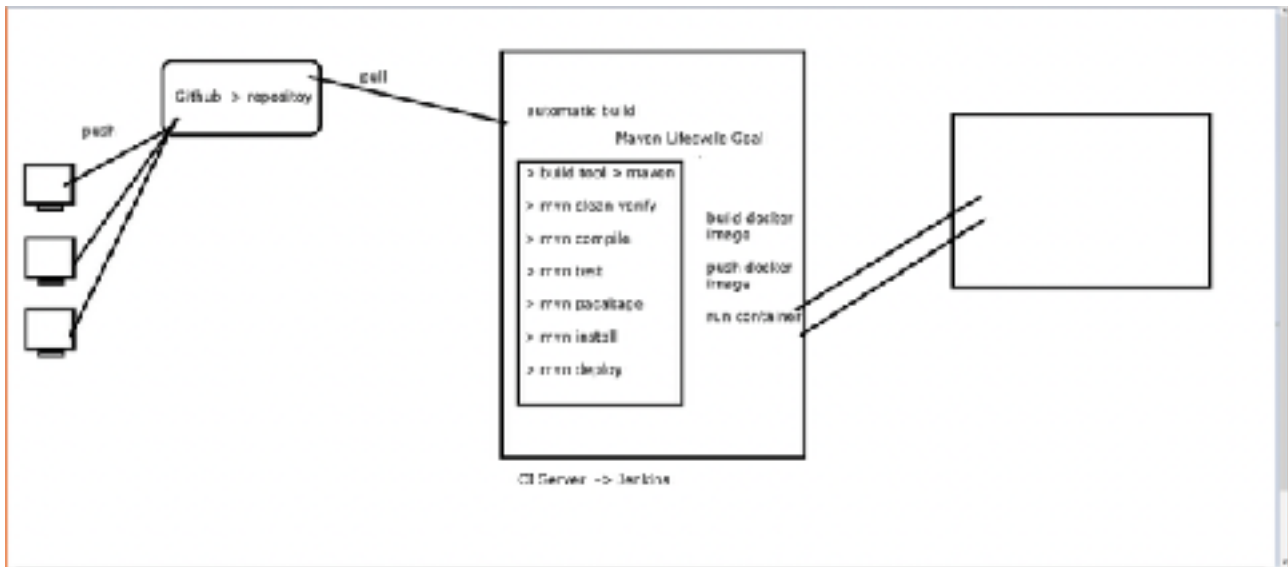


By Anupriya Sharma

<https://github.com/AnupriyaSharma294/ecom-webservice.git>

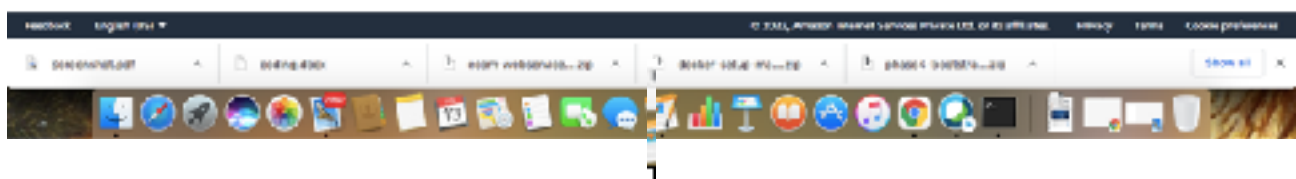
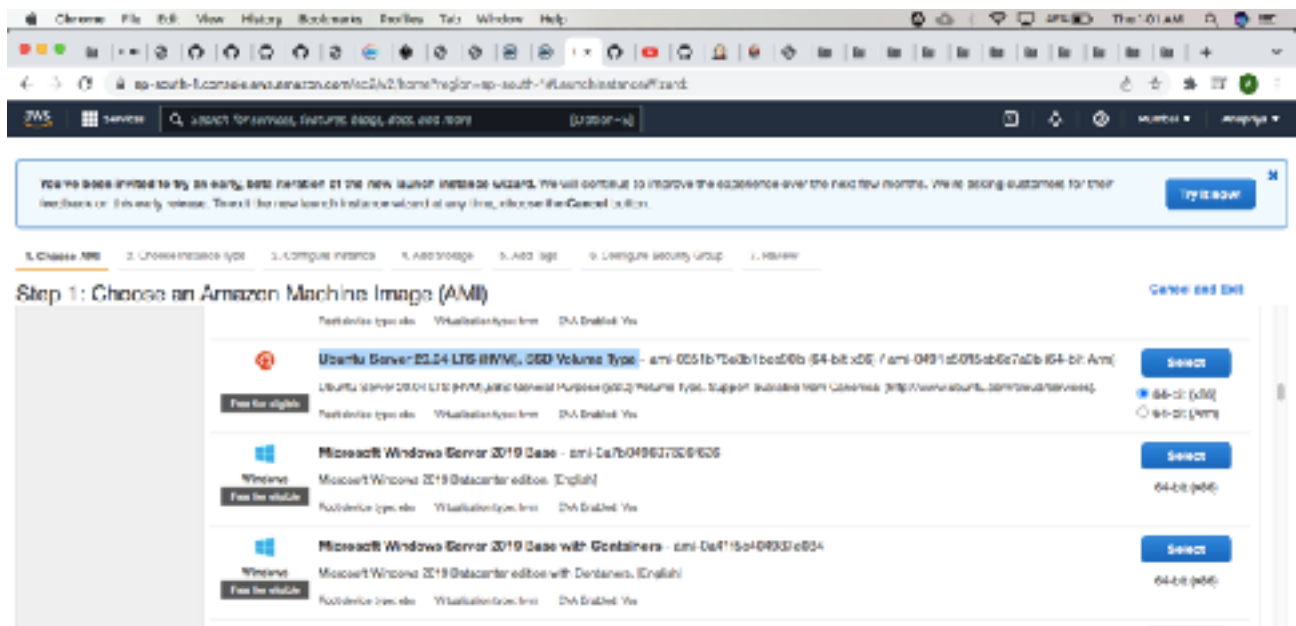
Screenshots

Application Overview Diagram

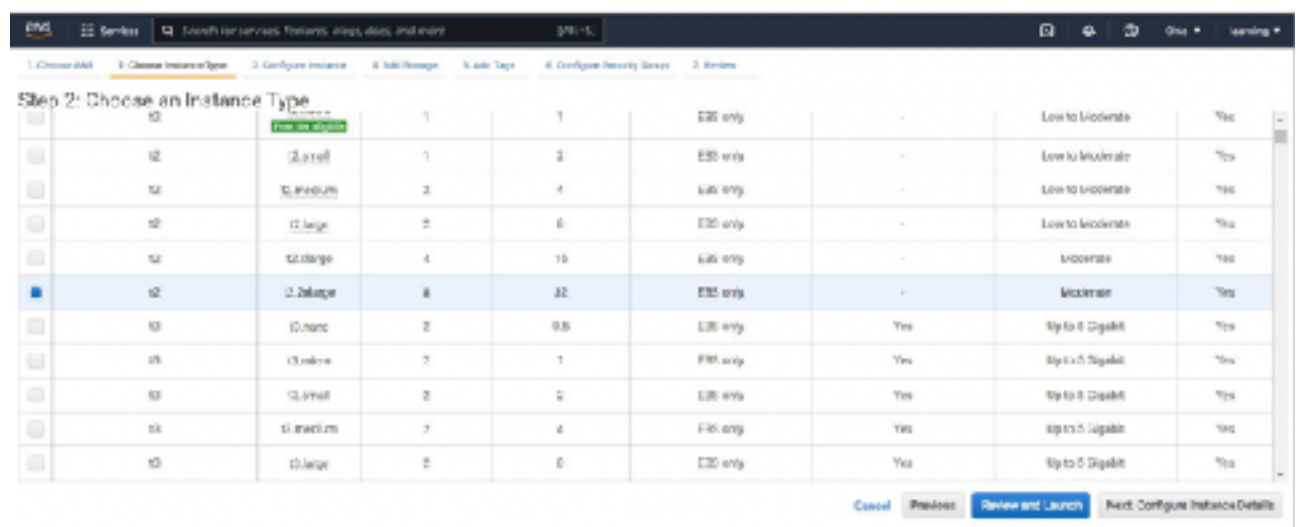


1. AWS Setup (EC2 Instance)

Selected Ubuntu Server 20.04 LTS AMI 64bit.



