**Docker Task -3**

1. **Create a image from running container.**
2. **Copy image from local machine to docker server and load the image.**
3. **Create Docker image using alpine and customize with tomcat.**
4. **Create single stage and multi stage docker file using this source code:**[**https://github.com/betawins/multi-stage-example.git**](https://github.com/betawins/multi-stage-example.git)
5. **Install docker compose and execute sample application.**
6. **Implement solution to scan images when pushed to docker registry.**
7. **Implement solution to scan images when pushed to AWS ECR.**
8. **Create a Jenkins pipeline to create a docker image and push the image to Docker hub.**

**1. Create a image from running container.**

**Step 1: List running containers**

docker ps

This will show all currently running containers.

### tep 2: Commit the running container as an image

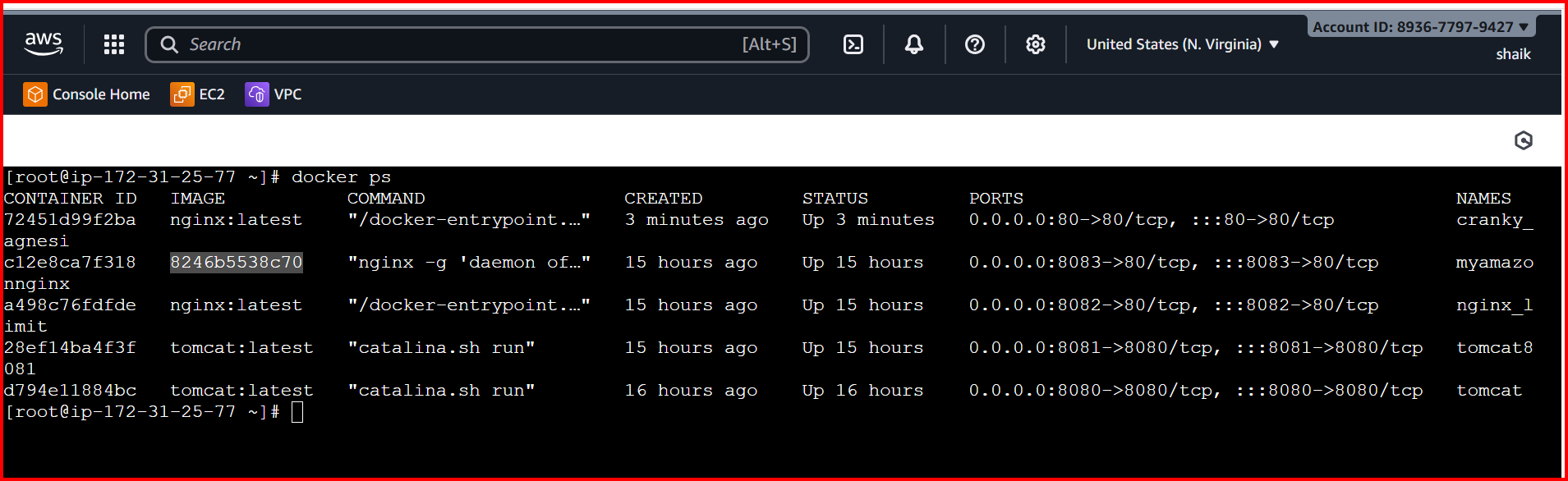
Use this command:

docker commit <container\_id\_or\_name> <new\_image\_name>:<tag>

### Step 3: Verify the new image

docker images

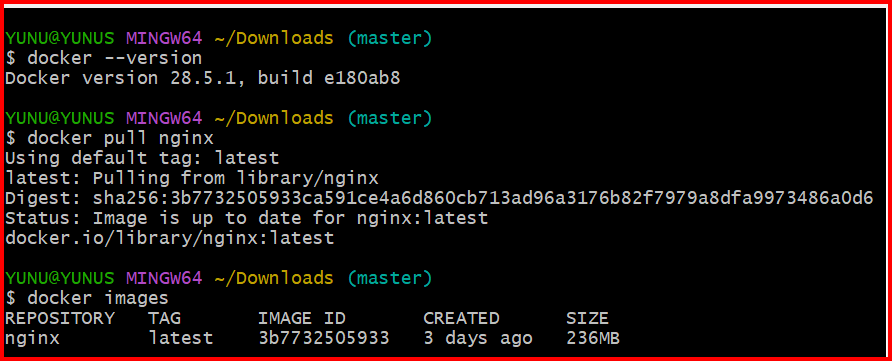
You should see your new image listed:



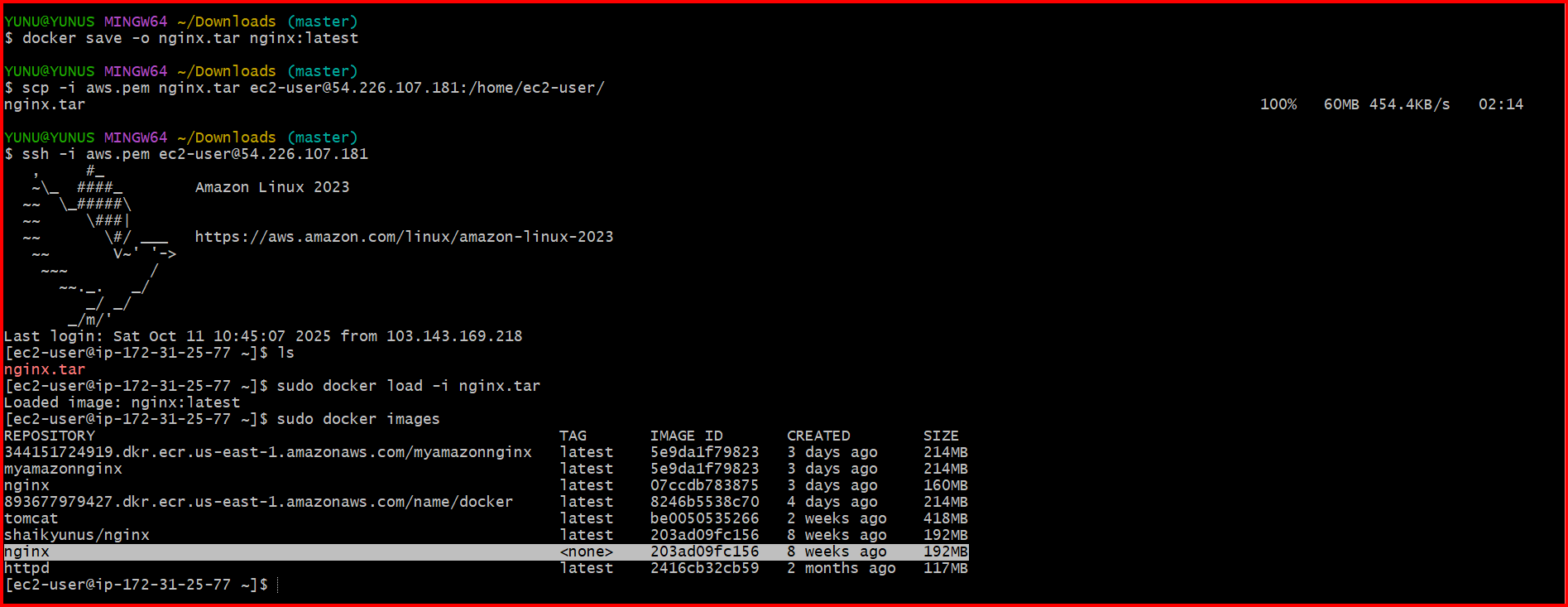
**2.Copy image from local machine to docker server and load the image.**

