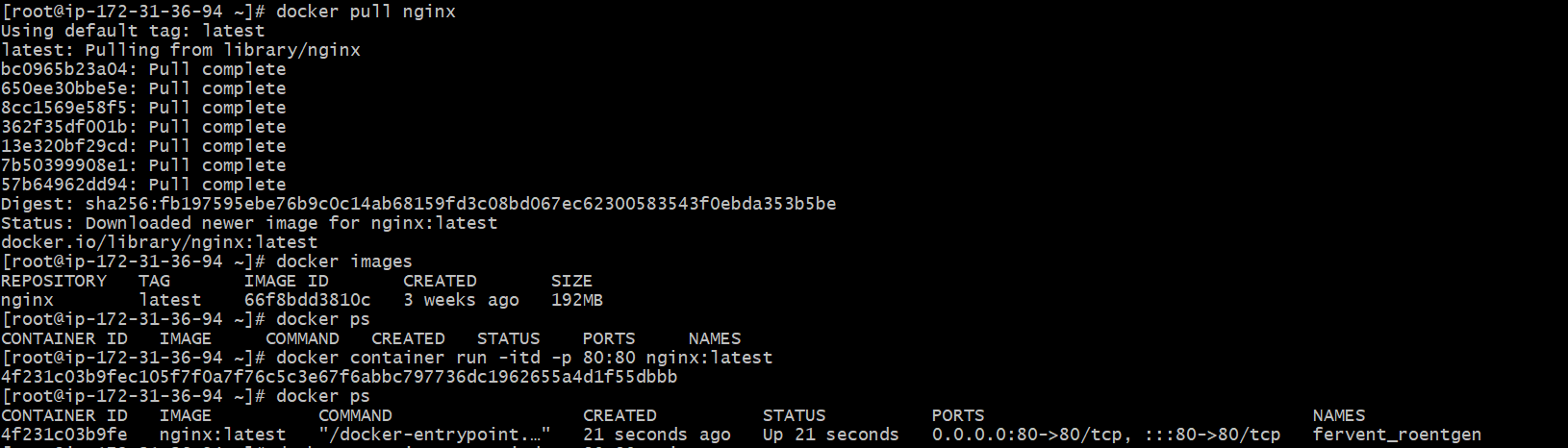
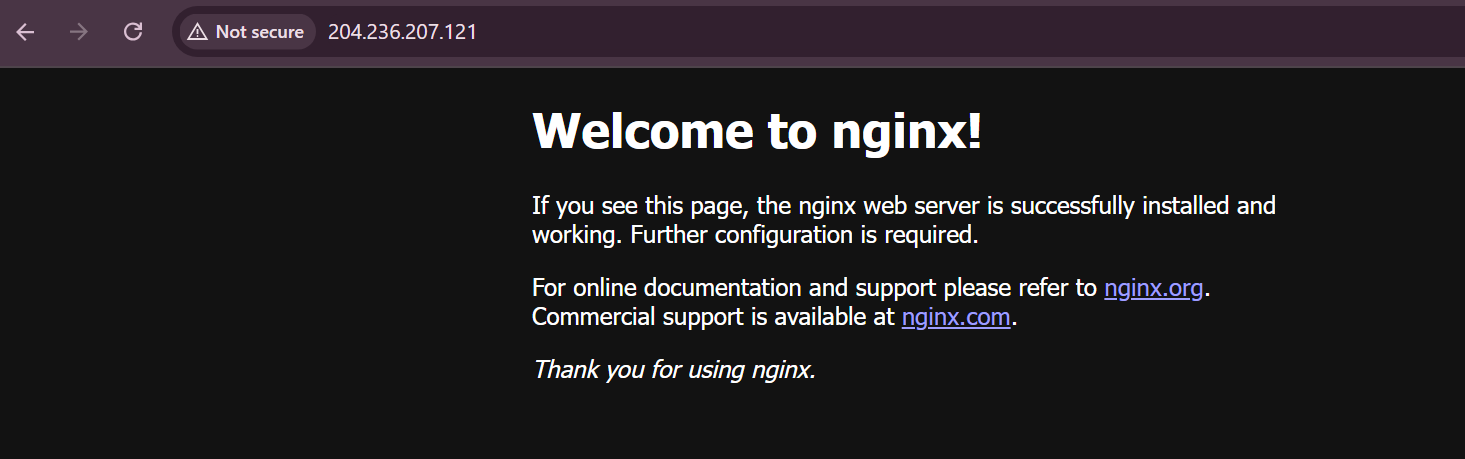
DOCKER[TASK-3]

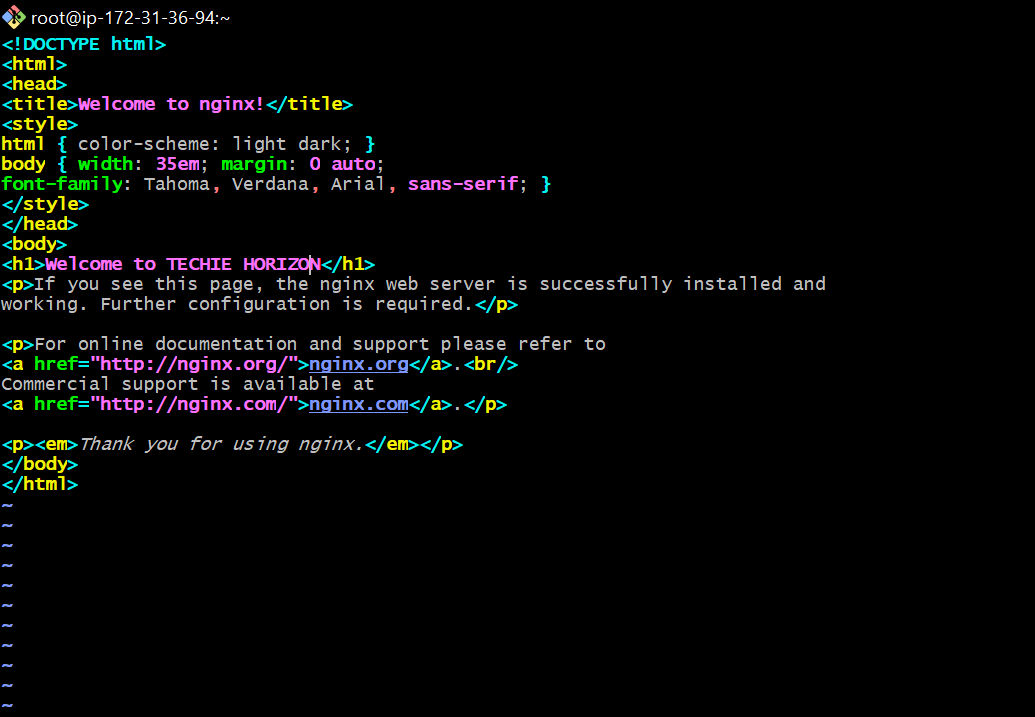
1) Create a image from running container.