Selected t2.xlarge instance since it has more memory and CPU for high performance



Disk Storage selection 20 Gb for the ubuntu instance

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more about storage options in Amazon EC2.](#)

Volume Type	Device	Snapshot	Size (GB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sdxt	Snapshot: us-east-1/ami-0a065630-93033982	20	General Purpose (SSD: gp2)	100 / 3000	125 / 1000	<input checked="" type="checkbox"/>	Not Encrypted

[Add New Volume](#)

See the eligible customers category to at least one of the categories: General Purpose (gp2) or Magnetic storage. [Learn more about the storage tier eligibility and usage restrictions.](#)

Shared file systems

The instance must have any of the options on this screen's "Select the file system" button below to use a file system.

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

Network /port configuration for SSH and Web application etc.

Step 6: Configure Security Group

☐ Select an existing security group

Security group name:

Description:

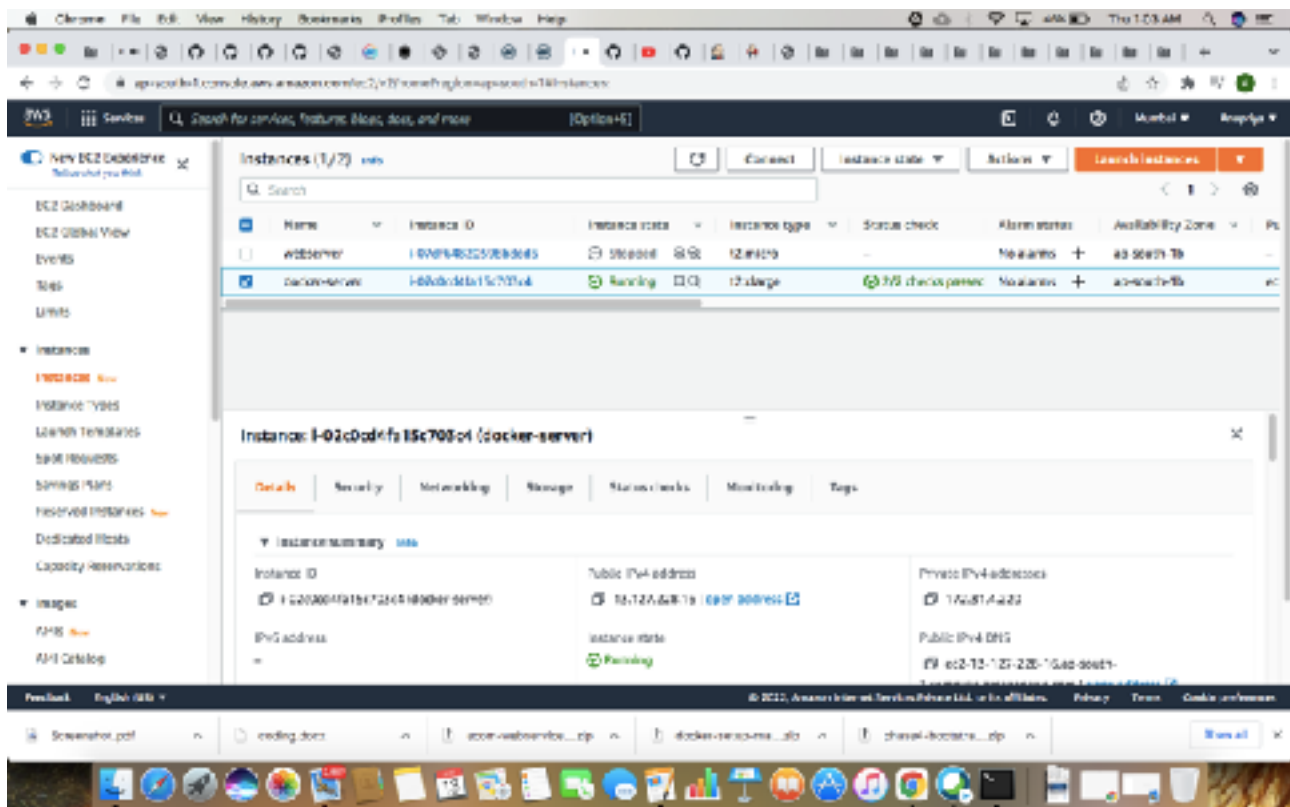
Type	Protocol	Port range	Source	Description
SSH	TCP	22	Anywhere - 0.0.0.0/0	e.g. SSH for Admin Access
Custom TCP	TCP	80	Anywhere - 0.0.0.0/0	e.g. HTTP for Admin Access
Custom TCP	TCP	8080	Anywhere - 0.0.0.0/0	e.g. HTTP for Admin Access
Custom TCP	TCP	8081	Anywhere - 0.0.0.0/0	e.g. HTTP for Admin Access
Custom TCP	TCP	3000	Anywhere - 0.0.0.0/0	e.g. SSH for Admin Access

[Add Rule](#)

