Docker 2 notes:

Base Image name:

FROM

It is used to specify base image for our application

FROM tomcat:9.0

FROM openJDK:17

FROM node:19.4

FROM mysql:8.5

FROM python:3.3

MAINTAINER

Maintainer decides the author of the Dockerfile

It is used to specify the author of the Dockerfile

MAINTAINER Abc [<abc@sldk.com>](mailto:<abc@sldk.com>)

First specify base image, followed by maintainer/author, run

RUN

At the time of Docker image creation, which instructions you want to execute

Run keyword is used to specify instructions to execute at the time of Docker image creation

Example: RUN ‘git clone<url>;’

RUN ‘mvn clean package’

If required, we can write multiple RUN instructions in a single Dockerfile and all those instructions will be processed in order

CMD

When does a Docker container get created?

When we try to execute the Docker image that time Docker container is created

CMD keyword is used to specify intructions to execute at the time of Docker container creation

At the time of image creation, we can write multiple RUN commands

CMD ‘java -jar app.jar’

CMD ‘app.py’

One Dockerfile can have multiple CMD instructions, however, Docker will process only last CMD instruction

RUN instructions will be executed at the time of Image creation, CMD instructions will be executed at the time of container creation

ENTRYPOINT

Keyword is used to specify instruction to execute at the time of docker container creation.

ENTRYPOINT is similar to CMD

ENTRYPOINT[“java”, “-jar”, “app.jar”]

ENTRYPOINT[“python”, “app.py”]

Note: CMD instructions we can override using command-line arguments however ENTRYPOINT instructions cannot be overridden

COPY

COPY any files from host machines to container machines

Docker container

Guest VM

Docker Engine

Linux

Host VM

COPY target/app.war /usr/app/tomcat/webapps/app.war

File must be available in host machine only then we can copy from host machine to guest VM

ADD

It is used to copy files from source to destination

ADD target/app.war /usr/app/tomcat/webapps/app.war

ADD <http-url> /usr/app/app.war

WORKDIR

Used to set working directory

COPY target/app.jar /usr/app/app.jar

WORKDIR /usr/app

It will go into /usr/app directory and execute further commands

CMD ‘java -jar app.jar’

EXPOSE

It is used to specify on which port# our application will run in our container

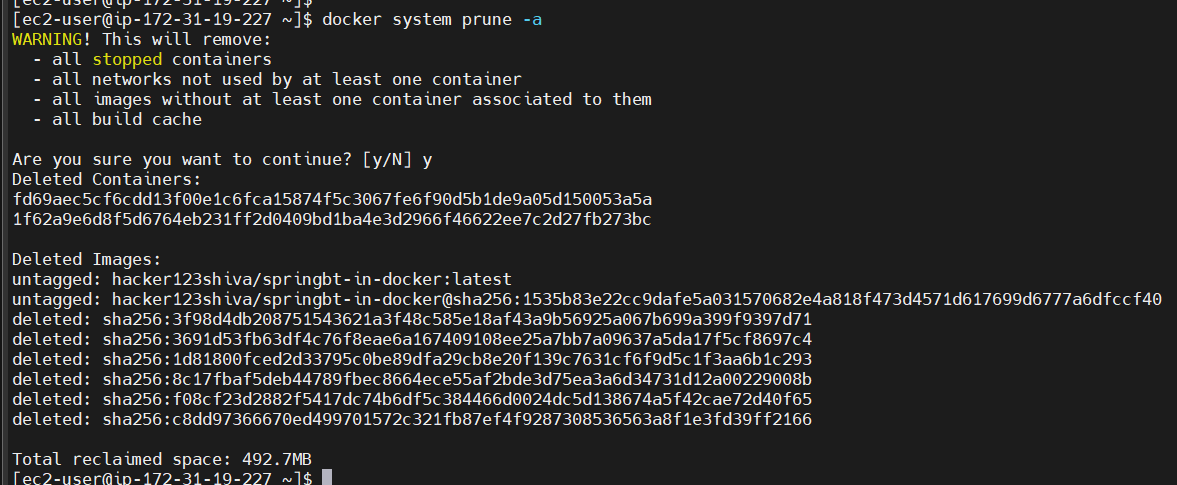
EXPOSE 8080 --> it is not to change port number it is only to provide information about port number to other team members

