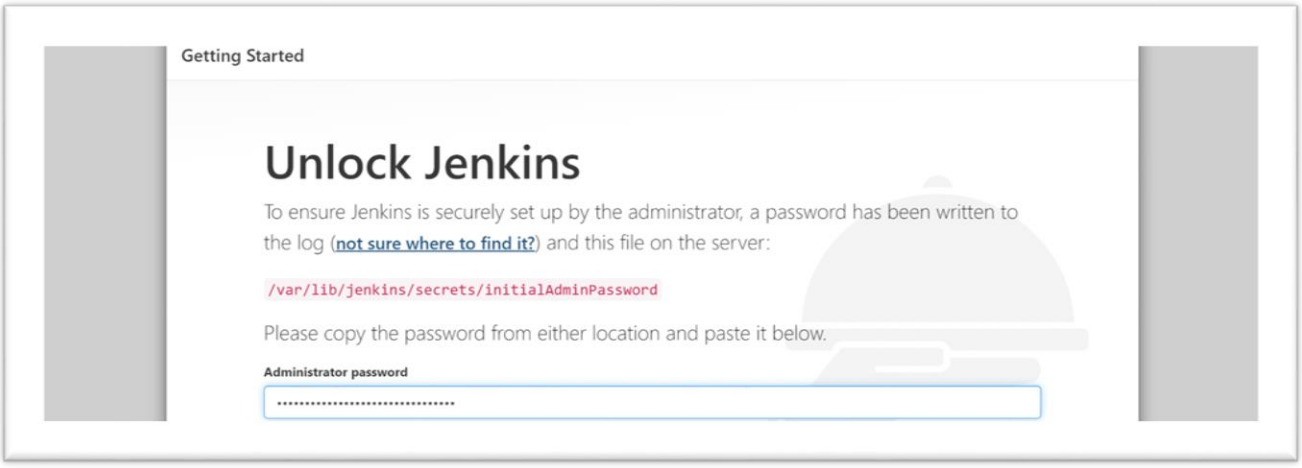
http://YOUR-SERVER-PUBLIC-IP:8080

Configure Jenkins:

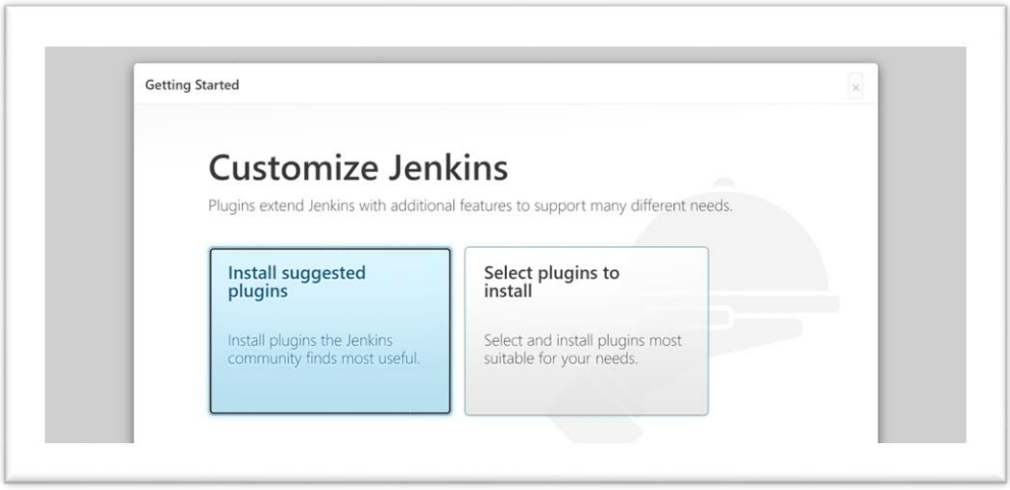
Grab the default password

Password Location:/var/lib/jenkins/secrets/initialAdminPassword, paste it

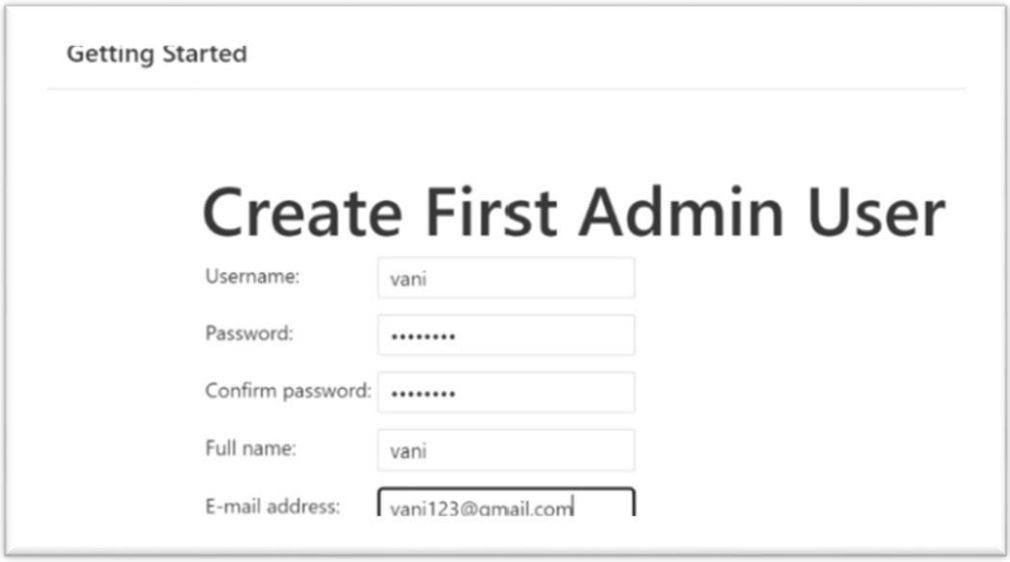
It is shown in following pic



Choose suggested plugins. Shown in below fig:23

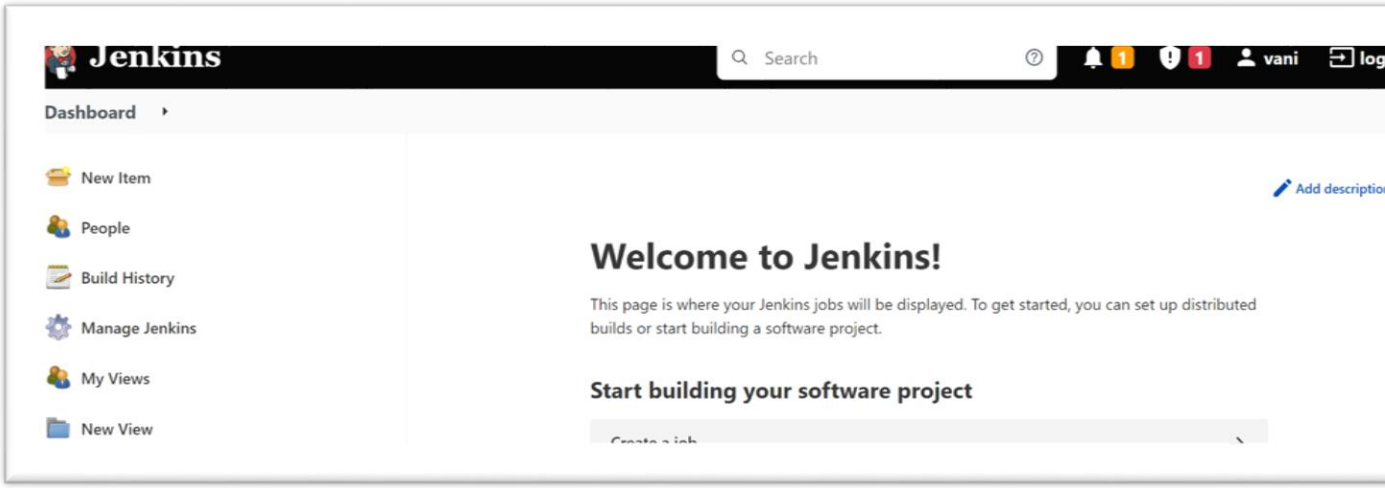


And enter username, password email address shown in below



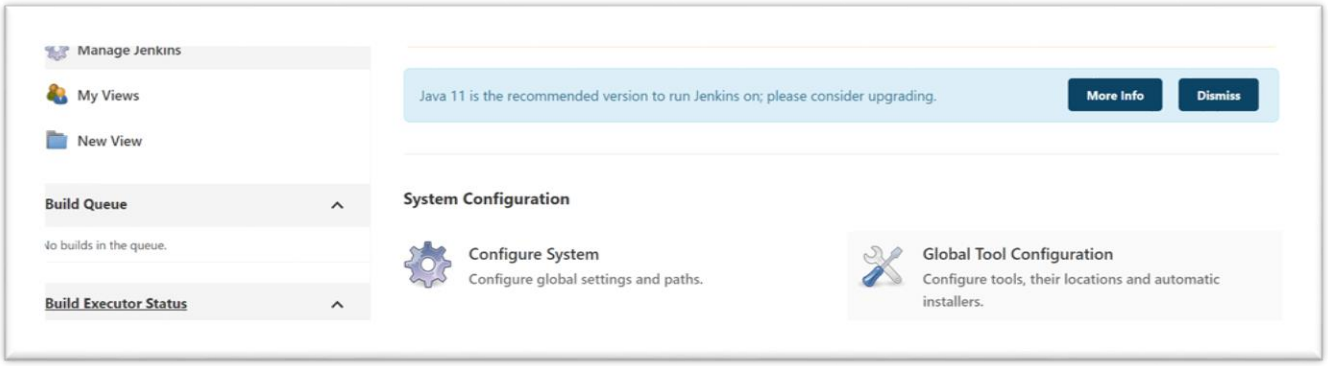
Click save and continue.it taken you into **jenkins dashboard**

Jenkins Dashboard:

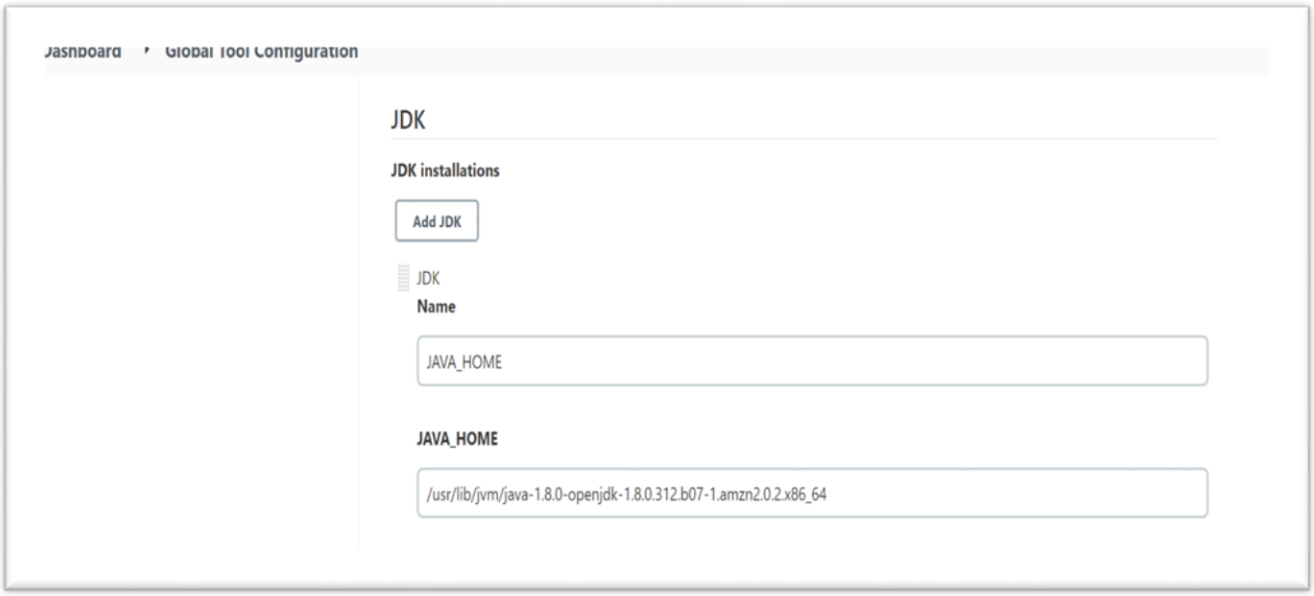


Configure java and maven path

Manage Jenkins > Global Tool Configuration > JDK

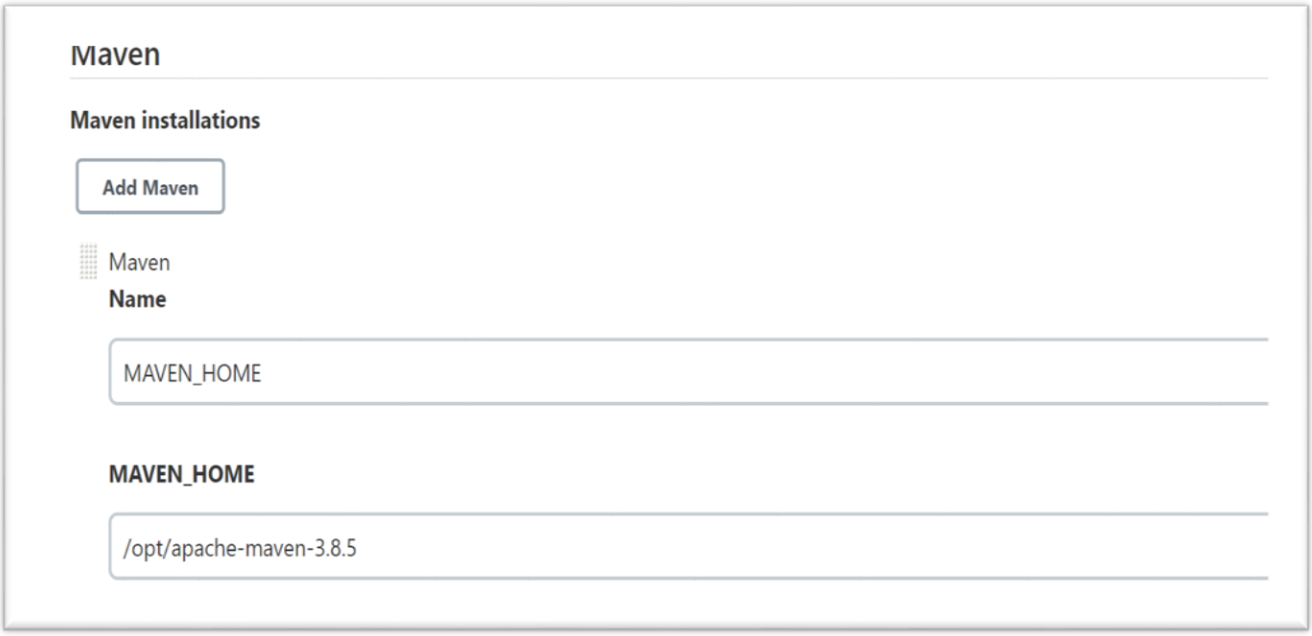


Click add JDK and give JDK name as JAVA_HOME and path of JDK



I gave path of jdk is **/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.amzn2.0.2.x86_64**

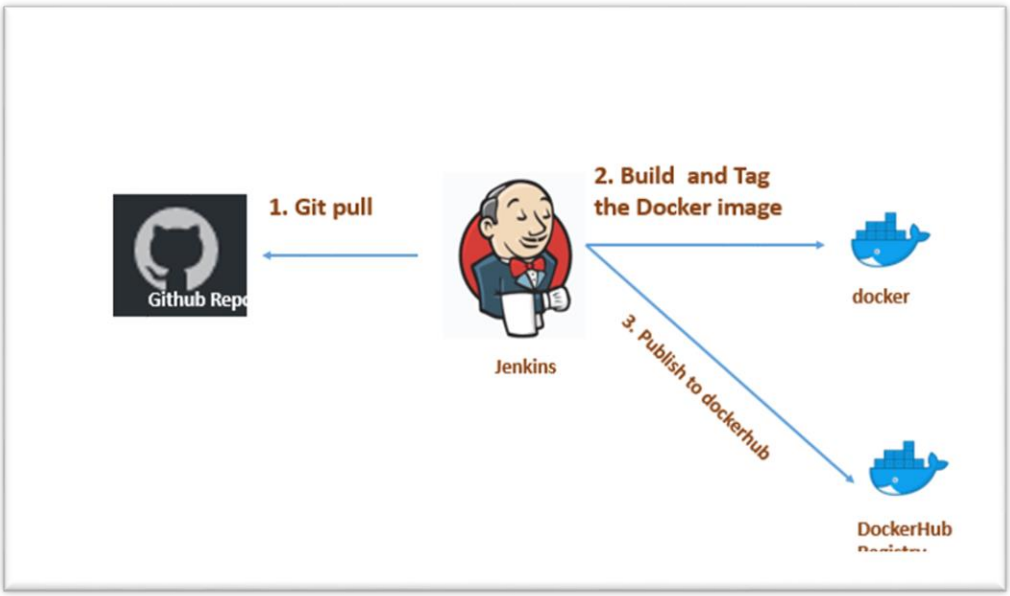
Next configure maven path, click maven give name for maven as MAVEN_HOME And path of maven



Maven _home /opt/apache-maven-3.8.5

After finishing all configure setting we are going to push the docker image into docker hub using jenkins.

Publish Docker image to Docker Hub using Jenkins: -



For creating docker image we need to write a **Dockerfile.** Go to the docker server, login as dockeradmin user using following command

sudo su – dockeradmin

Create a Docker file using vi editor

The Dockerfile added to the project will use a prebuilt tomcat image. Over this image, it will add the war file that was built inside the container to the tomcat webapps directory so that microservice api can be accessed through any web browser.

Dockerfile:

vi Dockerfile

FROM tomcat:latest

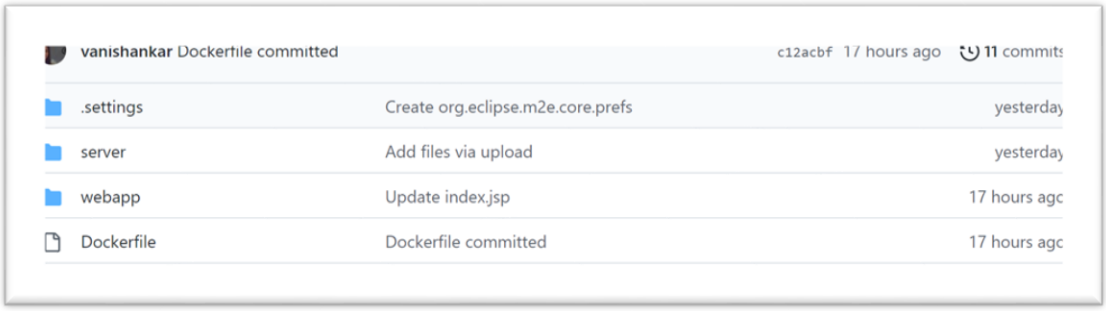
COPY ./webapp.war /usr/local/tomcat/webapps

RUN cp -r /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps

Dockerfile:-

```
FROM tomcat:latest
COPY ./webapp.war /usr/local/tomcat/webapps
RUN cp -r /usr/local/tomcat/webapps.dist/* /usr/local/tomcat/webapps
~
~
~
~
~
~
```

After docker file is created, I pushed into GitHub



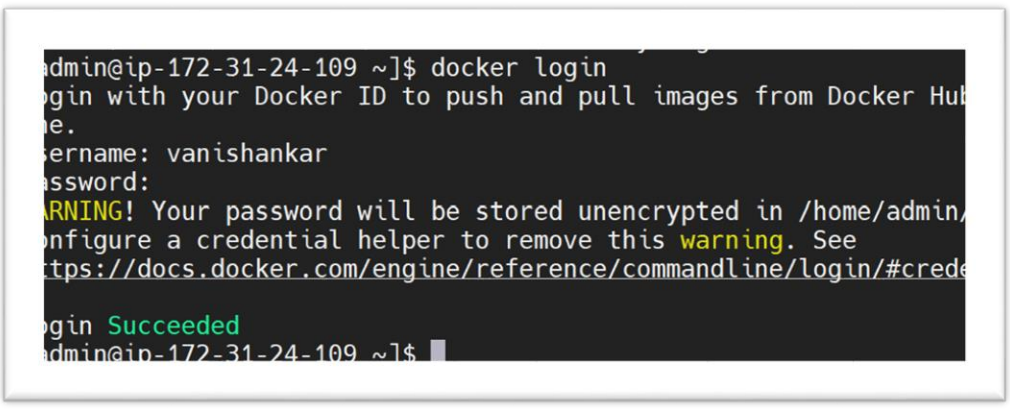
LOGIN TO DOCKERHUB

Next **login to docker hub** using following command

docker login,

I Gave username and password

It will allow us to login to docker hub to push image Shown in fig

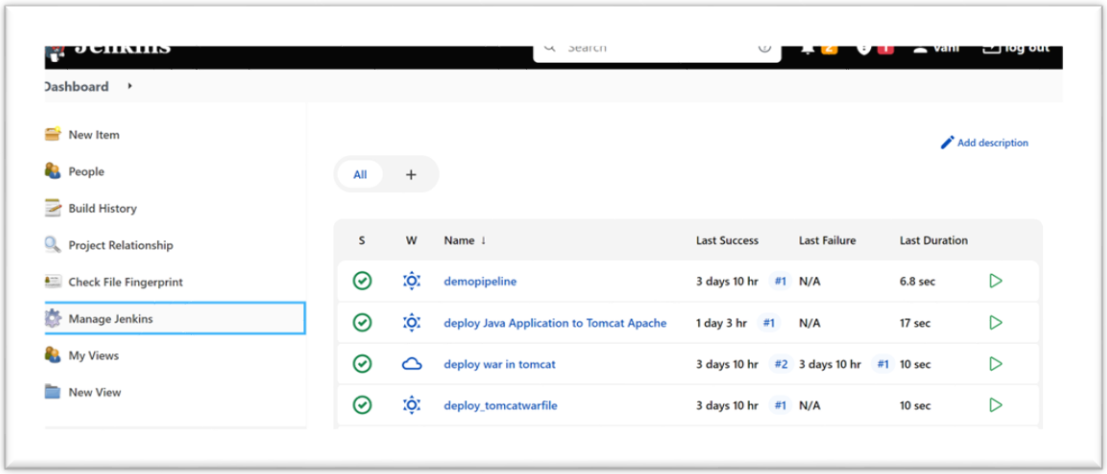


After I moved to browser (browse http://YOUR-SERVER-PUBLIC-IP:8080) And configure jenkins