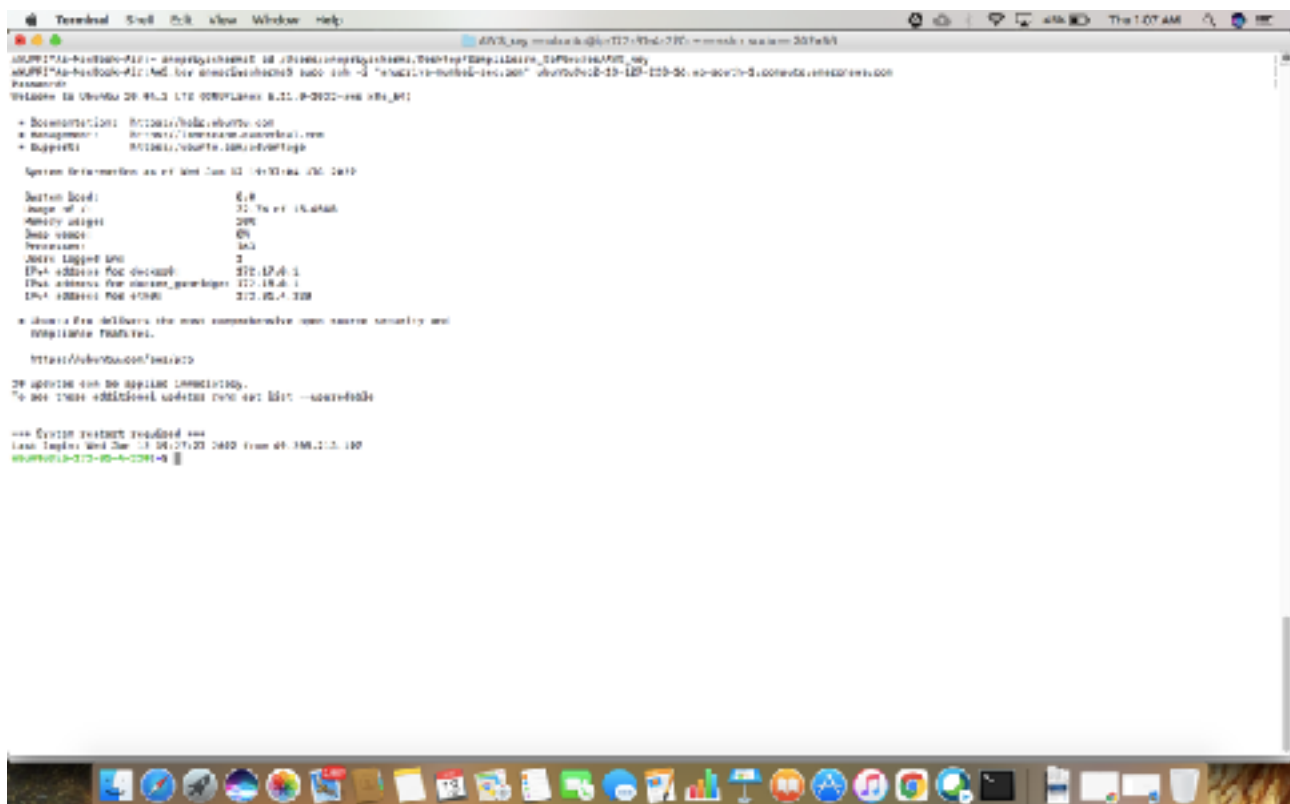
Warning
Rules with source of 0.0.0.0/0 allow IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Cancel](#) [Previous](#) [Review and Launch](#)

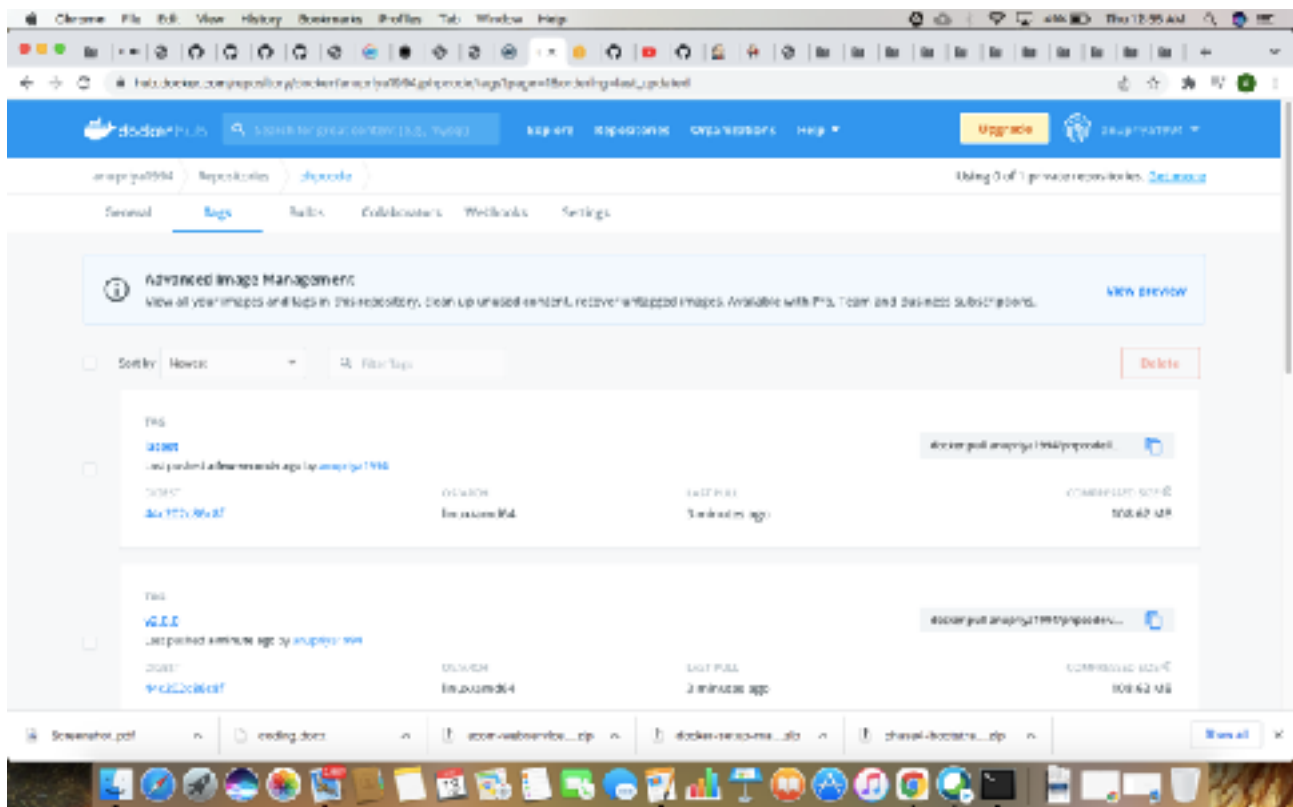
Instance created with the configurations



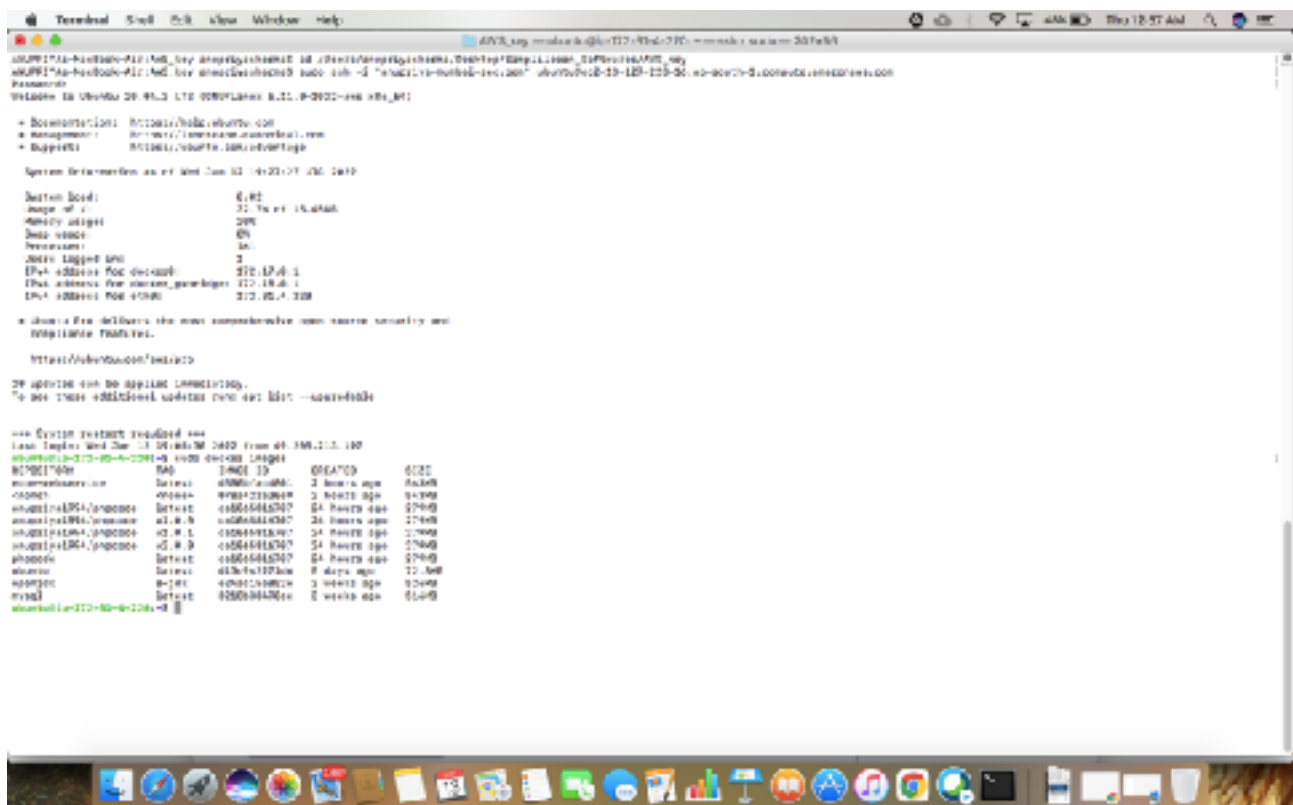
Connected using Terminal (SSH client)