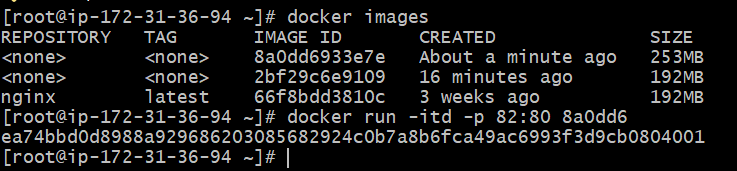
accessing the nginx in the browser with the port 80



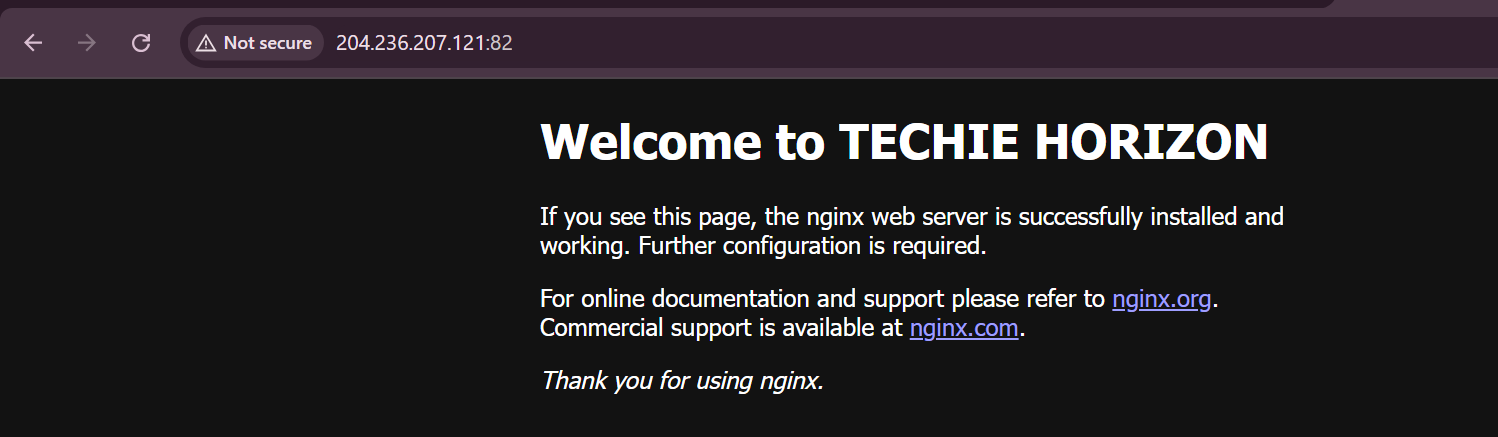
Changing the deployment page

* *Docker exec –it container\_id /bin/bash*
* *In that container : cd /usr/share/nginx/html*
* *Edit the index.html file*



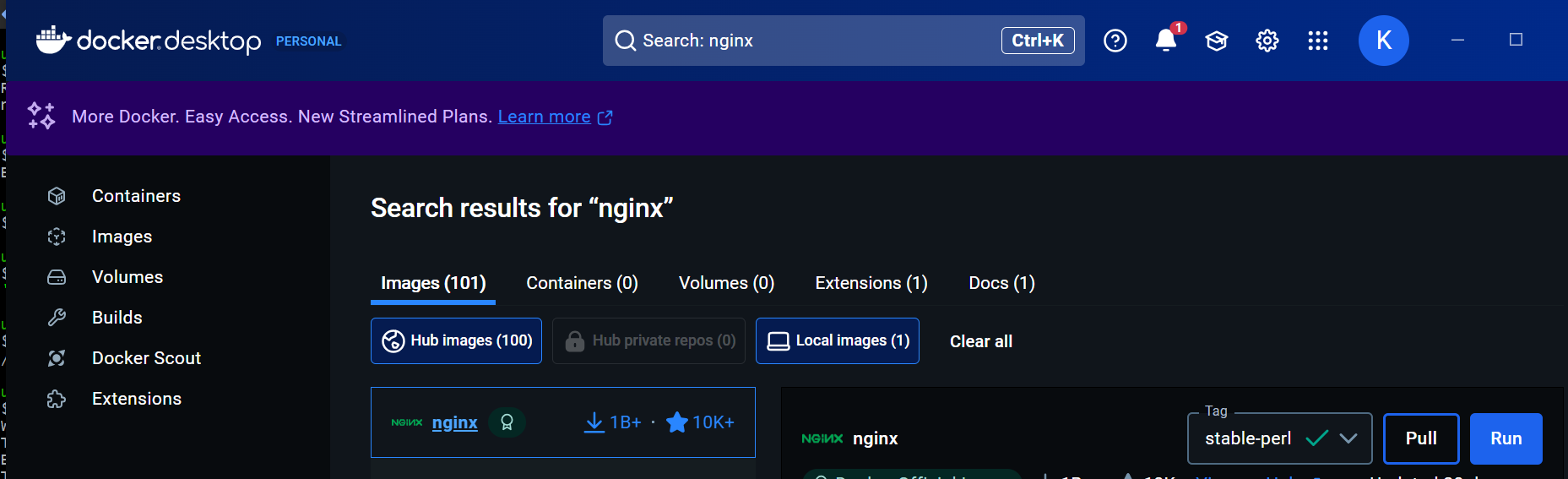


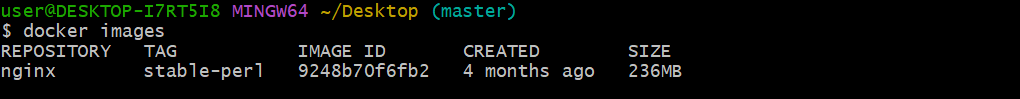
*Docker container run –itd –p 82:80 image\_id*

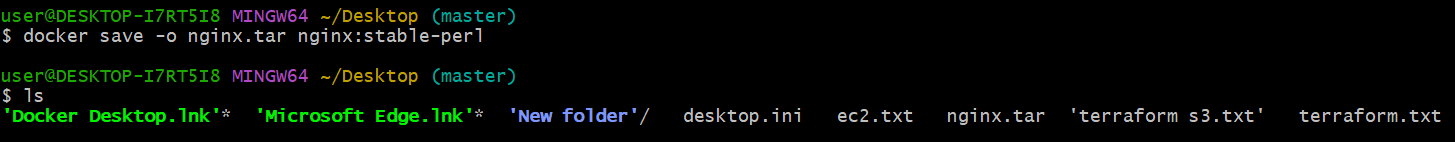


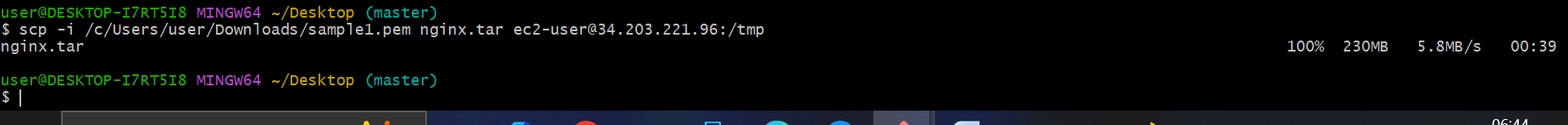
2) Copy image from local machine to docker server and load the image.

