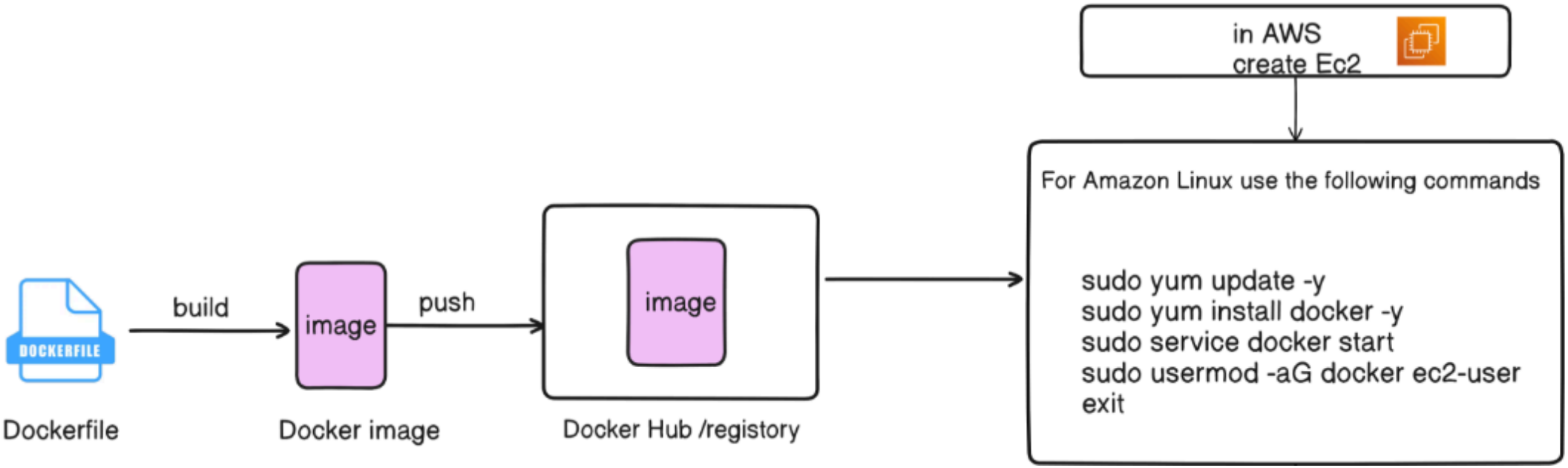
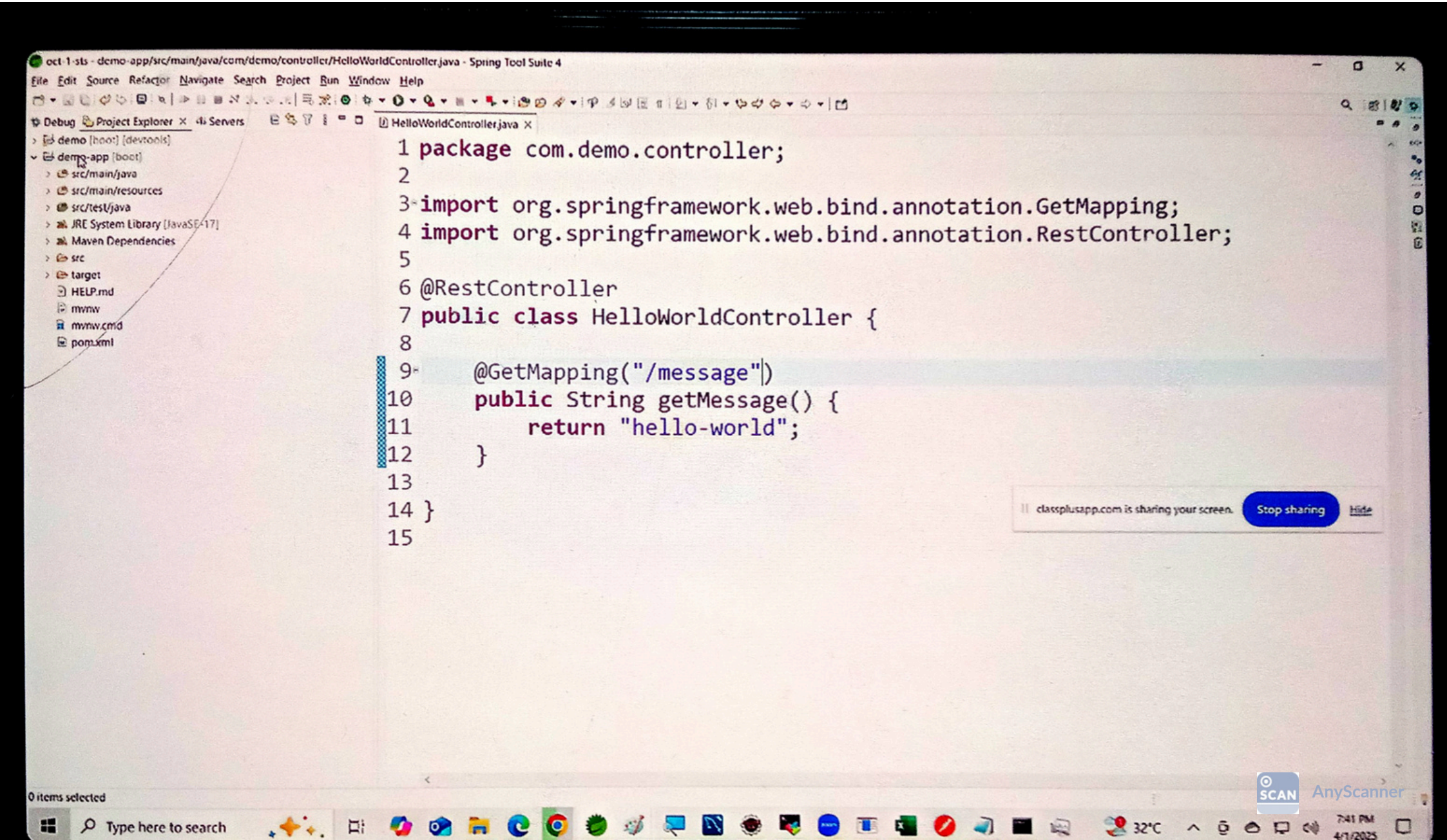
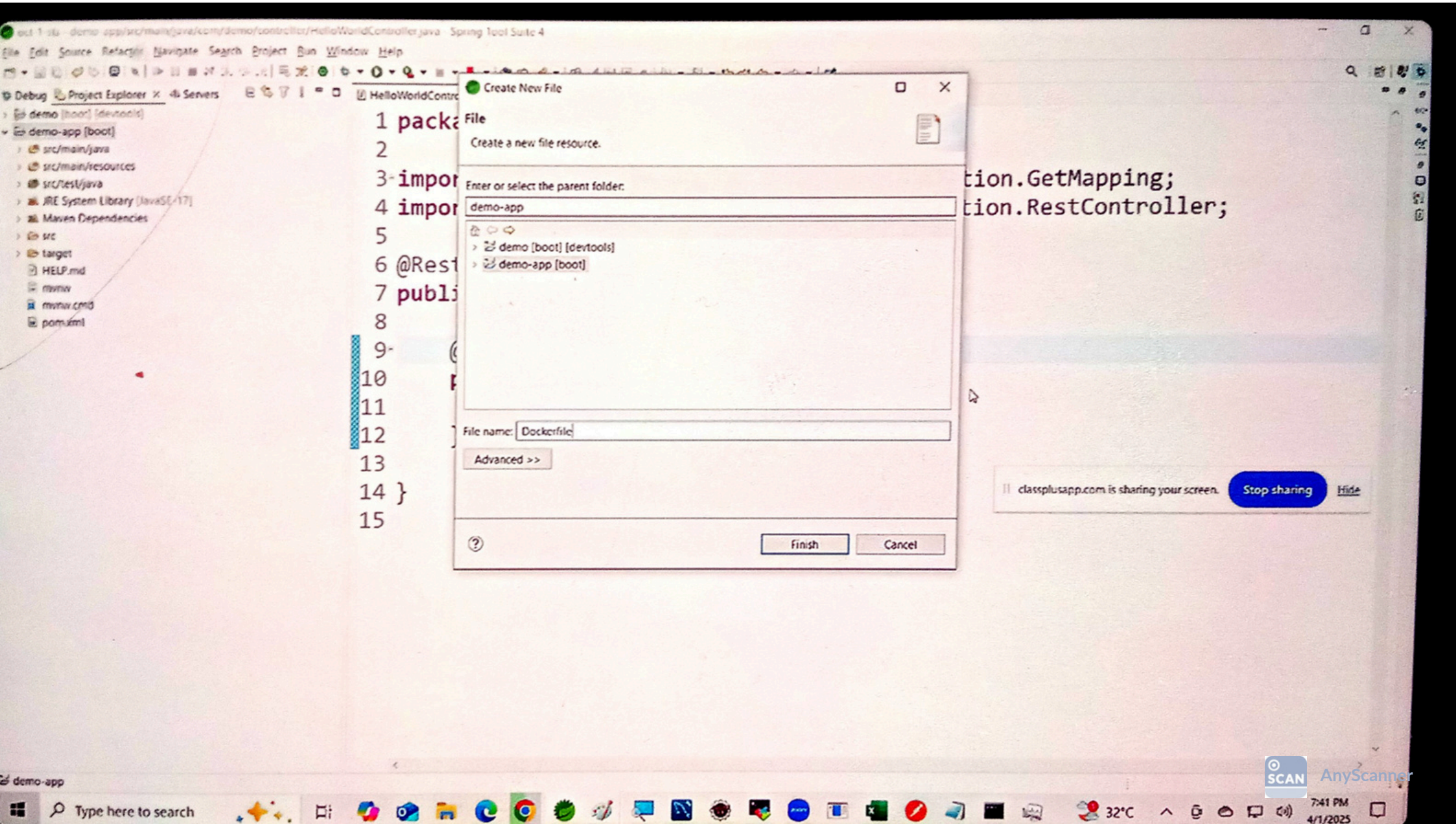