Download the docker desktop

pull any docker image in your local machine and push to your docker server

****

Copy the image to your remote Docker server

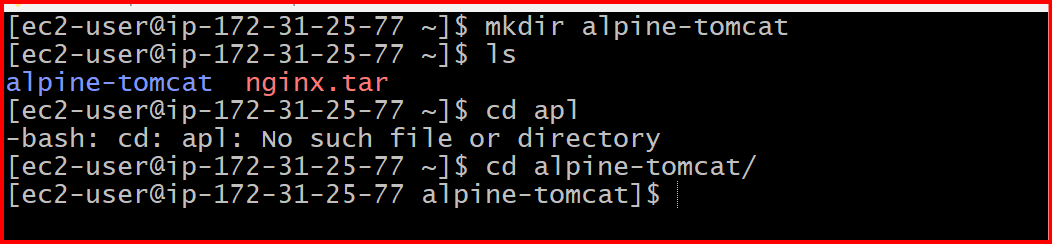
****

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

## Create a project directory

mkdir alpine-tomcat

cd alpine-tomcat



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

**Create a working directory**

mkdir tomcat-alpine

cd tomcat-alpine

**Create docker file**

Vi docker file

**Paste the following content inside**

FROM alpine:latest

LABEL maintainer="you@example.com"

LABEL description="Tomcat on Alpine with OpenJDK 17"

RUN apk update && apk add --no-cache openjdk17-jre curl tar bash

RUN mkdir -p /usr/local/tomcat

WORKDIR /usr/local/tomcat

RUN curl -fSL https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.110/bin/apache-tomcat-9.0.110.tar.gz -o apache-tomcat-9.0.110.tar.gz && \

tar -xzf apache-tomcat-9.0.110.tar.gz --strip-components=1 && \

rm apache-tomcat-9.0.110.tar.gz

EXPOSE 8080

CMD ["bin/catalina.sh", "run"]

**Build and test:**

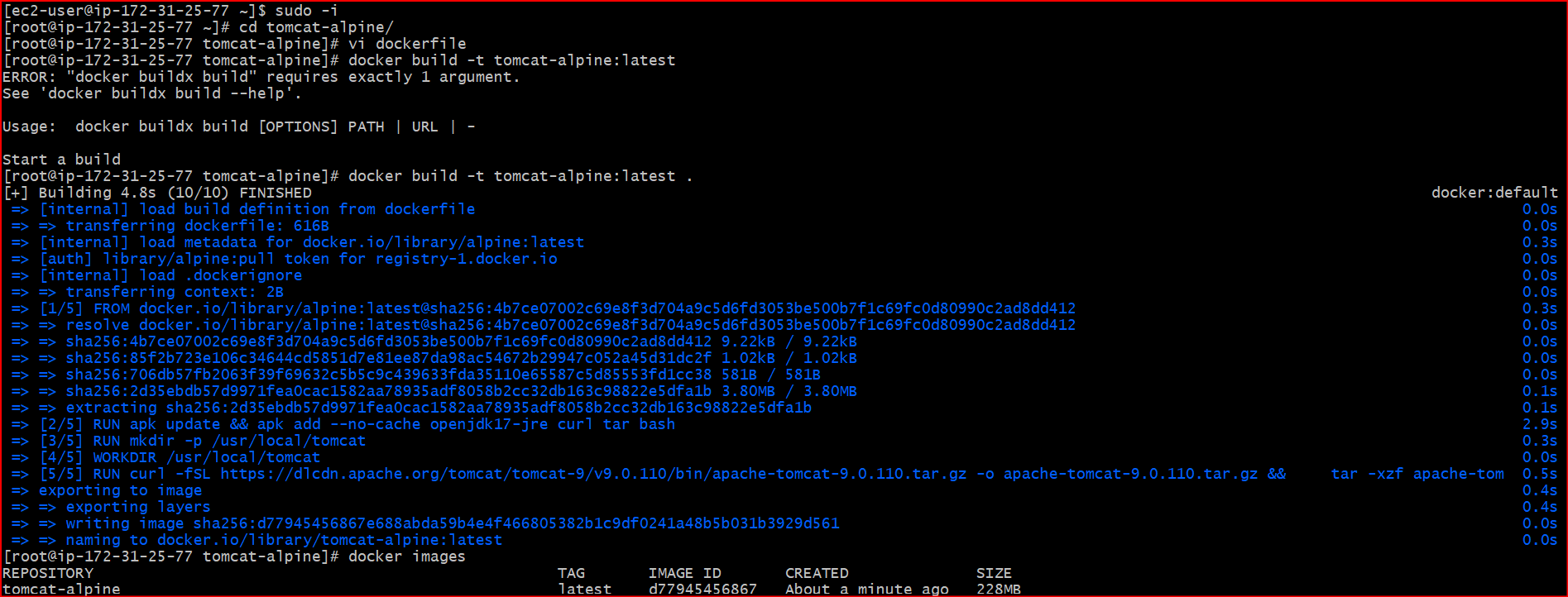
docker build -t tomcat-alpine:latest .

