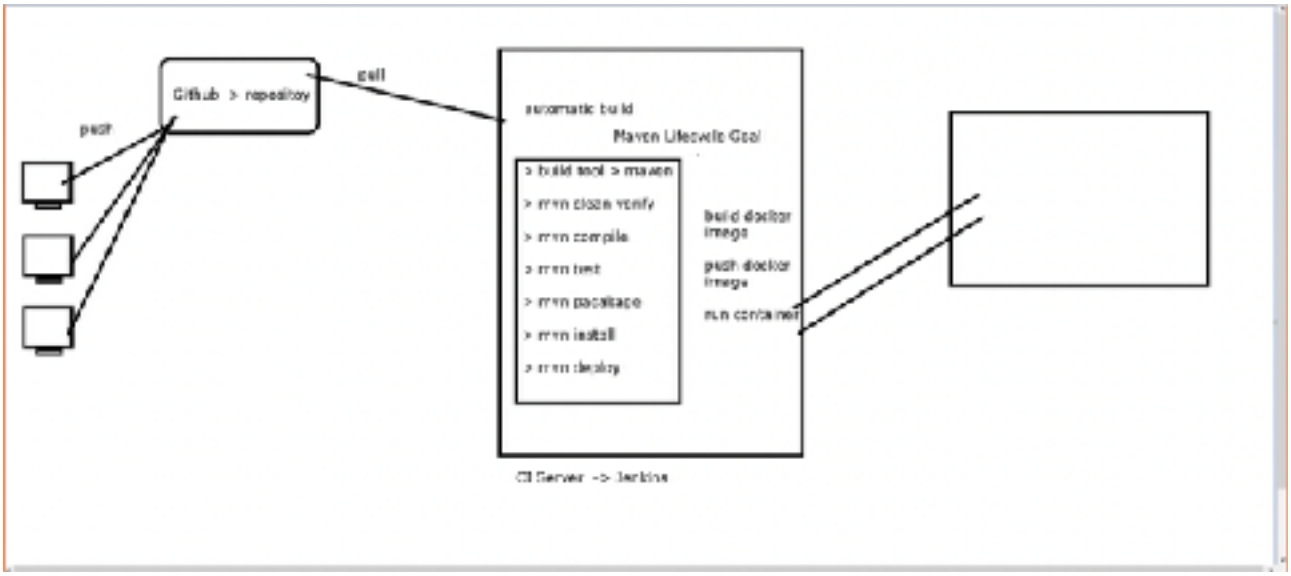
By Anupriya Sharma
https://github.com/AnupriyaSharma294/ecom-webservice.git

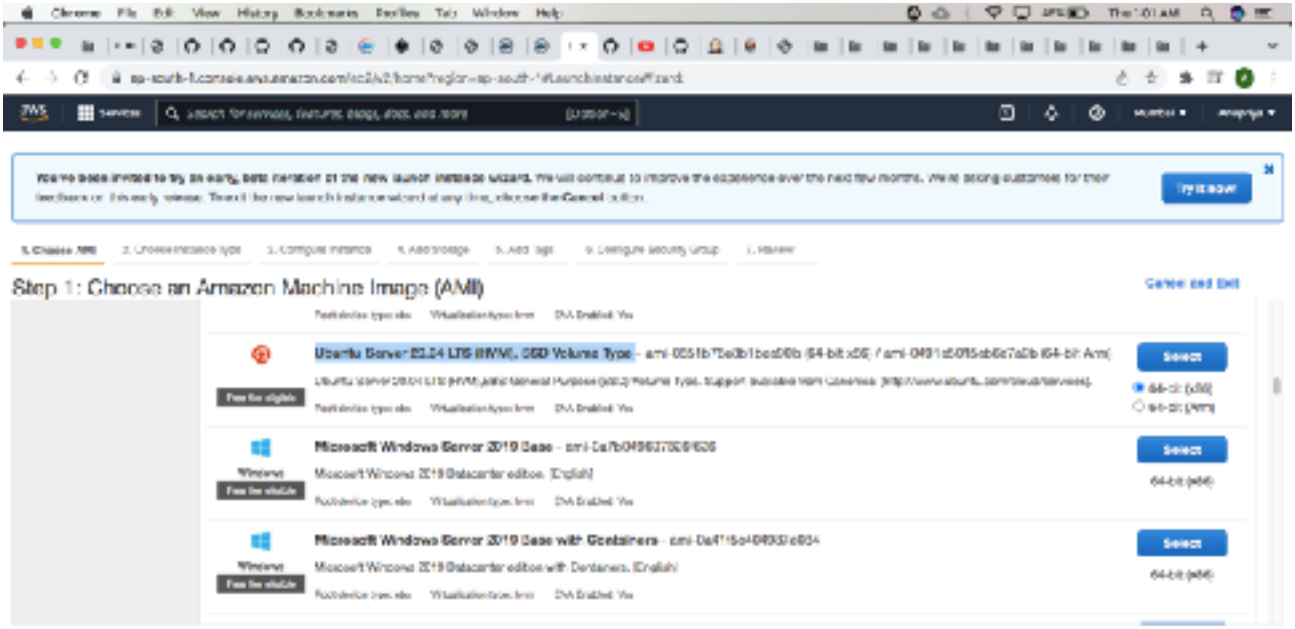
Screenshots

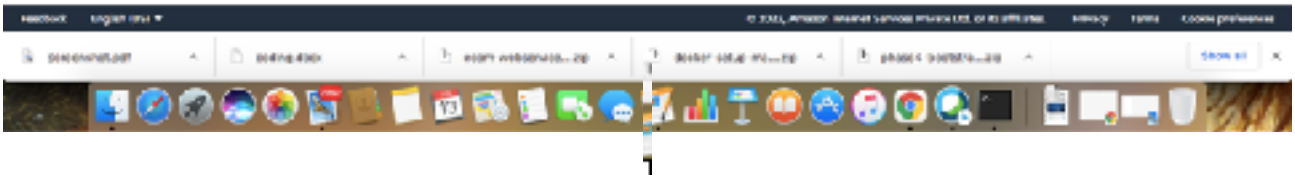
Application Overview Diagram



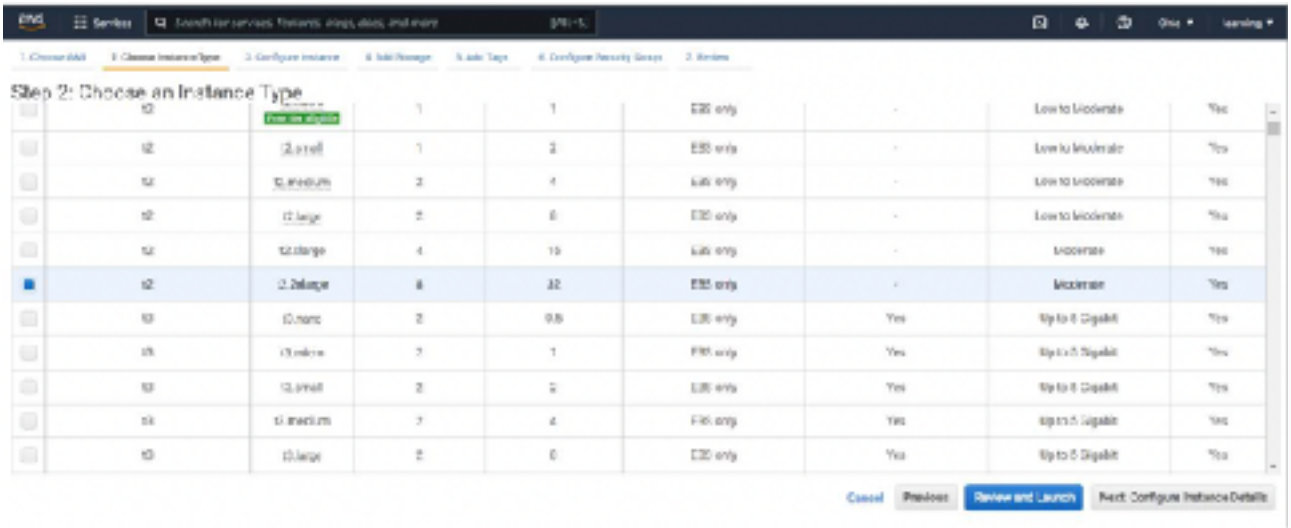
1. AWS Setup (EC2 Instance)

Selected Ubuntu Server 20.04 LTS AMI 64bit.

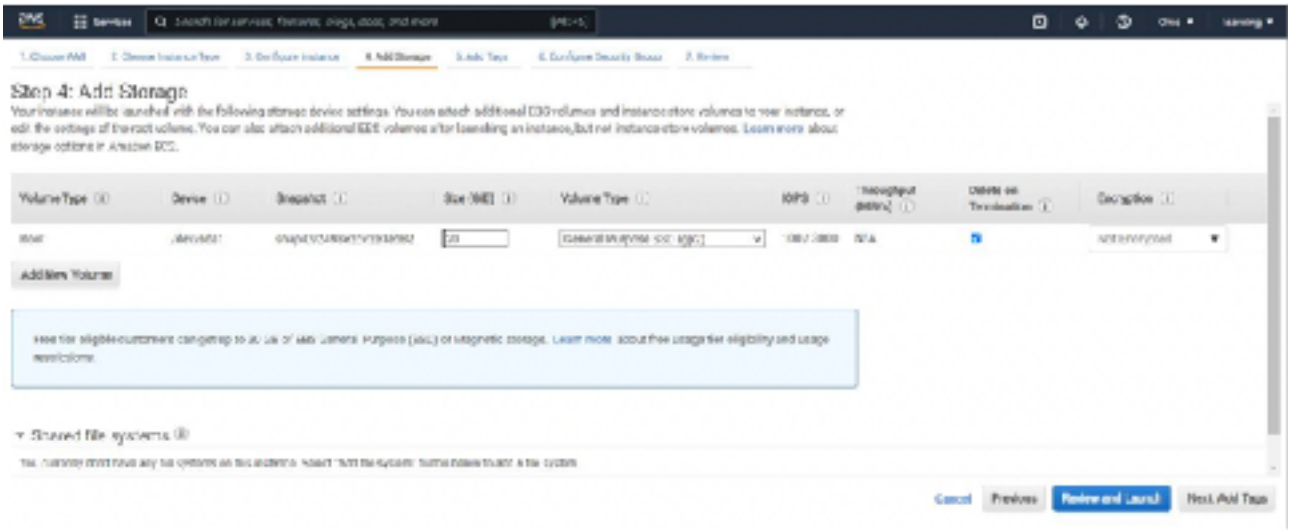




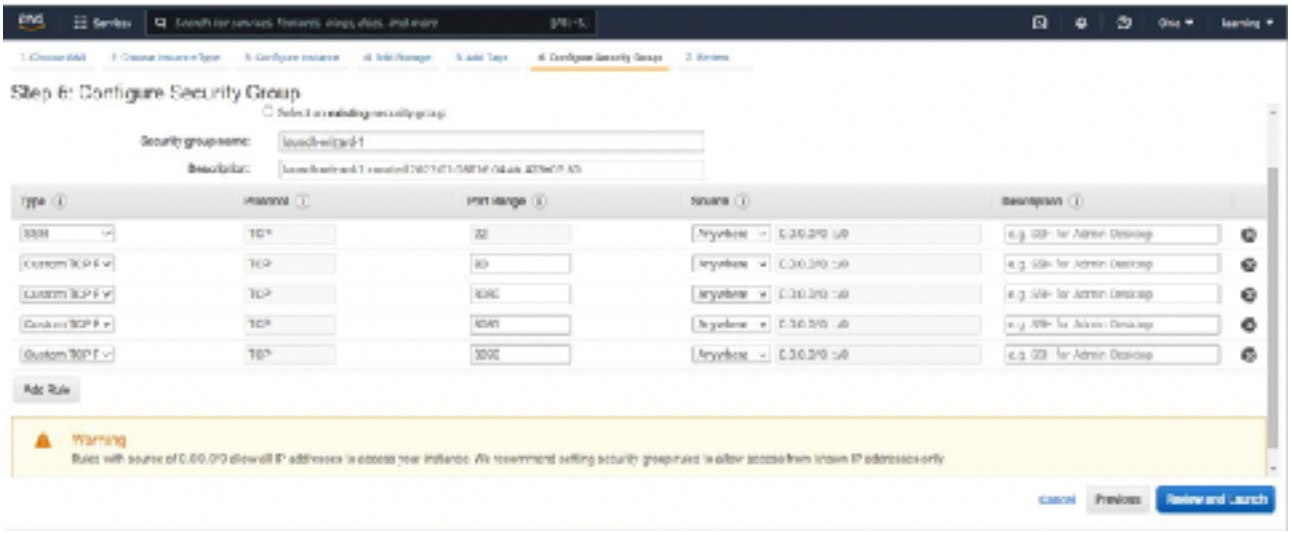
Selected t2.xlarge instance since it has more memory and CPU for high performance