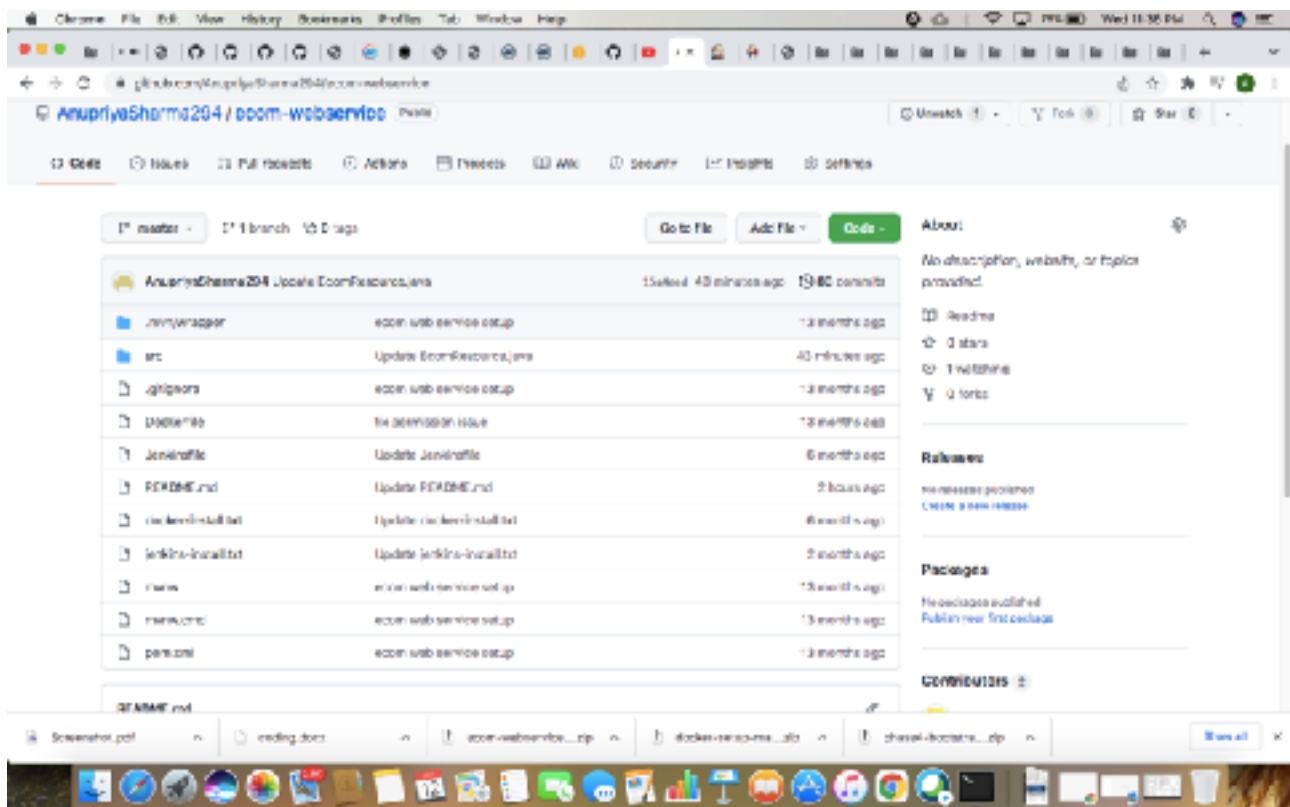
2. Docker



DOCKER HUB Dashboard and Image pushed via command line

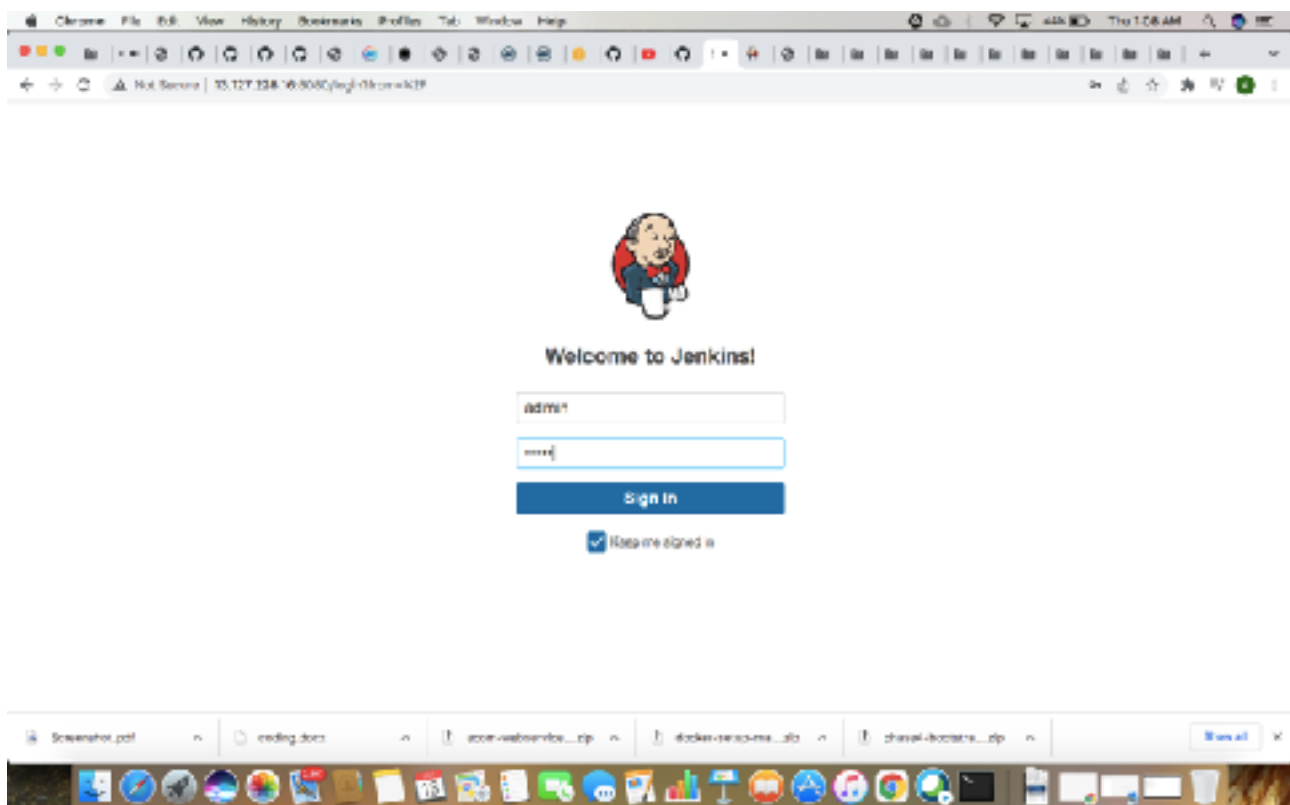


3. Github (Sample Spring boot application)