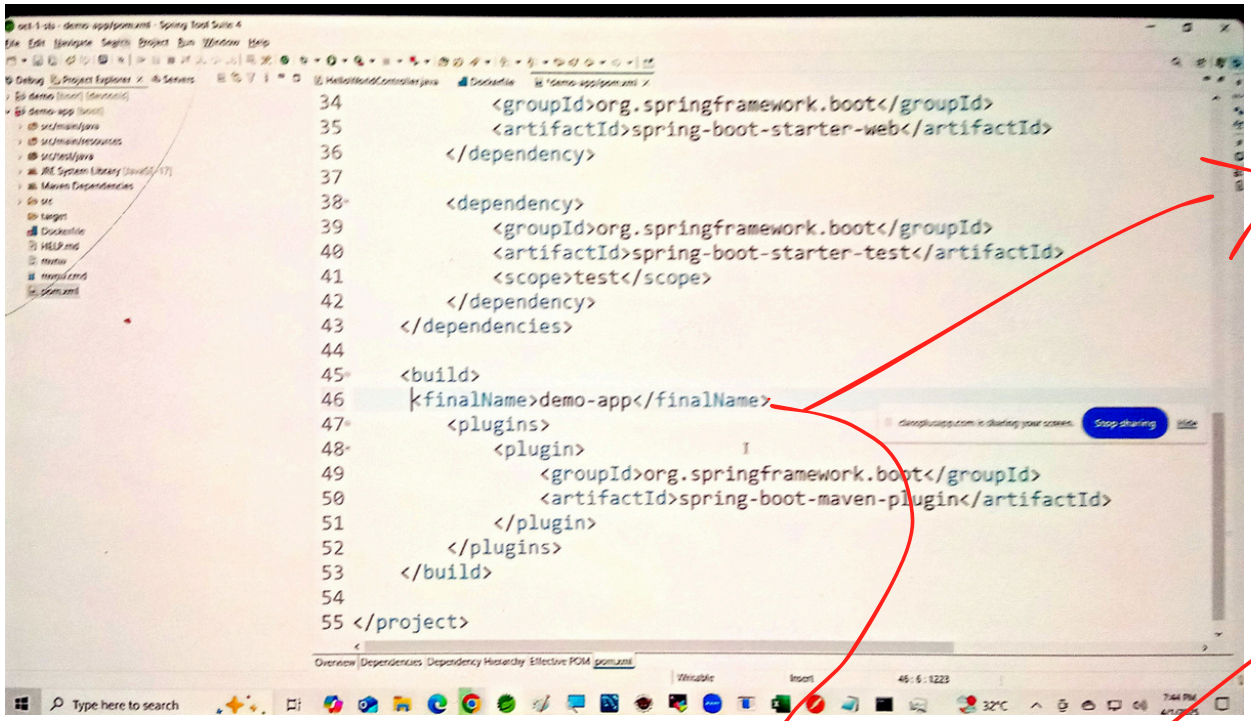
1 create spring boot project



2 in spring boot project create Dockerfile

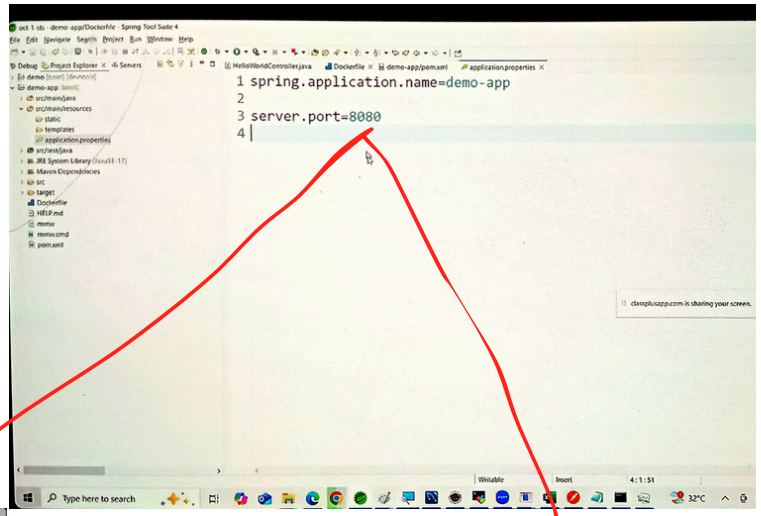


3 configure Dockerfile as shown below

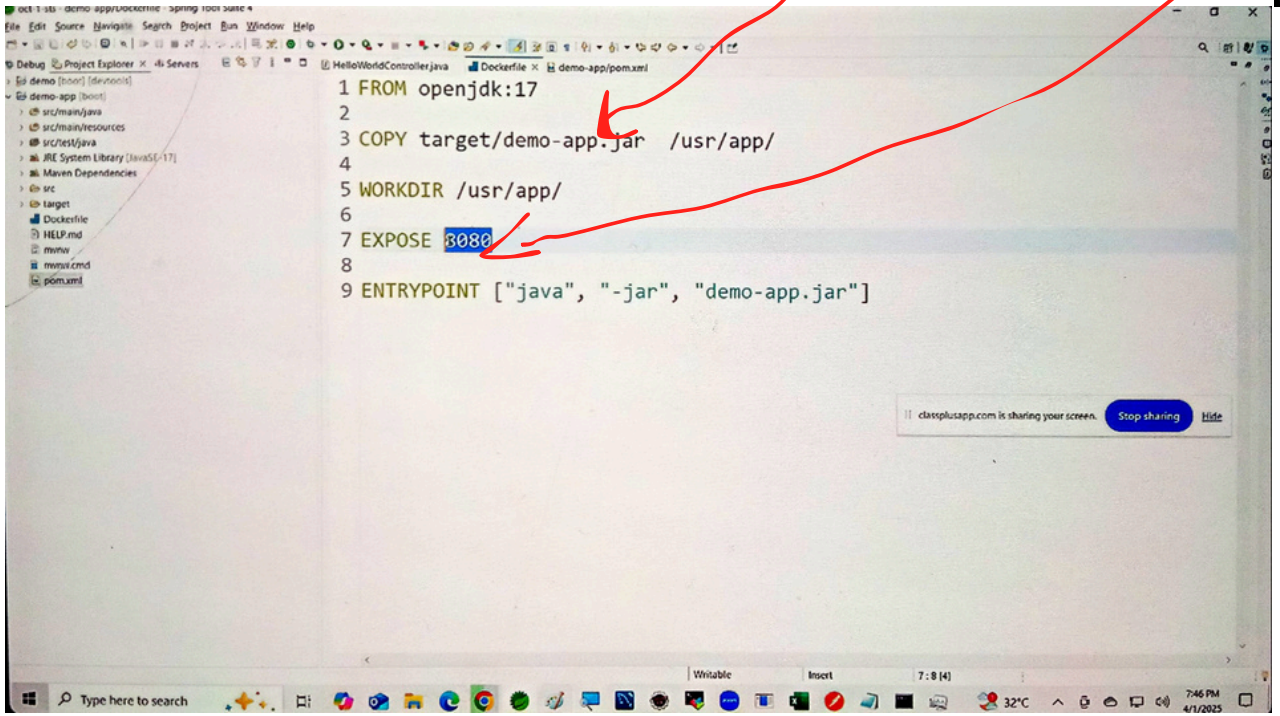


```
34            <groupId>org.springframework.boot</groupId>
35            <artifactId>spring-boot-starter-web</artifactId>
36        </dependency>
37
38        <dependency>
39            <groupId>org.springframework.boot</groupId>
40            <artifactId>spring-boot-starter-test</artifactId>
41            <scope>test</scope>
42        </dependency>
43    </dependencies>
44
45    <build>
46        <finalName>demo-app</finalName>
47        <plugins>
48            <plugin>
49                <groupId>org.springframework.boot</groupId>
50                <artifactId>spring-boot-maven-plugin</artifactId>
51            </plugin>
52        </plugins>
53    </build>
54
55 </project>
```