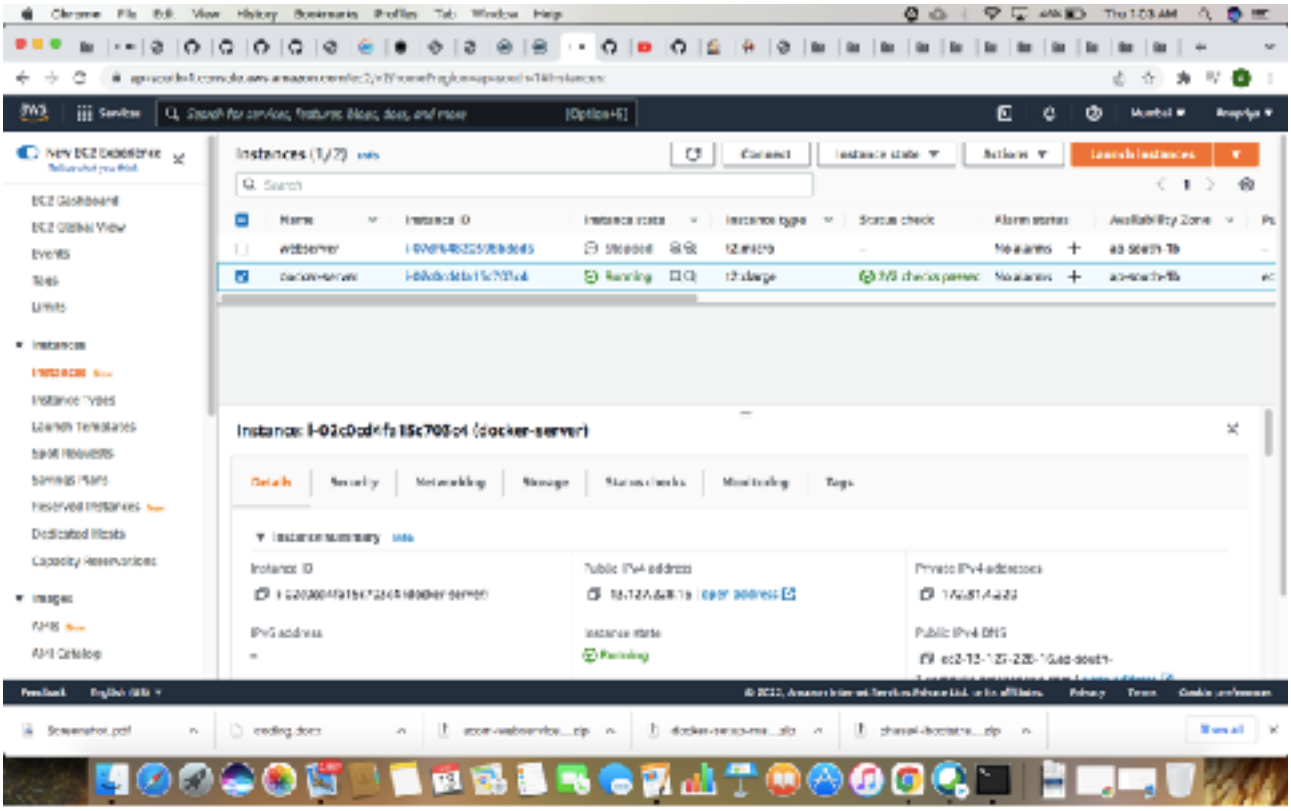
Disk Storage selection 20 Gb for the ubuntu instance



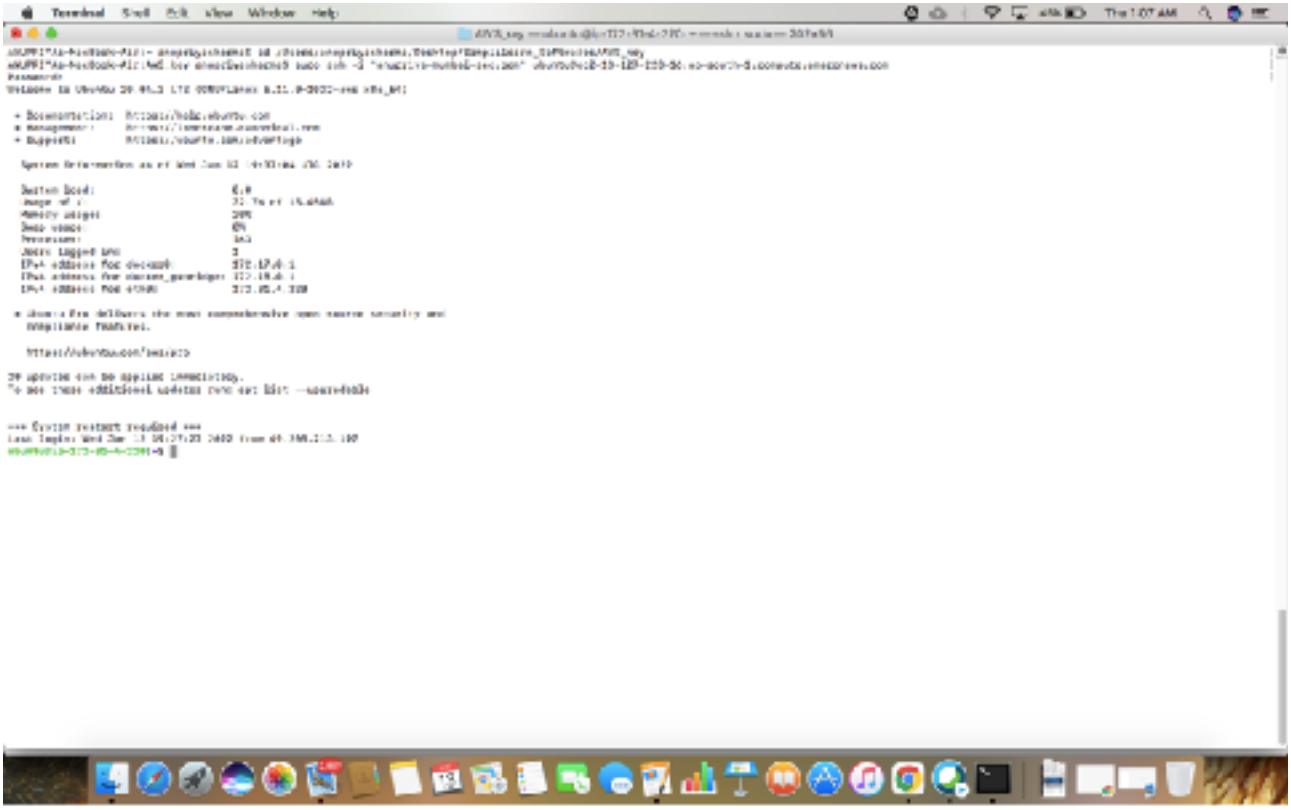
Network /port configuration for SSH and Web application etc.



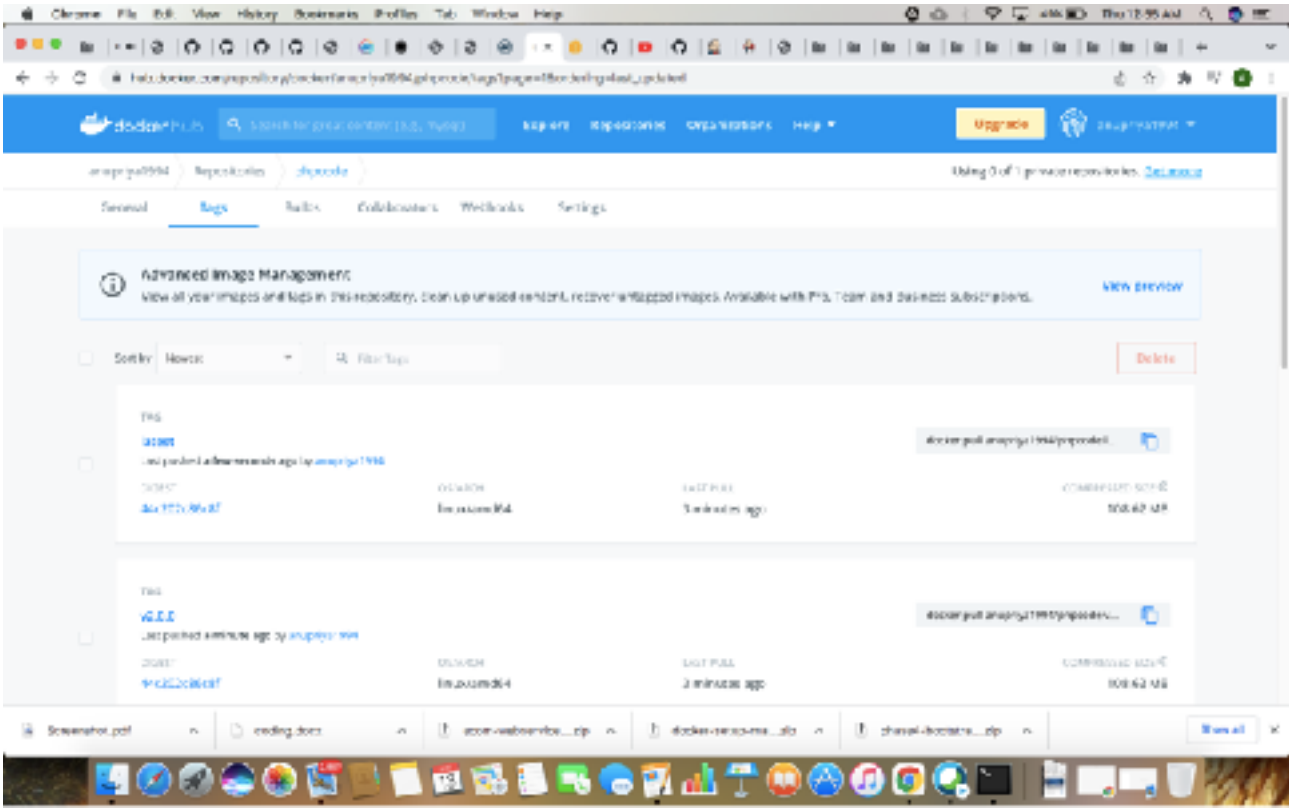
Instance created with the configurations



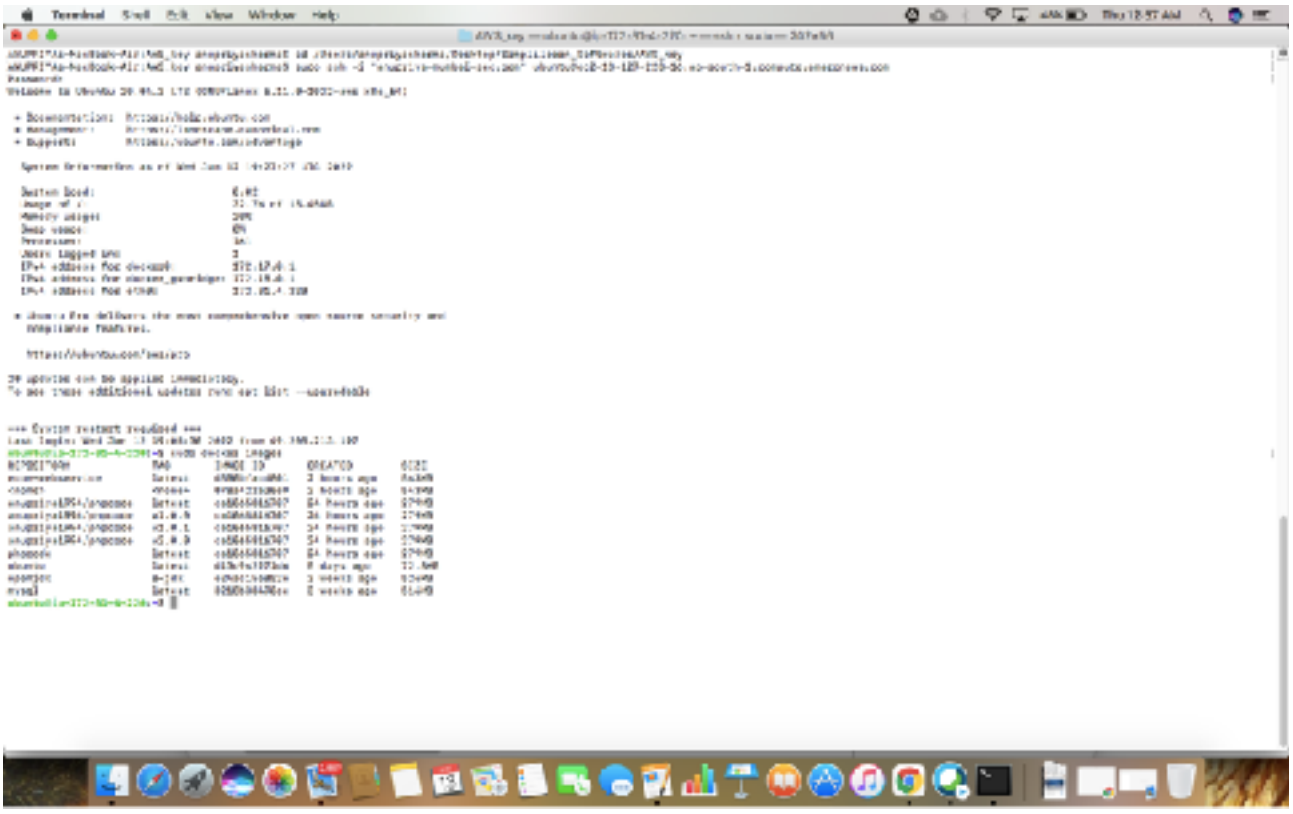
Connected using Terminal (SSH client)