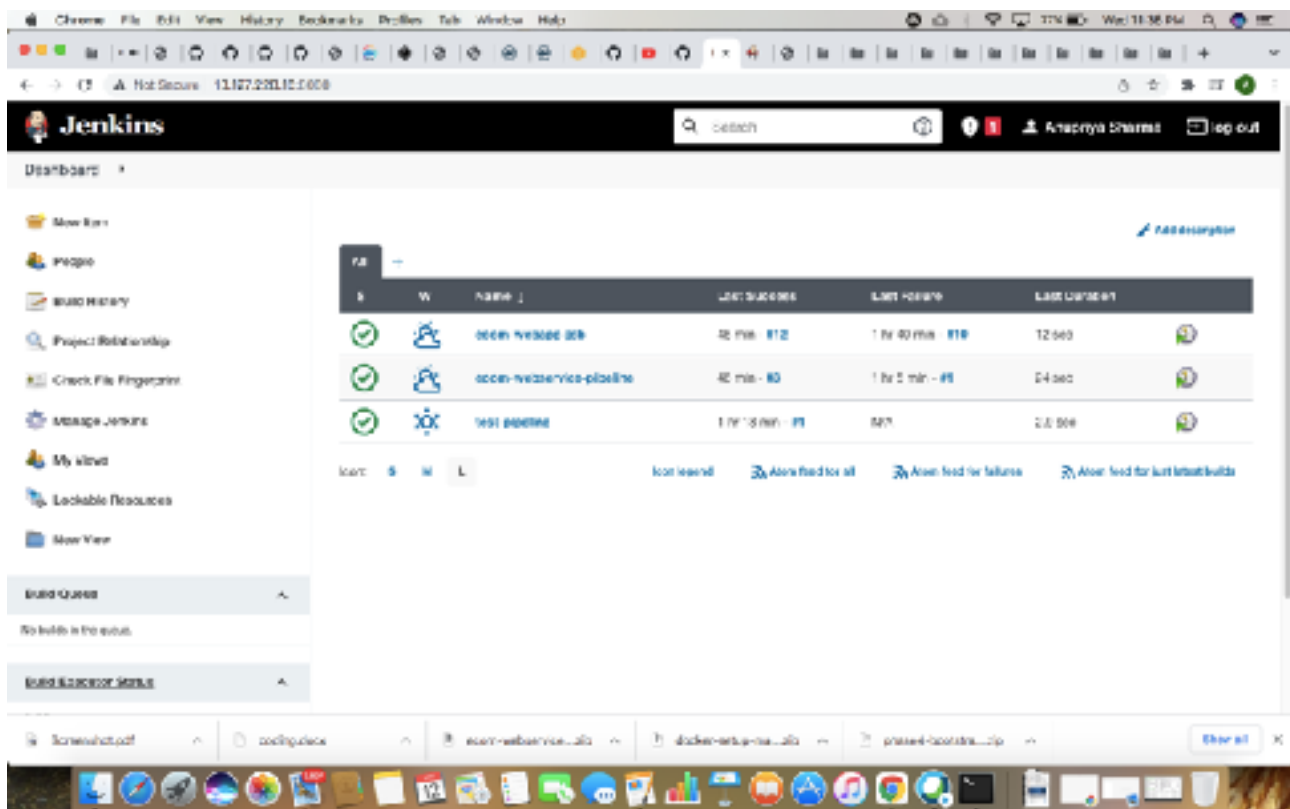


4. Jenkins Setup

Access using ec2 public ip:8080 port <http://13.127.228.16:8080/login?from=%2F>

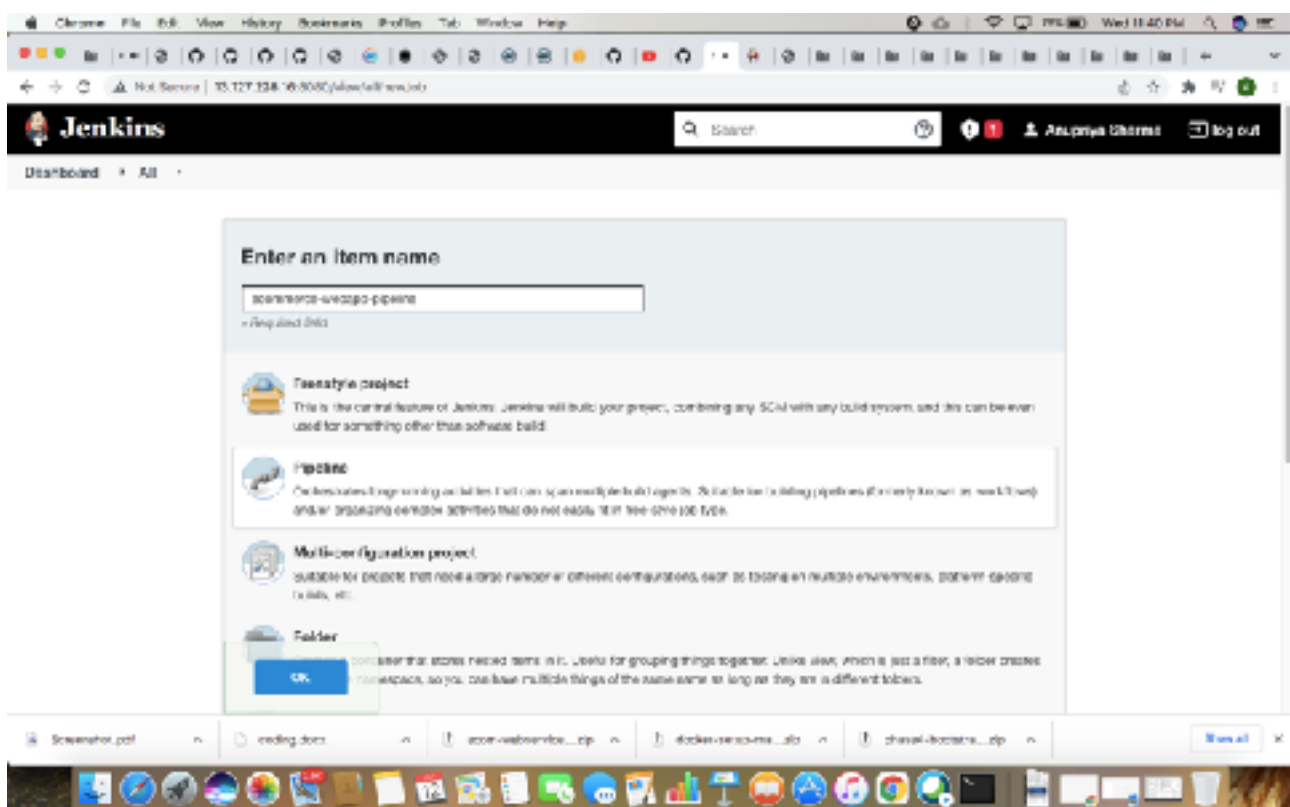