USER

Used to set USER to run commands

Prune

[ec2-user@ip-172-31-19-227 ~]$ docker system prune -a



[ec2-user@ip-172-31-19-227 ~]$ vi Dockerfile

[ec2-user@ip-172-31-19-227 ~]$

[ec2-user@ip-172-31-19-227 ~]$ cat Dockerfile

FROM ubuntu

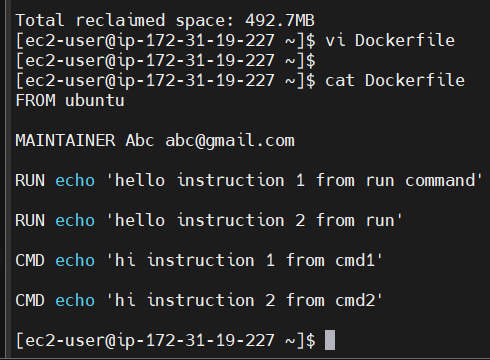
MAINTAINER Abc abc@gmail.com

RUN echo 'hello instruction 1 from run command'

RUN echo 'hello instruction 2 from run'

CMD echo 'hi instruction 1 from cmd1'

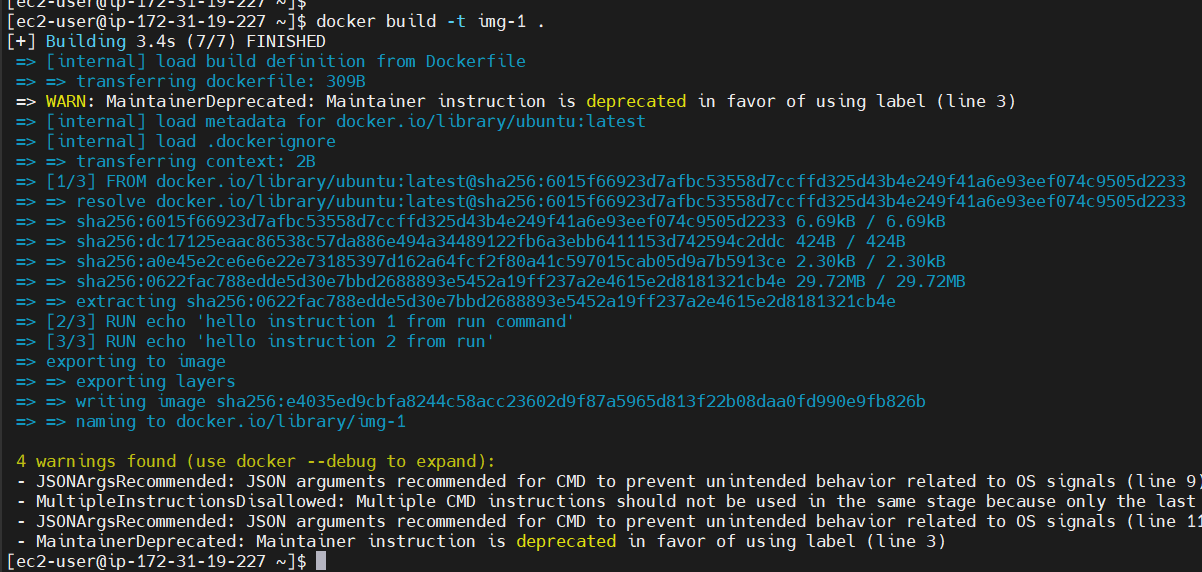
CMD echo 'hi instruction 2 from cmd2'



[ec2-user@ip-172-31-19-227 ~]$ docker build -t img-1 .