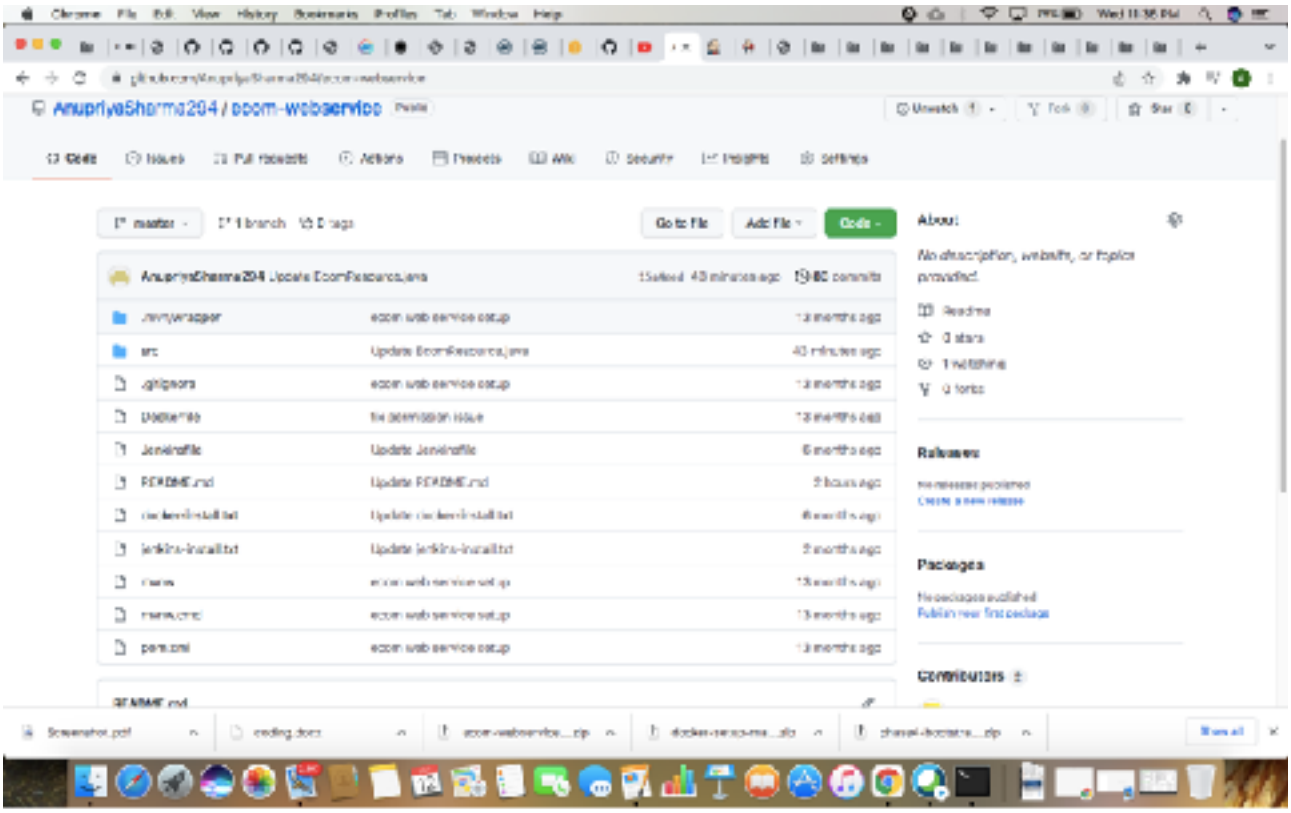
2. Docker



DOCKER HUB Dashboard and Image pushed via command line

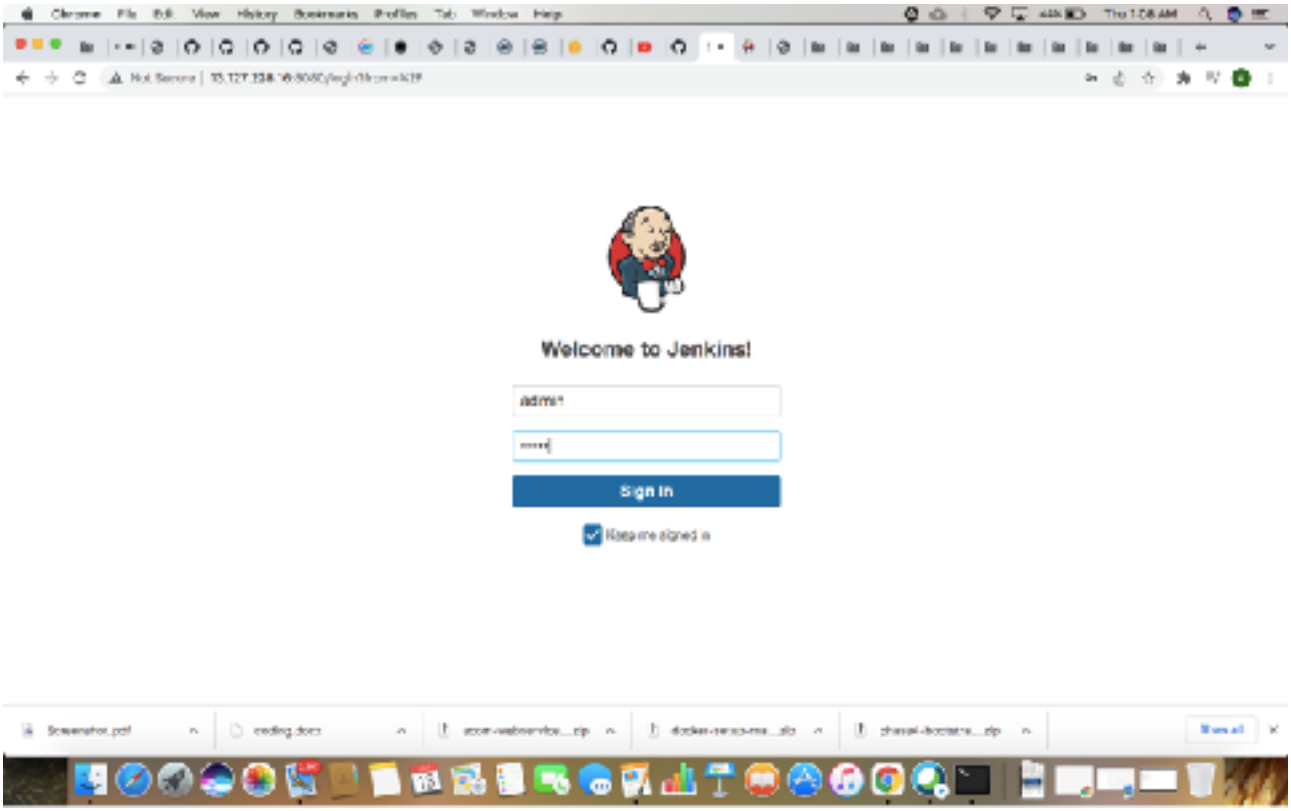


3.**Github** (Sample Spring boot application)

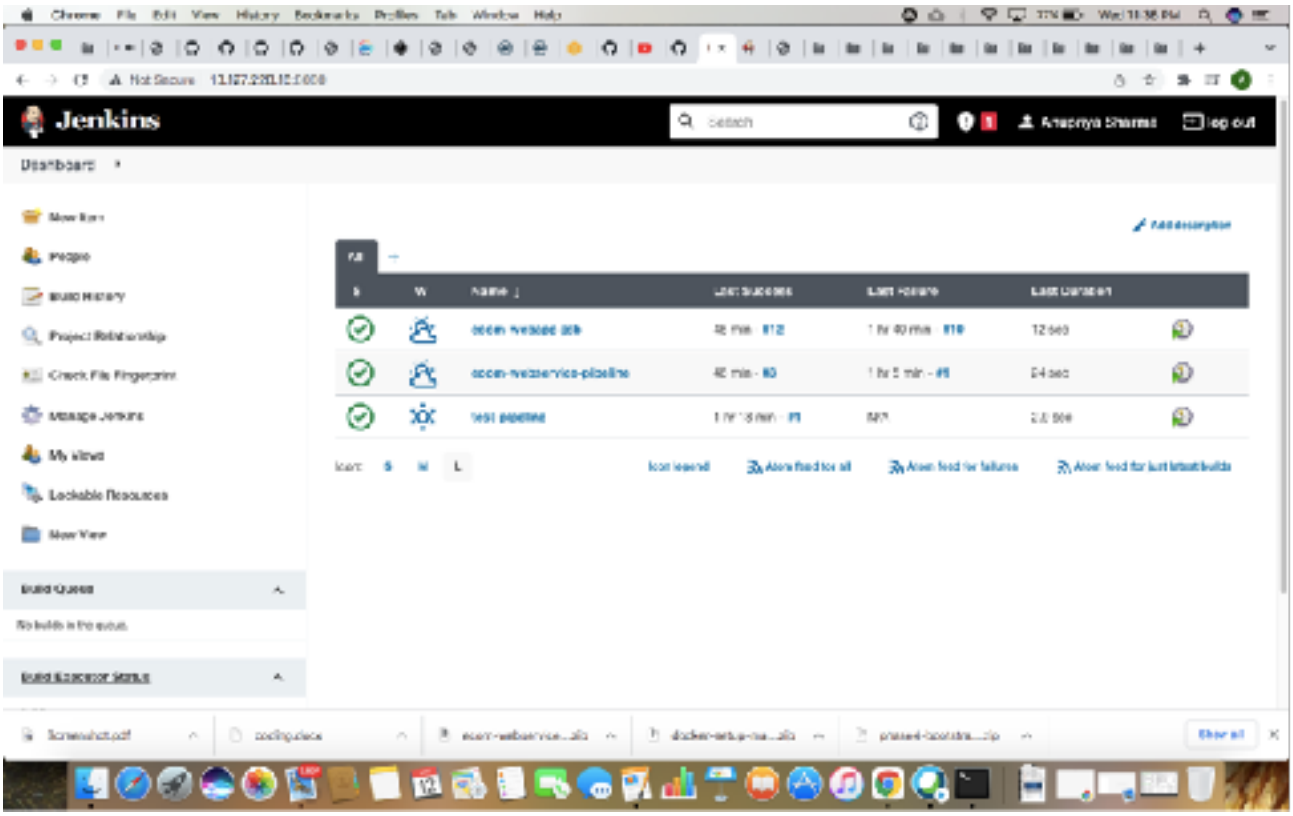


4. Jenkins Setup

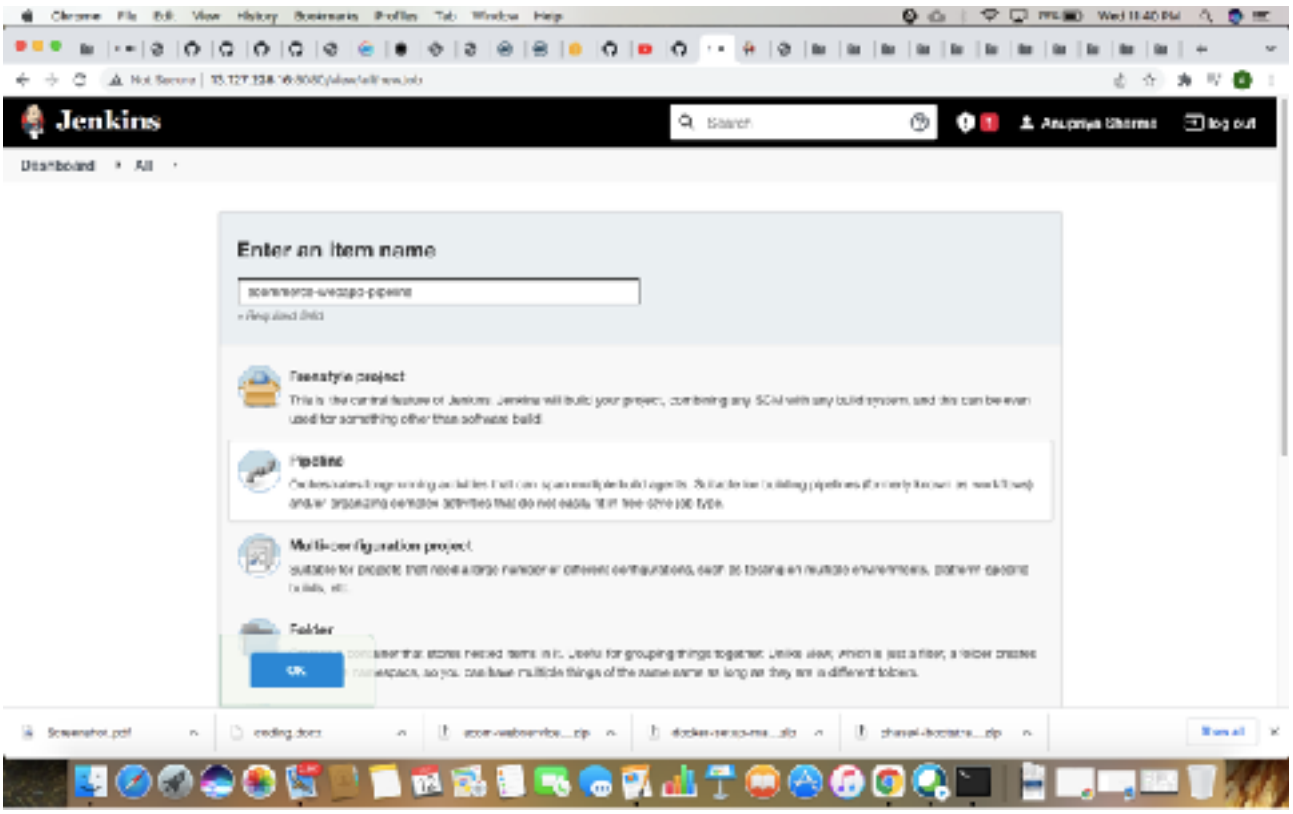
Access using ec2 public ip:8080 port http://13.127.228.16:8080/login?from=%2F