Here i am giving my project name



```
1 spring.application.name=demo-app
2
3 server.port=8080
4
```

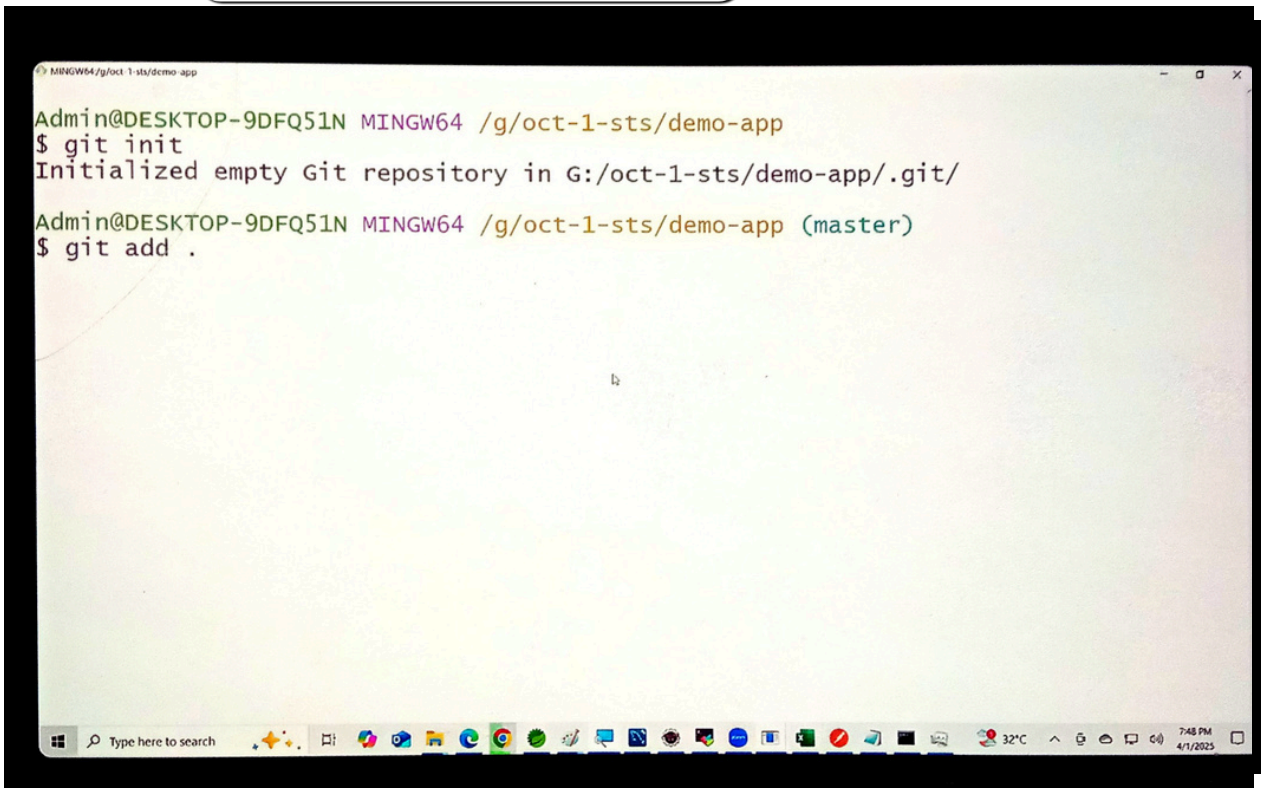


```
1 FROM openjdk:17
2
3 COPY target/demo-app.jar  /usr/app/
4
5 WORKDIR /usr/app/
6
7 EXPOSE 8080
8
9 ENTRYPOINT ["java", "-jar", "demo-app.jar"]
```

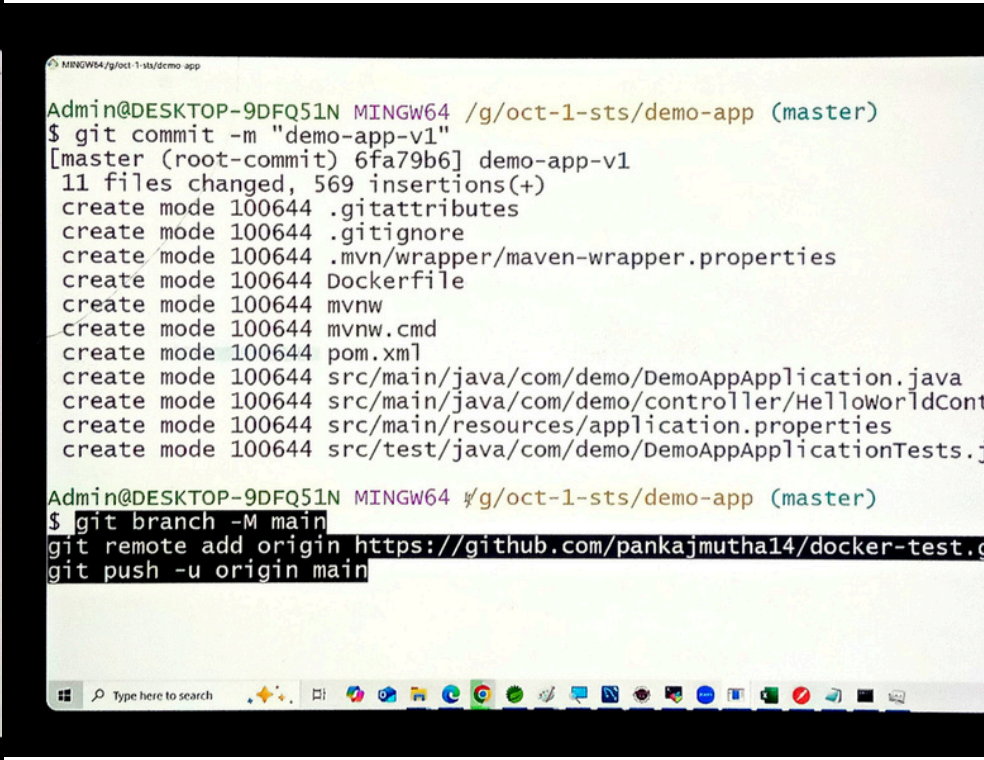
it is optional just reminder which port i'm using