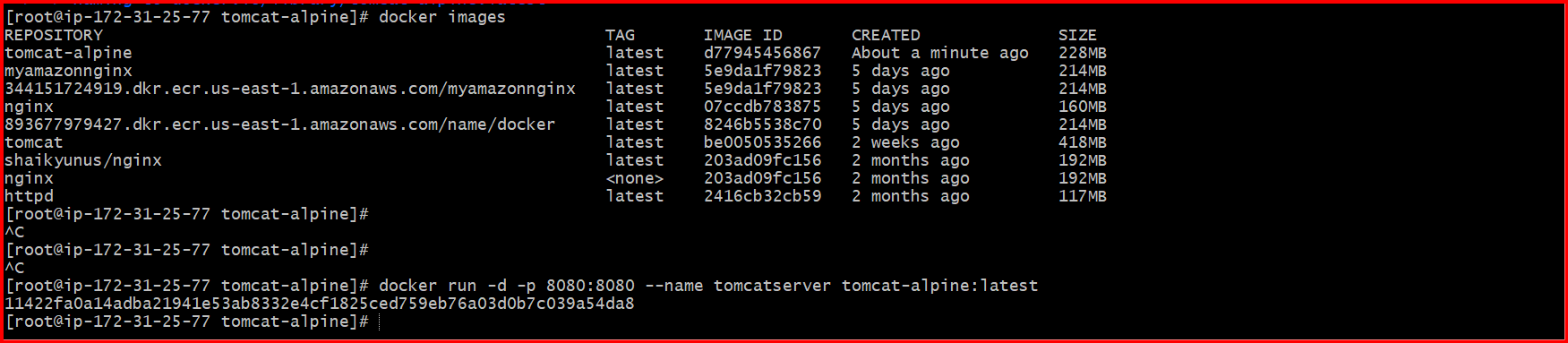
**If build succeeds, run:**

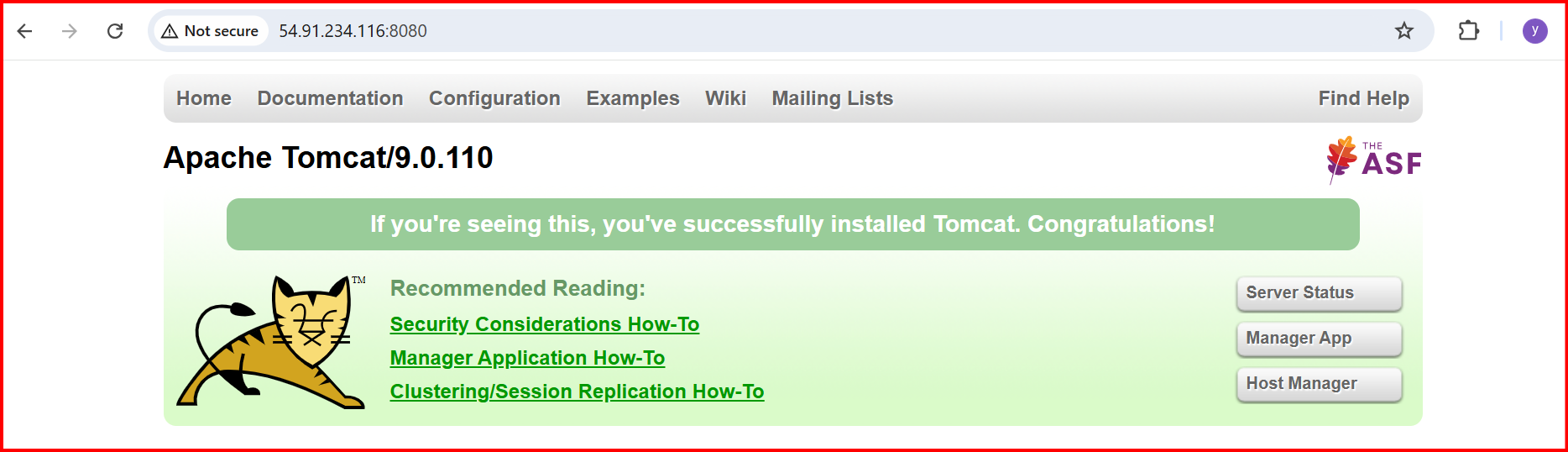
docker run -d -p 8080:8080 --name tomcatserver tomcat-alpine:latest

**Then check:**

http://<your-ec2-ip>:8080





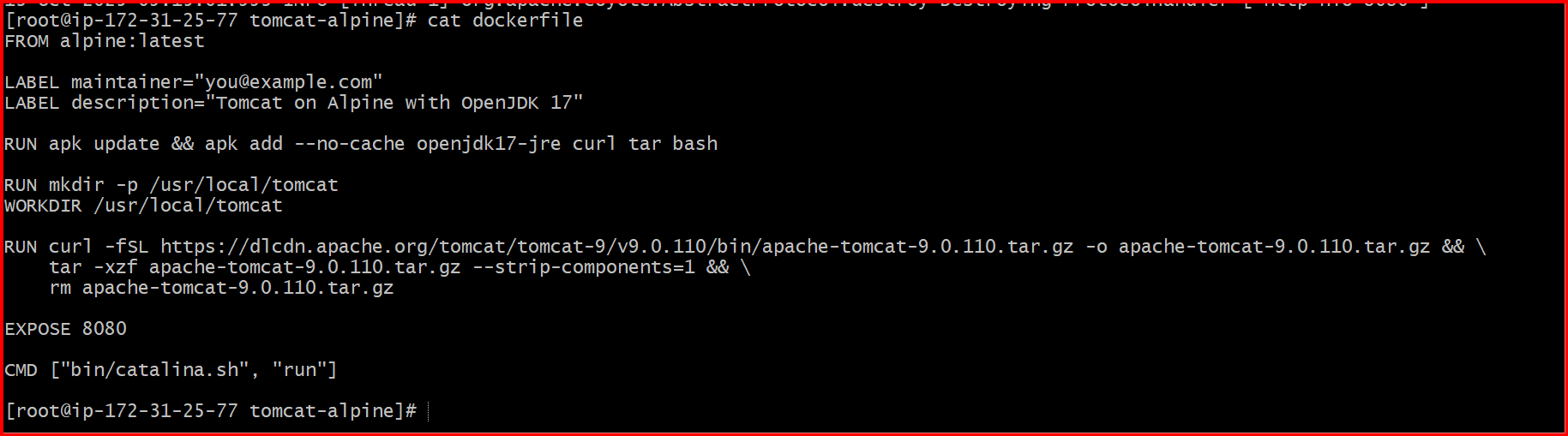


**4.Create single stage and multi stage docker file using this source code:**[**https://github.com/betawins/multi-stage-example.git**](https://github.com/betawins/multi-stage-example.git)