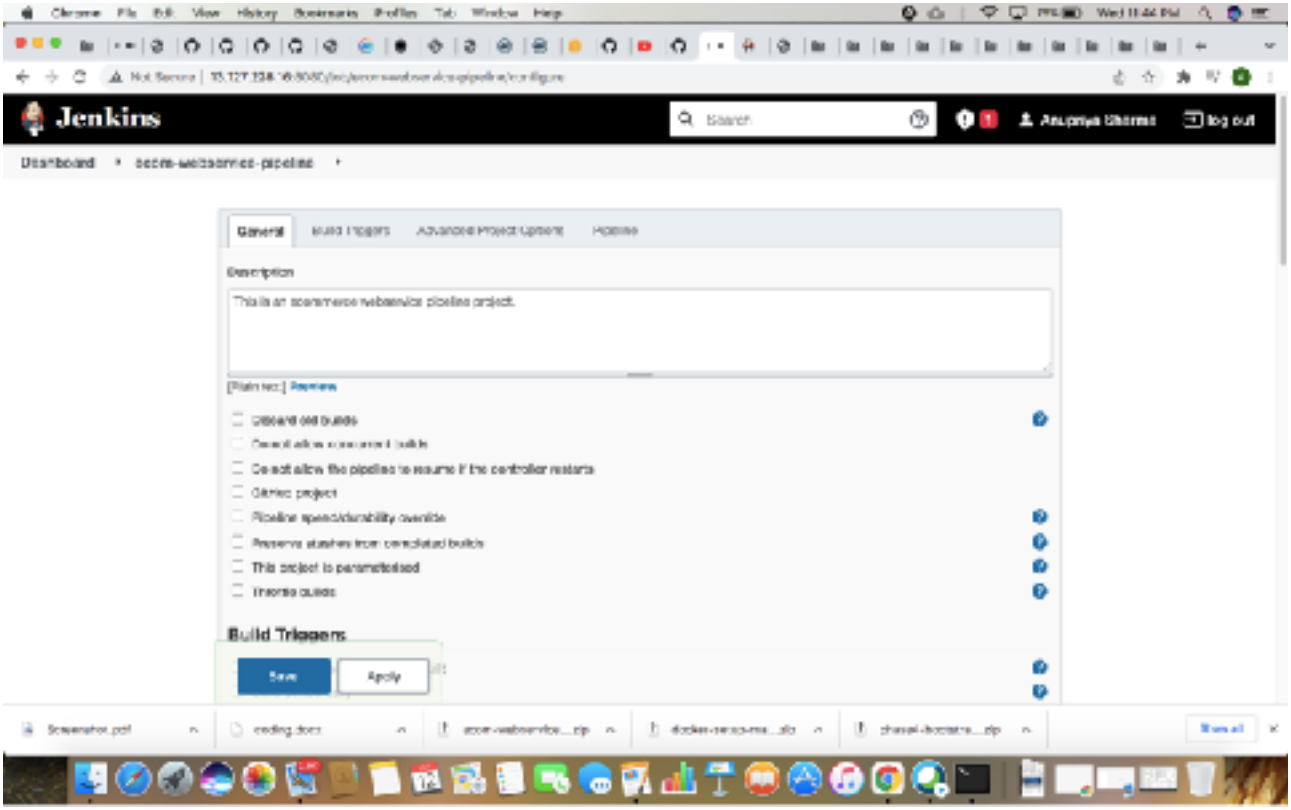
once login it will show dashboard page



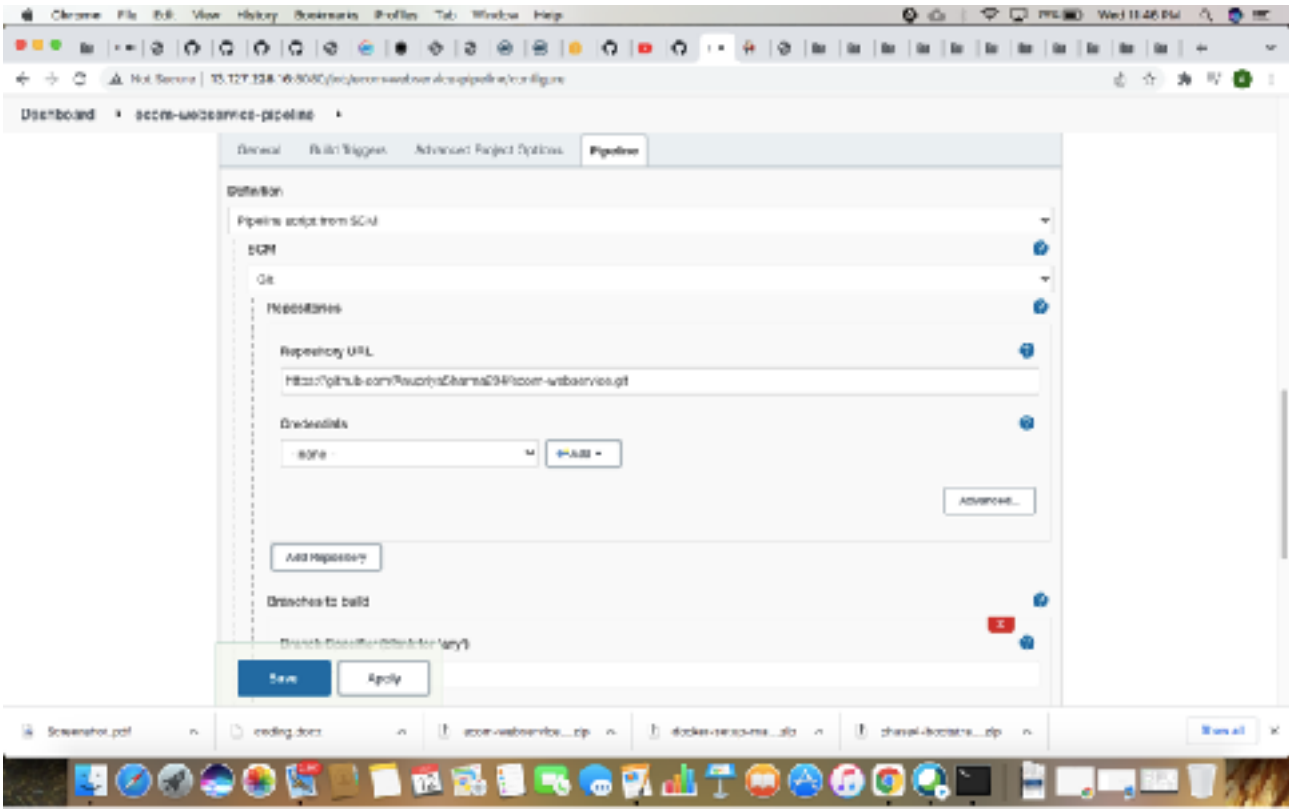
Click on new item to add new Pipeline.
Enter the name and select pipeline project below



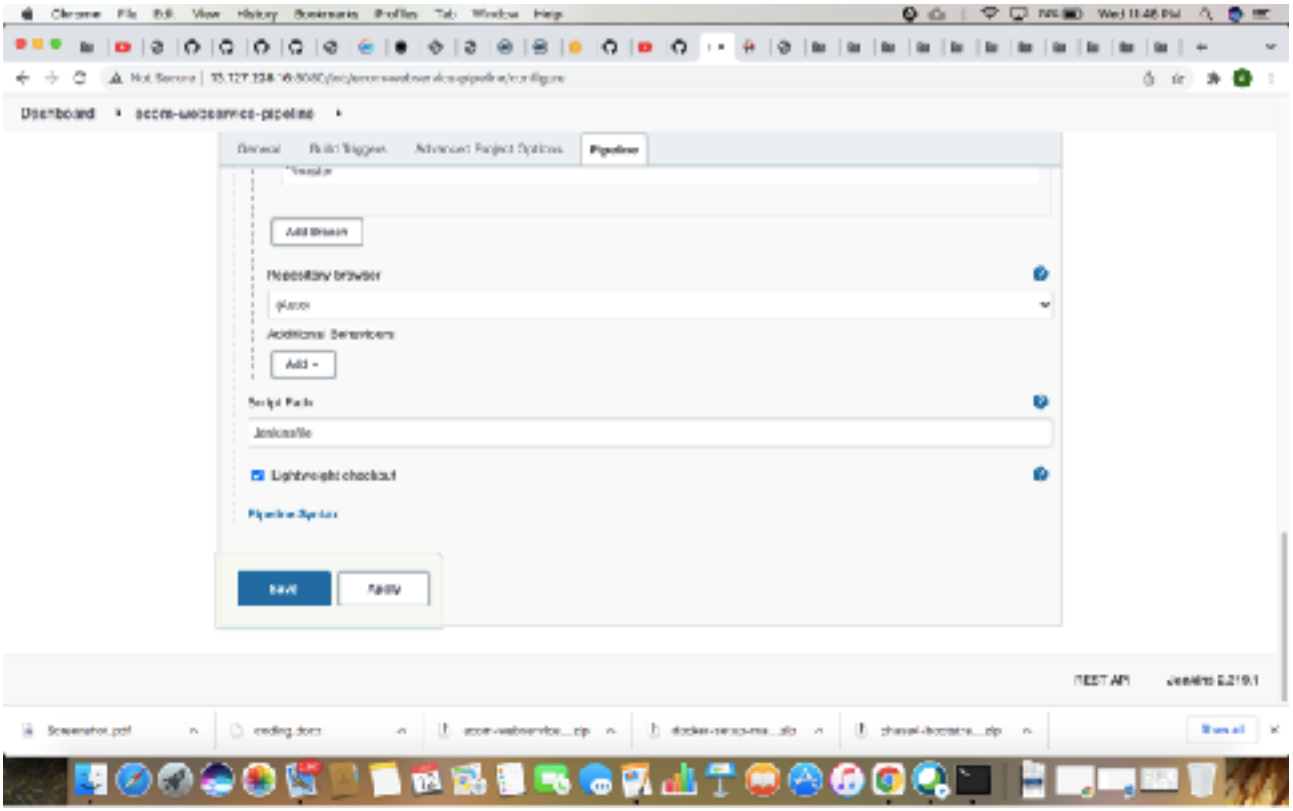
Enter description



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') {

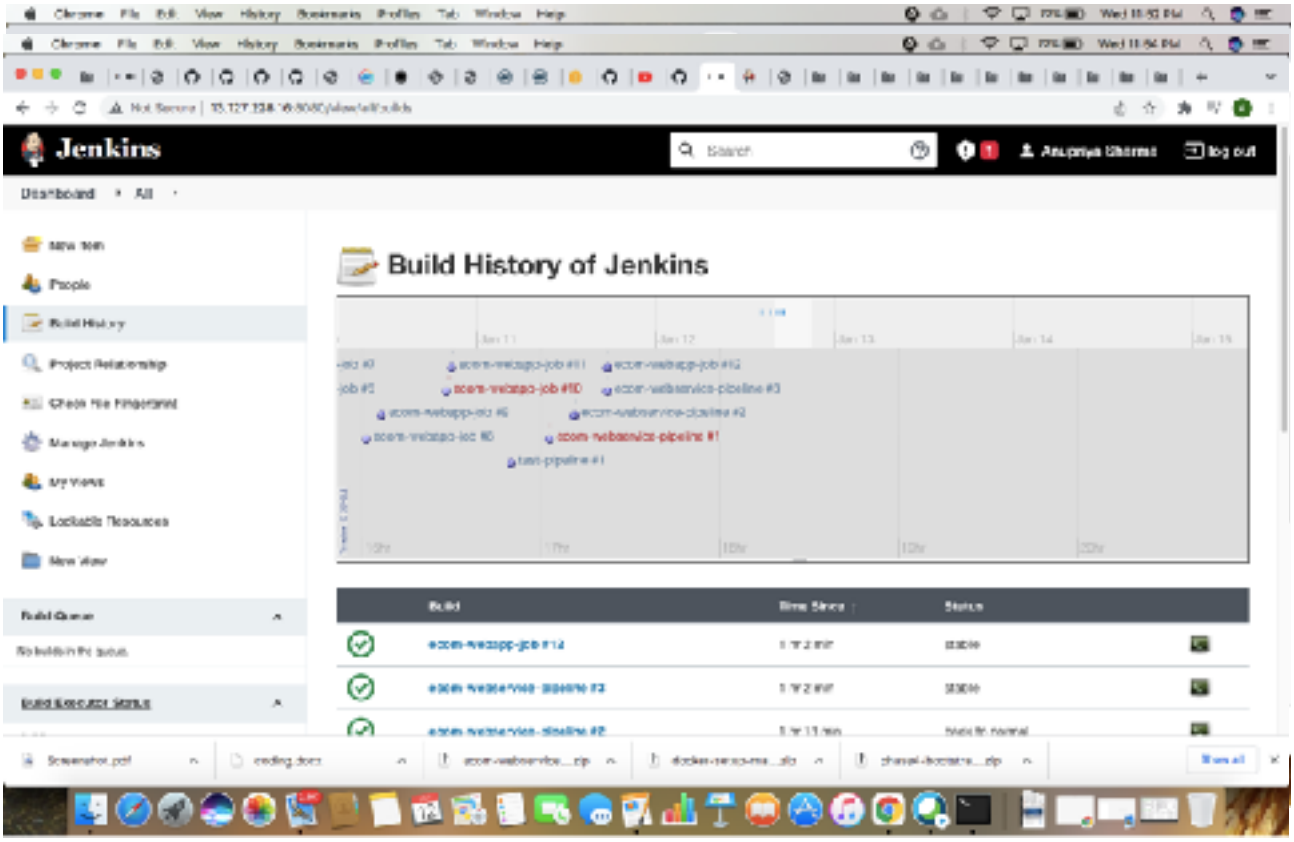
steps {

```
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 deploment 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 \
```