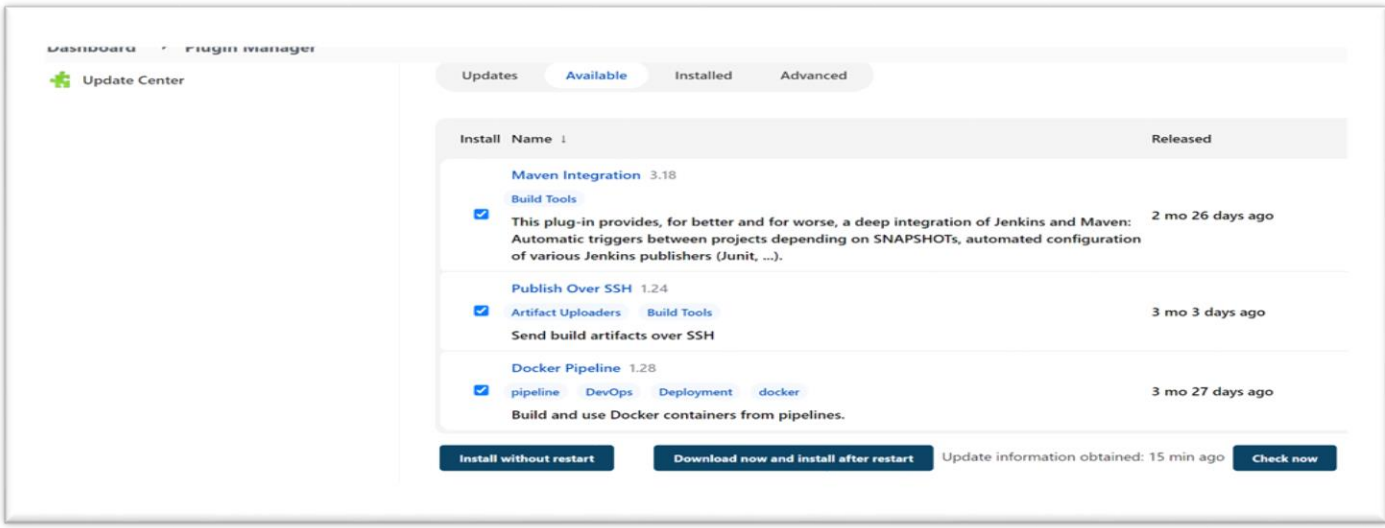
Setting Up Environment in jenkins

Install Maven Integration plugin and publish over ssh

This plugin is required to used integrate maven (maven integration) and publish over ssh is used to Send build artifacts over SSH.
Click on Manage Jenkins -> Manage Plugins-> search for maven integration and publish over ssh -> Install it without restart. Shown in below fig

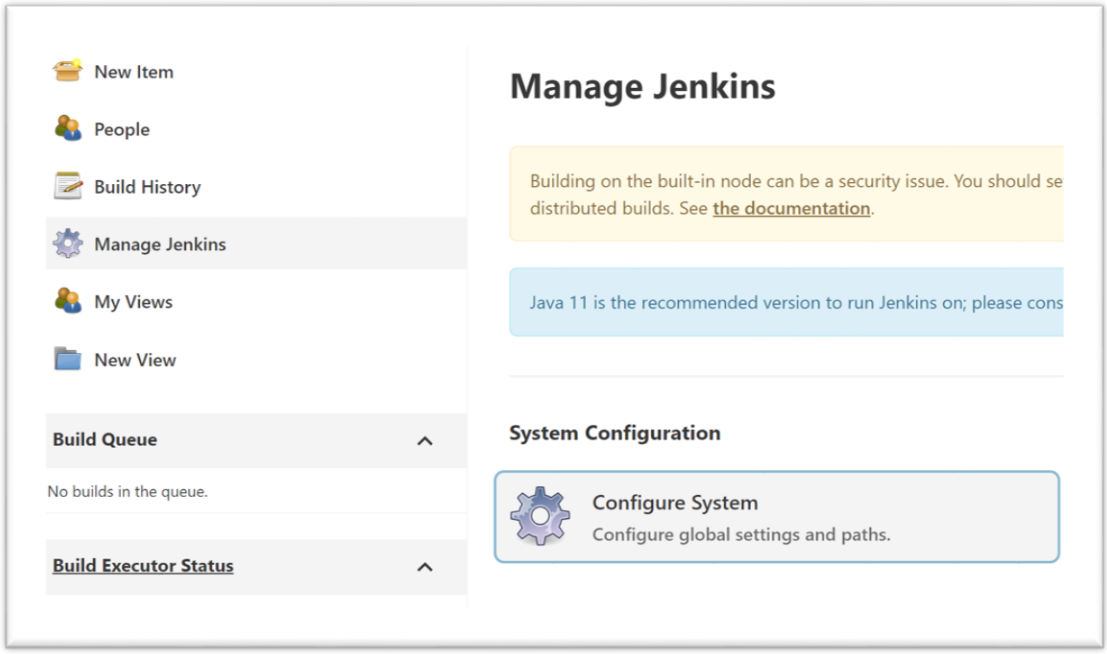


Plugins Manager:



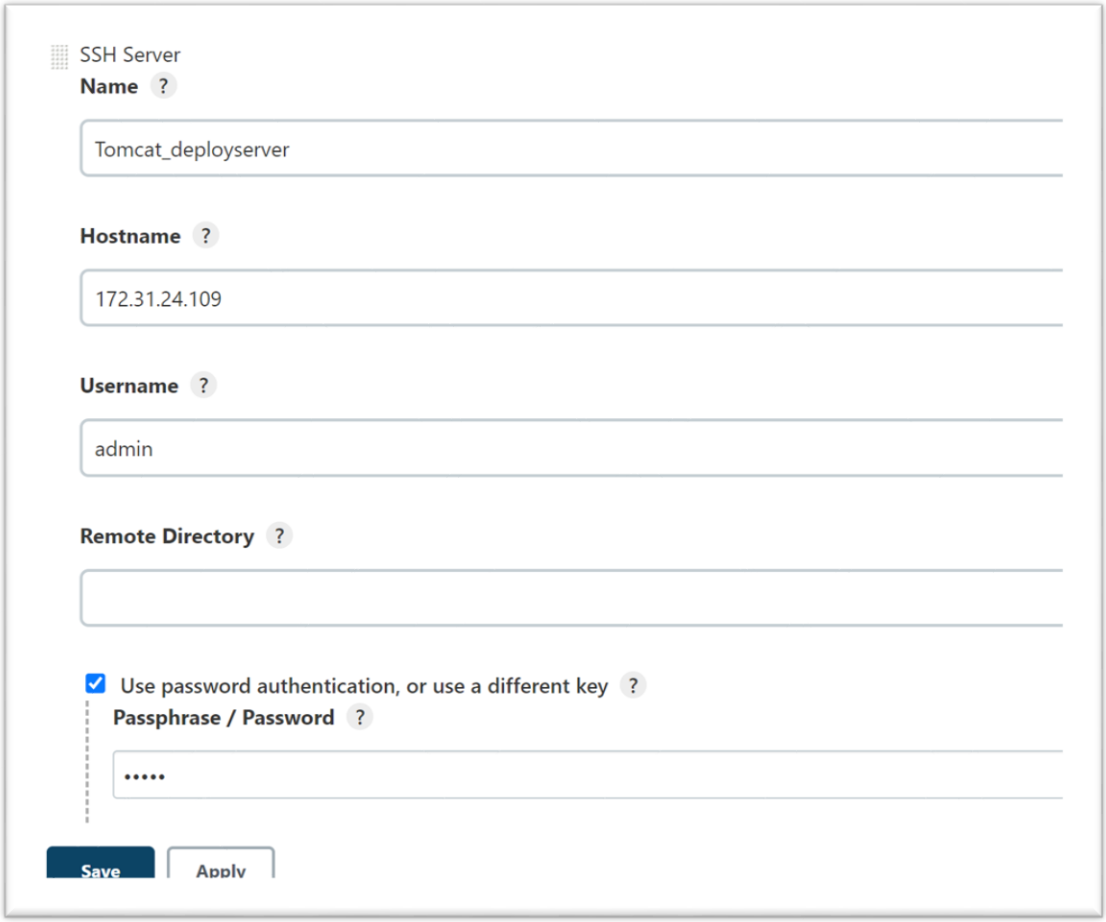
Click on **Manage Jenkins ->system configuration-> ssh server** and configure all the details shown in foll fig.