once login it will show dashboard page

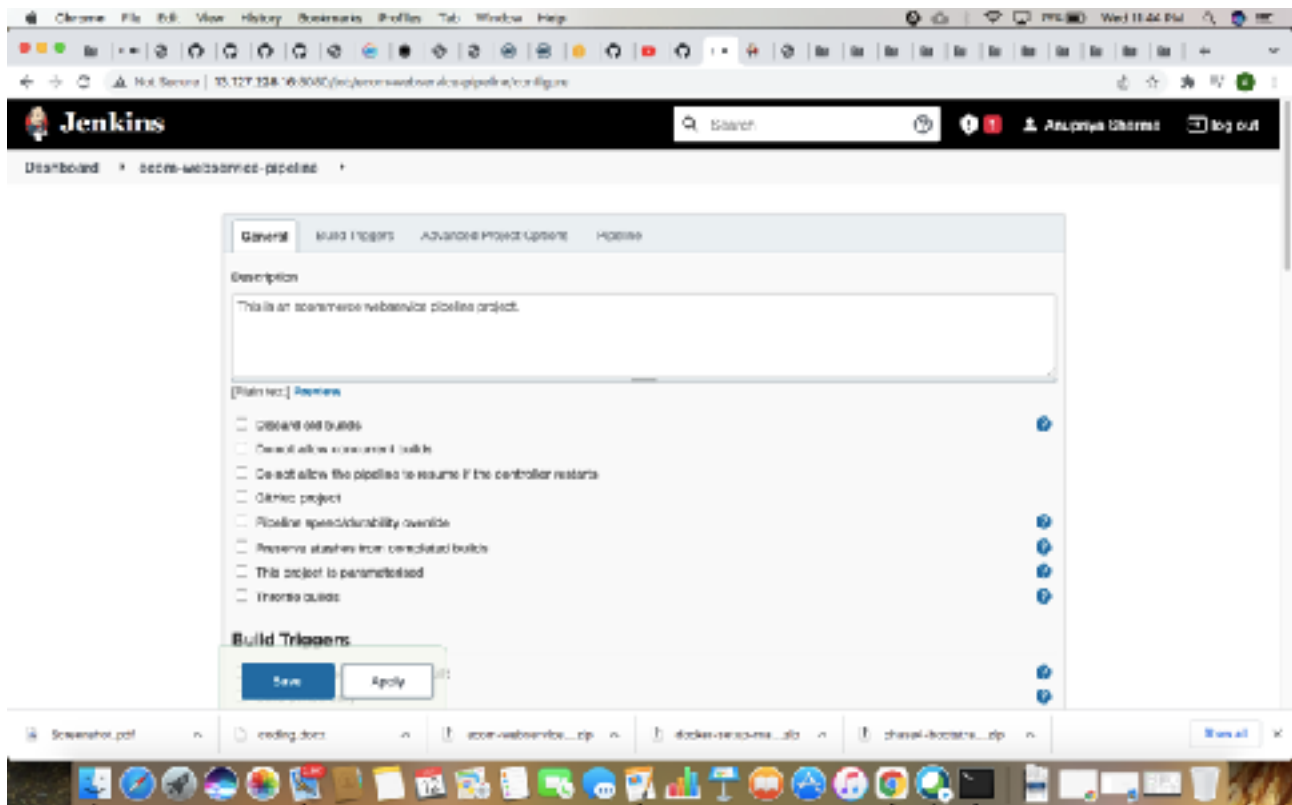


Click on new item to add new Pipeline.

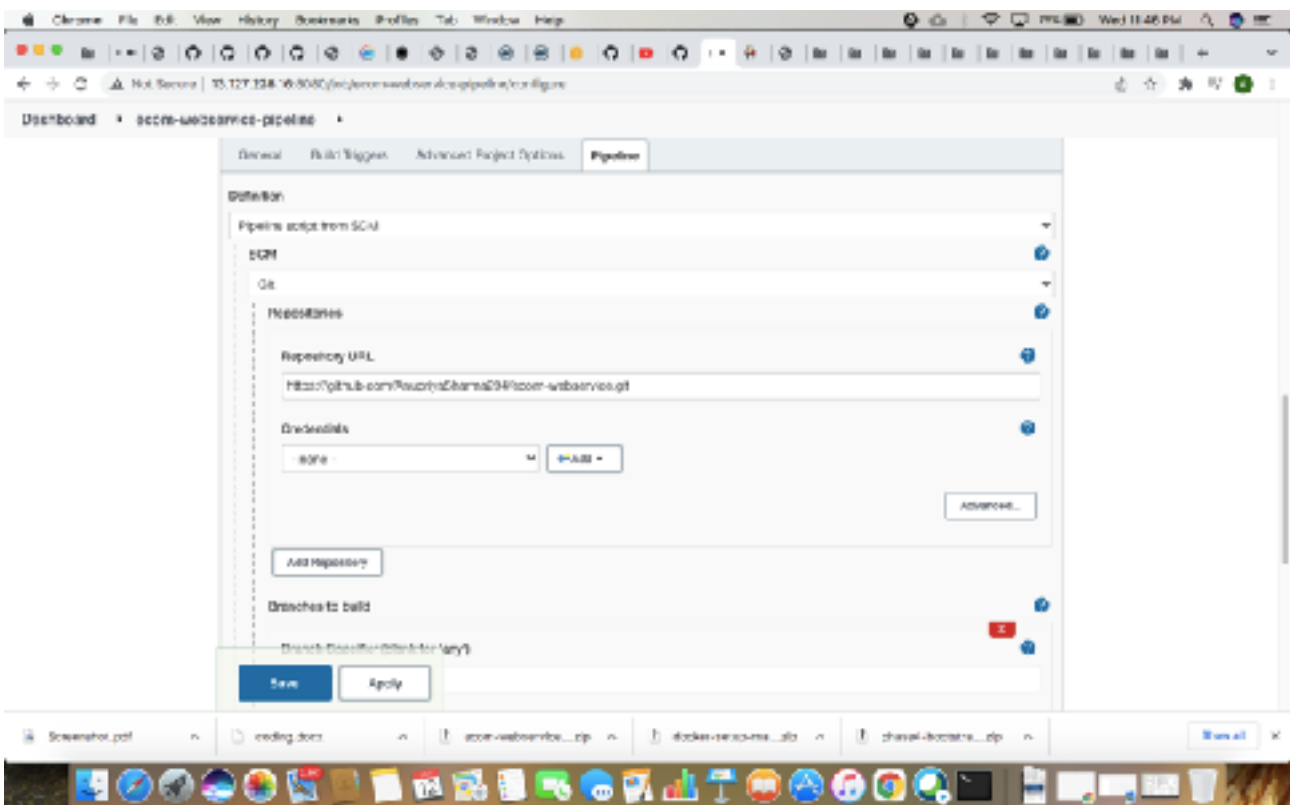
Enter the name and select pipeline project below