Download the docker desktop

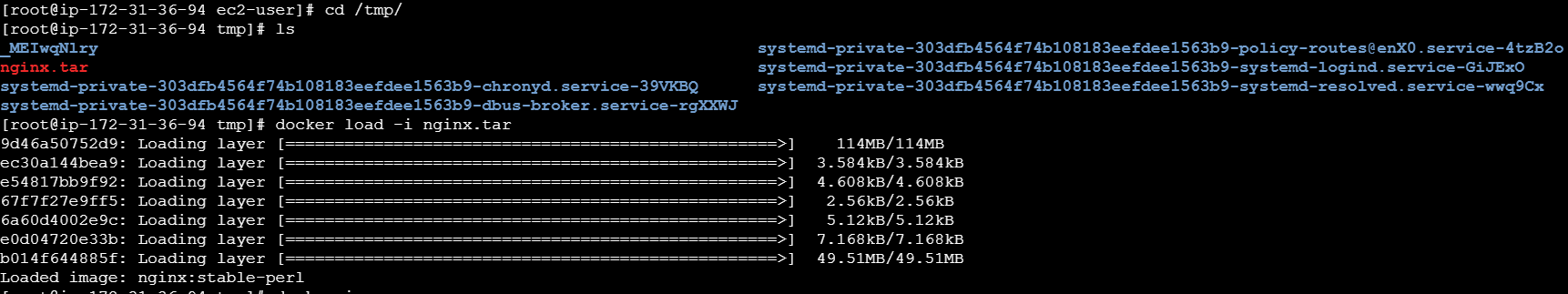
And pull the nginx image



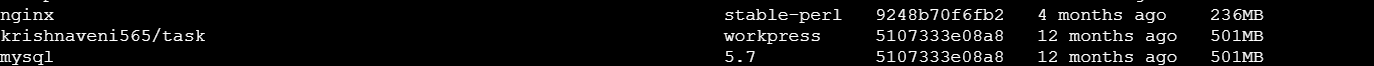




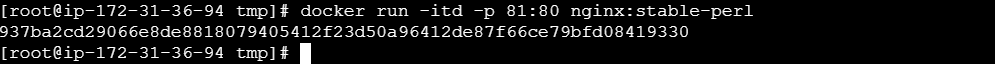
On docker engine

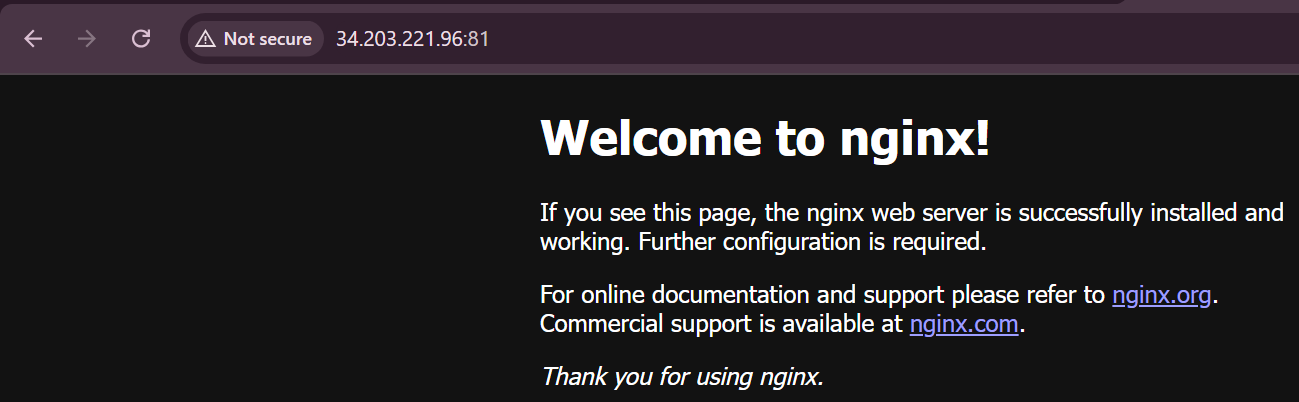


Next if we do ,docker ps



Running the containers





3) Create Docker image using alpine and customize with tomcat.

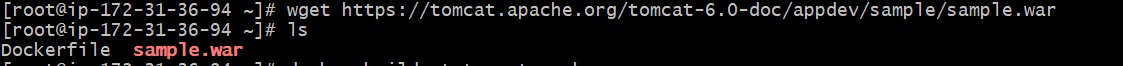
Using alpine:[docker pull alpine]