4 go to your project path in your local system and push code in github

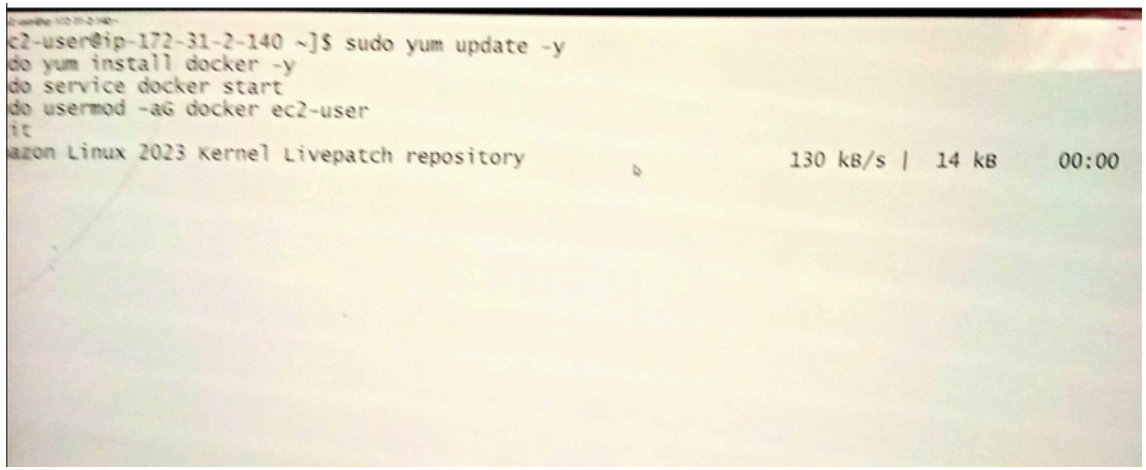
1 step



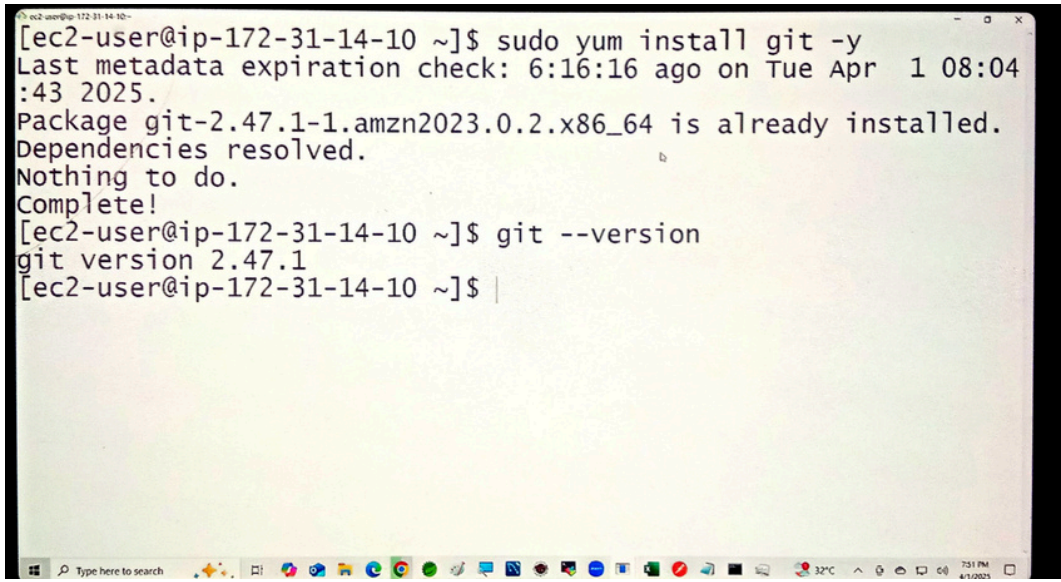
2 step



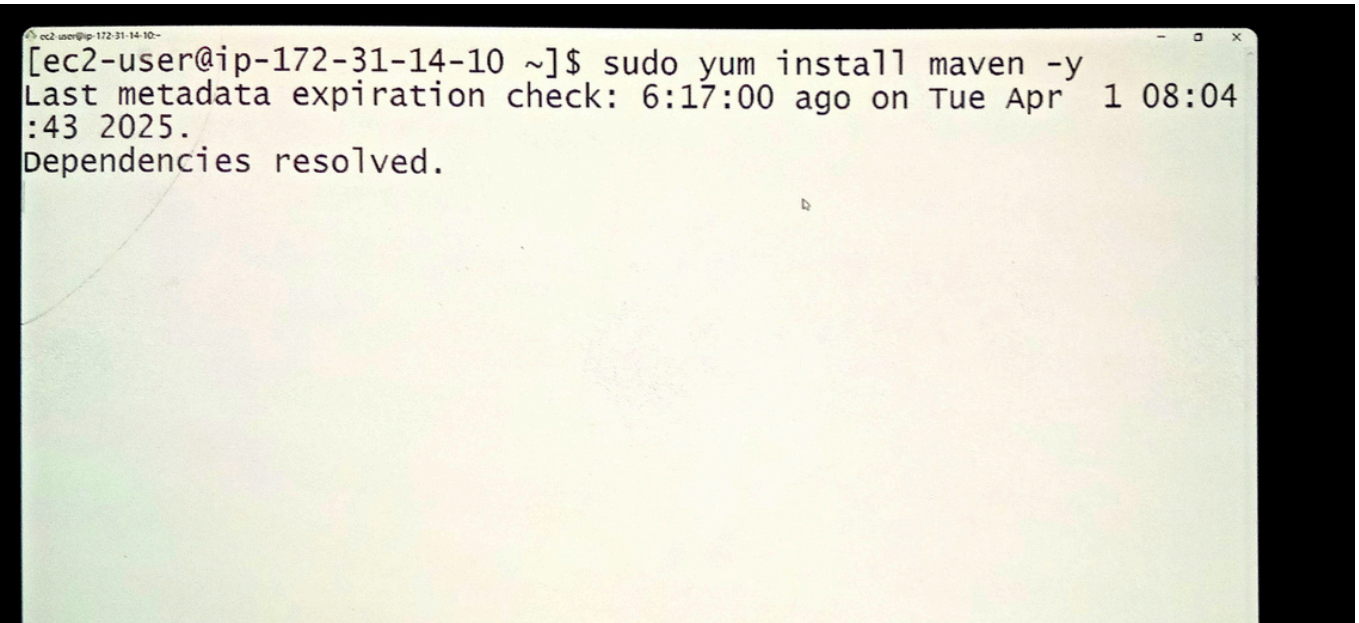
5 install docker in ec2



install github in ec2



install maven in ec2

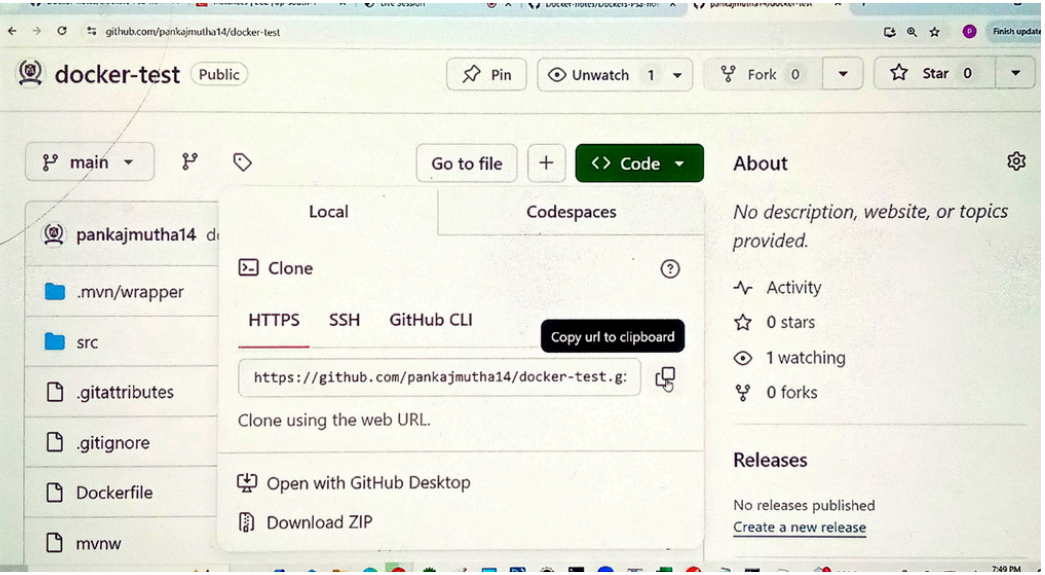


check the version of docker,git ,maven which we installed in step 5

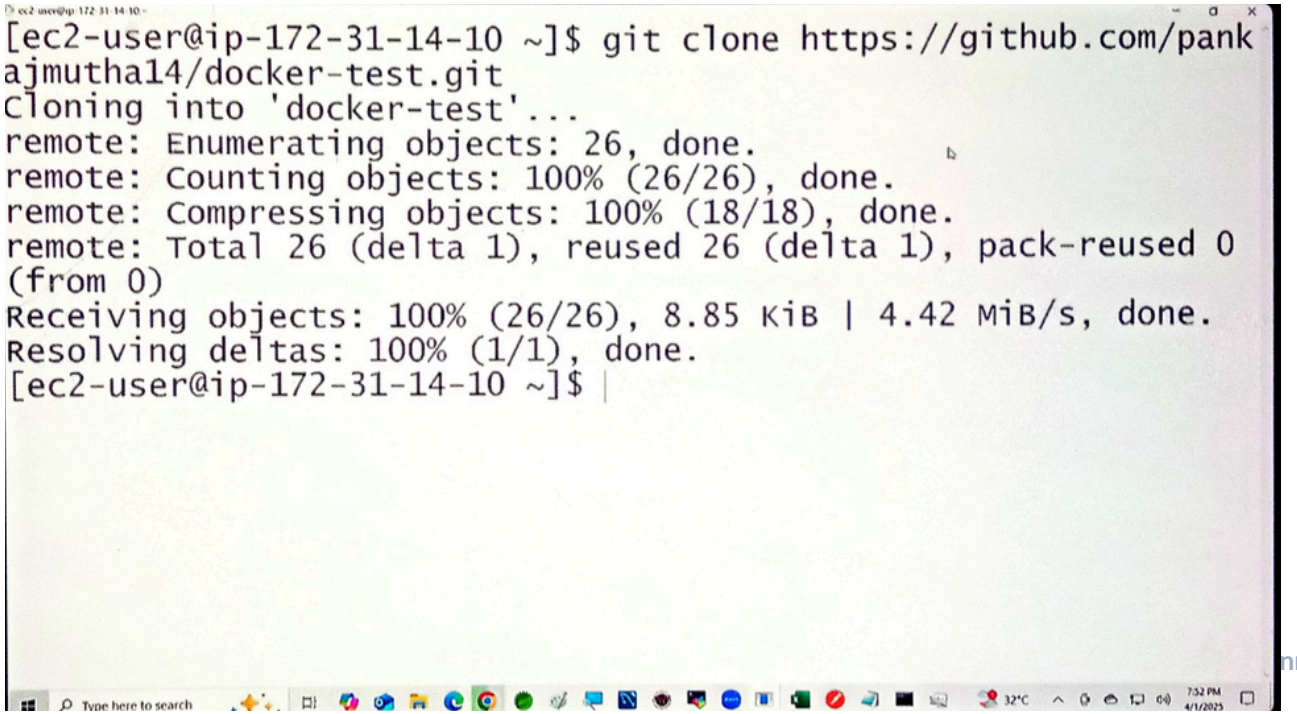
```
[ec2-user@ip-172-31-14-10 ~]$ git --version
git version 2.47.1
[ec2-user@ip-172-31-14-10 ~]$ whereis git
git: /usr/bin/git /usr/share/man/man1/git.1.gz
[ec2-user@ip-172-31-14-10 ~]$ mvn -v
Apache Maven 3.8.4 (Red Hat 3.8.4-3.amzn2023.0.5)
Maven home: /usr/share/maven
Java version: 17.0.14, vendor: Amazon.com Inc., runtime: /usr/l
ib/jvm/java-17-amazon-corretto.x86_64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.1.130-139.222.amzn2023.x86_64", a
rch: "amd64", family: "unix"
[ec2-user@ip-172-31-14-10 ~]$
```

7 go to your github repository copy url and

step 1



step 2



8

check the repository has existed or not which we already cloned

```
[ec2-user@ip-172-31-14-10 ~]$ git clone https://github.com/pank
ajmutha14/docker-test.git
Cloning into 'docker-test'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 26 (delta 1), reused 26 (delta 1), pack-reused 0
(from 0)
Receiving objects: 100% (26/26), 8.85 KiB | 4.42 MiB/s, done.
Resolving deltas: 100% (1/1), done.
[ec2-user@ip-172-31-14-10 ~]$ ls
docker-test  f1                                psa.zip
example.zip  jdk-21_windows-x64_bin.msi
[ec2-user@ip-172-31-14-10 ~]$ cd docker-test
[ec2-user@ip-172-31-14-10 docker-test]$
```

now go to cd docker-test path

now you docker-test path

```
[ec2-user@ip-172-31-14-10 docker-test]$ pwd
/home/ec2-user/docker-test
[ec2-user@ip-172-31-14-10 docker-test]$ ls -l
total 28
-rw-r--r--. 1 ec2-user ec2-user   129 Apr  1 14:22 Dockerfile
-rw-r--r--. 1 ec2-user ec2-user 10665 Apr  1 14:22 mvnw