Create one docker image with image name img-1, . means in the current directory one Dockerfile is available

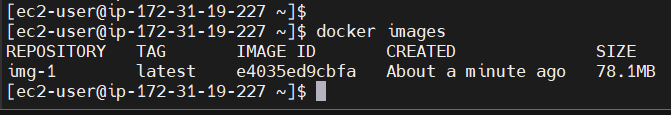
-t img-1 specifies the tagname (Image name is img-1),



[ec2-user@ip-172-31-19-227 ~]$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

img-1 latest e4035ed9cbfa About a minute ago 78.1MB



Now run that image and see

[ec2-user@ip-172-31-19-227 ~]$ docker run e4035ed9cbfa

hi instruction 2 from cmd2

[ec2-user@ip-172-31-19-227 ~]$ cat Dockerfile

FROM ubuntu

MAINTAINER Abc abc@gmail.com

RUN echo 'hello instruction 1 from run command'

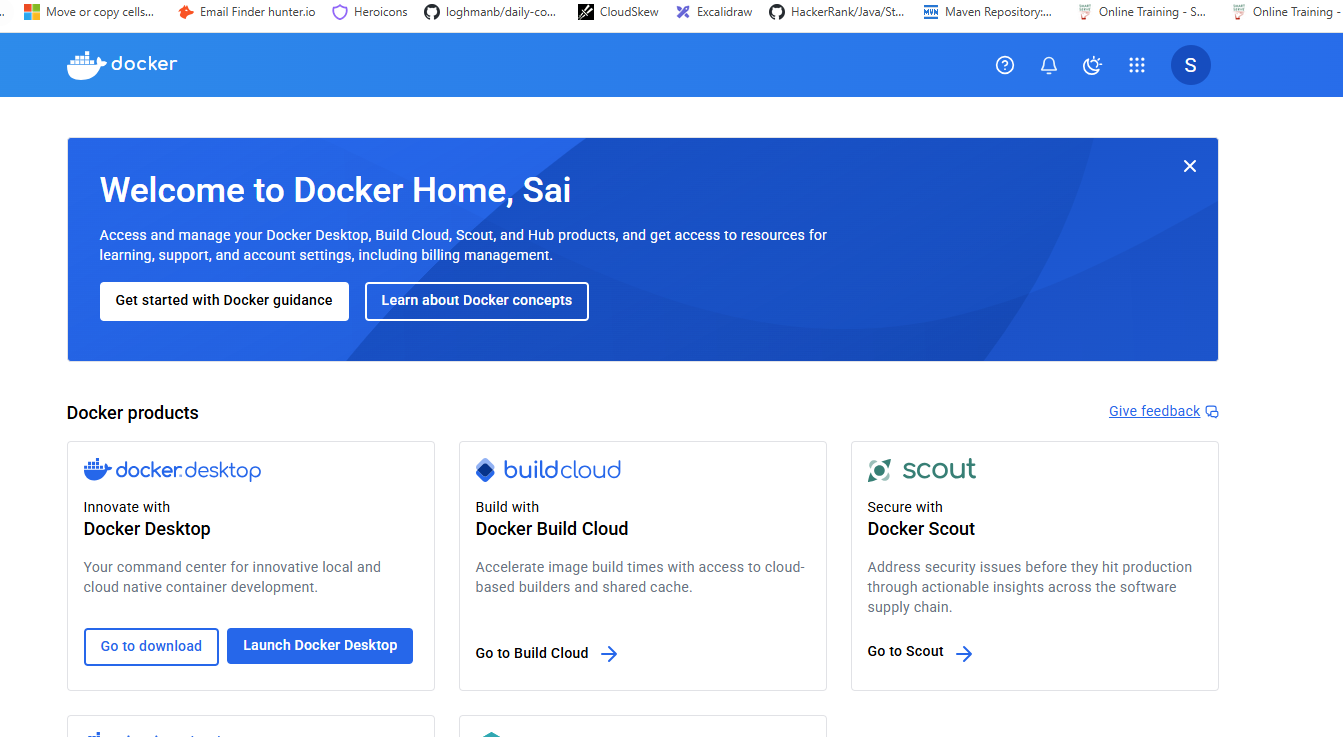
RUN echo 'hello instruction 2 from run'

CMD echo 'hi instruction 1 from cmd1'