System configuration



Configure publish over ssh

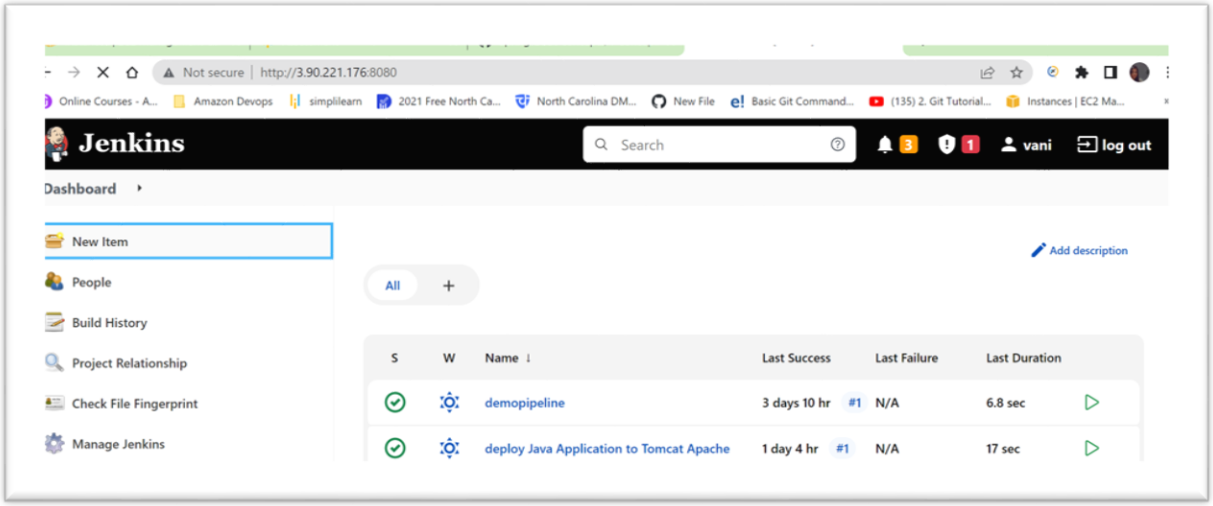
I gave name for server is Tomcat_deployer,hostname is private Ip address of docker machine, then I gave username as admin which I already created and password which help to connect user



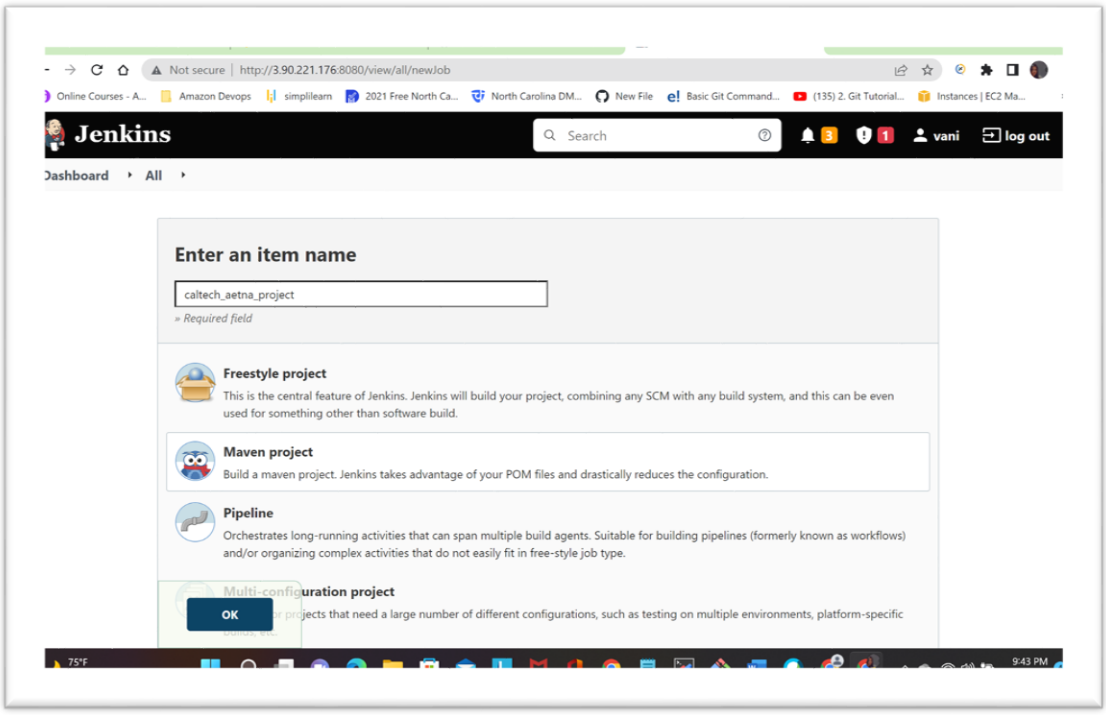
Once I finished all configuration. Next, I created a new job for pushing docker image using docker hub.

CREATING A NEW JOB

Navigate to dashboard click new item



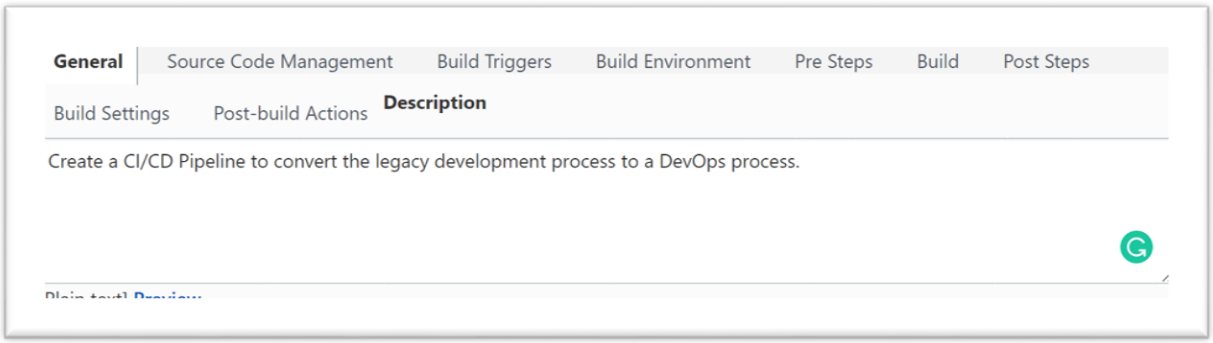
It will navigate you to the following screen:



Click on Maven Project and I gave a name for project(caltech_aetna_project).

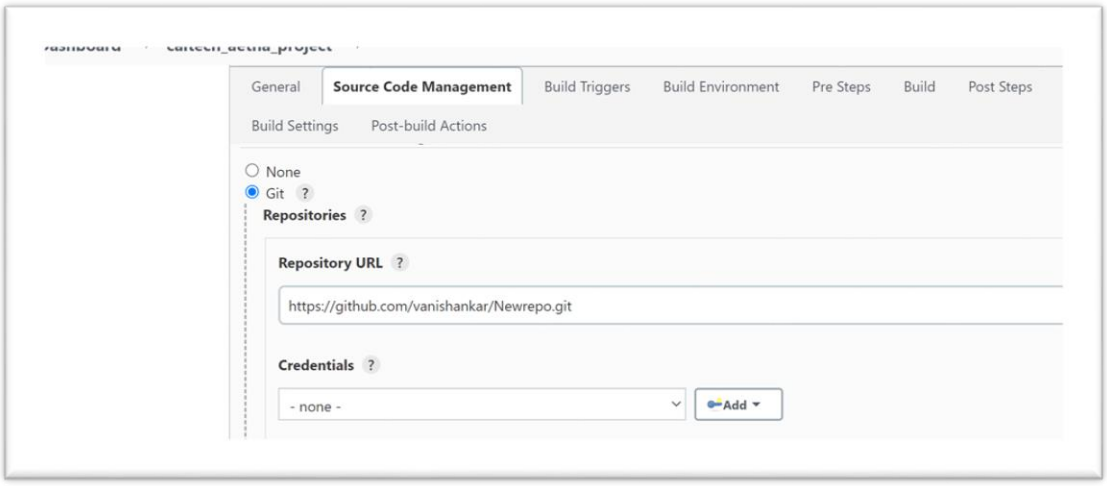
Once done, click on Ok button.

Navigate down to general Section and give description about project, next



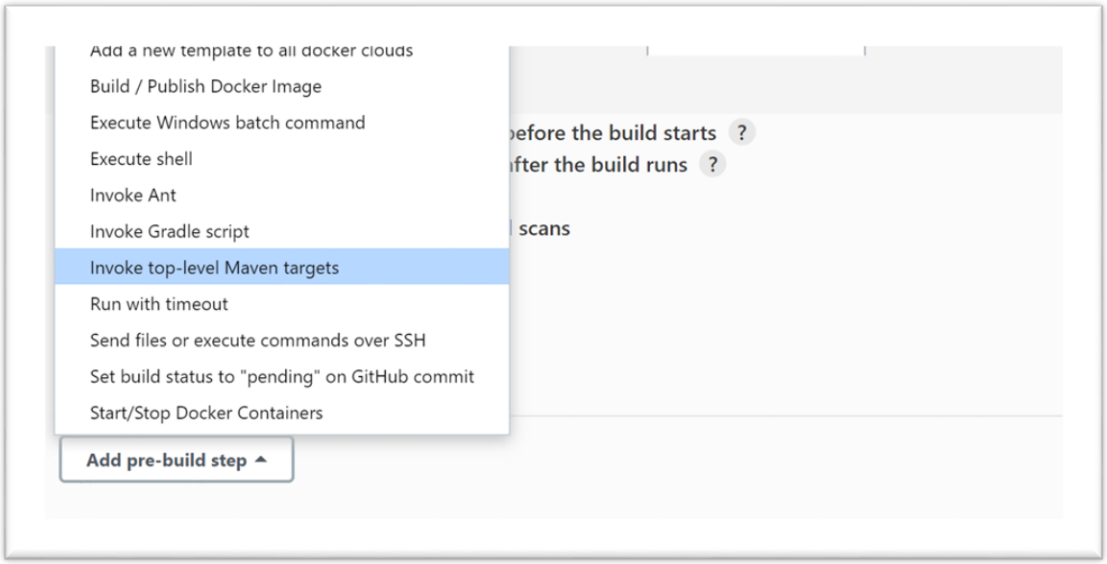
Source code management

navigate to source code management give git hub repo URL GitHub:https://github.com/vanishankar/Newrepo.git, shown in below fig

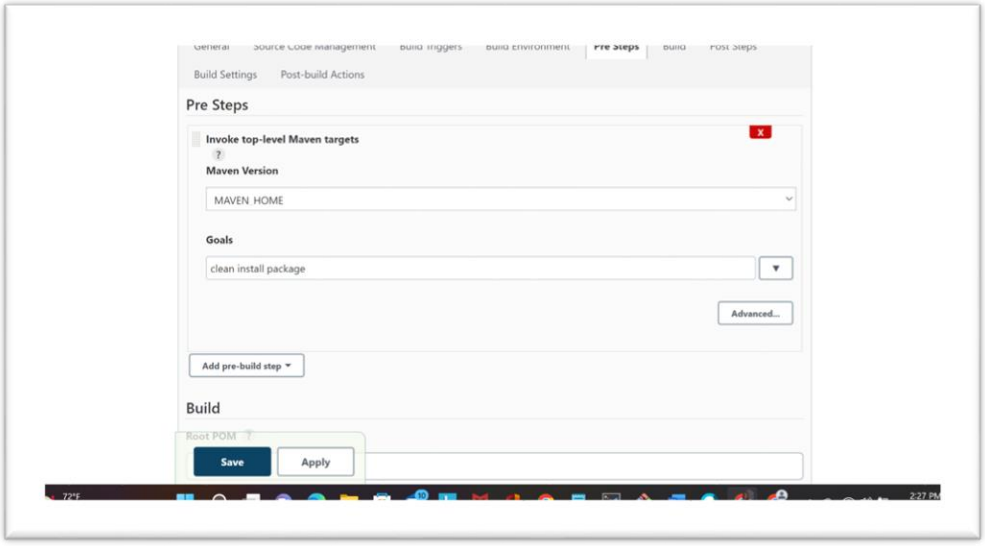


Next, I navigate to pre steps select top level maven goals shown in foll fig

Pre build steps:

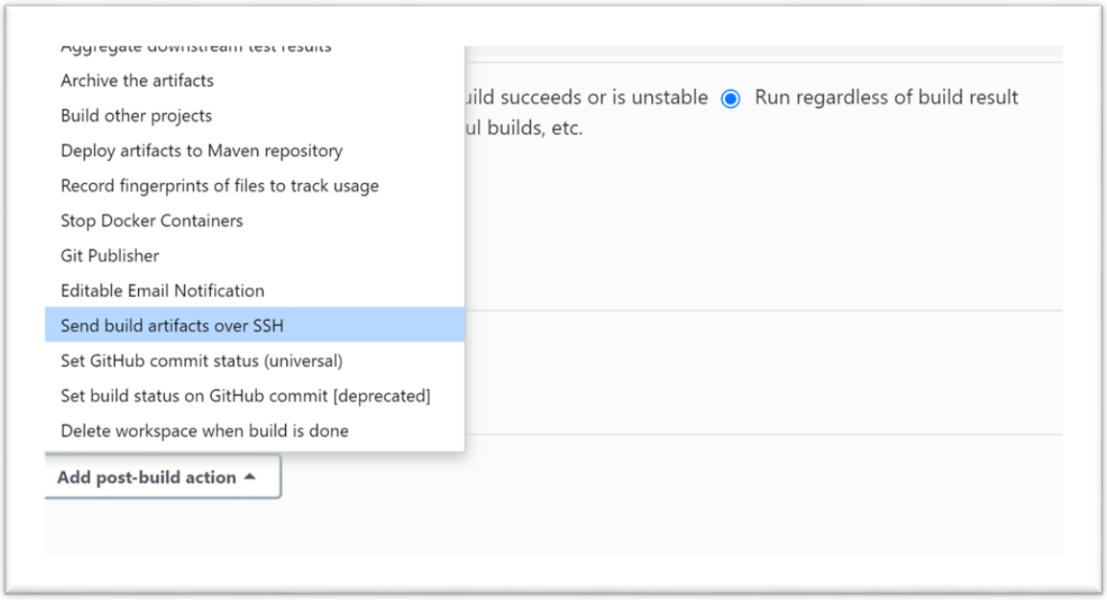


and I gave maven goals (clean, install, package.) Clean will delete all previously compiled Java .class files and resources (like. properties) in your project. Your build will start from a clean slate. *Install* will then compile, test & package your Java project and even *install*/copy your built .jar/.war file into your local Maven repository



package: Converts your .java source code into a .jar/.war file and puts it into the /target folder. **Next, I navigate to post steps I click send build artifacts over ssh(to build artifacts to docker server)**

Post Build Action



Select **Send build artifacts over SSH** under **Post-Build Actions** section. I Selected my remote machine from the dropdown Tomcat_deployerserver

Under **Transfer Set**

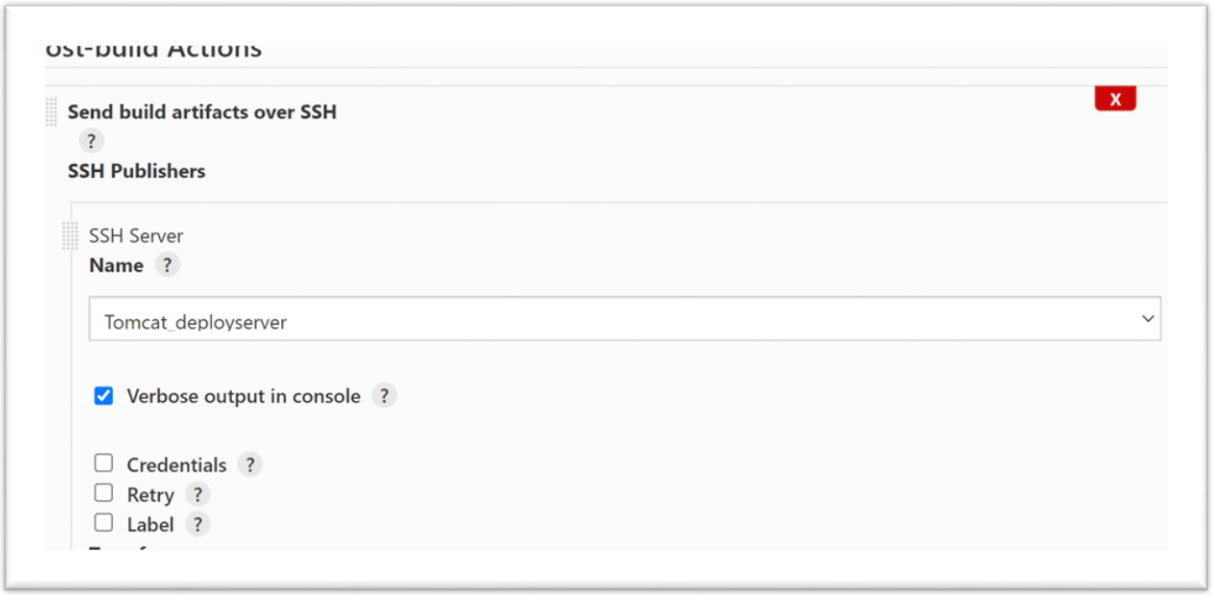
- **Source Files** - add relative path to the build directory where build artifacts reside.

(I add source file –webapp/target/webapp.war)

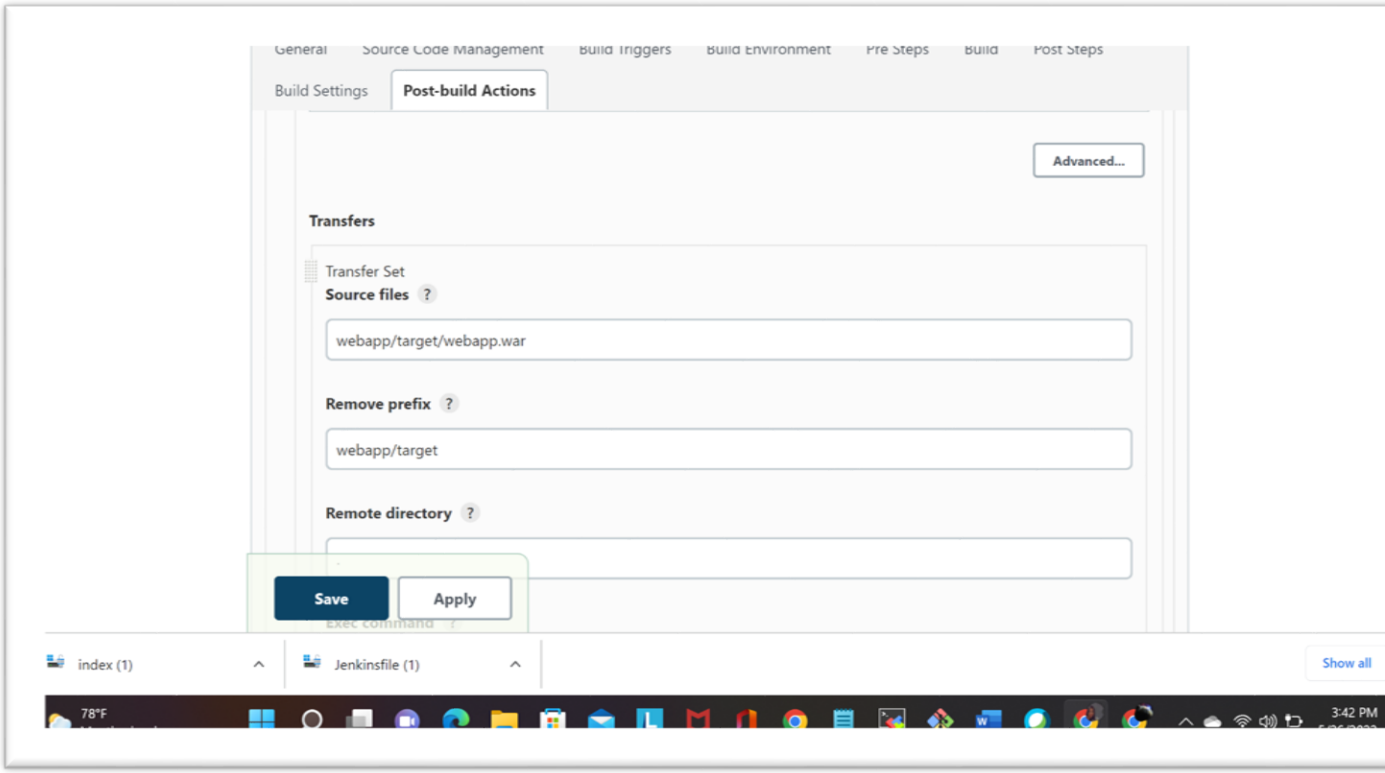
- **Remove Prefix** - First part of the file path that should not be created on the remote server.