-rw-r--r--. 1 ec2-user ec2-user  7061 Apr  1 14:22 mvnw.cmd
-rw-r--r--. 1 ec2-user ec2-user  1429 Apr  1 14:22 pom.xml
drwxr-xr-x. 4 ec2-user ec2-user    30 Apr  1 14:22 src
[ec2-user@ip-172-31-14-10 docker-test]$ mvn clean package
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/
org/springframework/boot/spring-boot-starter-parent/3.4.4/sprin
g-boot-starter-parent-3.4.4.pom
```

check your working pwd if it is matching docker-test go ahead

create .jar file

once everything is correct you will see BUILD SUCCESS


```
---------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------
---------------
[INFO] Total time:  24.270 s
[INFO] Finished at: 2025-04-01T14:24:07Z
```

9 once BUILD is done go to cd target path

```
[ec2-user@ip-172-31-14-10 docker-test]$ ls -l
total 44
-rw-r--r--. 1 ec2-user ec2-user   129 Apr  1 14:22 Dockerfile
-rw-r--r--. 1 ec2-user ec2-user 10665 Apr  1 14:22 mvnw
-rw-r--r--. 1 ec2-user ec2-user  7061 Apr  1 14:22 mvnw.cmd
-rw-r--r--. 1 ec2-user ec2-user  1429 Apr  1 14:22 pom.xml
drwxr-xr-x. 4 ec2-user ec2-user    30 Apr  1 14:22 src
drwxr-xr-x. 9 ec2-user ec2-user 16384 Apr  1 14:24 target
[ec2-user@ip-172-31-14-10 docker-test]$ cd target
[ec2-user@ip-172-31-14-10 target]$ ls -l
```

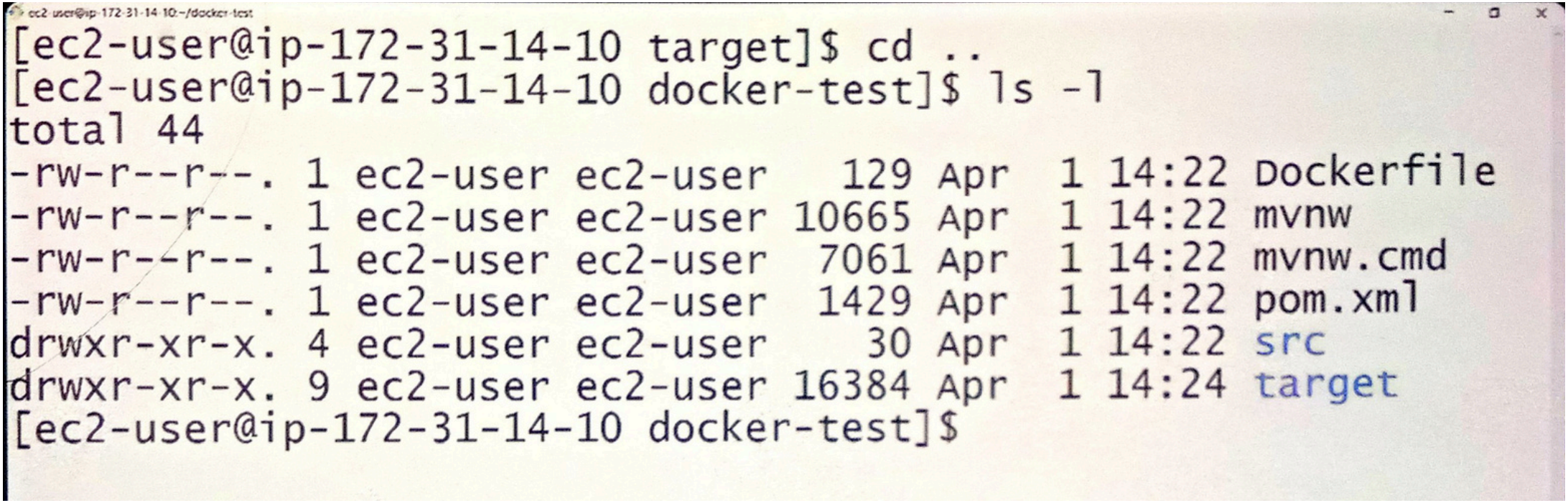
once you inside target folder ls -l you will see .jar file

```
total 20260
drwxr-xr-x. 3 ec2-user ec2-user       47 Apr  1 14:23 classes
-rw-r--r--. 1 ec2-user ec2-user 20742068 Apr  1 14:24 demo-app.
jar
-rw-r--r--. 1 ec2-user ec2-user     3218 Apr  1 14:24 demo-app.
jar.original
drwxr-xr-x. 3 ec2-user ec2-user       25 Apr  1 14:23 generated
-sources
drwxr-xr-x. 3 ec2-user ec2-user       30 Apr  1 14:23 generated
-test-sources
drwxr-xr-x. 2 ec2-user ec2-user       28 Apr  1 14:24 maven-arc
hiver
drwxr-xr-x. 3 ec2-user ec2-user       35 Apr  1 14:23 maven-sta
tus
drwxr-xr-x. 2 ec2-user ec2-user       99 Apr  1 14:24 surefire-
reports
drwxr-xr-x. 3 ec2-user ec2-user       17 Apr  1 14:23 test-clas
ses
```