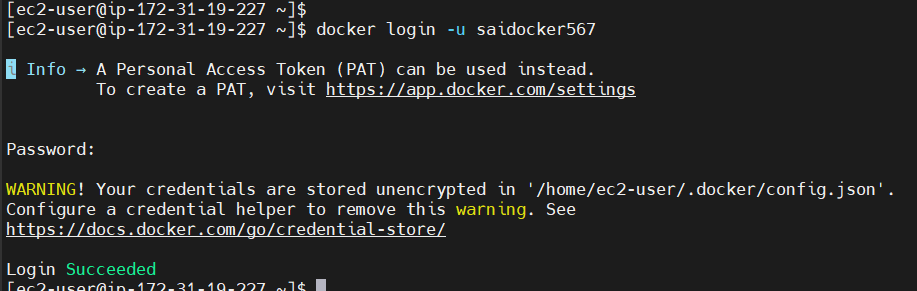
CMD echo 'hi instruction 2 from cmd2'

It is printing the second CMD

Create a Docker account



docker login -u <username>

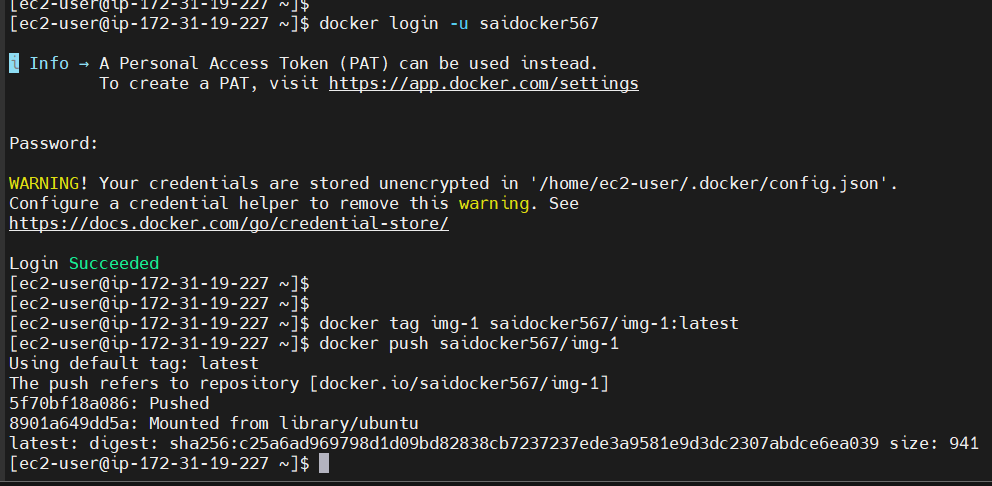


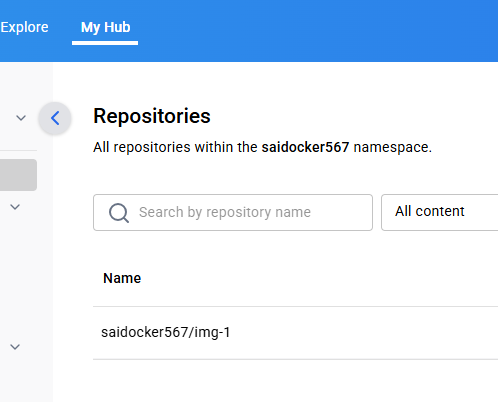
docker tag img-1 <username>/img-1:v1

[ec2-user@ip-172-31-19-227 ~]$ docker tag img-1 saidocker567/img-1:v1

[ec2-user@ip-172-31-19-227 ~]$ docker tag img-1 saidocker567/img-1:latest

[ec2-user@ip-172-31-19-227 ~]$ docker push saidocker567/img-1





We can see the pushed image in Docker hub

Push Docker image into Docker hub account:

# Create an account in Docker Hub and make sure to note username and password

--> Login into Docker hub account

docker login

It will ask username and password please do specify

--> tag the docker image

docker tag <image-name><tag-name>

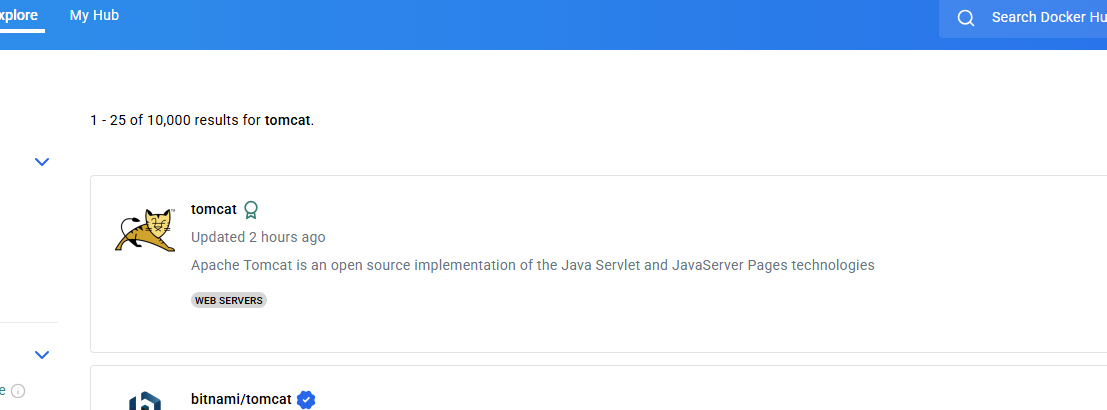
docker tag img-1 <username>/<image-name>

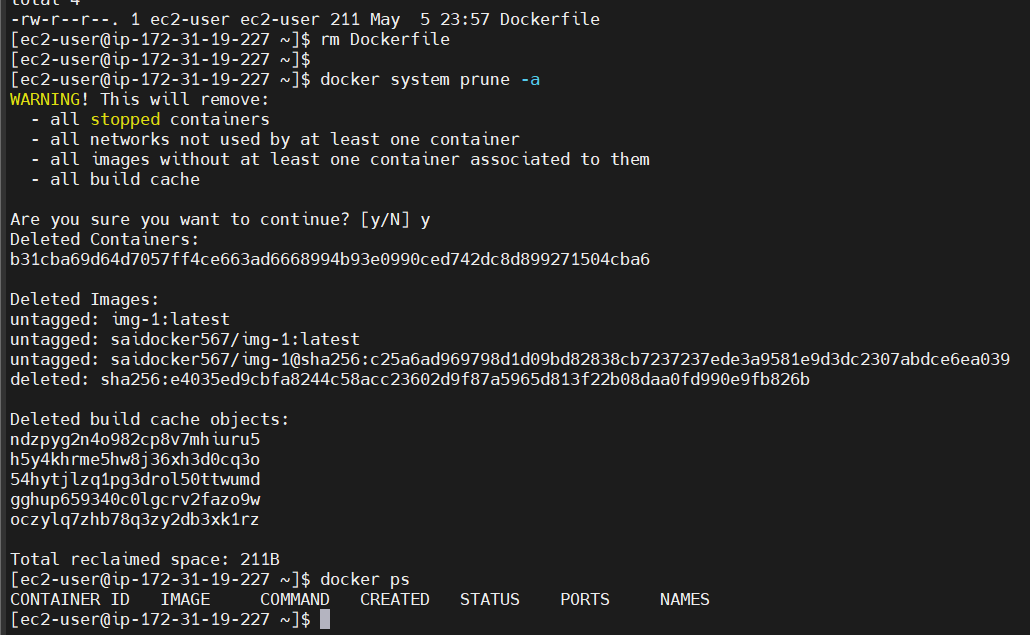
--> Finally push the Docker image to Docker hub

docker push <tag-name>

Dockerizing Java Web App

Search for tomcat image





Maven will install Java as well

[ec2-user@ip-172-31-19-227 ~]$ sudo yum install maven

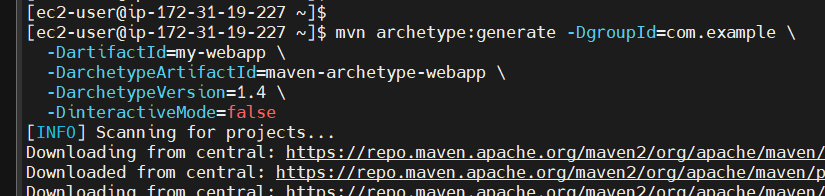
mvn archetype:generate -DgroupId=com.example \

-DartifactId=my-webapp \

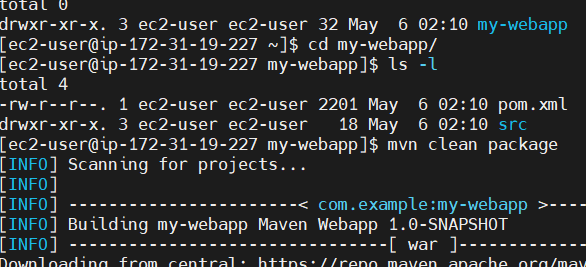
-DarchetypeArtifactId=maven-archetype-webapp \