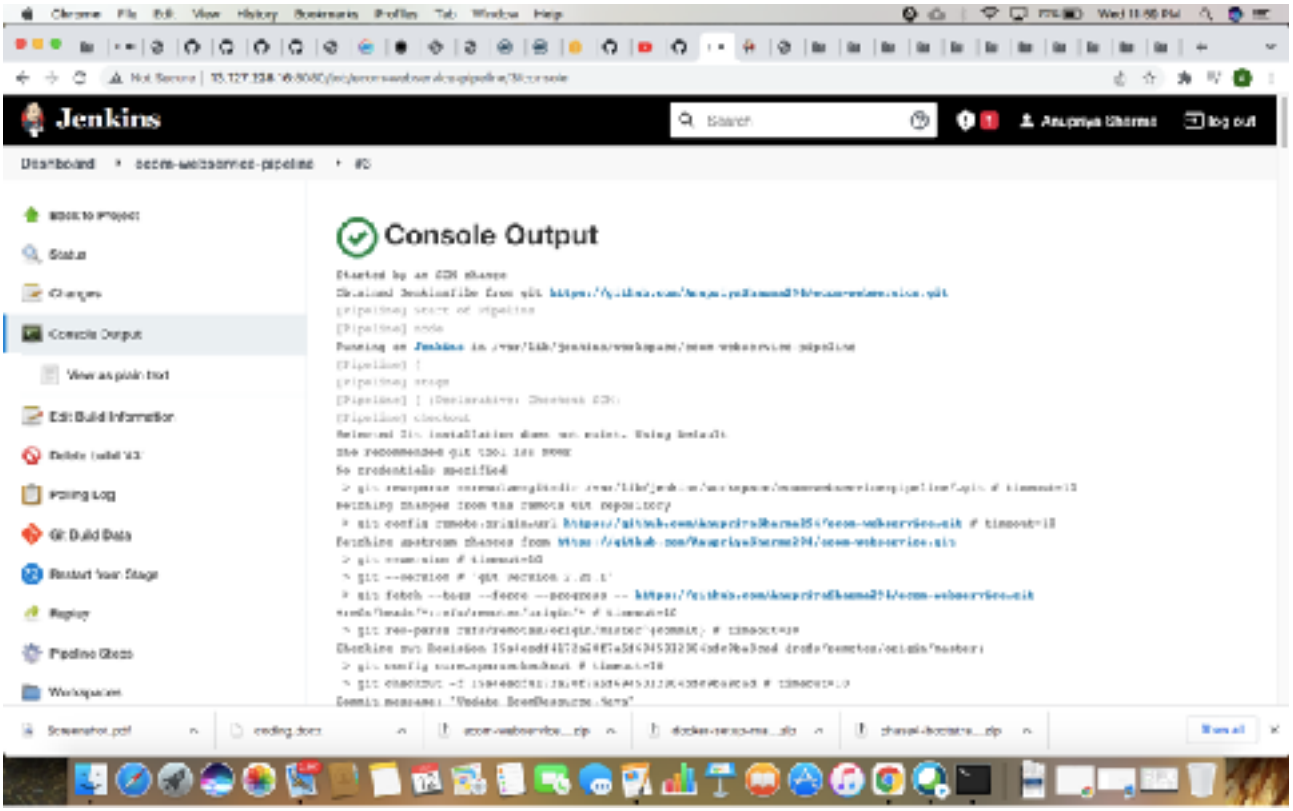
```
echo OK; \
fi;);
docker container run --restart always --name ecom-webservice -p 8081:8081 -d ecom-webservice
'''
}
}
}
}
```

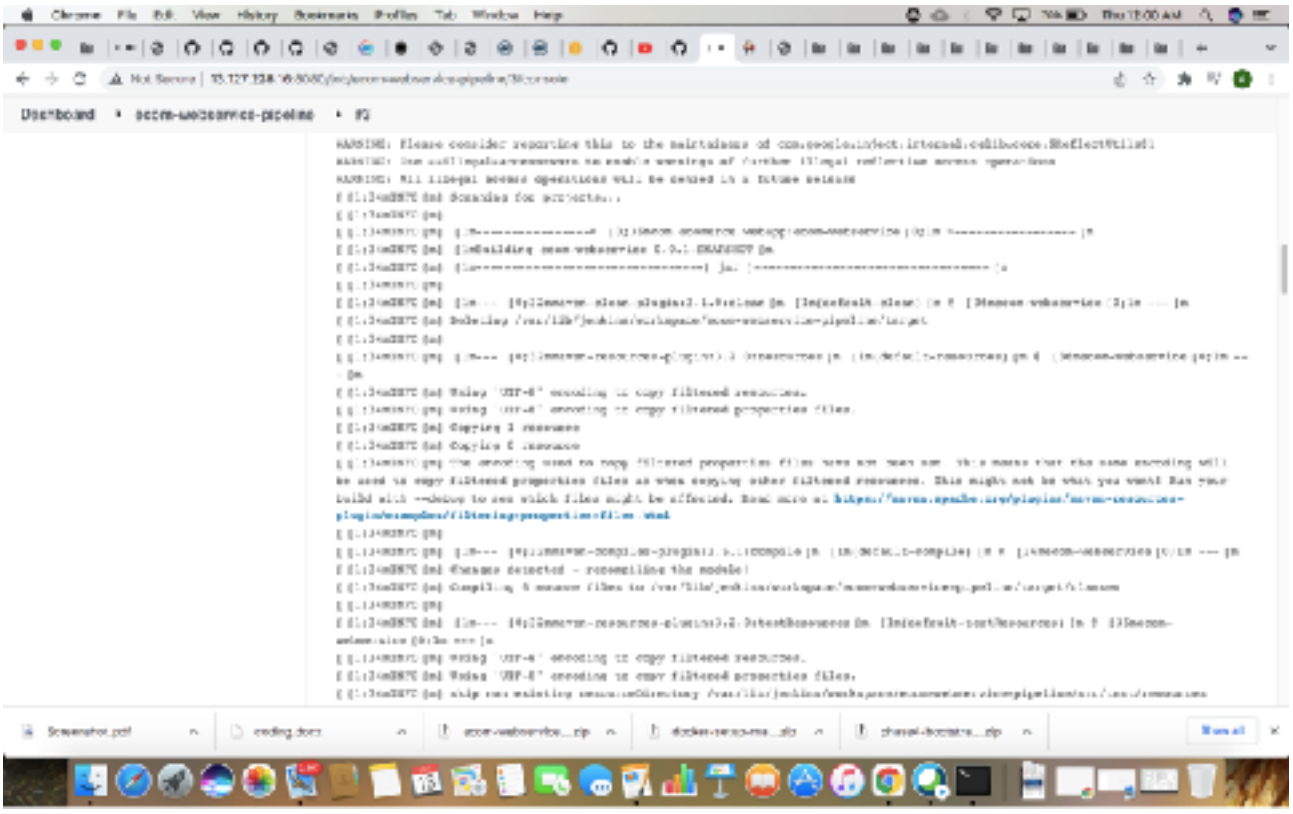
After build it will show the progress in different phases as shown below