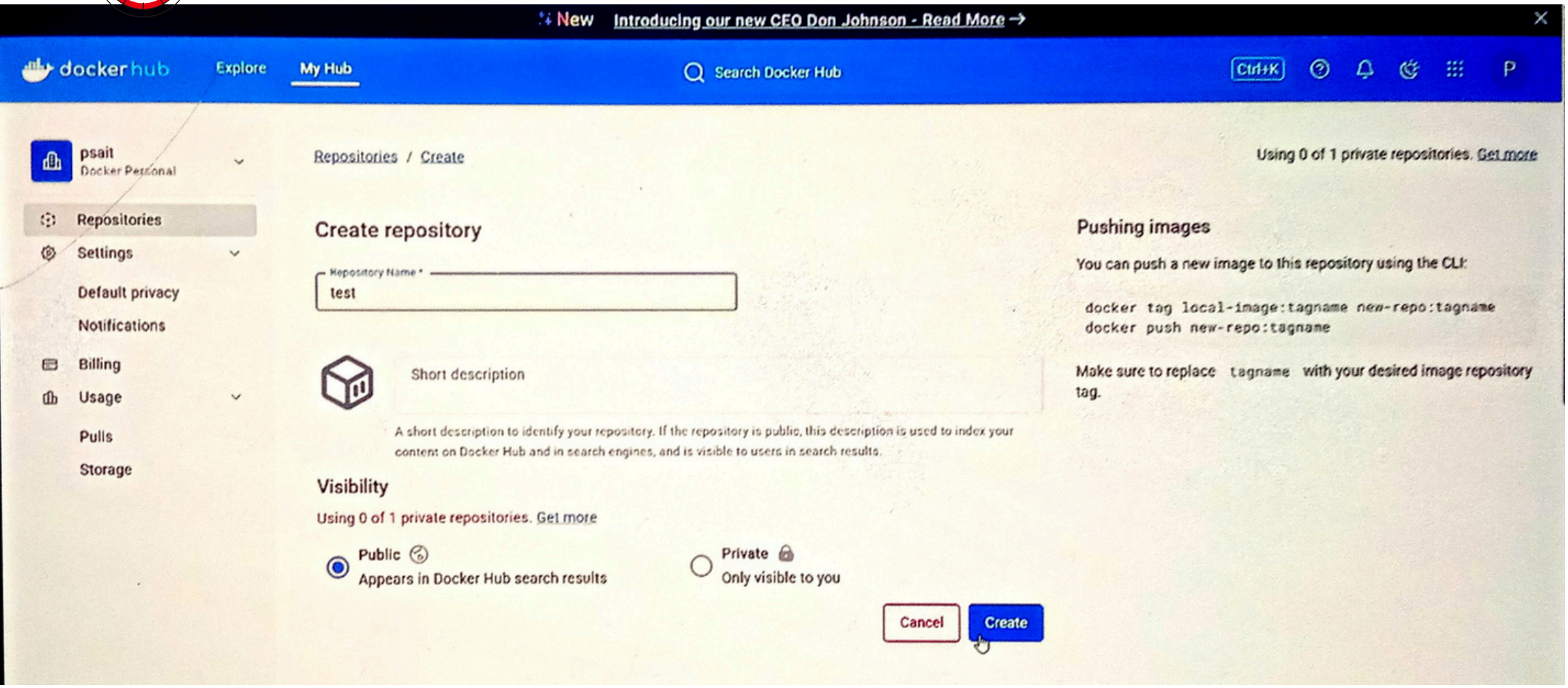
Your paragraph text

10

once you checked .jar is created inside target folder now come back to docker-test path i.e. cd ..

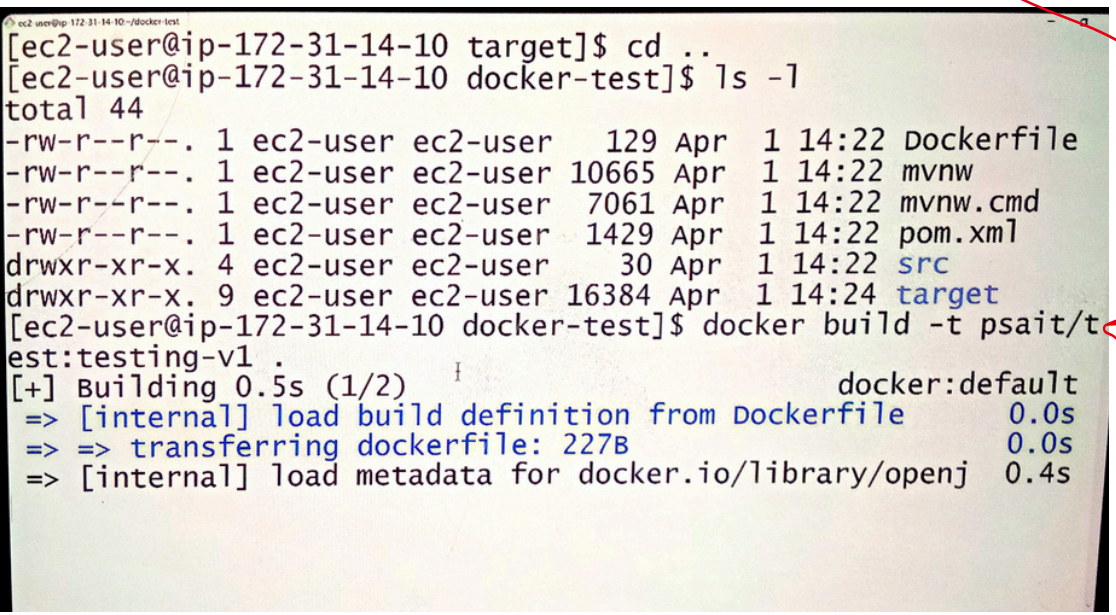


create repository in Docker hub/ Docker registory



12

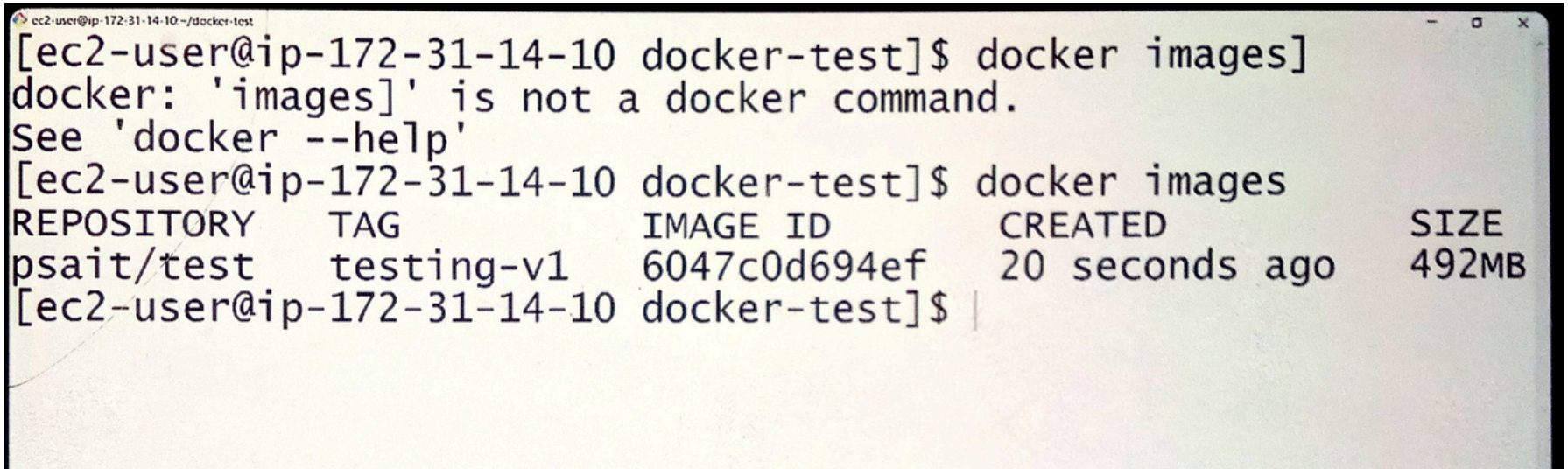
build image here docker-test path



-> docker build -t psait/test:testing-v1 .