-DarchetypeVersion=1.4 \

-DinteractiveMode=false



[ec2-user@ip-172-31-19-227 my-webapp]$ mvn clean package



[ec2-user@ip-172-31-19-227 my-webapp]$ vi Dockerfile

FROM openjdk:17

MAINTAINER Abc abc@gmail.com

COPY target/my-webapp.war /usr/app/my-webapp.war

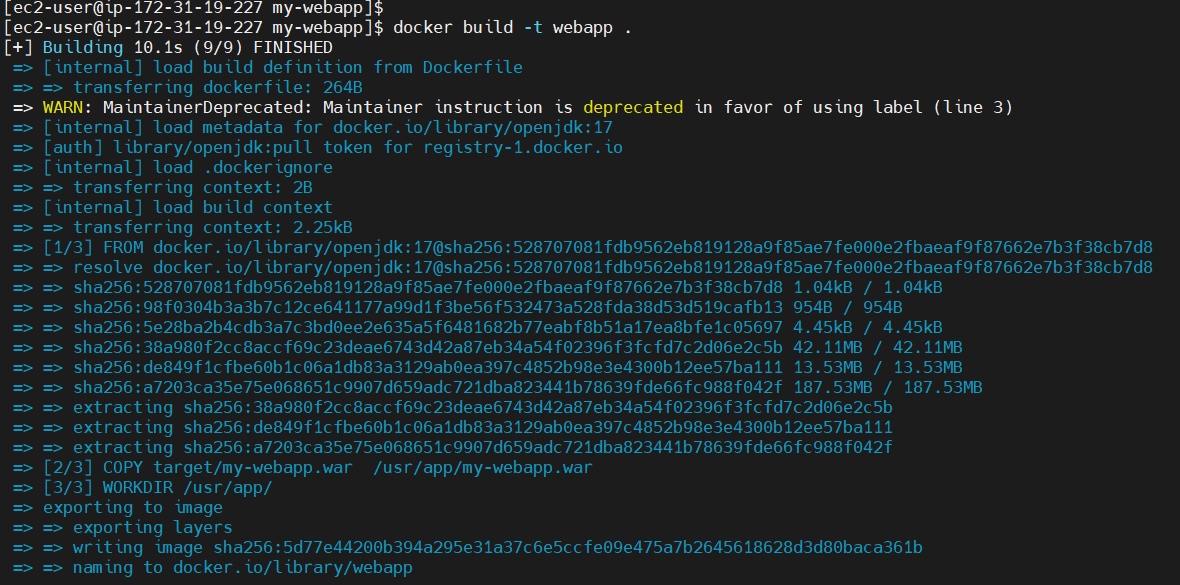
WORKDIR /usr/app/

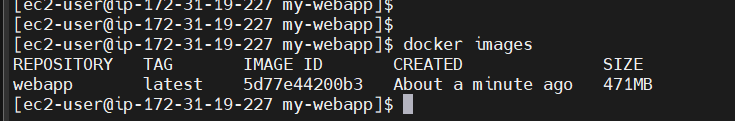
EXPOSE 8080

ENTRYPOINT ["java" , "-jar" , ""]



[ec2-user@ip-172-31-19-227 my-webapp]$ docker build -t webapp .





[ec2-user@ip-172-31-19-227 my-webapp]$ docker run -d -p 8181:8080 webapp

Some error re-running Maven

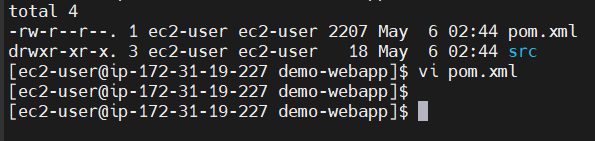
[ec2-user@ip-172-31-19-227 my-webapp]$ mvn archetype:generate -DgroupId=com.example -DartifactId=demo-webapp -DarchetypeArtifactId=maven-archetype-webapp -DarchetypeVersion=1.4 -DinteractiveMode=false