**First clone the git repo**

****

**Check the docker file**

****

**cd multi-stage-example**

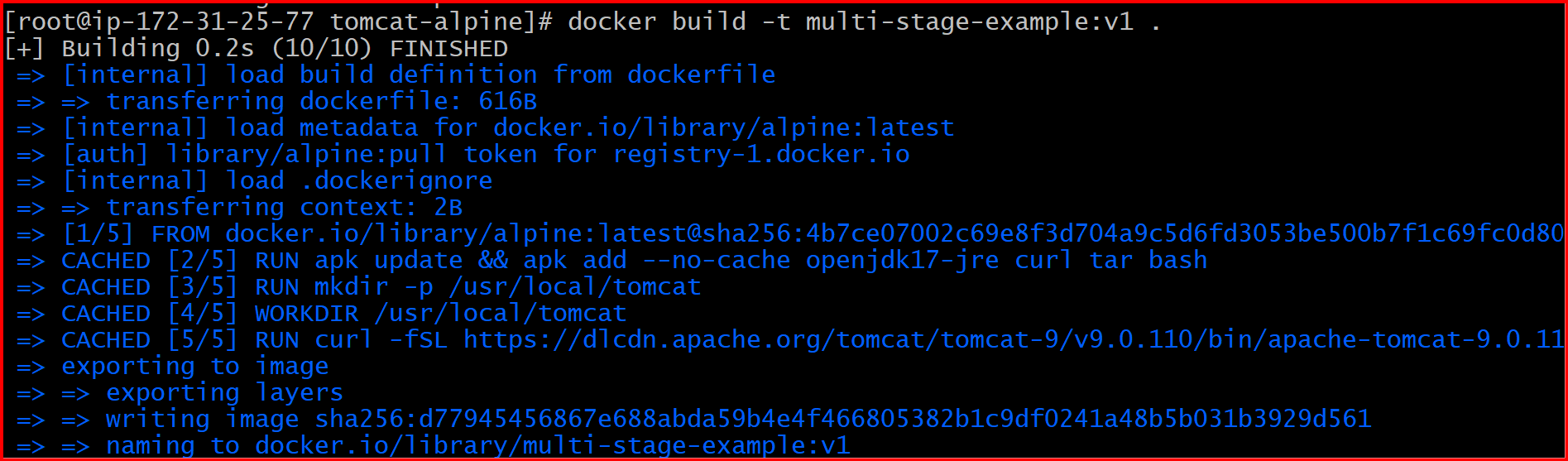
**Build the Docker Image:**

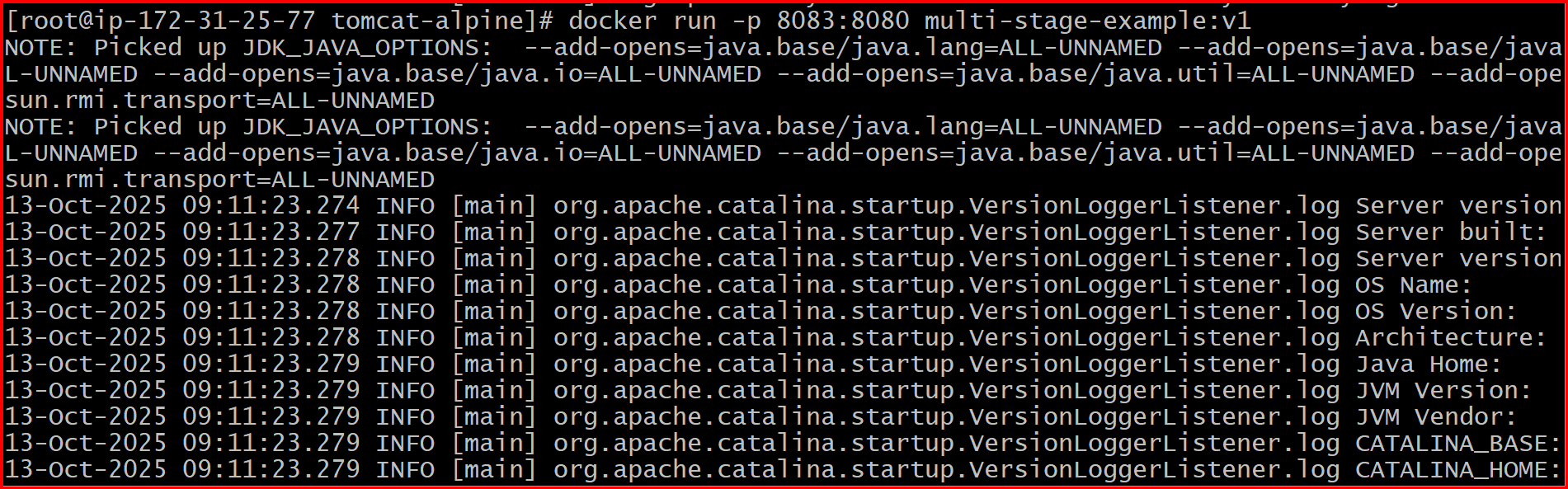
**For the multi-stage Dockerfile:**

docker build -t multi-stage-example:v1 .

**Run the Docker Container:**

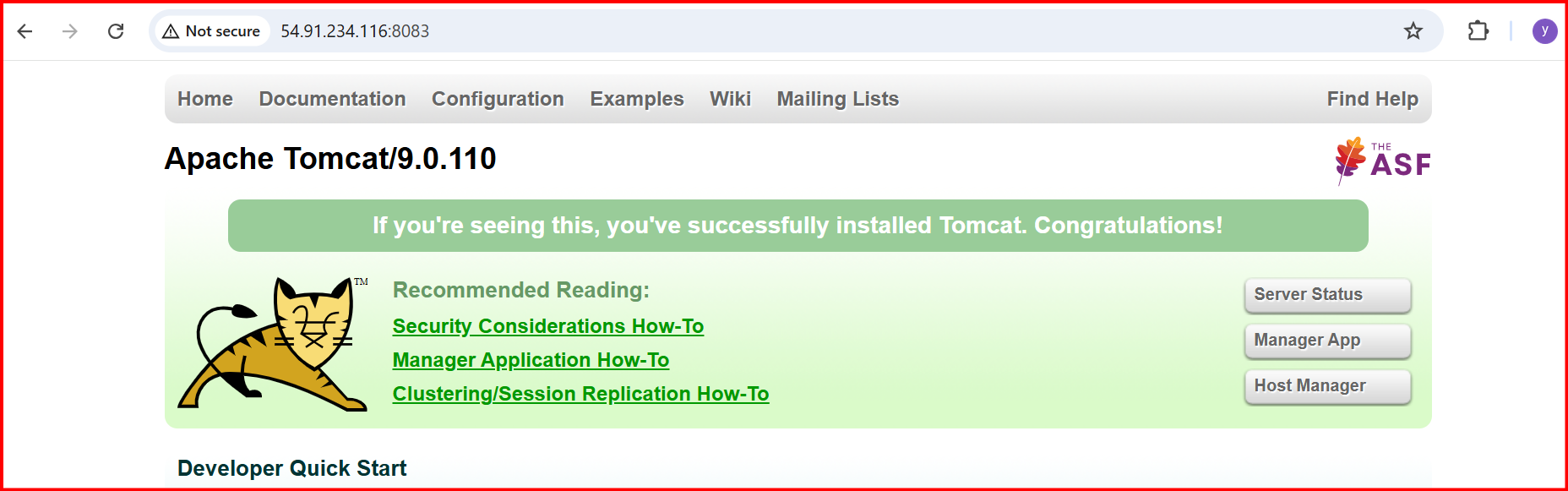
docker run -p 8083:8080 multi-stage-example:v1