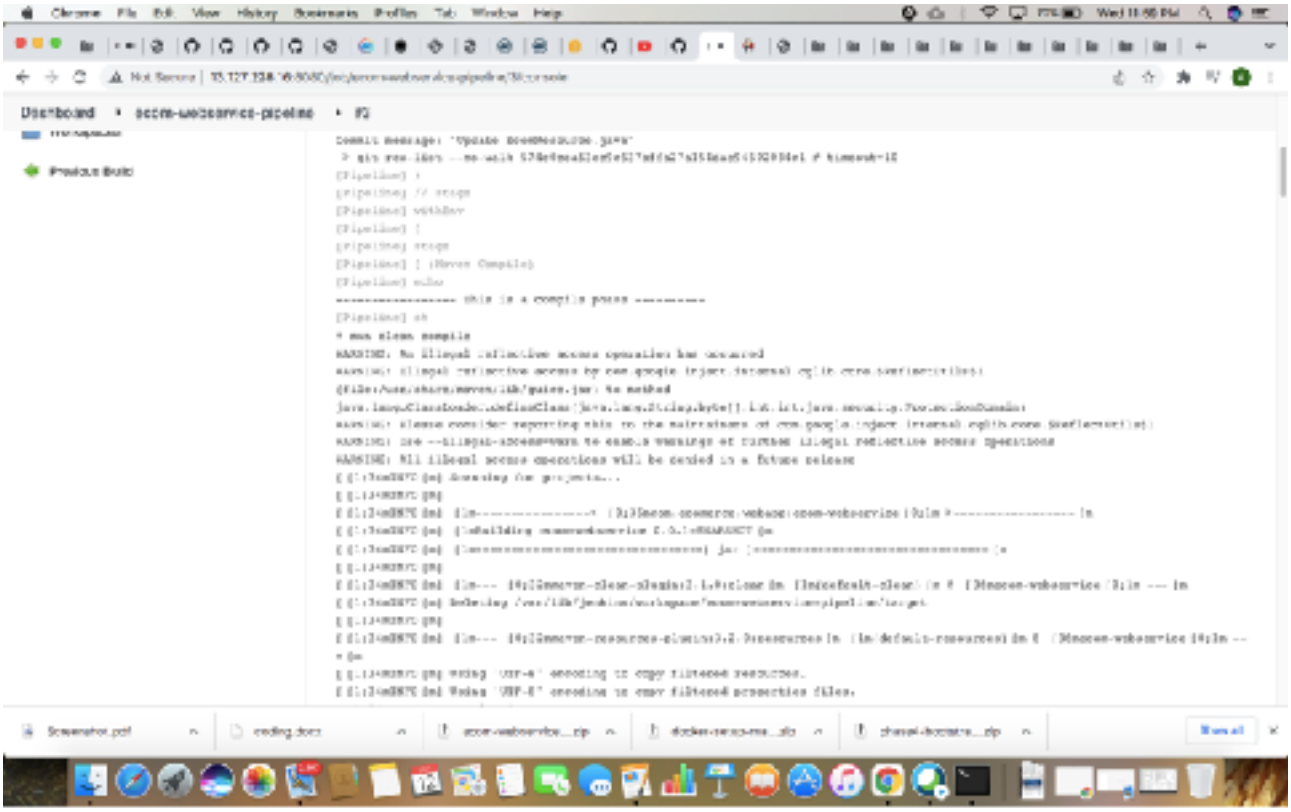
Build History



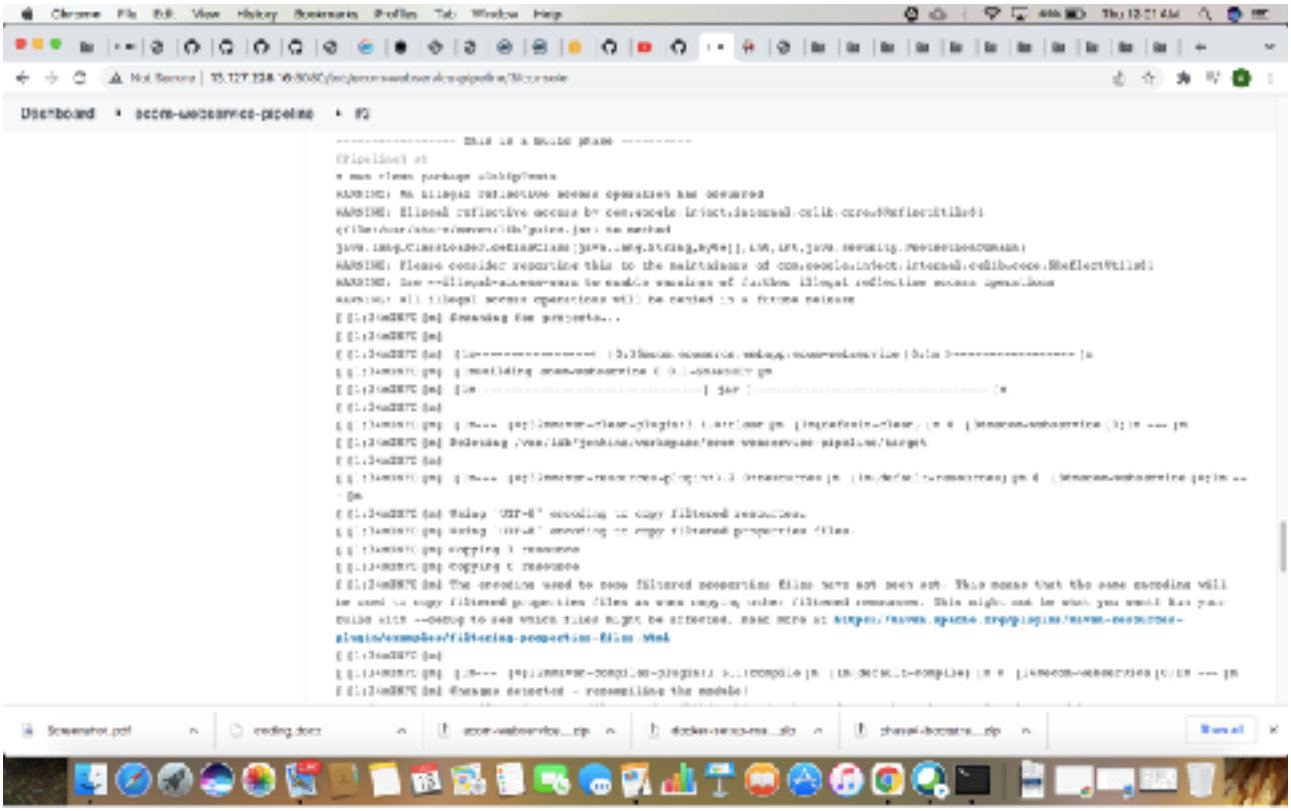
Console Output

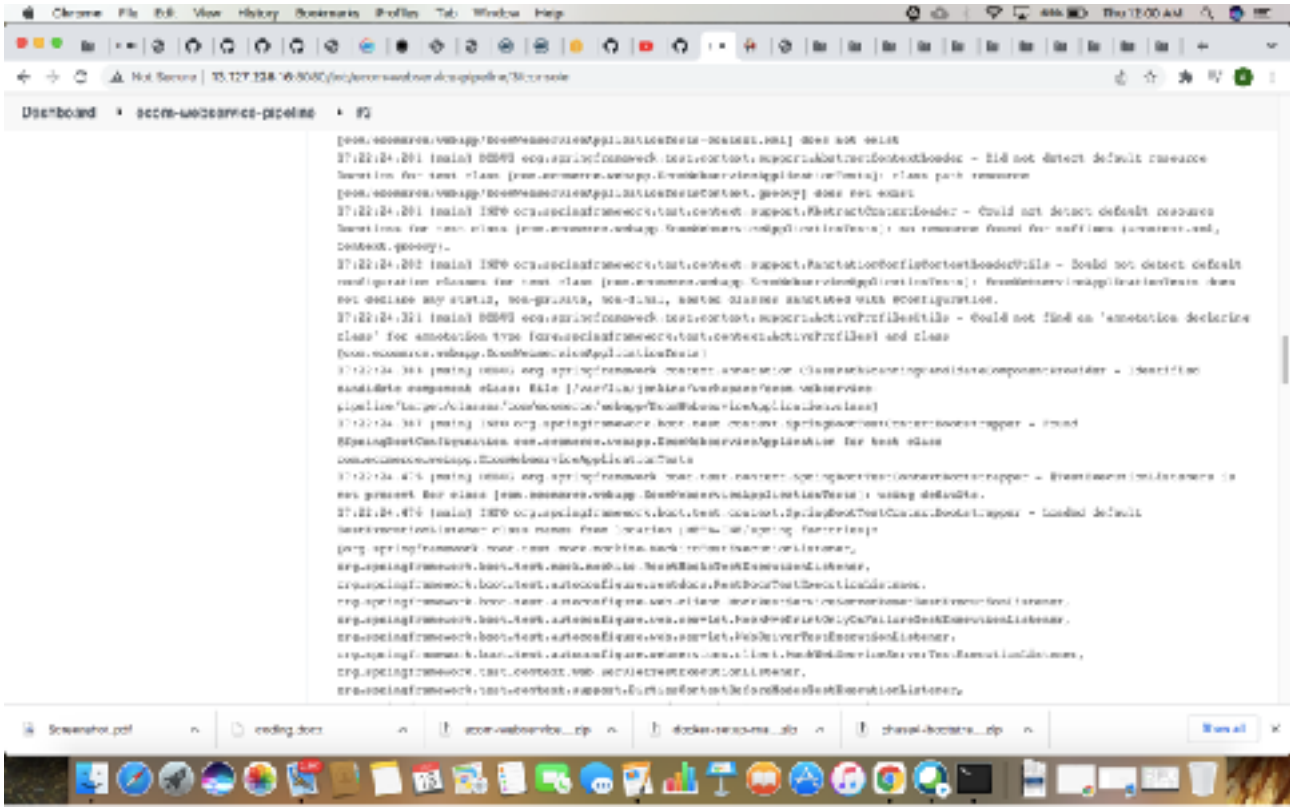


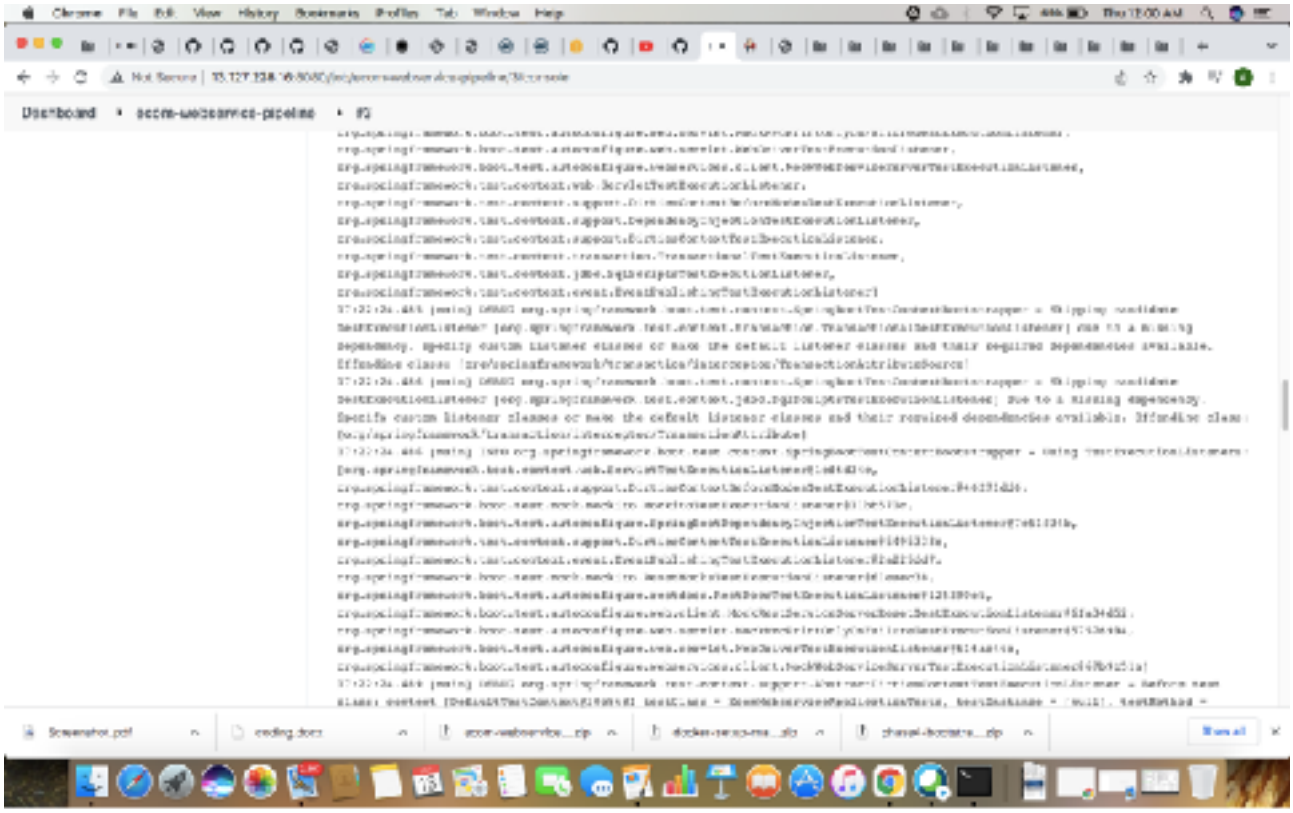


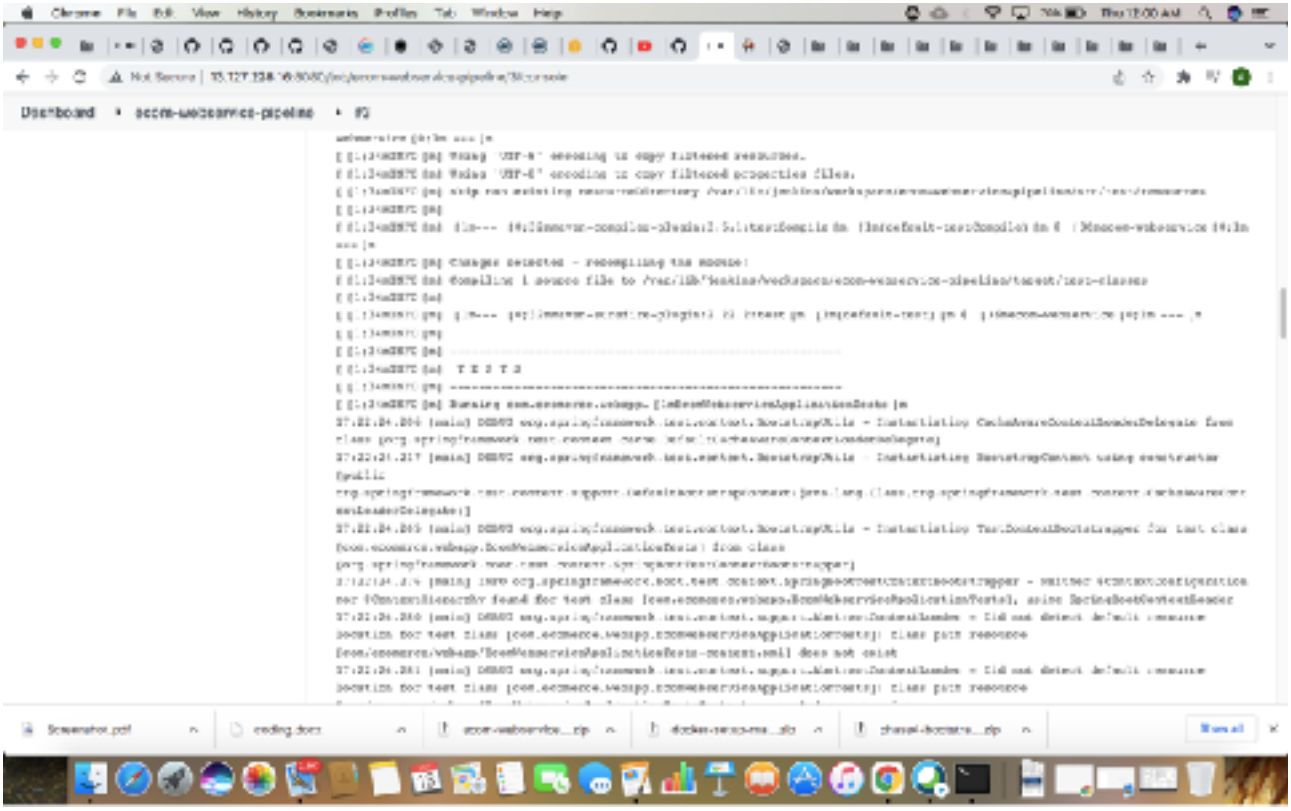

Dashboard
ecom-webservice-pipeline
#2
Previous Build

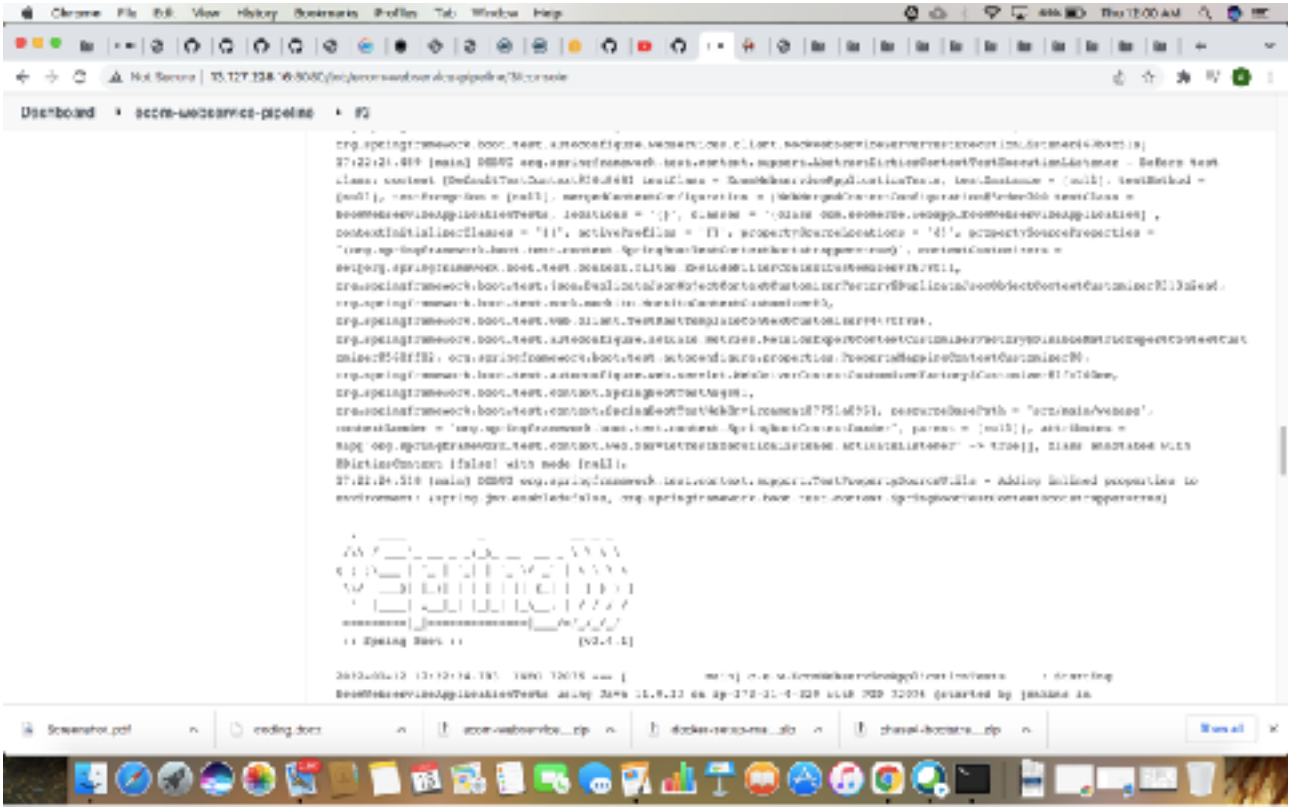

Dashboard
ecom-webservice-pipeline
#2

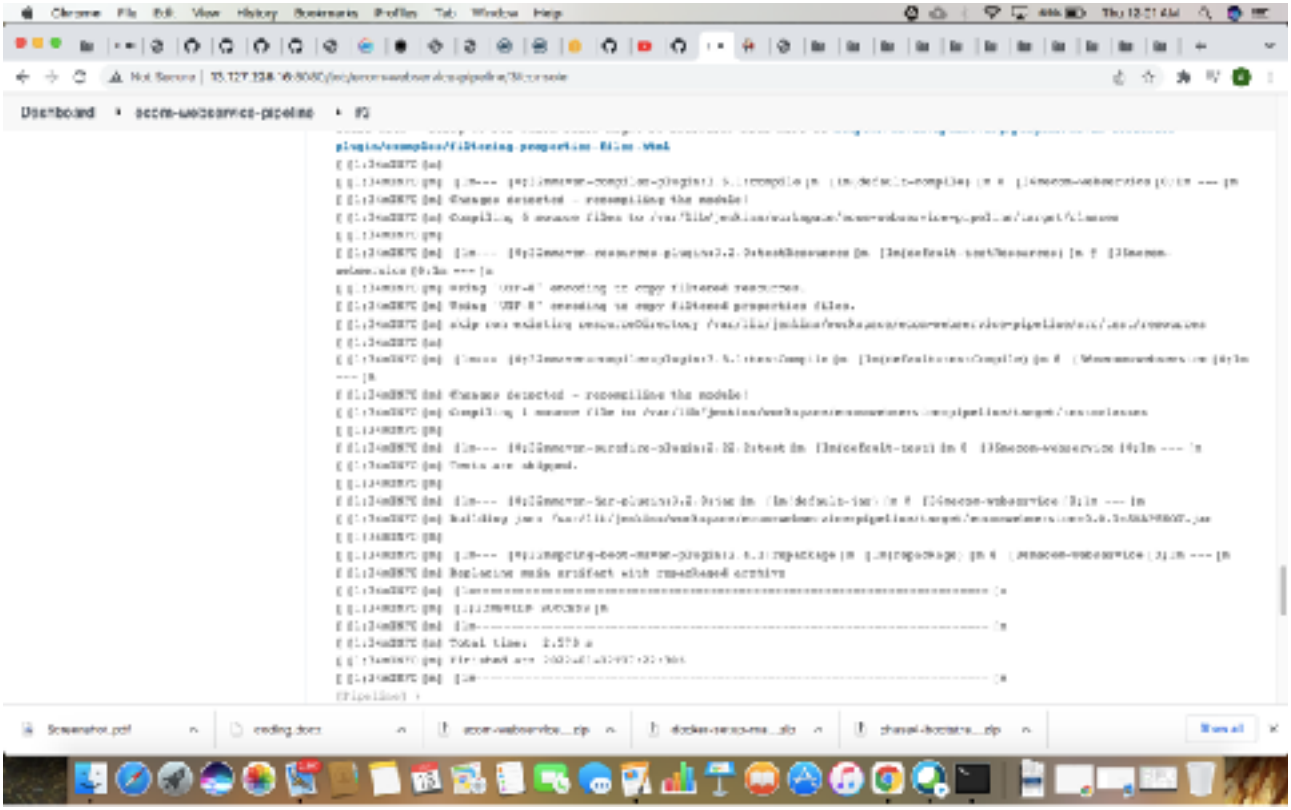