dot indicates build image on current path

check image is build or not by doing docker images



13 now run docker container: docker run -d -p 8080:8080 psait/tets:testing-v1

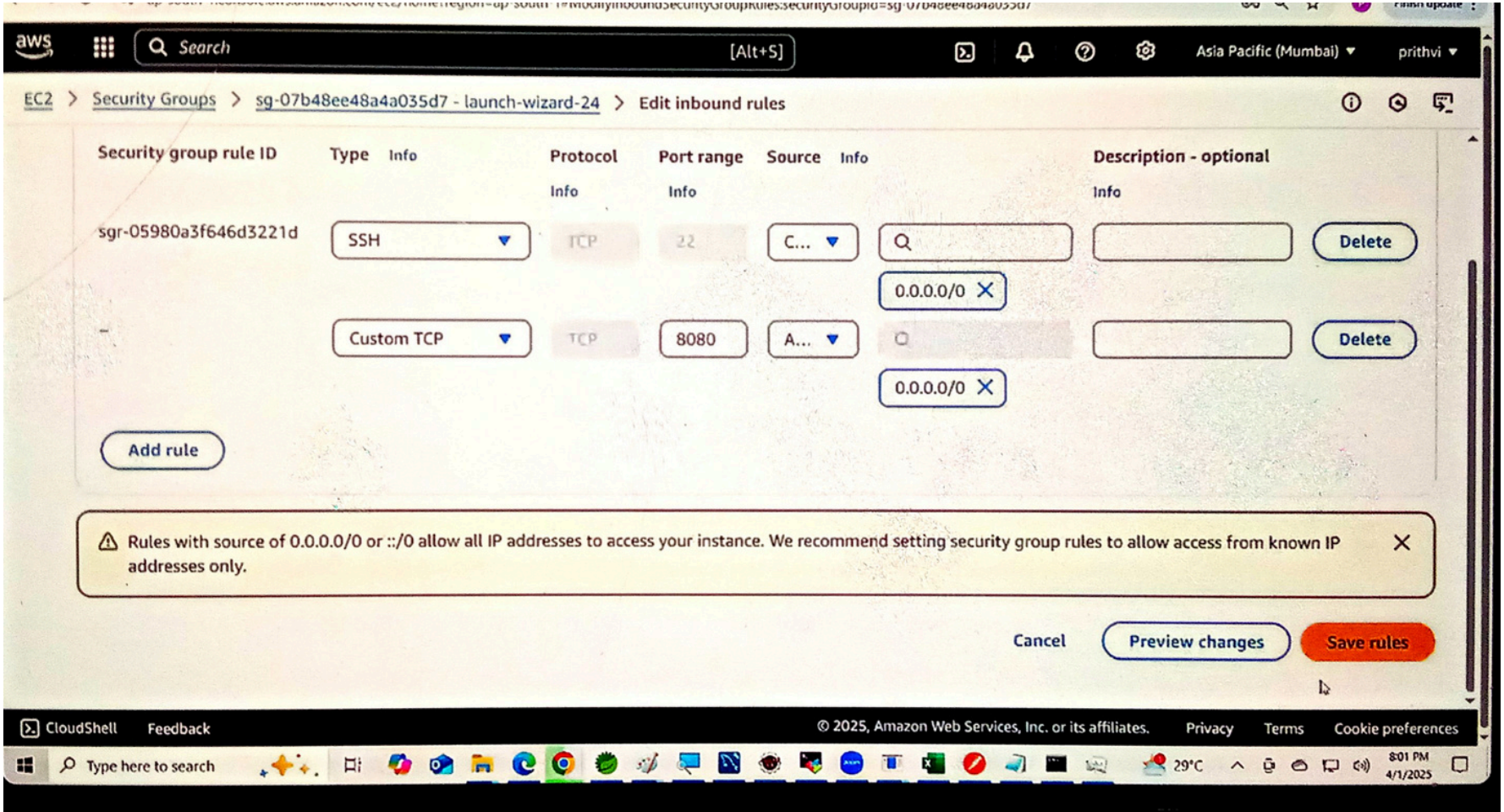
```
REPOSITORY    TAG           IMAGE ID       CREATED              SIZE
psait/test    testing-v1    6047c0d694ef   20 seconds ago       492MB
[ec2-user@ip-172-31-14-10 docker-test]$ docker run -d -p 8080:8
080 psait/test
Unable to find image 'psait/test:latest' locally
docker: Error response from daemon: manifest for psait/test:lat
est not found: manifest unknown: manifest unknown.
See 'docker run --help'.
[ec2-user@ip-172-31-14-10 docker-test]$ docker run -d -p 8080:8
080 psait/test:tesrting-v1
Unable to find image 'psait/test:tesrting-v1' locally
docker: Error response from daemon: manifest for psait/test:tes
rting-v1 not found: manifest unknown: manifest unknown.
See 'docker run --help'.
[ec2-user@ip-172-31-14-10 docker-test]$ docker run -d -p 8080:8
080 psait/test:testing-v1
9ef59f26e646c179490bb2b581116add64b20085961df7803f26c4914d6bd41
6
[ec2-user@ip-172-31-14-10 docker-test]$
```

Now check docker container running or not

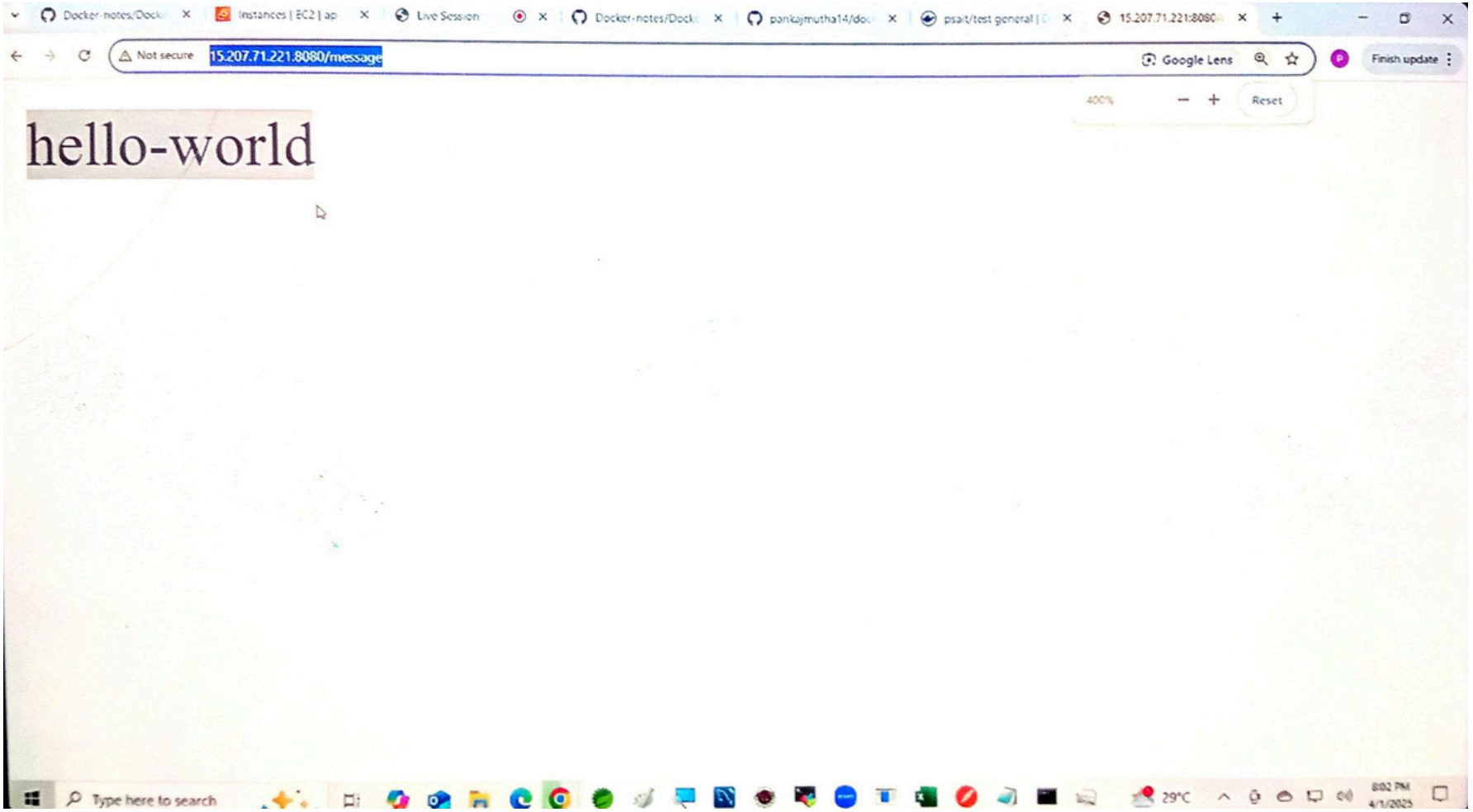
```
[ec2-user@ip-172-31-14-10 docker-test]$ docker ps
CONTAINER ID   IMAGE                    COMMAND
 CREATED              STATUS              PORTS
              NAMES
9ef59f26e646   psait/test:testing-v1    "java -jar demo-app.…"
 11 seconds ago   Up 9 seconds   0.0.0.0:8080->8080/tcp, :::808
0->8080/tcp   romantic_solomon
[ec2-user@ip-172-31-14-10 docker-test]$
```