Check in browser

<http://54.91.234.116:8083/>



**5.Install docker compose and execute sample application.**

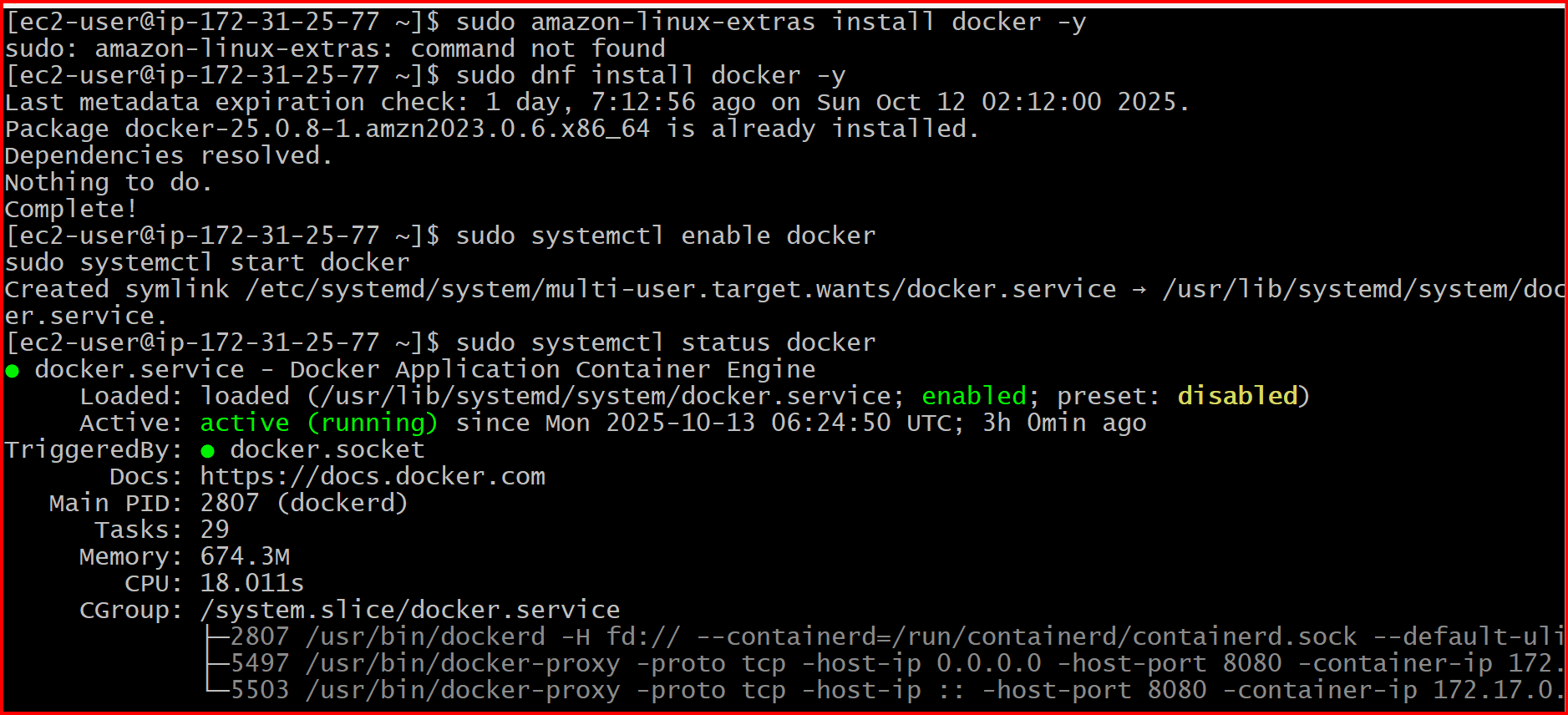
Install Docker

**sudo dnf install docker -y**

**sudo systemctl enable docker**

**sudo systemctl start docker**

**Sudo systemctl status docker**

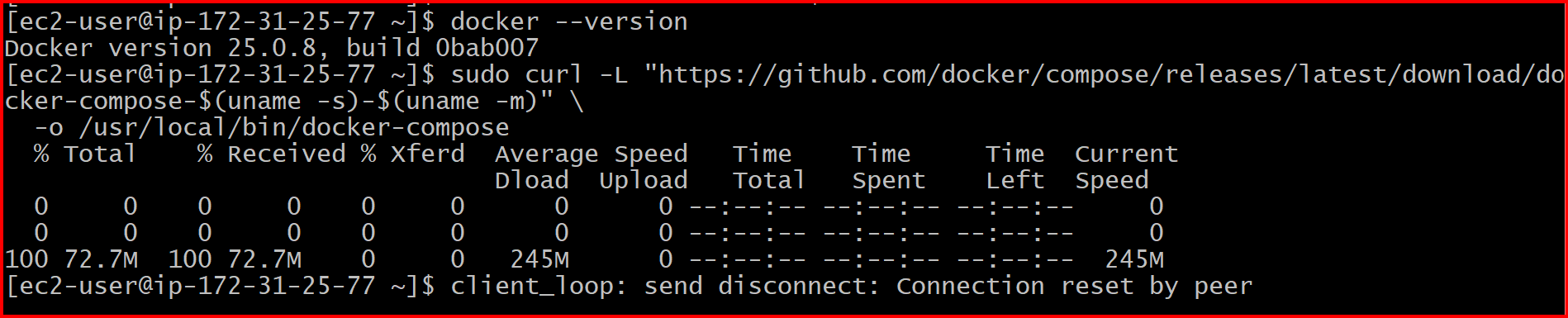
****

**docker –version**

**Install Docker Compose**

sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" \

-o /usr/local/bin/docker-compose

****

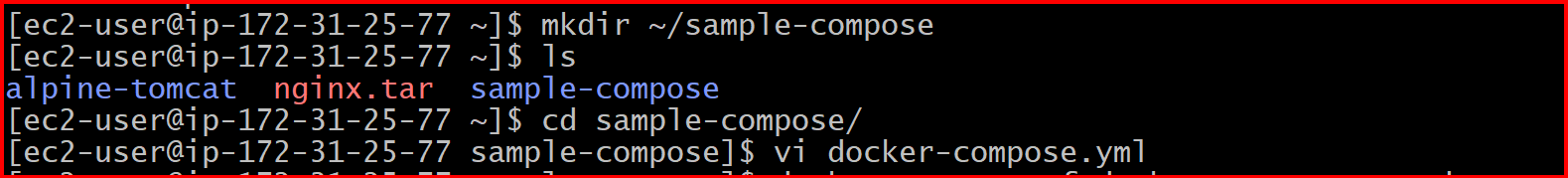
sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version