Editing the Dockerfile



Save the file

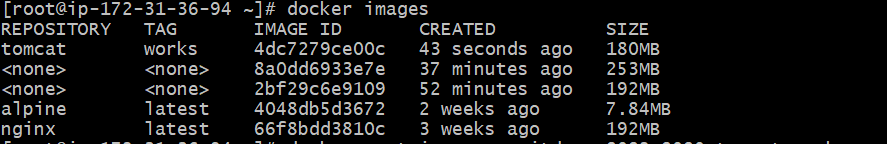
Downloaded sample.war file



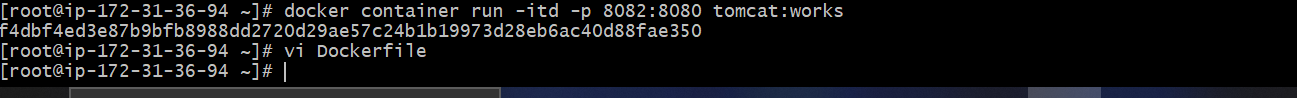
Execution



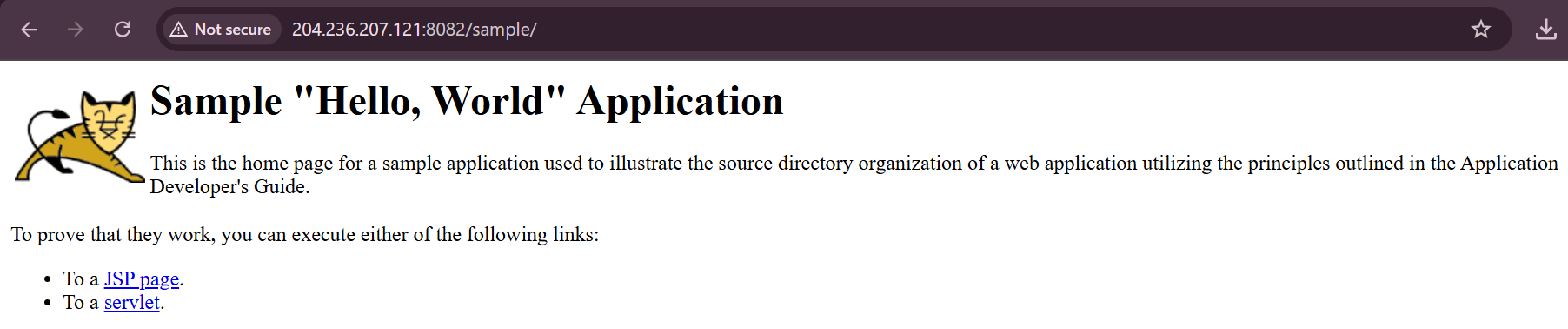
Now we can check whether the image has created using :docker ps



Running the container

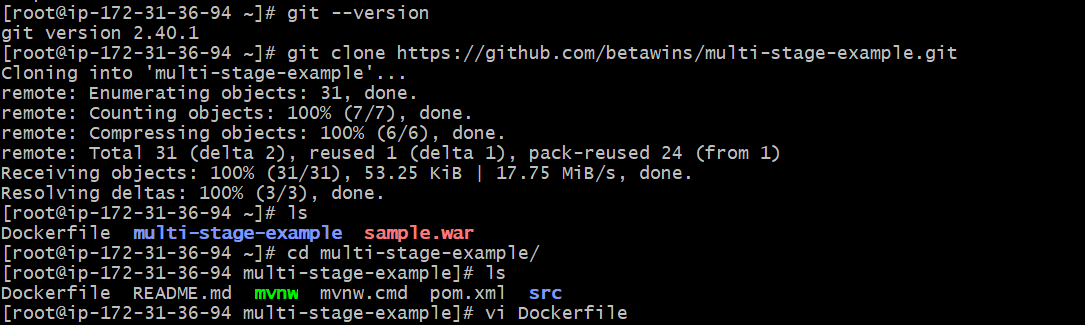


We can also accessin the browser with the port :8082

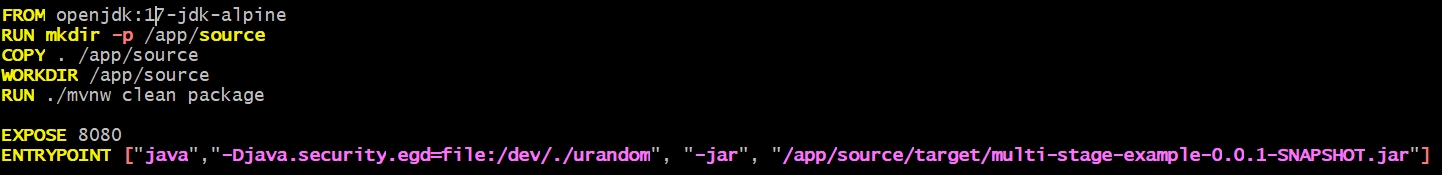


4) Create single stage and multi stage docker file using the below source code.  
   <https://github.com/betawins/multi-stage-example.git>

Single stage docker file



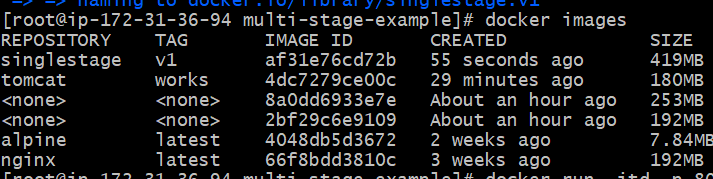
Edited the dockerfile



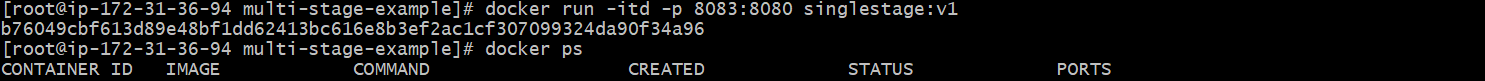
Executing the build



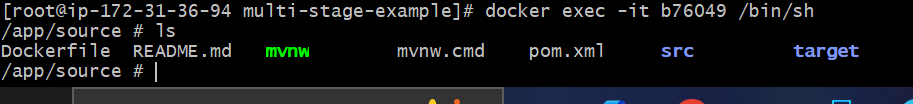
Ckecking the whether the image has created or not[singlestage]