[ec2-user@ip-172-31-19-227 ~]$ ls -l

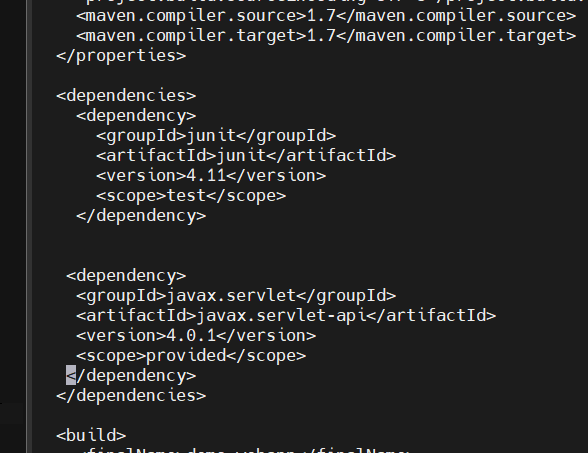
total 0

drwxr-xr-x. 3 ec2-user ec2-user 32 May 6 02:44 demo-webapp

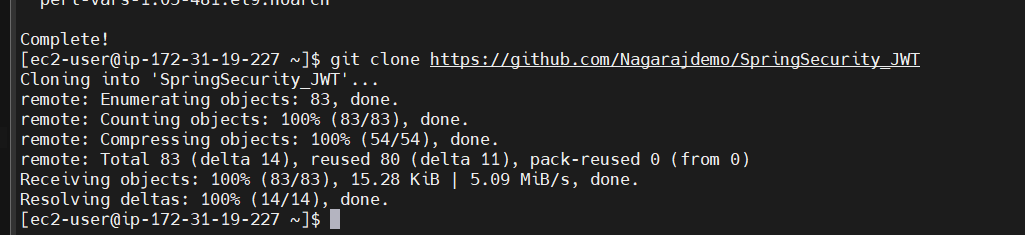
drwxr-xr-x. 5 ec2-user ec2-user 83 May 6 02:43 my-webapp



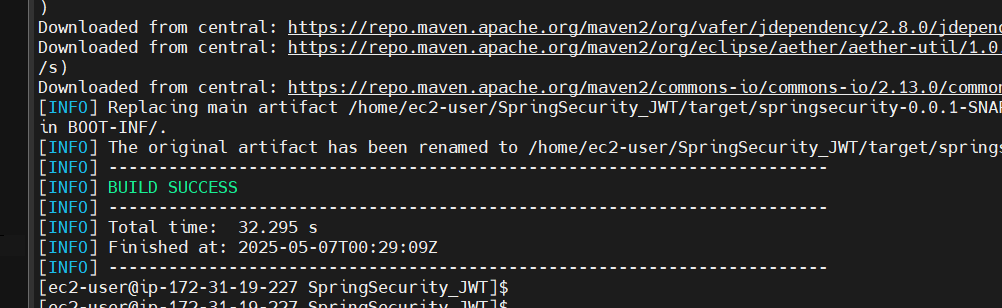
https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api/4.0.1