**Create a Sample Docker Compose Project**

mkdir ~/sample-compose

cd ~/sample-compose



**Create a docker-compose.yml file:**

Vi docker-compose.yml

version: "3.8"

services:

web:

image: nginx:latest

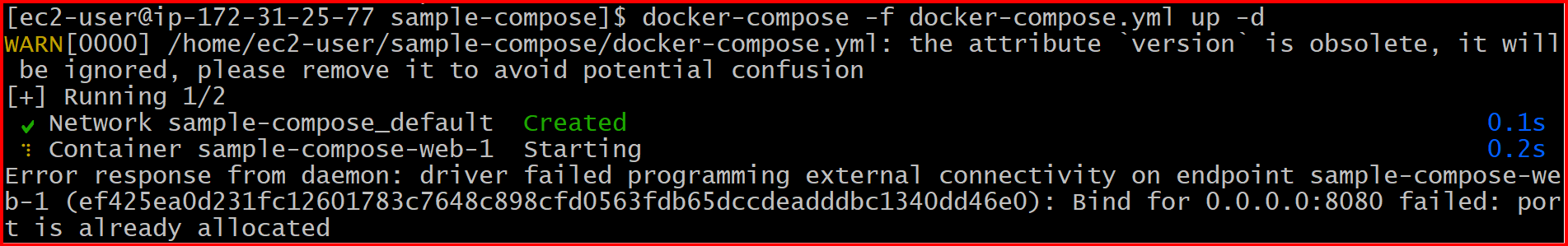
ports:

- "8080:80"

**Start the Application**

**docker-compose -f docker-compose.yml up -d**

**docker ps**

****

**docker-compose up -d**

[**http://54.91.234.116:8080/**](http://54.91.234.116:8080/)

****

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

**First check docker is running or not if not install docker**

sudo yum update -y

sudo yum install docker -y

sudo systemctl start docker

check the docker version

docker –version

**Install dependencies:**

sudo yum update -y

sudo yum install wget curl git -y

**Download the latest Trivy binary:**

wget https://github.com/aquasecurity/trivy/releases/download/v0.55.2/trivy\_0.55.2\_Linux-64bit.tar.gz