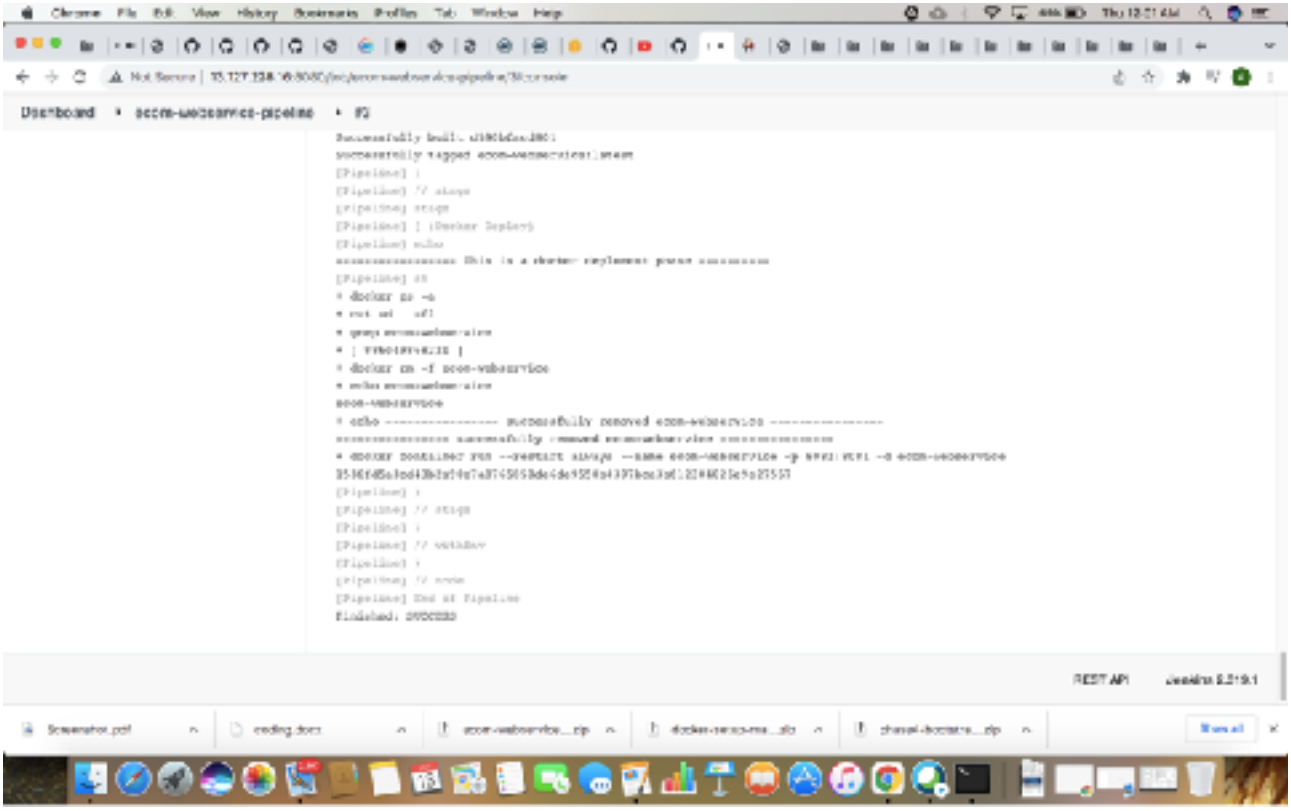



Dashboard
ecom-webservice-pipeline
#2

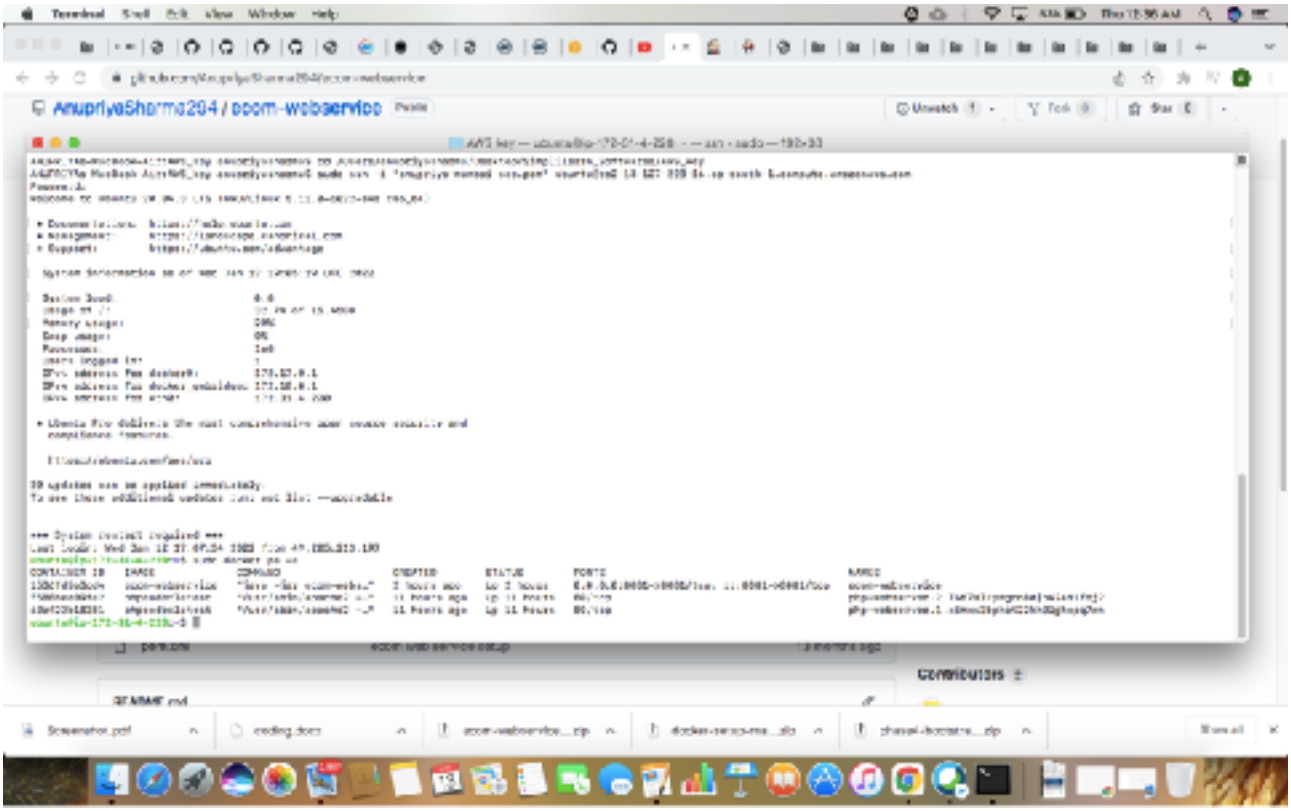

Dashboard
ecom-webservice-pipeline
#2


Dashboard
ecom-webservice-pipeline
#2
plugin/compiler/filtering properties files.xhtml
Changes detected - recompiling the module!
Tests are skipped.
Replacing main artifact with repackaged archive
Total time: 2.273 s