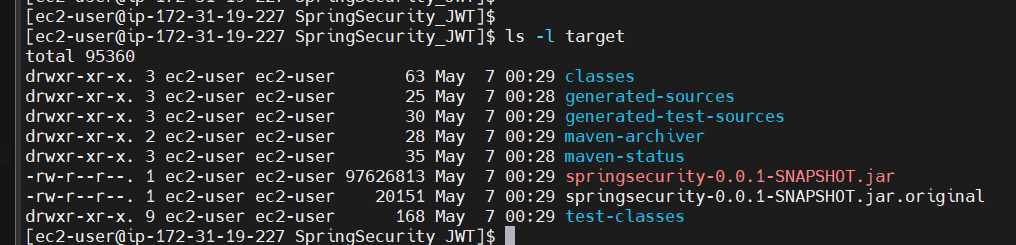
Retrying with a different repo



[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ mvn clean package



There is a jar file inside the target folder



[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ vi Dockerfile

[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ cat Dockerfile

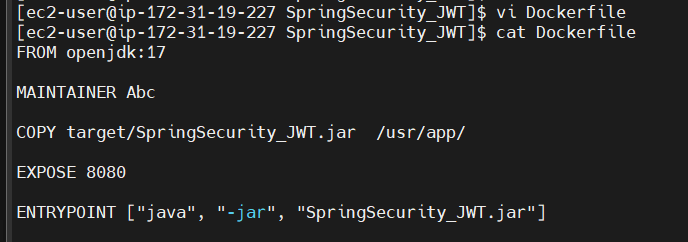
FROM openjdk:17

MAINTAINER Abc

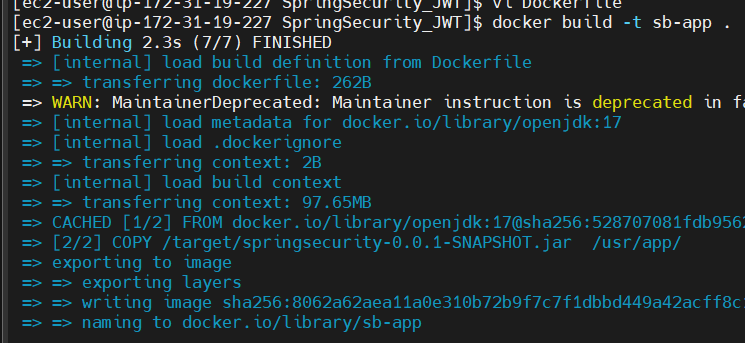
COPY /target/springsecurity-0.0.1-SNAPSHOT.jar /usr/app/

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "SpringSecurity\_JWT.jar"]

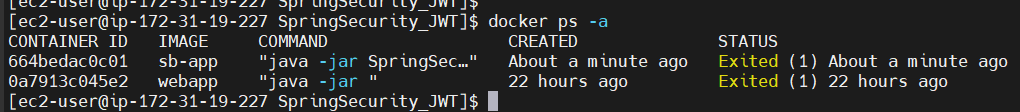


[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ docker build -t sb-app .



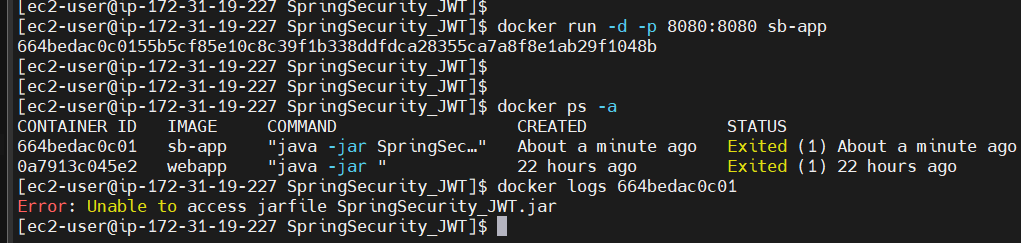
[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ docker run -d -p 8080:8080 sb-app

664bedac0c0155b5cf85e10c8c39f1b338ddfdca28355ca7a8f8e1ab29f1048b



[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ docker logs 664bedac0c01

Error: Unable to access jarfile SpringSecurity\_JWT.jar



[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ vi Dockerfile

[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$

[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ cat Dockerfile

FROM openjdk:17

MAINTAINER Abc

COPY /target/springsecurity-0.0.1-SNAPSHOT.jar /usr/app/

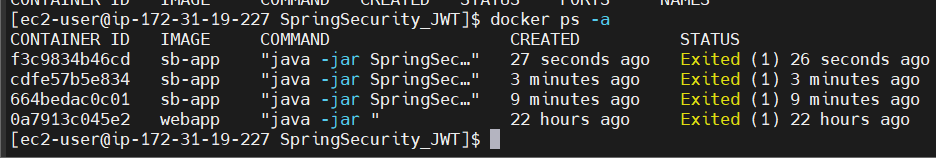
EXPOSE 8080

ENTRYPOINT ["java", "-jar", "springsecurity-0.0.1-SNAPSHOT.jar"]

Updated Dockerfile

[ec2-user@ip-172-31-19-227 SpringSecurity\_JWT]$ docker run -d -p 8080:8080 sb-app

f3c9834b46cd86f7866f41e319910ed1f48f556af1b6d6d1de0efc5fee9f0853



There is some problem accessing with public IP

Next class