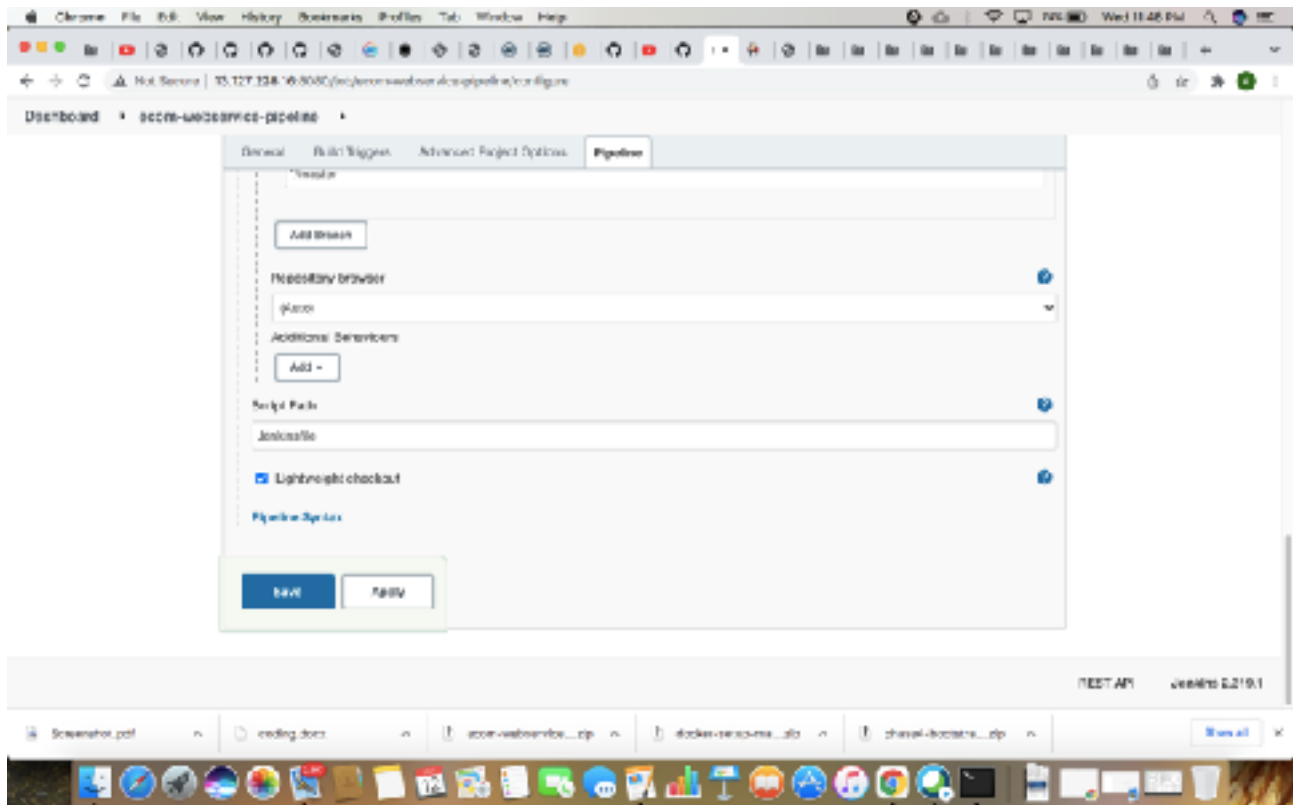
Enter description



Select Pipeline script from SCM , select git and enter git url



Specify Branch and jenkins script file in Script Path text box. And click on Apply and Save button.



Jenkins Scripts

agent any

triggers {

pollSCM('* * * * *')

}

// Got permission denied while trying to connect to the Docker daemon socket at unix.

// sudo usermod -a -G docker jenkins

// restart jenkins server -> sudo service jenkins restart

stages {

pipeline {

stage('Maven Compile') {

steps {

```

echo '----- This is a compile phase -----'

sh 'mvn clean compile'

}

}

stage('Maven Test') {

steps {

echo '----- This is a Test phase -----'

sh 'mvn clean test'

}

}

stage('Maven Build') {

steps {

echo '----- This is a build phase -----'

sh 'mvn clean package -DskipTests'

}

}

stage('Docker Build') {

steps {

echo '----- This is a build docker image phase -----'

sh '''

docker image build -t ecom-webservice .

'''

}

}

stage('Docker Deploy') {

steps {

echo '----- This is a docker deployment phase -----'

sh '''

(if [ $(docker ps -a | grep ecom-webservice | cut -d " " -f1) ]; then \

echo $(docker rm -f ecom-webservice); \

echo "----- successfully removed ecom-webservice -----"

else \

```

```
echo OK;\
```

```
fi);
```

```
docker container run --restart always --name ecom-webservice -p 8081:8081 -d ecom-webservice
```

```
'''
```

```
}
```