**Extract and install:**

tar zxvf trivy\_0.55.2\_Linux-64bit.tar.gz

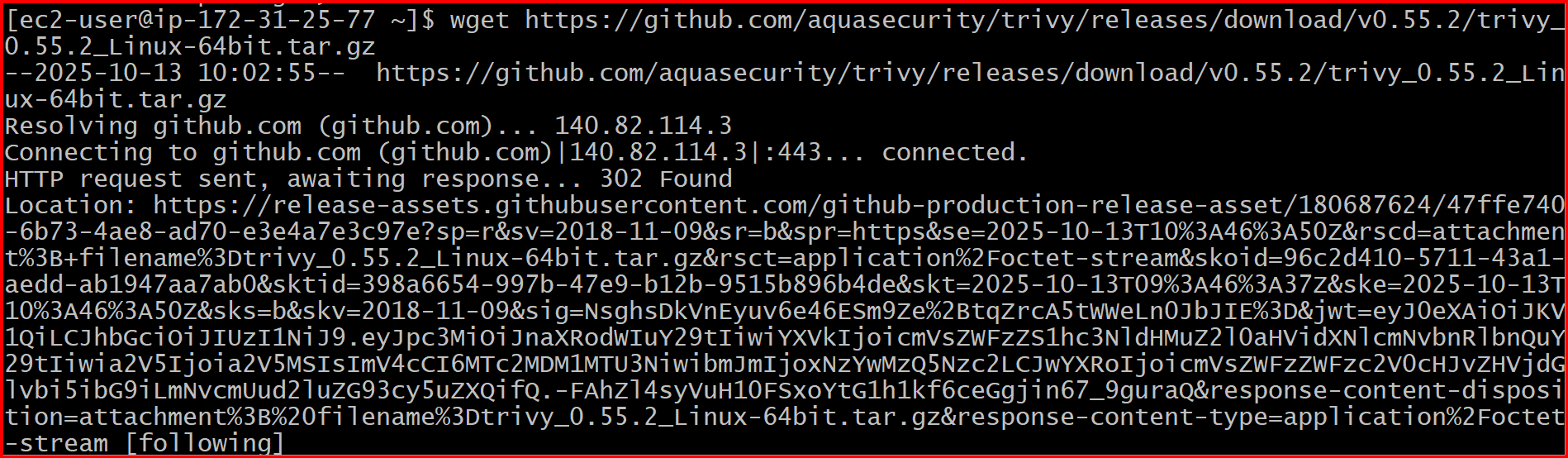
sudo mv trivy /usr/local/bin/

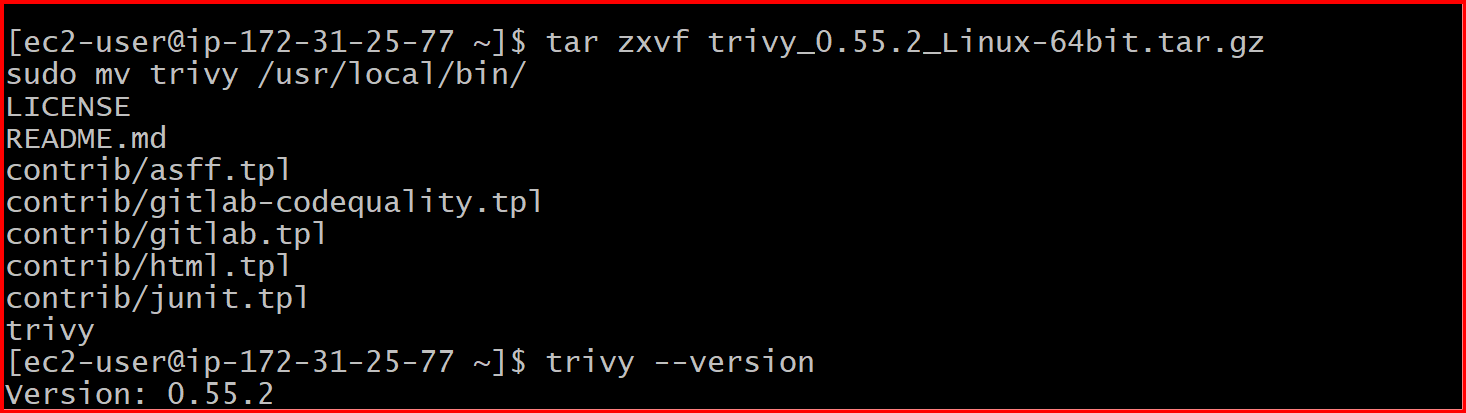
**Verify installation:**

trivy --version

**Expected output:**

Version: 0.55.2

****

****

**Pull, Tag, and Push an Image to Docker Hub**

**Docker login**

**Pull the base image:**

docker pull nginx:latest

**Run it to verify:**

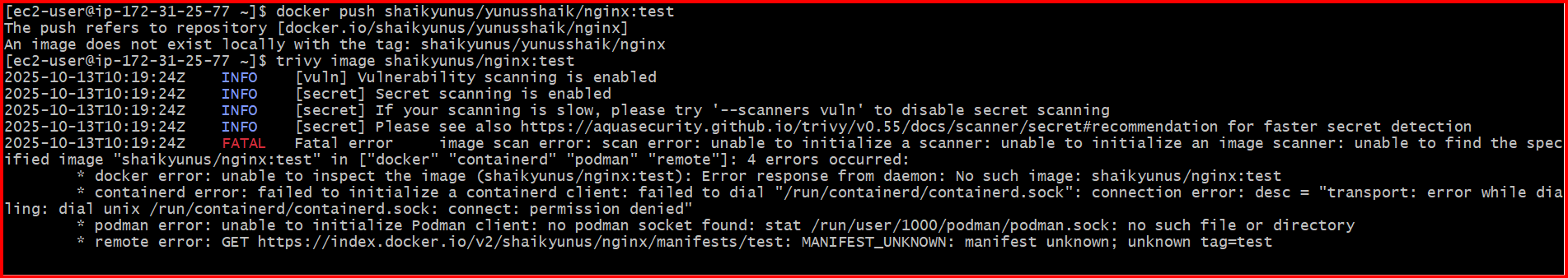
docker container run -itd -p 80:80 nginx:latest

**Tag the image for your Docker Hub account:**

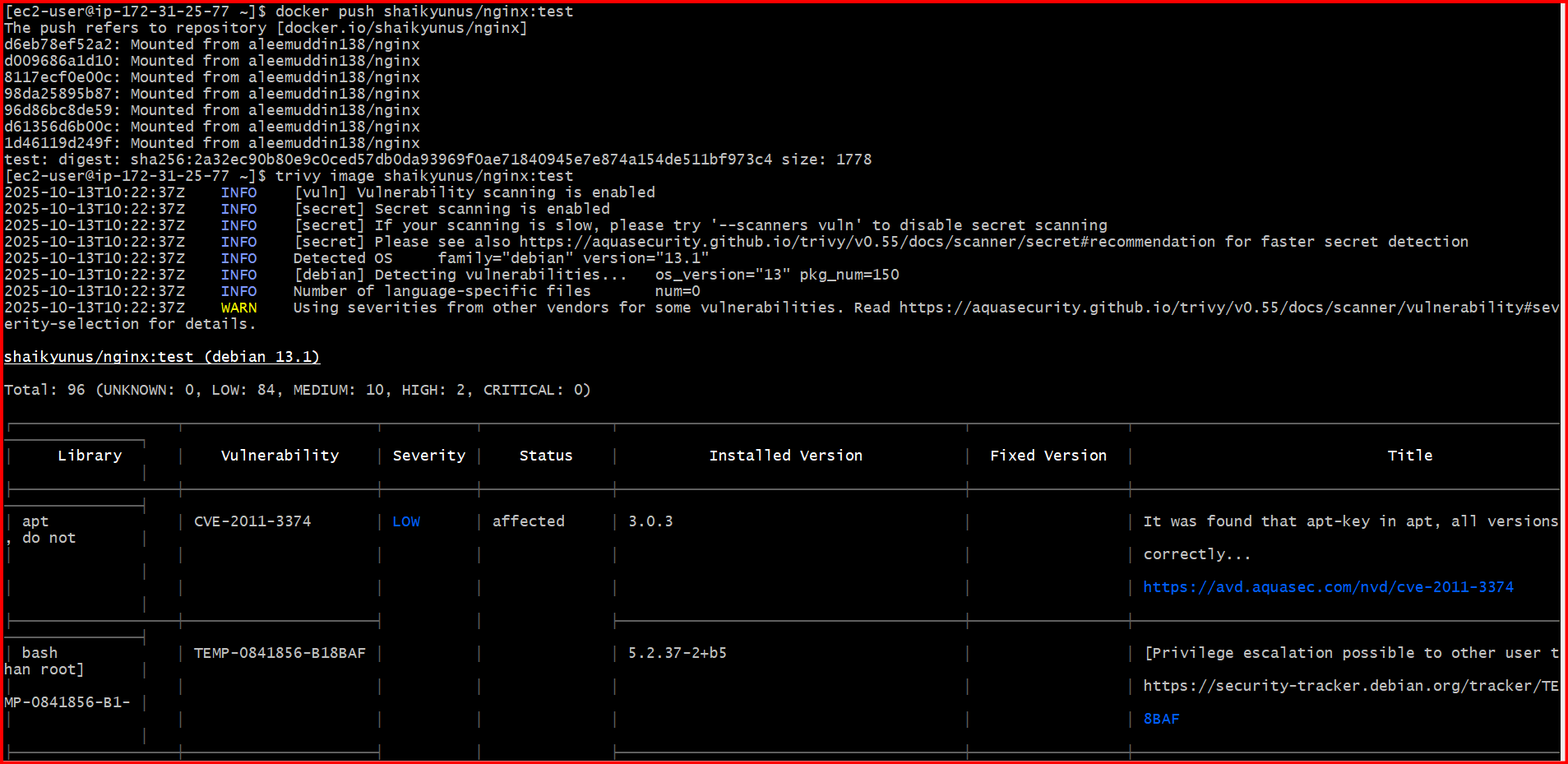
docker tag nginx:latest aleemuddin138/nginx:test