- **Remote Directory** - Destination folder where all these stuffs should be placed. The directory name mentioned here will be created under your SFTP home path configured on your SSH server. (I select current directory to store webapps.war file)

POST-BUILD-ACTIONS:



POST-BUILD-ACTIONS



Enter **Exec command** which you want to run on a remote server.

and to automate docker file and give the following command

docker build –t . ----------> **Build an image from a Dockerfile and tag the name**

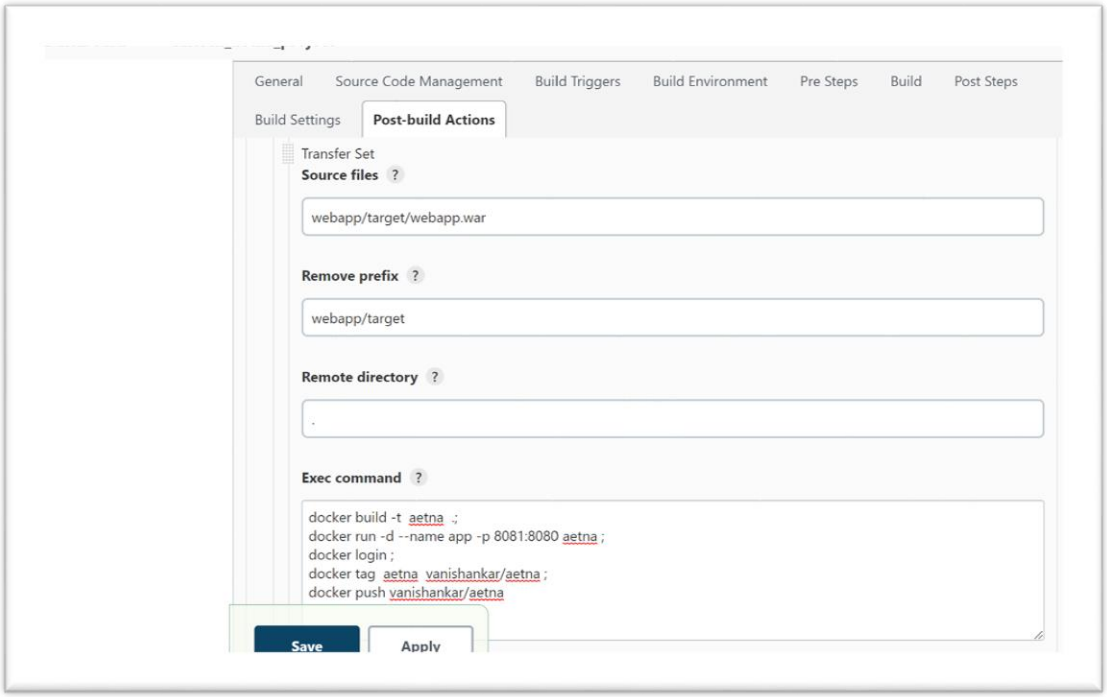
docker run -d -- name app –p 8081:8080 aetna ----------> **Run a command in a new container in a port 8081 externally**

docker login ----------> **Log in to a Docker registry**

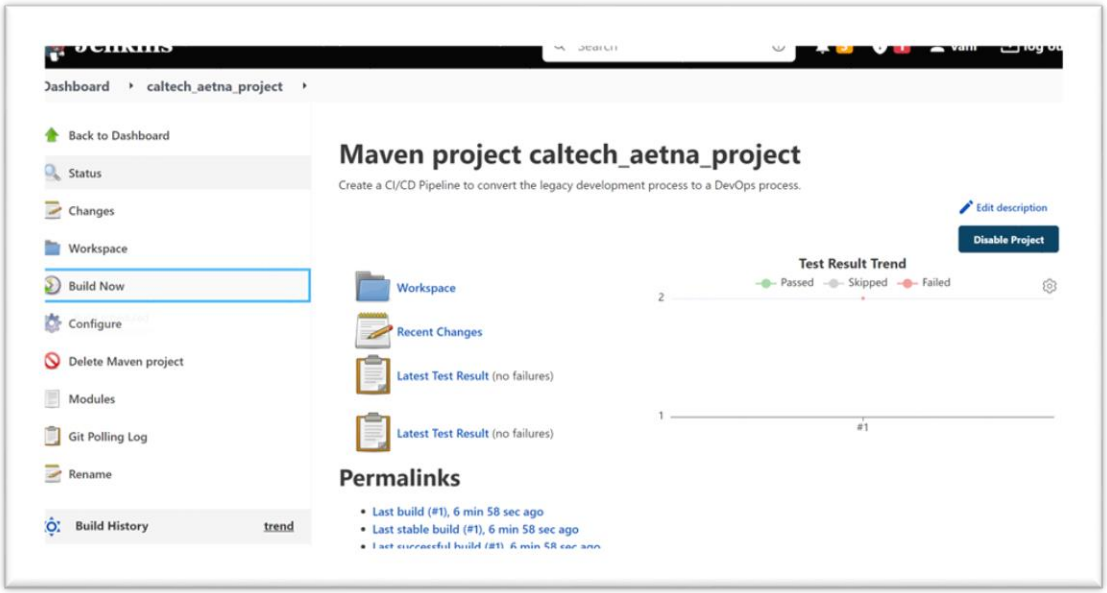
docker tag aetna vanishankar/aetna ----------> **it Create a tag aetna that refers to SOURCE_IMAGE**

docker push vanishankar/aetna ----------> **Push an aetna image to a repository**

This docker command will help to build image, run the container, login to docker hub and pushes the image what we created



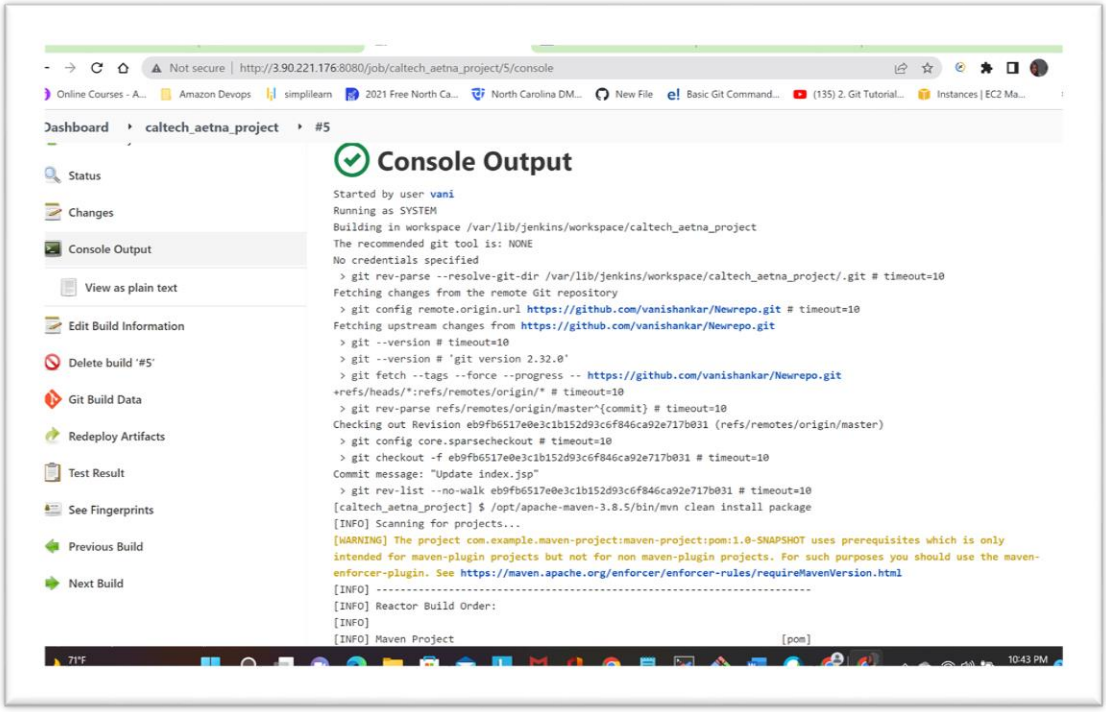
Click save and move to dashboard click Build now



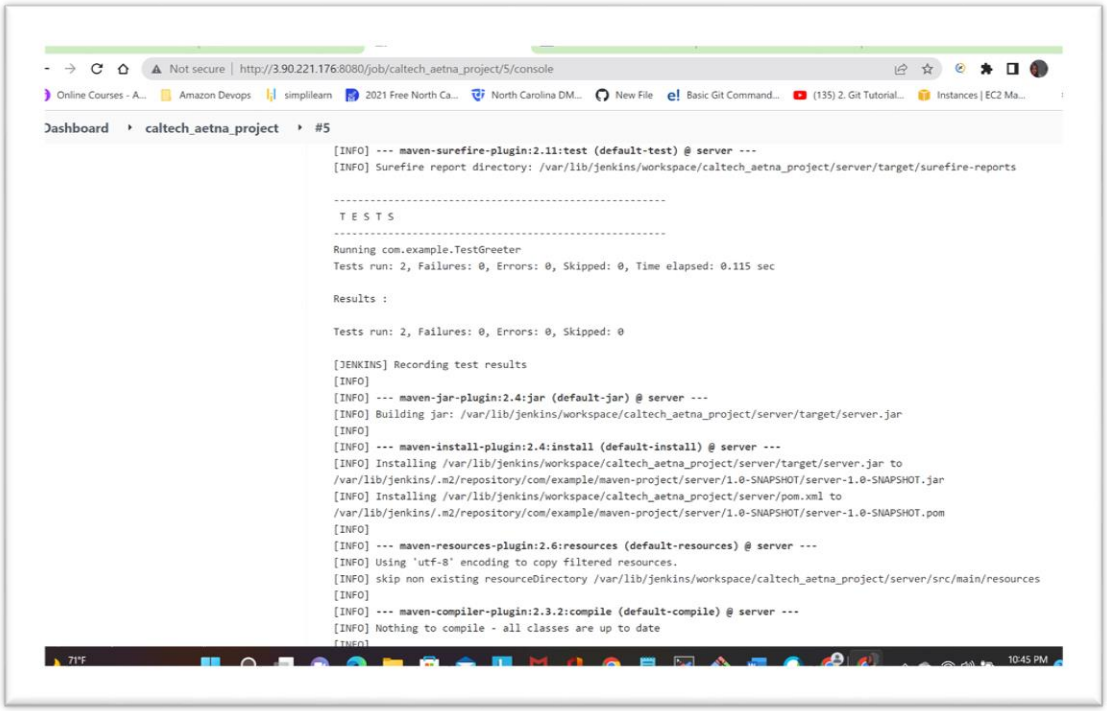
Finally, our caltech_aetna_project job is success. Below fig shows output of our job

Console output:

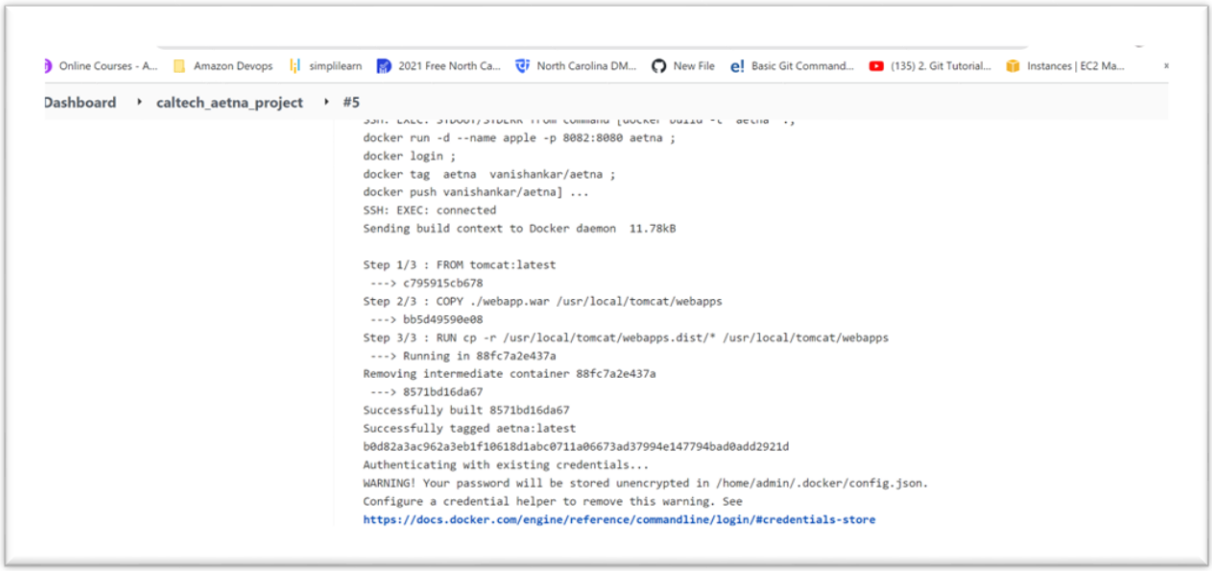
The foll fig shows output of our job



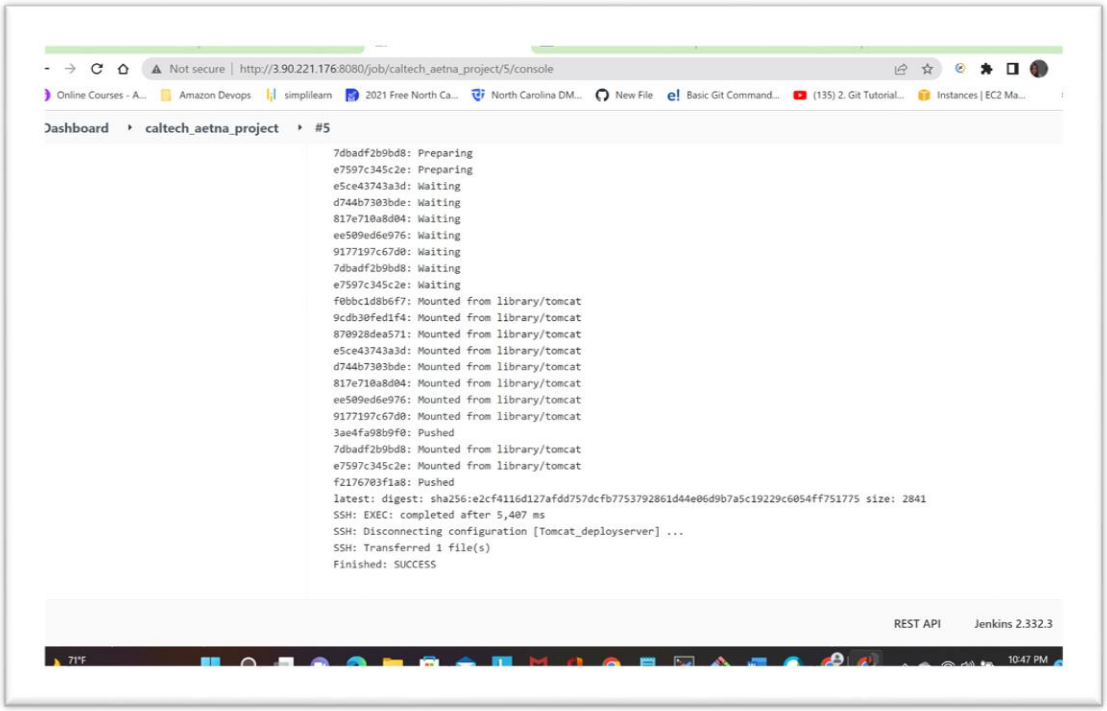
Console output



Console output:

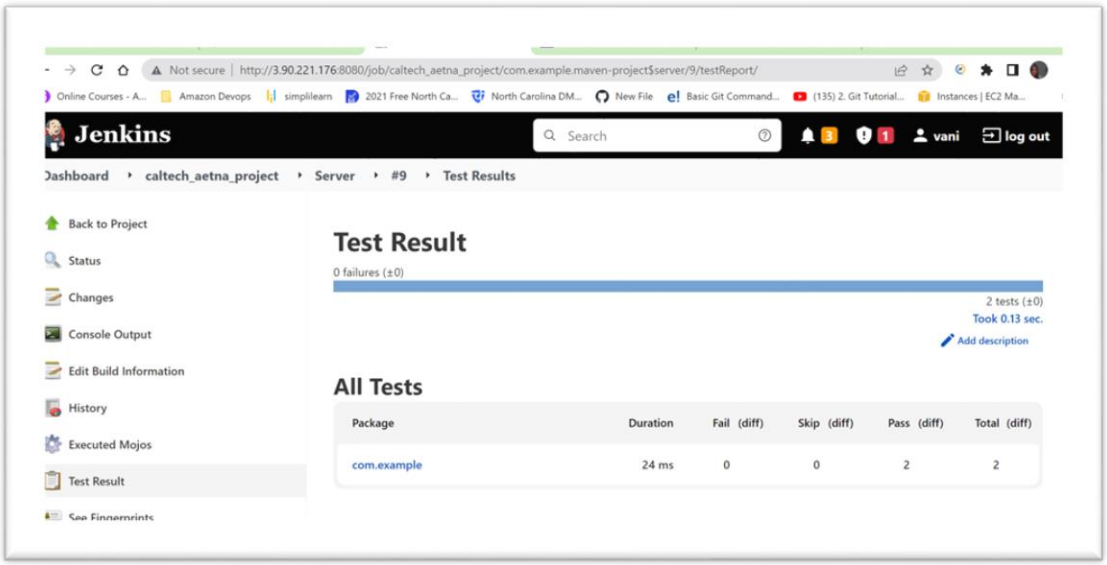


Console output

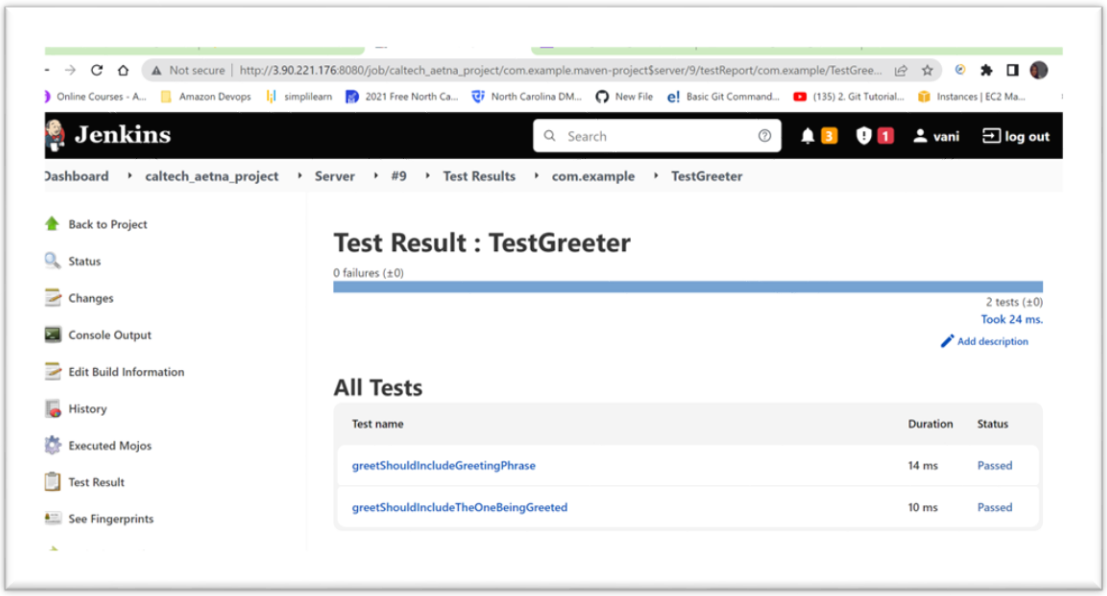


Test result:

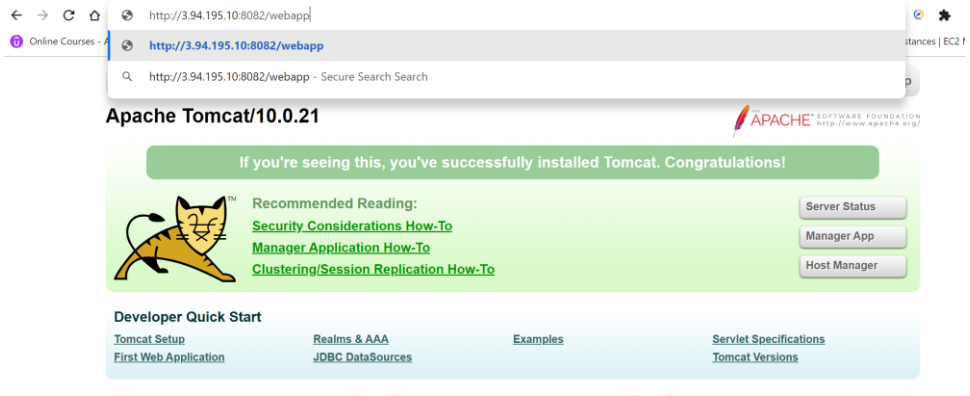
The below fig shows test results



Test Result:



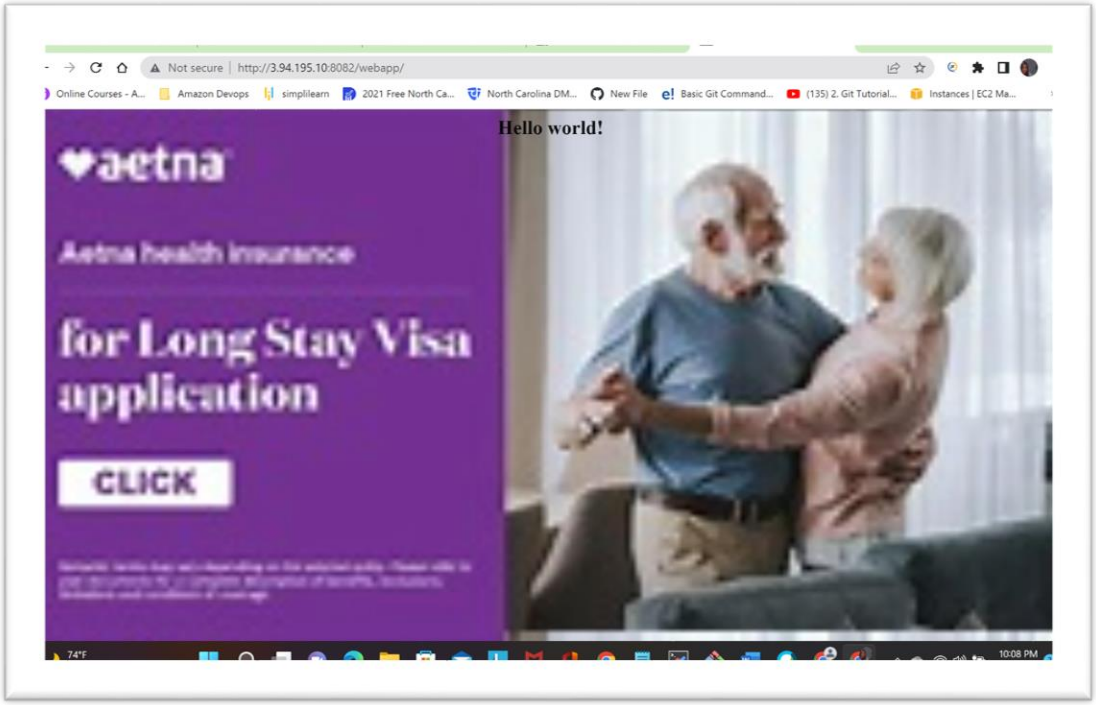
Next, we can access from browser with docker_server Ip address with port (we gave 8081) ACCESSING FROM BROWSER



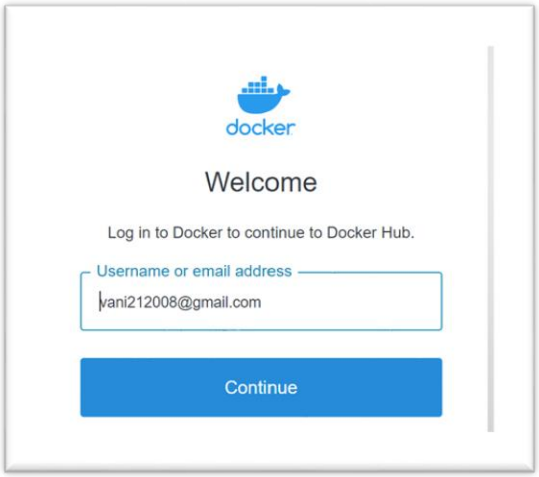
Our tomcat is running

FINAL OUTPUT:

finally, we accessed from browser



Now login to docker hub with id our image aetna is pushed to **Docker hub**



Now login to docker hub with id and clock continue button.it will navigate into Docker hub repo home page

DOCKERHUB REPOSITORY

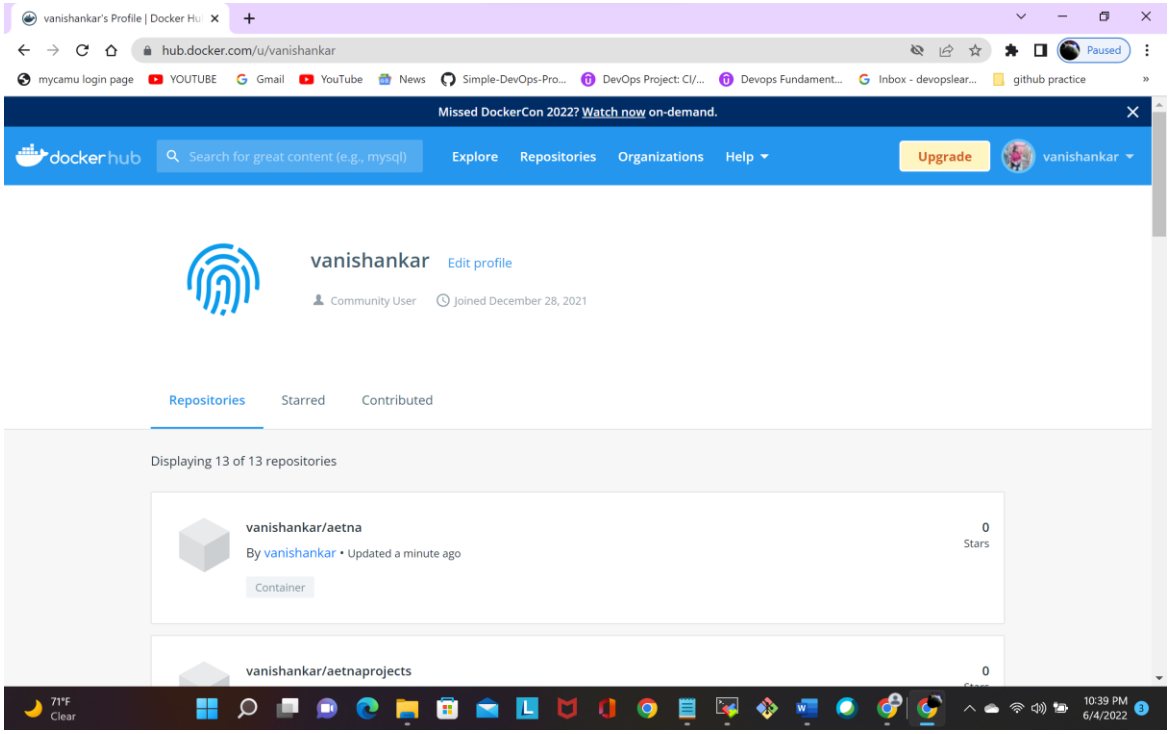


image aetna is pushed to **Docker hub. Successfully** pushed our docker image into docker hub using jenkins.

Concepts used in the project is containerization(docker) it helps Faster delivery of your applications, Deploy and scale more easily Get higher density and run more workloads

Docker Troubleshooting

If you see the below error, it means docker service is not running on the machine. Start the docker service and try the build again

docker: Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?

If you see the below error "Permission Denied"

```
+ docker run hello-world
docker: Got permission denied while trying to connect to the Docker daemon
socket at unix:///var/run/docker.sock: Post
http://%2Fvar%2Frun%2Fdocker.sock/v1.40/containers/create: dial unix
/var/run/docker.sock: connect: permission denied.
See 'docker run --help'.
```

Then add jenkins user to the docker group and restart the jenkins service as shown below:

sudo usermod -aG docker admin

sudo service jenkins stop

sudo service jenkins start

Finally, I deleted the all images and container running in the Docker server using following commands

To check or list out what are the images are present in our server I used following command

docker images

To list out all container

docker ps –a or docker container ls

To list out only running container:

docker ps

Delete docker images

docker rmi –f image id

Delete docker container

docker rm –f container id

Stop docker service

docker stop container id

Delete unused image

docker image prune

Delete unused container

docker container prune

Check the status

Service docker status

Following fig shows deleted, pruned images and container

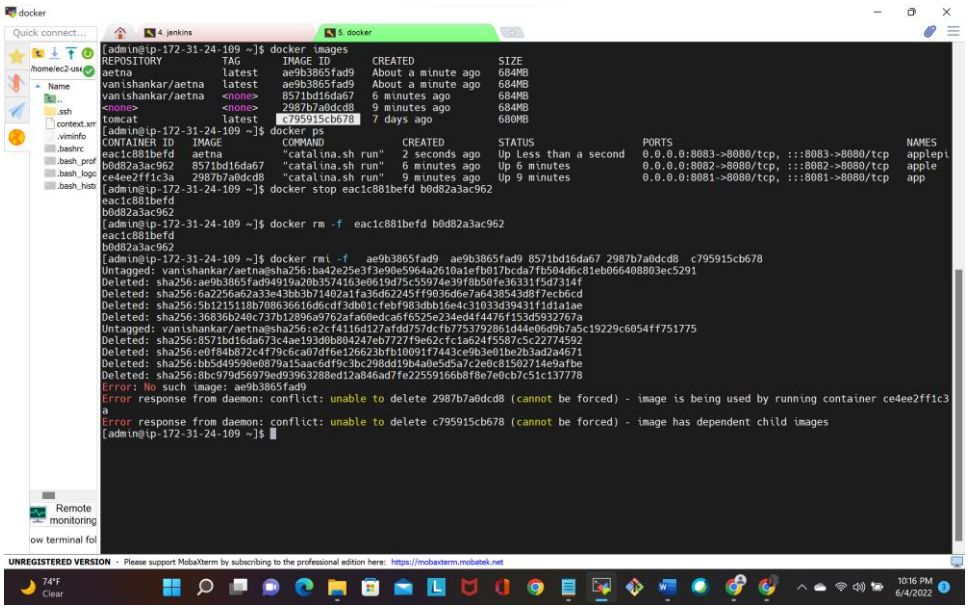
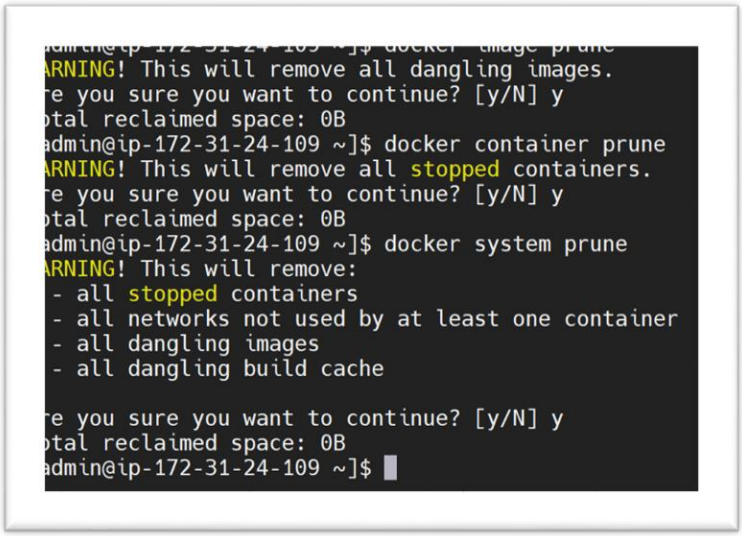


Image and container prune:



Conclusion

Advantages and USPs: Docker The main reason is that Jenkins work really well with Docker. Without Docker you need to install additional tools and add different agents to Jenkins. With Docker, there is no need to install additional tools, you just use images of these tools. Jenkins will download them from internet for you (Docker Hub). Jenkins is a popular continuous integration and delivery (CI/CD) tool. Jenkins is open source and has a great developer community that maintains its existing software and builds plugins to increase its functionality. Though with excellent community support, while using Jenkins, you may suffer from issues like configuration files clashes which you can resolve by running Jenkins in a container technology like Docker.

Kalaivani **Ravishankar**

GitHub: https://github.com/vanishankar/Newrepo.git