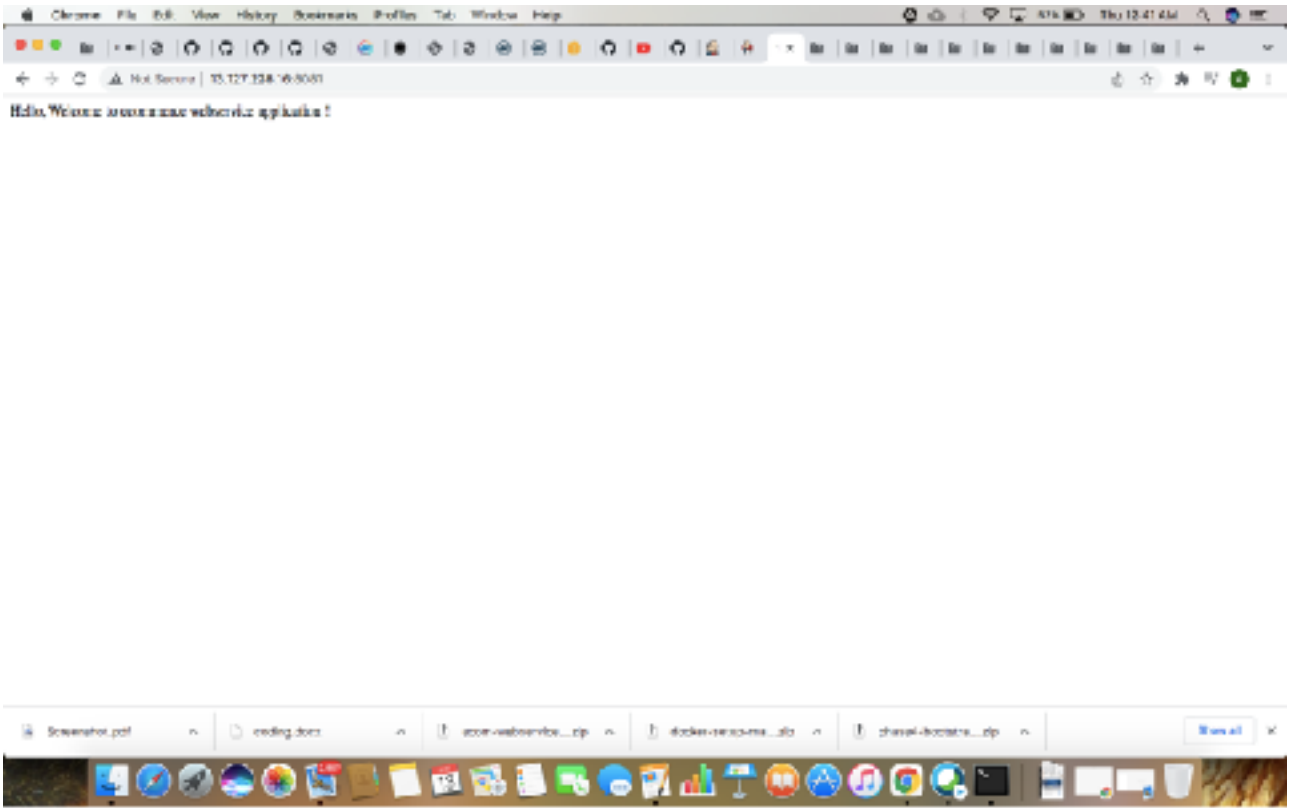

Dashboard
ecom-webservice-pipeline
#2
---------- This is a build docker image phase ----------
+ docker image build -t ecom-webservice .
Sending build context to Docker daemon 17.34kB
Step 1/4 : FROM openjdk:8-jdk
Step 2/4 : COPY ./target/ecom-webservice-0.0.1-SNAPSHOT.jar ecom-webservice-0.0.1-SNAPSHOT.jar
Step 3/4 : CMD ["java" "-jar" "ecom-webservice-0.0.1-SNAPSHOT.jar"]
Successfully built d1901bfdc3801
Successfully tagged ecom-webservice:latest
---------- This is a docker container phase ----------
+ docker ps -a



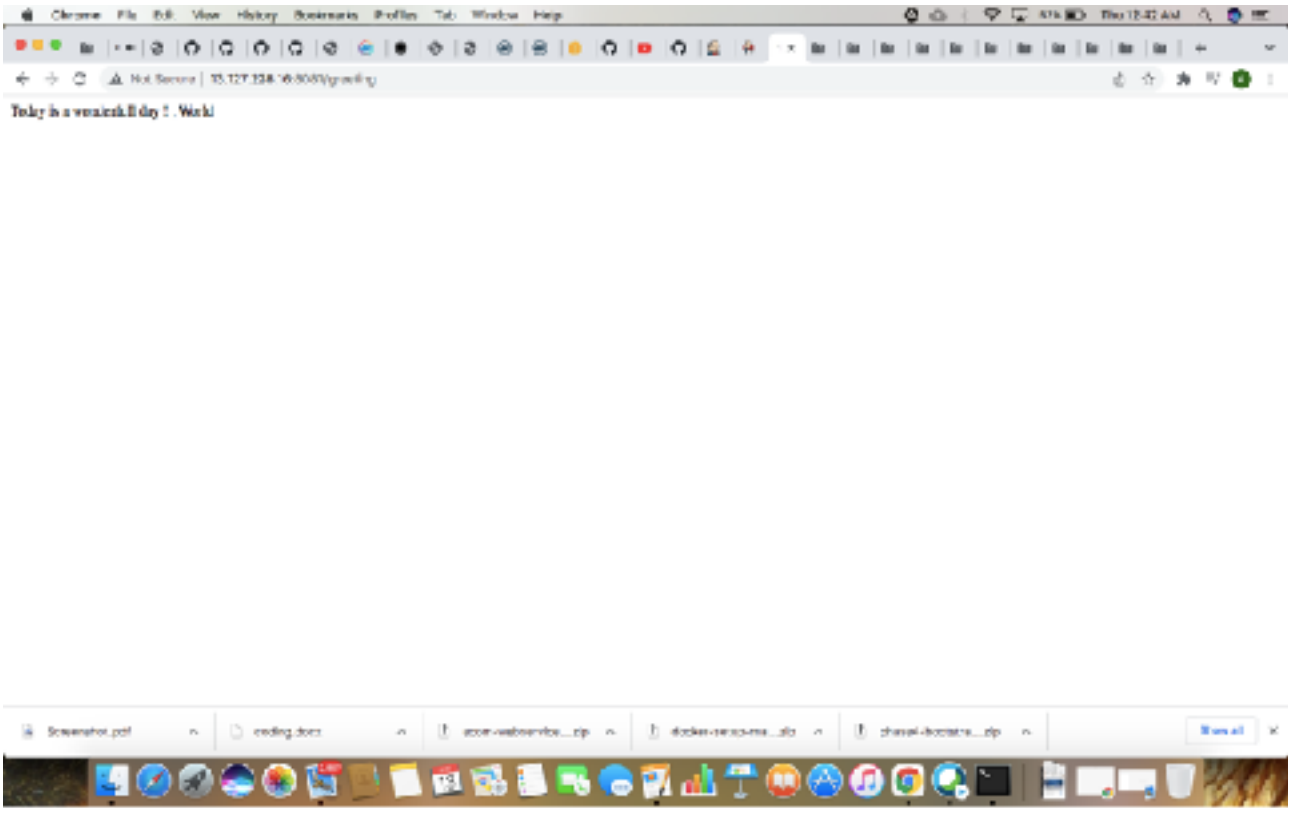
Deployed docker container can be viewed



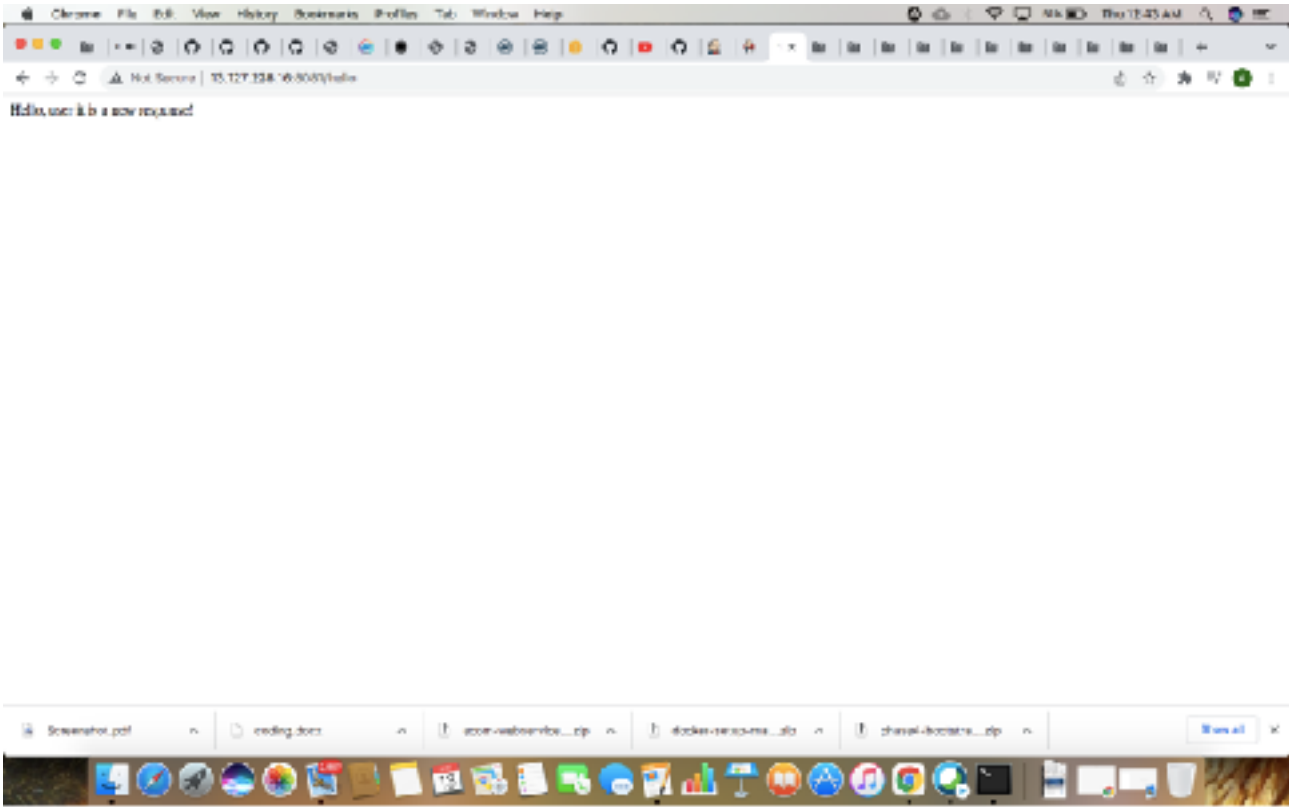
WebApplication can be accessed by port 8081 as shown below:



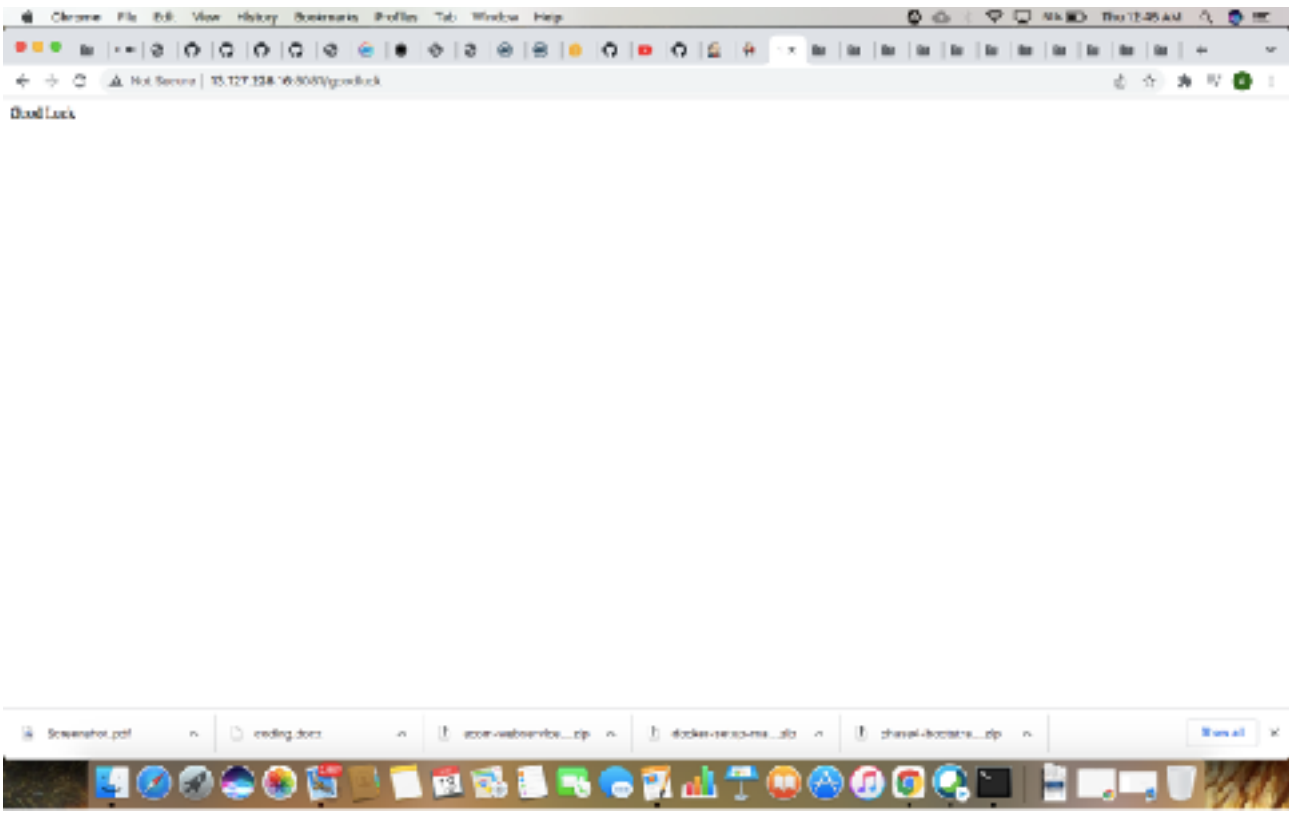
/greeting



/hello



/goodluck



Workspace View