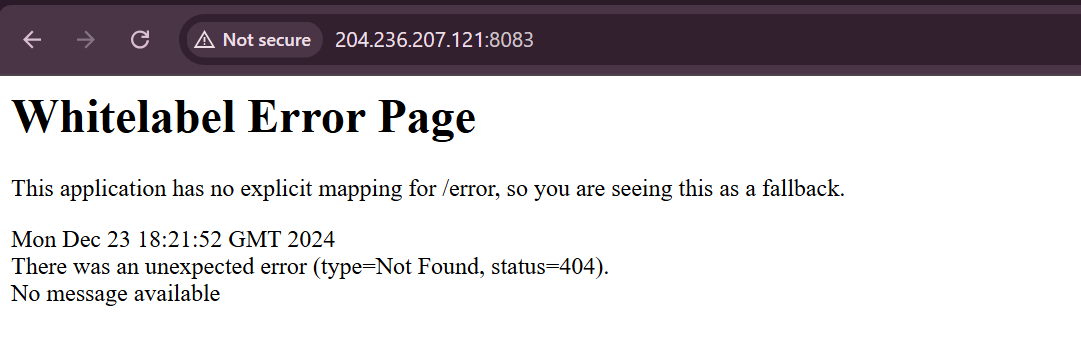
Now running



We can also see that source code is also copied in this image



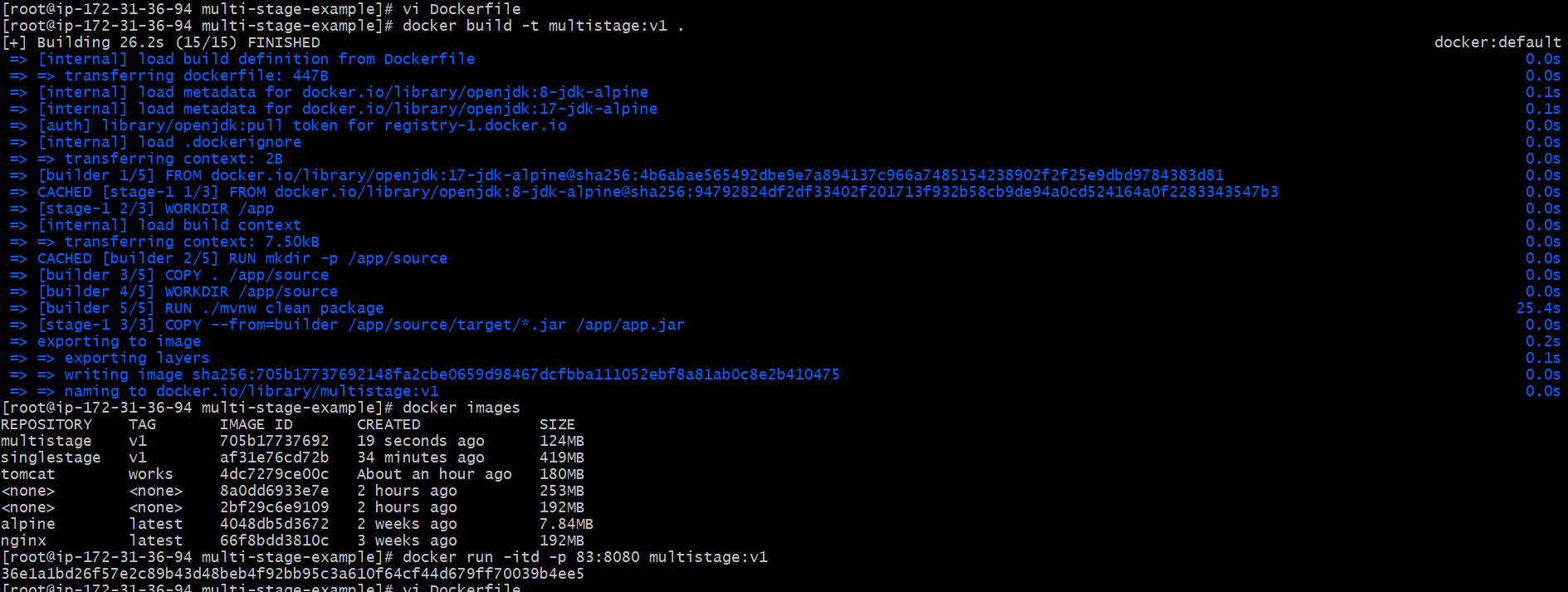
Accessing in the browser



MULTI-STAGE DOCKER FILE

Edit the dockerfile



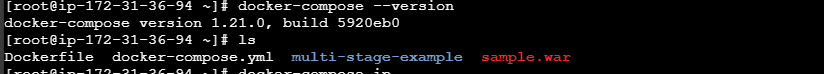


Accessing in the browser



5) Install docker compose and execute sample applciation.

Installing the docker compose



Creating a script docker-compose.yml file and write a script

version: '3'

services:

db:

image: mysql:5.7

volumes:

- db\_data:/var/lib/mysql

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=somewordpress

- MYSQL\_DATABASE=wordpress

- MYSQL\_USER=wordpress

- MYSQL\_PASSWORD=wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

ports:

- "8000:80"

restart: always

environment:

- WORDPRESS\_DB\_HOST=db:3306

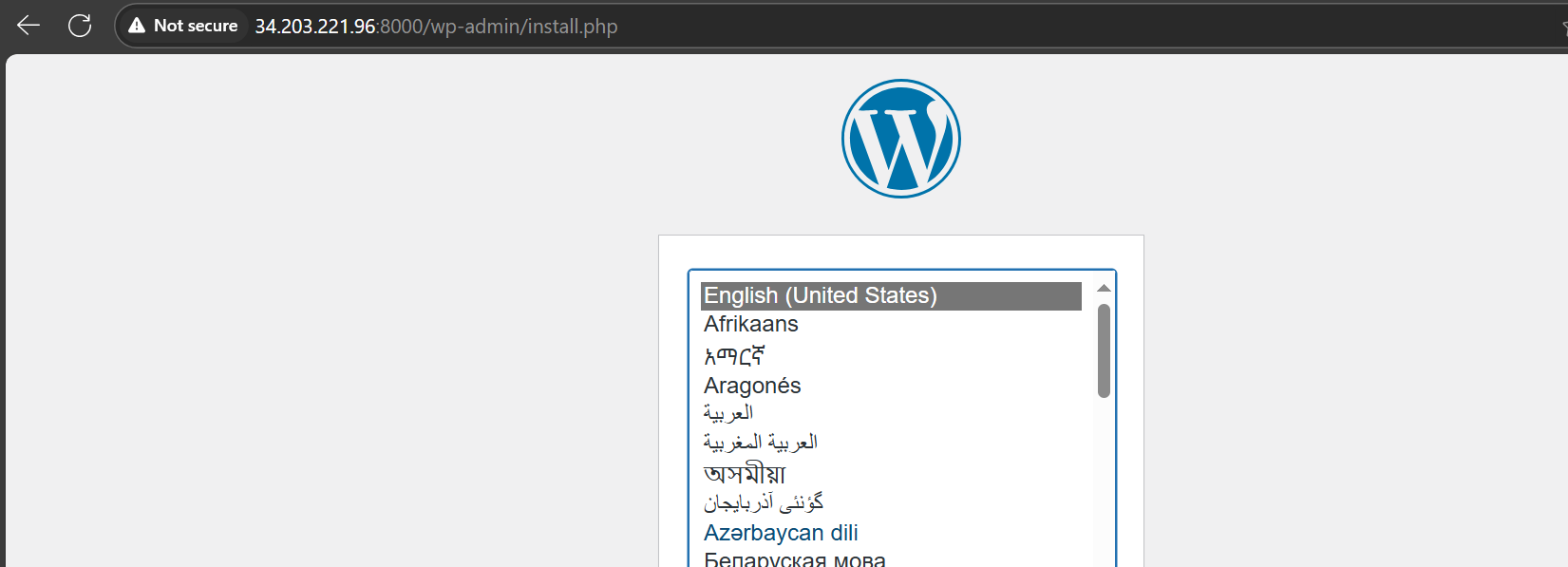
- WORDPRESS\_DB\_USER=wordpress

- WORDPRESS\_DB\_PASSWORD=wordpress

- WORDPRESS\_DB\_NAME=wordpress

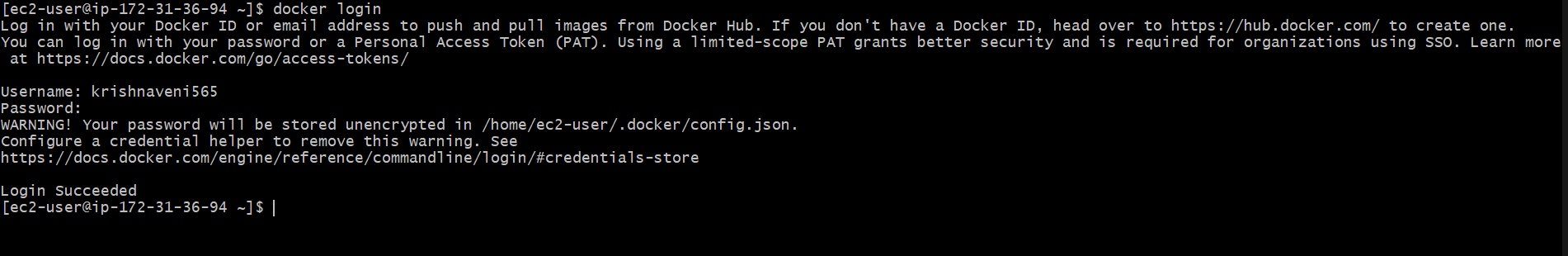
volumes:

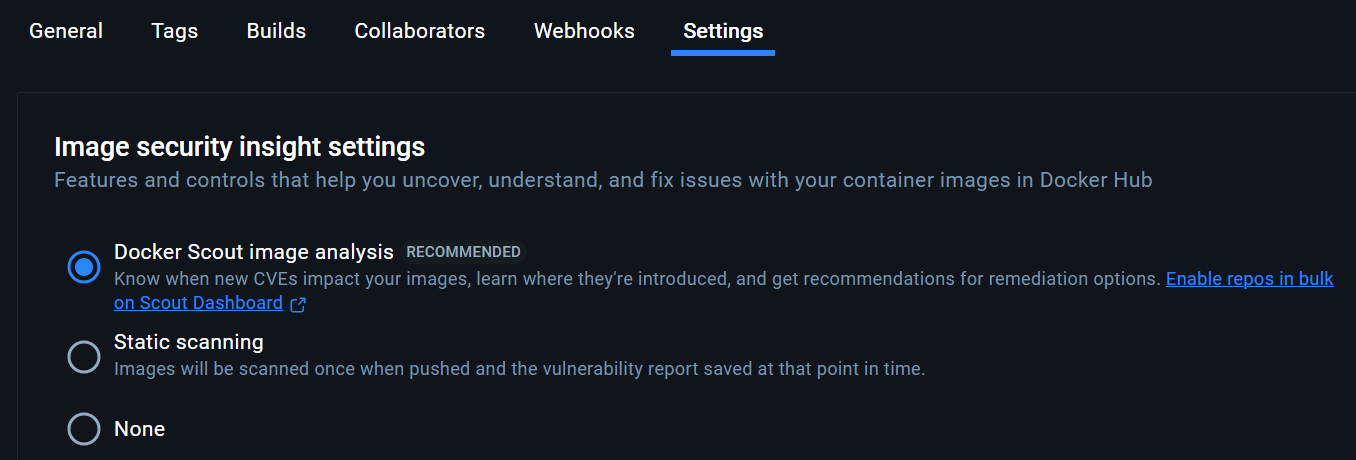
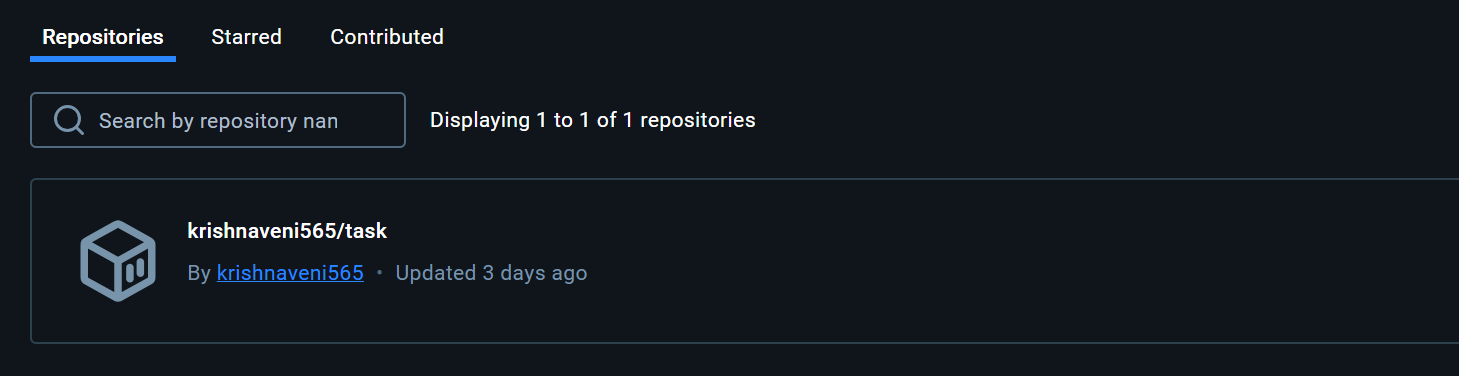
db\_data: { }



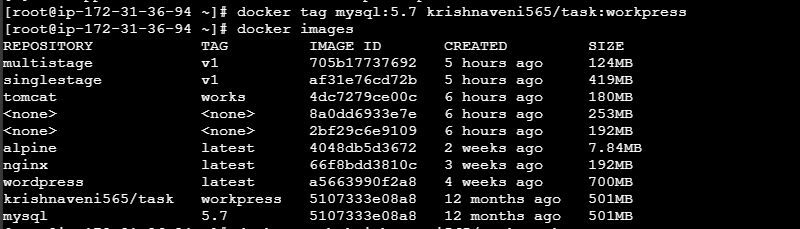
6) Implement solution to scan images when pushed to docker registry.