14

Go to Security group add inbound rule i.e. 8080 because our spring boot is running on 8080 port



15 Go to browser copy public ip address from ec2 and give port number like below



16

Now i want to push my docker image into Docker hub, before you need to login Docker hub as shown below

```
[ec2-user@ip-172-31-14-10 docker-test]$ docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants better security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: psait
Password:
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[ec2-user@ip-172-31-14-10 docker-test]$
```

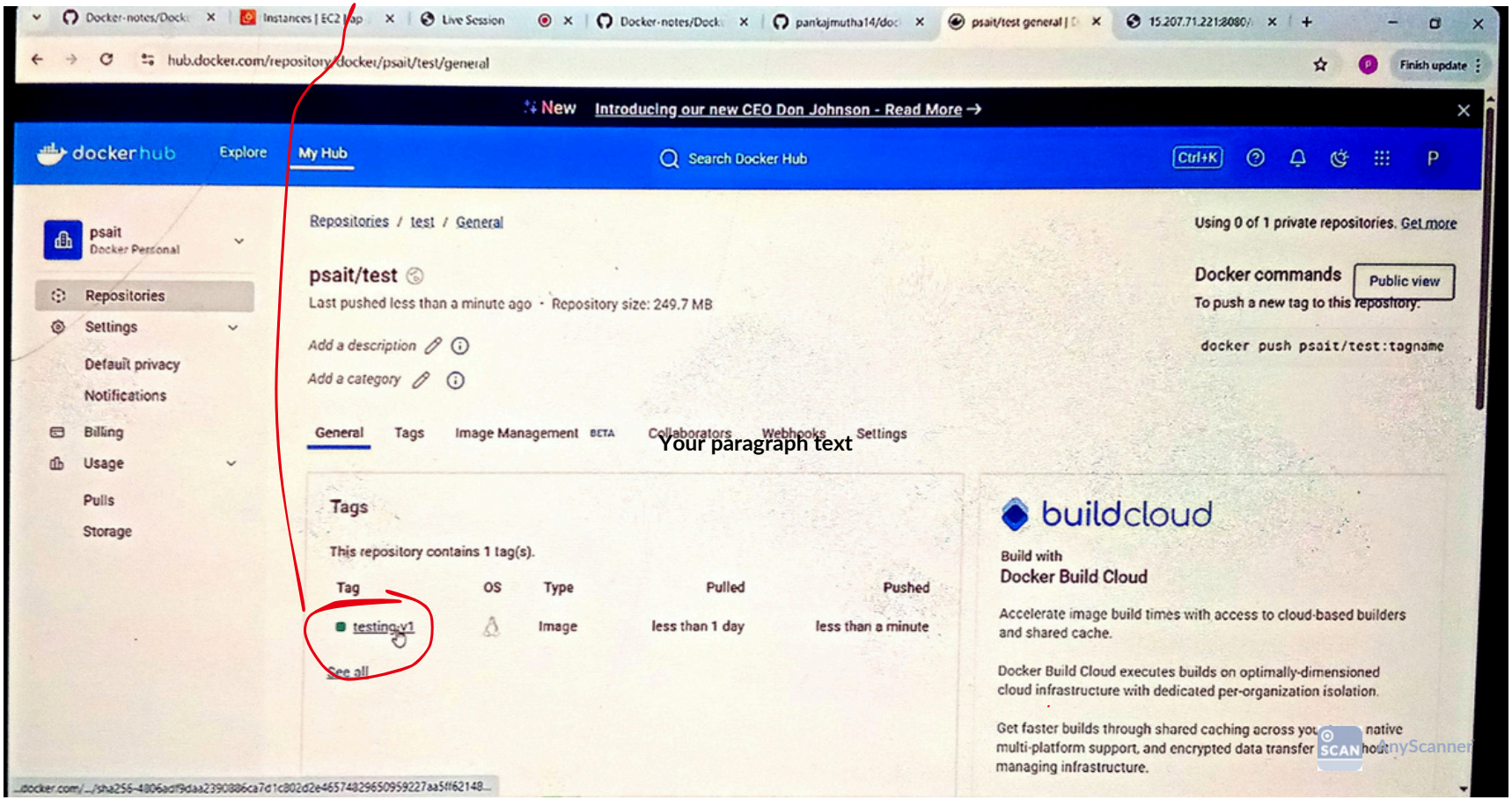
docker login
Enter username: psait
Enter password

Once you log in you need to push docker image now as shown below

```
[ec2-user@ip-172-31-14-10 docker-test]$ docker push psait/test:testing-v1
The push refers to repository [docker.io/psait/test]
5f70bf18a086: Pushing  1.024kB
122005d28155: Preparing
dc9fa3d8b576: Preparing
27ee19dc88f2: Preparing
c8dd97366670: Preparing
```

17

Once you pushed image in Docker hub you will see below like



Further you can pull,run repeat what we did earlier

-> Spring Boot: framework which is used to develop enterprise based applications.
-> Spring Boot applications will be packaged as a jar file for deployment.
-> To run the jar file we will use command : java -jar <file-name.jar>
-> To run springboot application jar file we will use tomcat server as "embedded server".
-> By default spring boot application will run on port number 8080

Step 1: Create Docker File

```
FROM openjdk:17

COPY target/demo-app.jar  /usr/app/

WORKDIR /usr/app/

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "demo-app.jar"]
```

Step 2: Note (Install maven and git in linux VM first)

1) Clone git repo in docker host machine (linux): git clone <http-url>

2) Point to project root folder: cd <app-name> and run to generate jar: mvn clean package

3) Create docker image and check that:
-> docker build -t psait/pankajsiracademy:<tag> .
(
Here
-> psait is username
-> repositoryname of docker hub
-> <tag> can
 a. prod-v1 or prod-v2v2 for production
 b. dev-v1 for development environment
 c. test-v1 for testing environment
 d. staging-v1 for staging environment
)
-> docker images
4) run docker container: docker run -d -p 8080:8080 --name psa psait/your-app
5)See the image is running or not: docker ps
6) docker logs <container-id>
5) Access application URL in browser: http://public-ip:8080/
Steps to push docker image to docker hub
_____
-> login into docker hub account from bash
docker login
Enter username: psait
Enter password
-> push docker image
docker push psait/pankajsiracademy:<tag>

Thank you