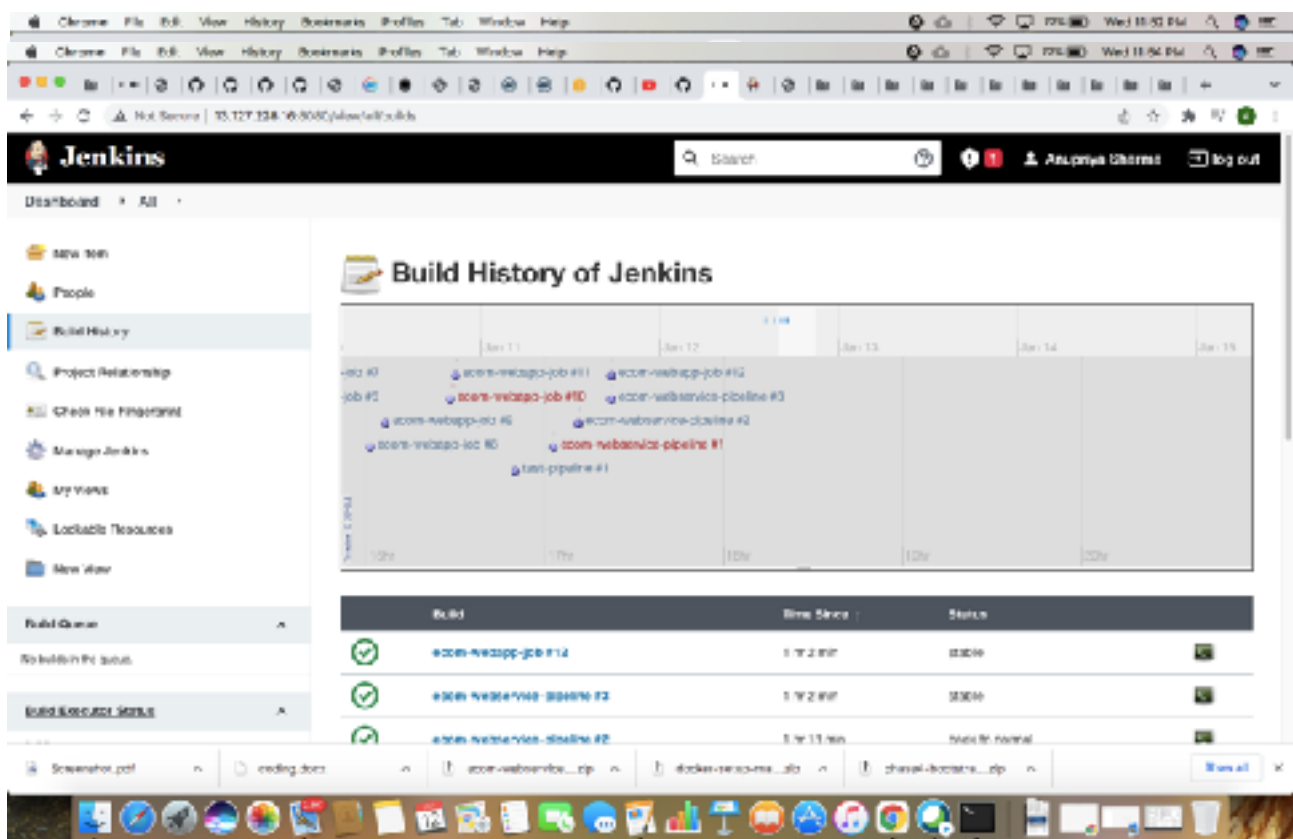
```
}
```

```
}
```

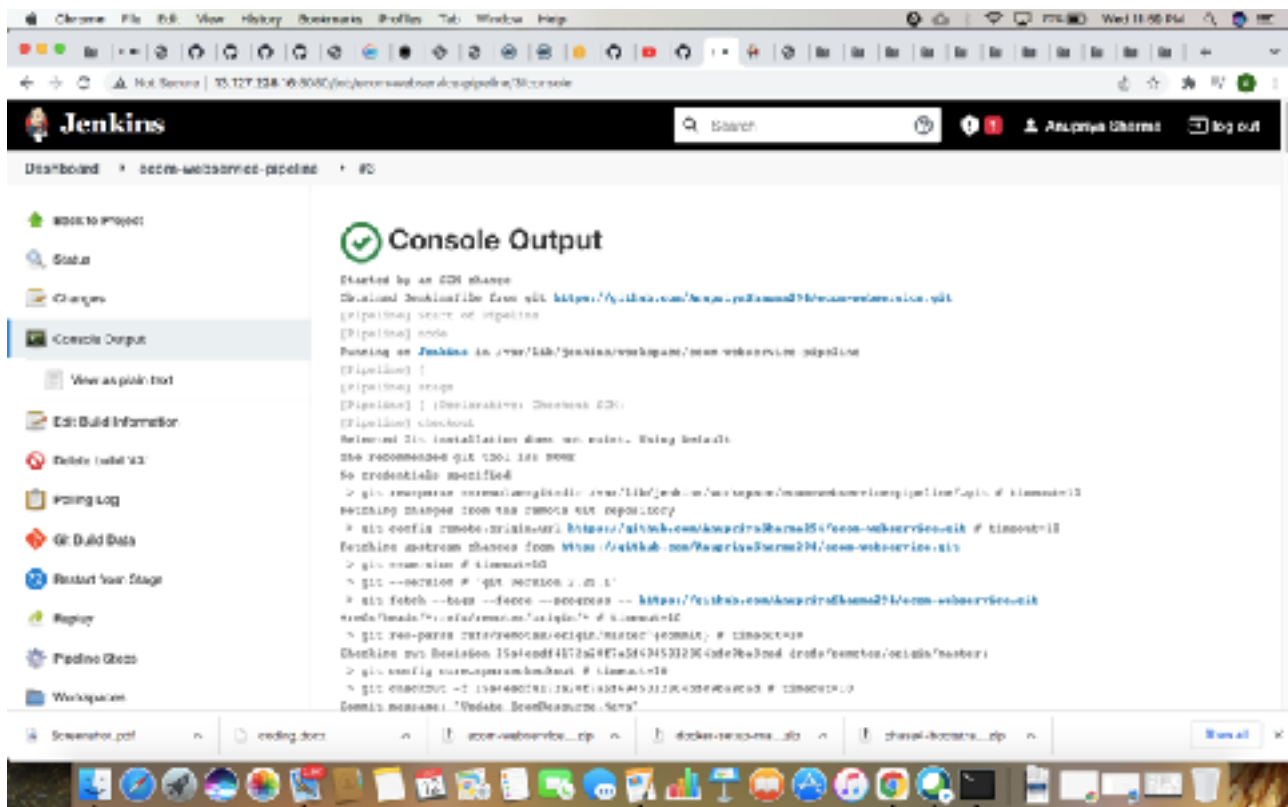
```
}
```

After build it will show the progress in different phases as shown below

Build History

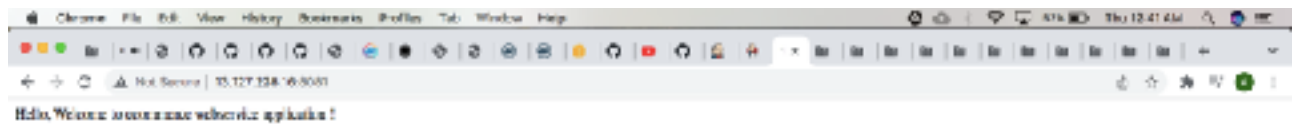


Console Output

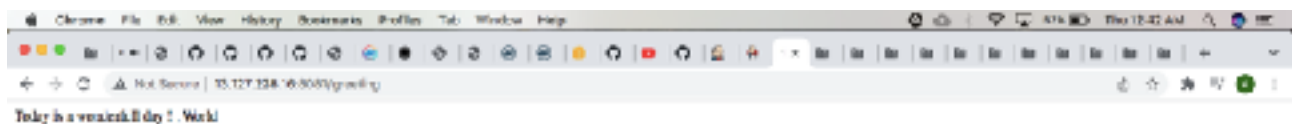




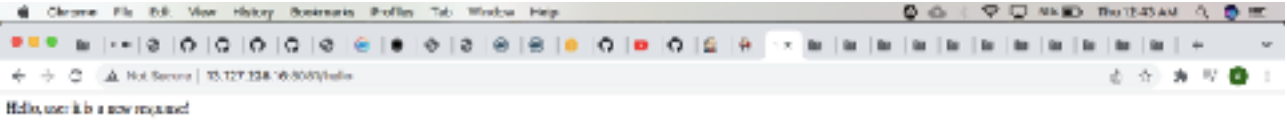
WebApplication can be accessed by port 8081 as shown below:



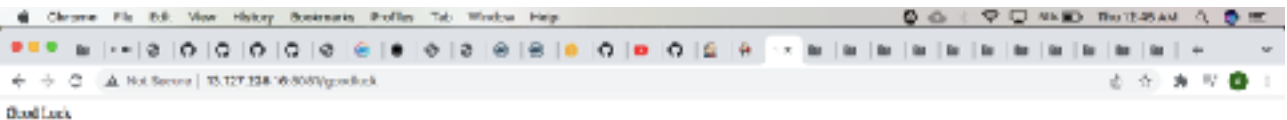
/greeting



/hello



/goodluck



Workspace View