Login to docker hub

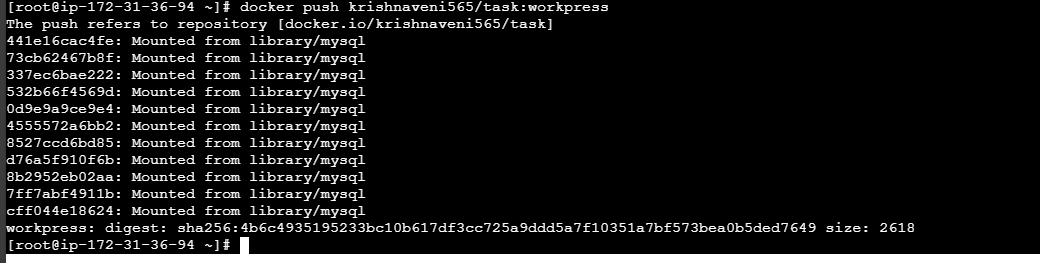
In the docker hub,create a repo

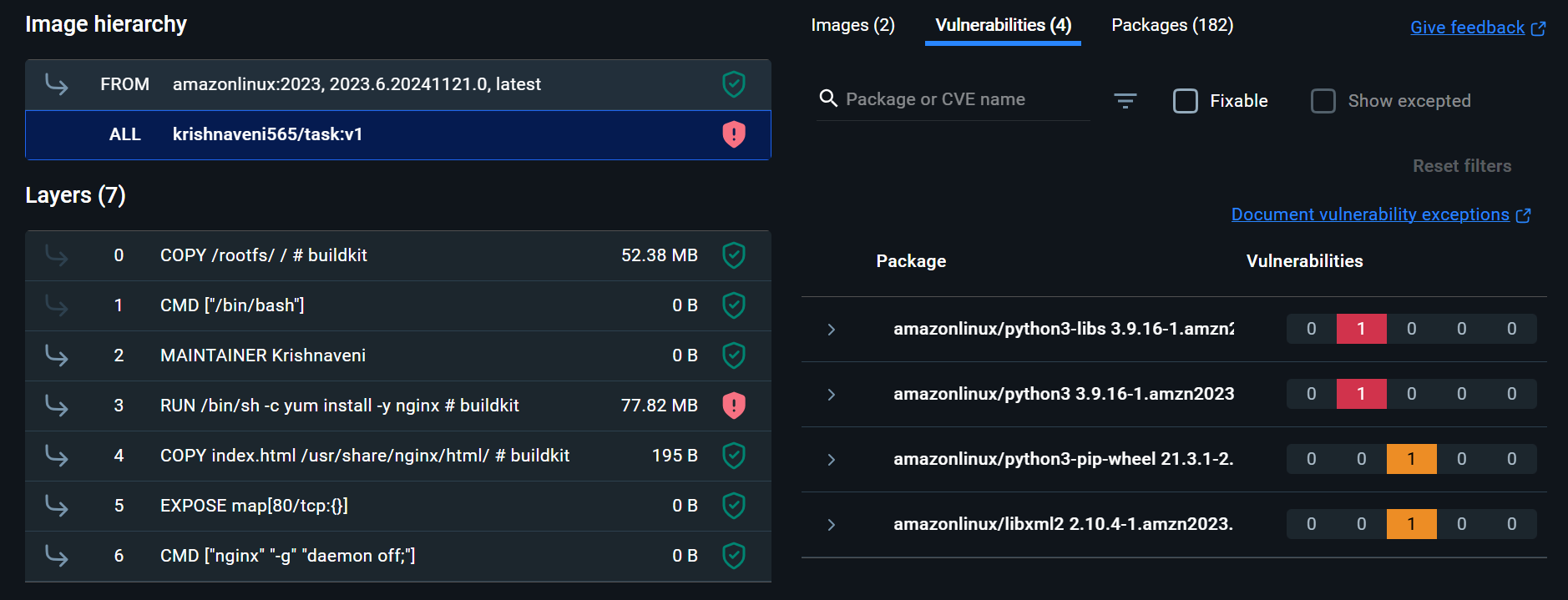


Tagging the image to push into the hub

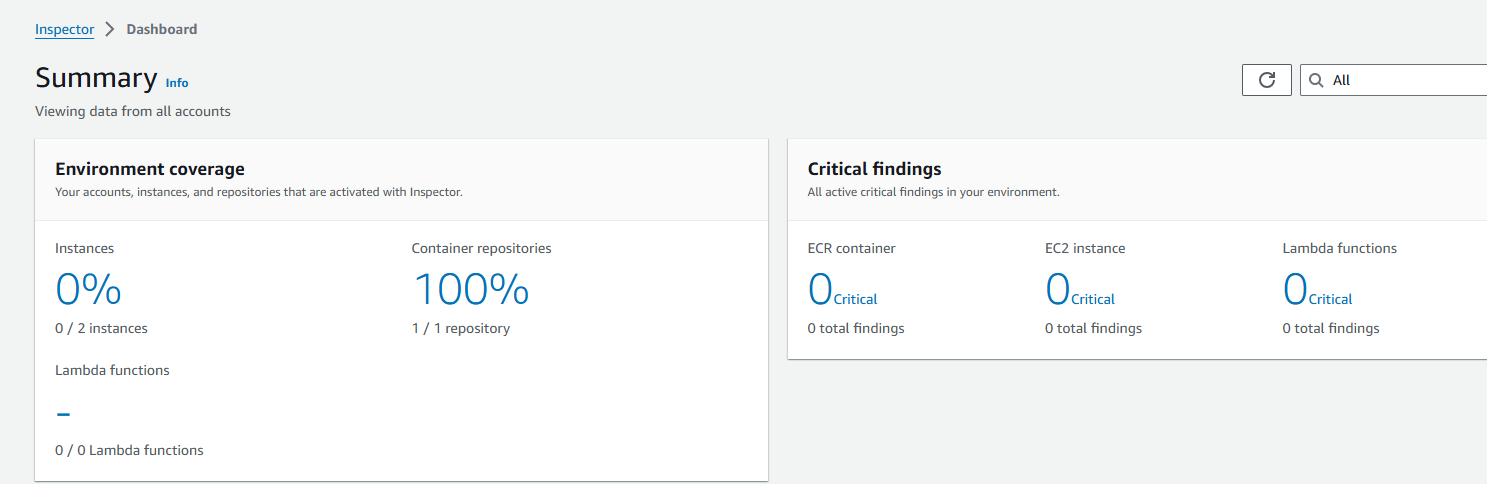
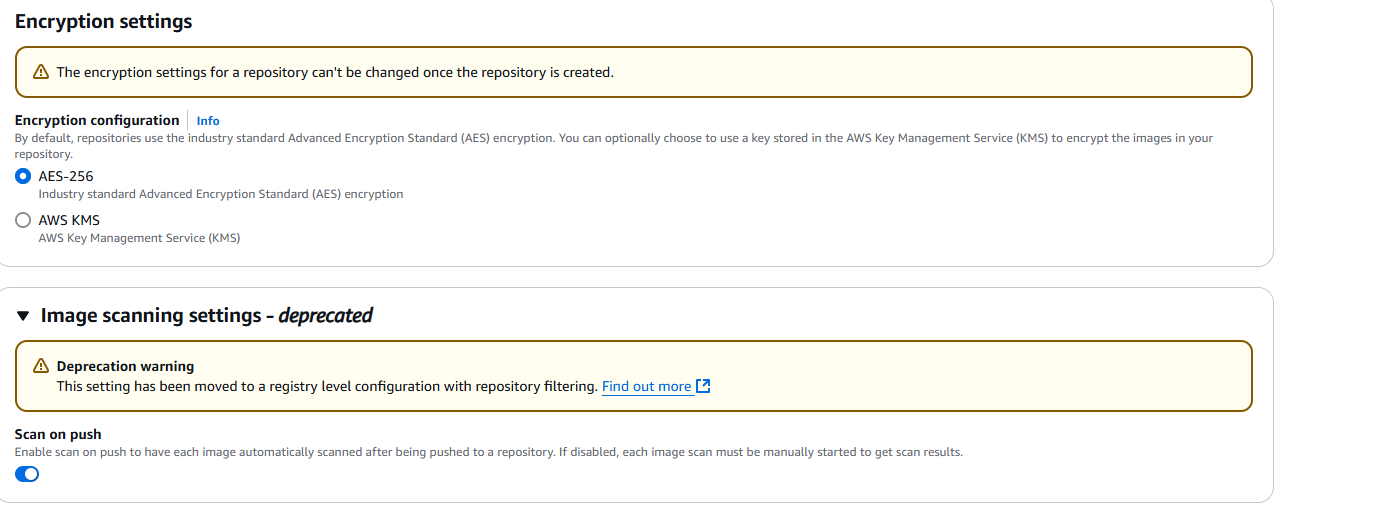


Then push into the docker hub

Now ,in dockerhub we can see the scanning

  
7) Implement solution to scan images when pushed to aws ecr.

Enable “scan on push” while creating the repo

  
8) Create a jenkins pipeline to create a docker image and push the image to dockerhub.

[GitHub - betawins/multi-stage-example: Multi Stage docker build example](https://github.com/betawins/multi-stage-example.git)

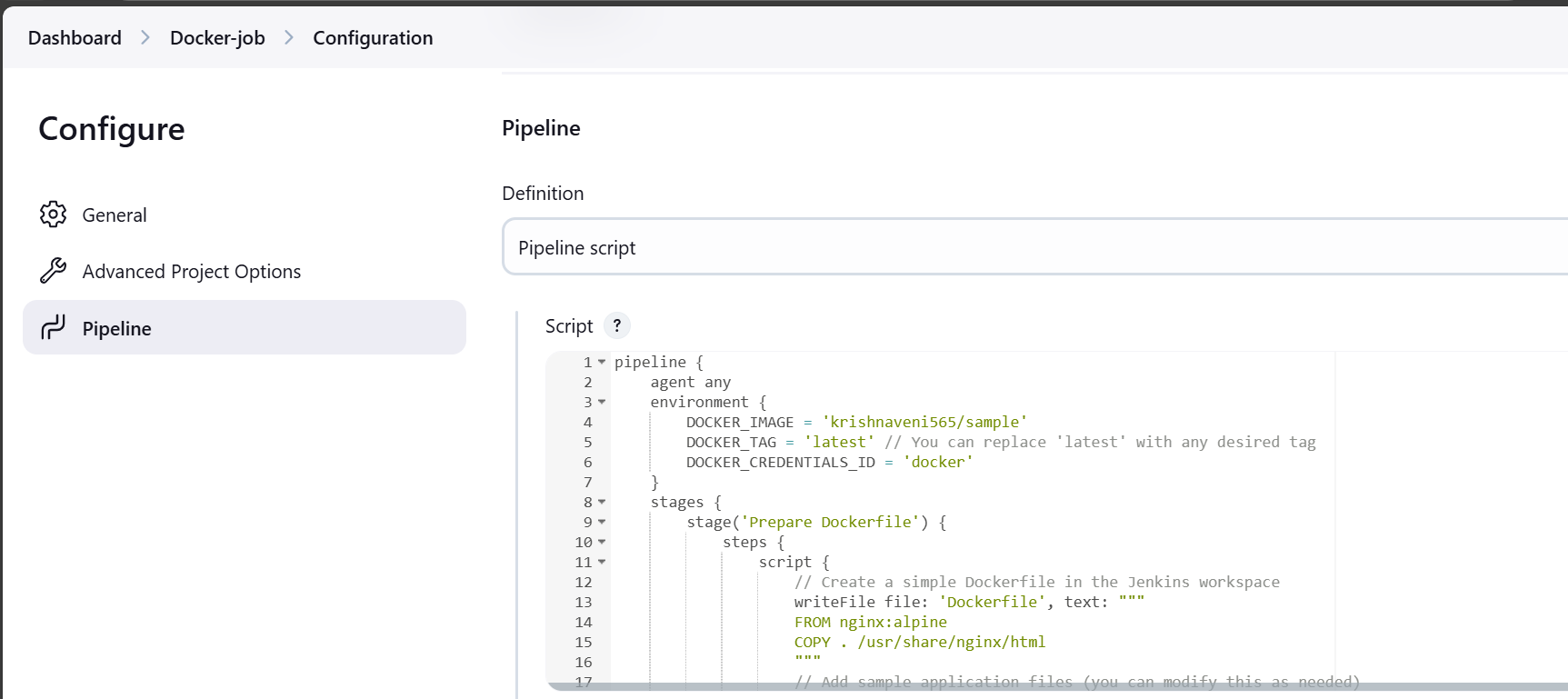
Multi Stage docker build example. Contribute to betawins/multi-stage-example development by creating an account on GitHub. (43 kB)

Install Jenkins ,and download plugins related to the docker

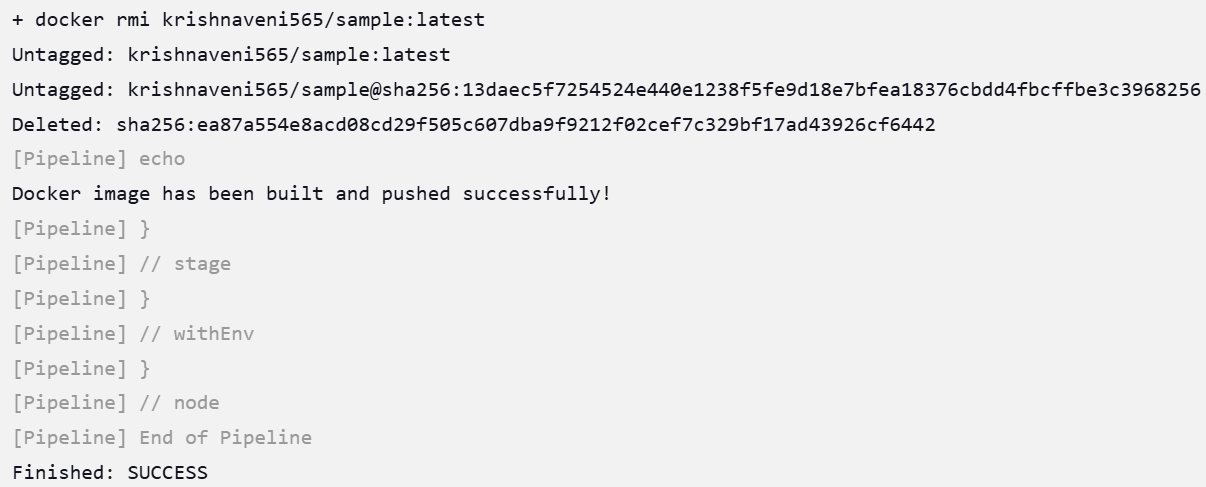
Now add the credentials in the Jenkins server with username and password of dockerhub

In Jenkins server, install docker and add permissions

Restart the Jenkins



Console output



We can check in dockerhub registry