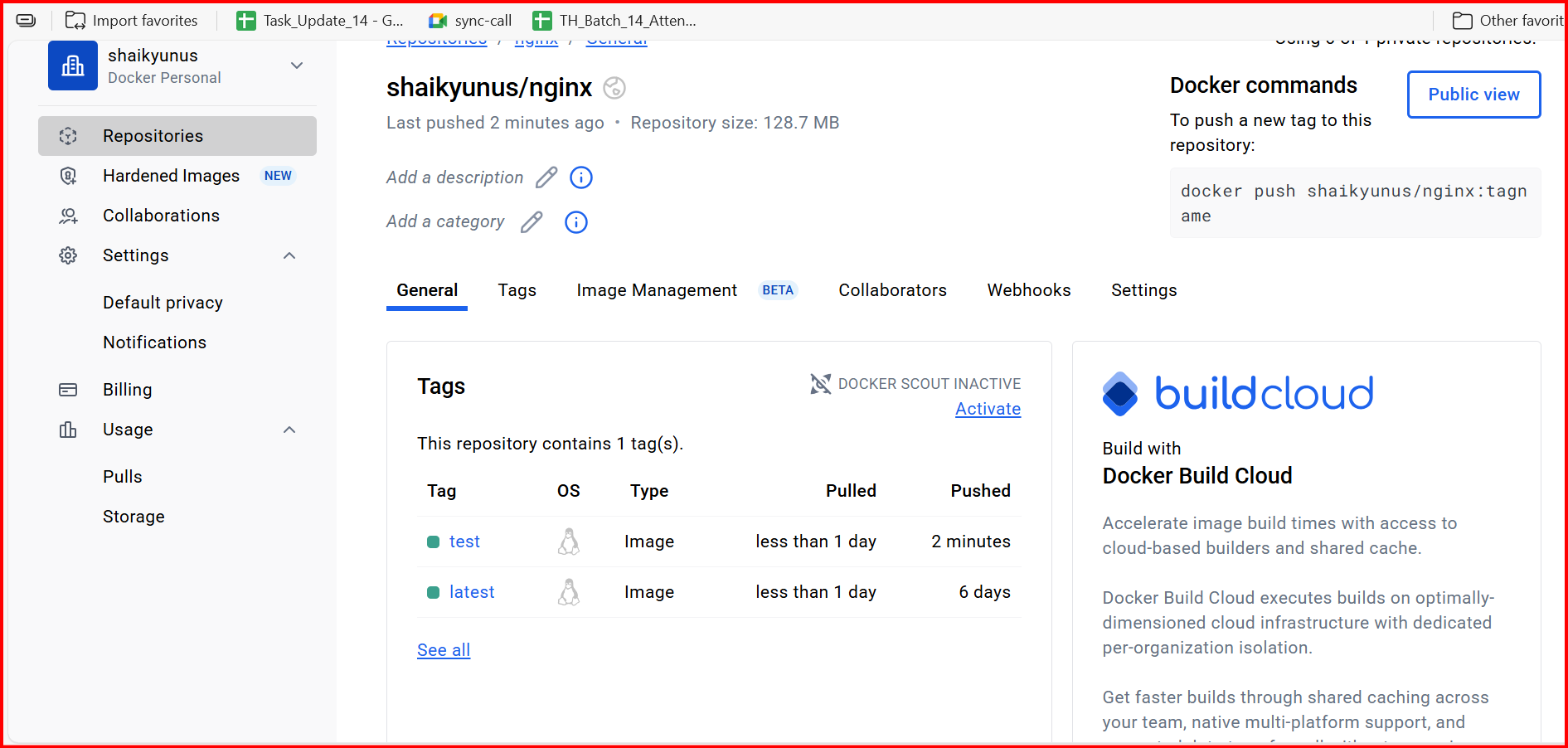
Push it to your Docker Hub repository:

docker push shaikyunus/nginx:test



trivy image shaikyunus/nginx:test



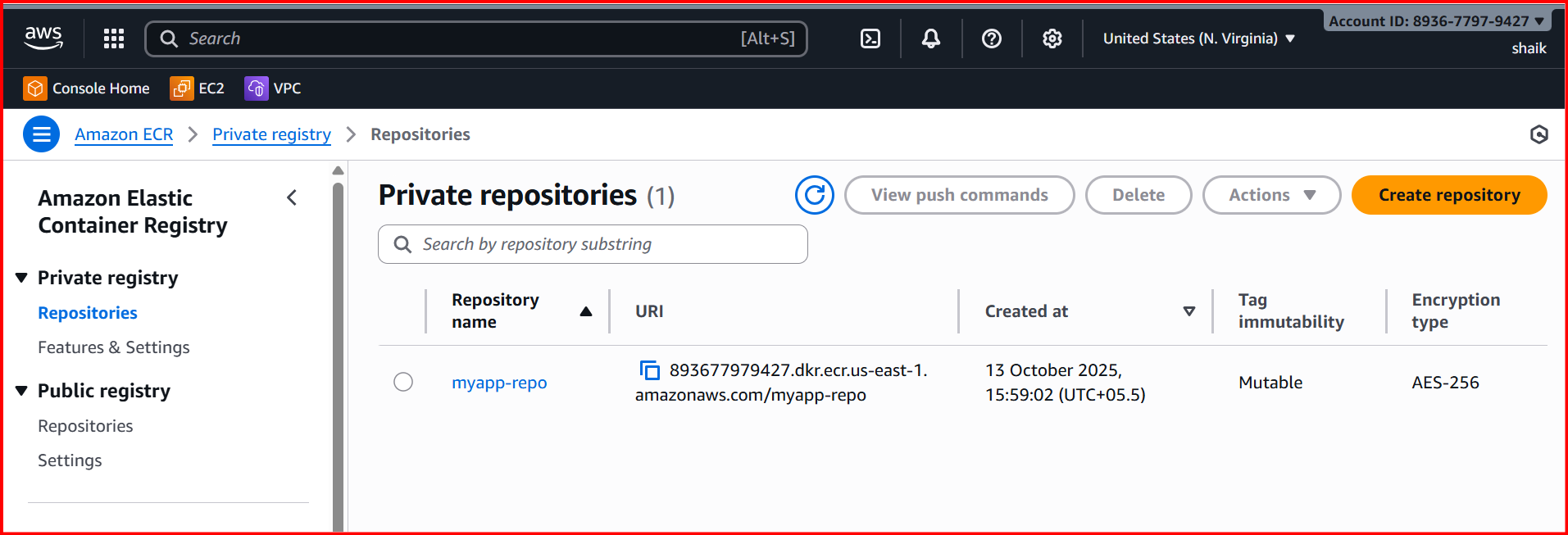
****

**7.Implement solution to scan images when pushed to AWS ECR.**

**Create the repository in *your own* ECR**

aws ecr create-repository --repository-name myapp-repo --region us-east-1

****

****

1. **Tag your image with your own account ID:**

docker tag nginx:latest <account id>.dkr.ecr.us-east-1.amazonaws.com/myapp-repo:latest

1. **Log in:**

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin <account id>.dkr.ecr.us-east-1.amazonaws.com

1. **Push:**

docker push <account id>.dkr.ecr.us-east-1.amazonaws.com/myapp-repo:latest

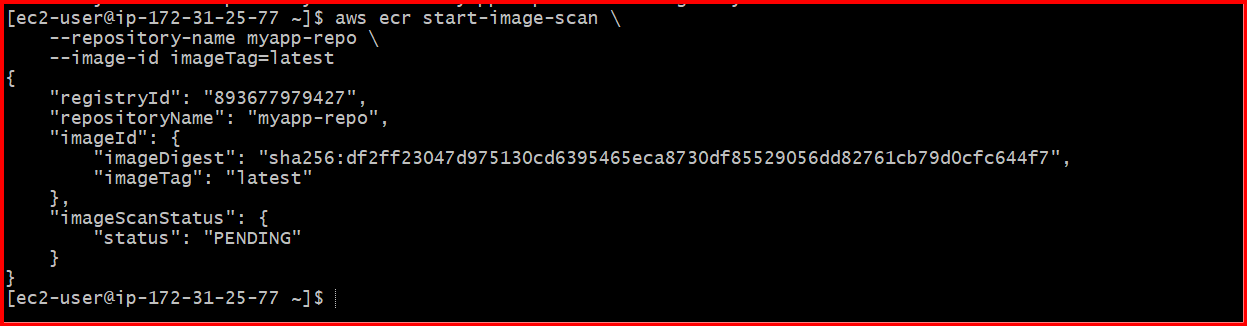


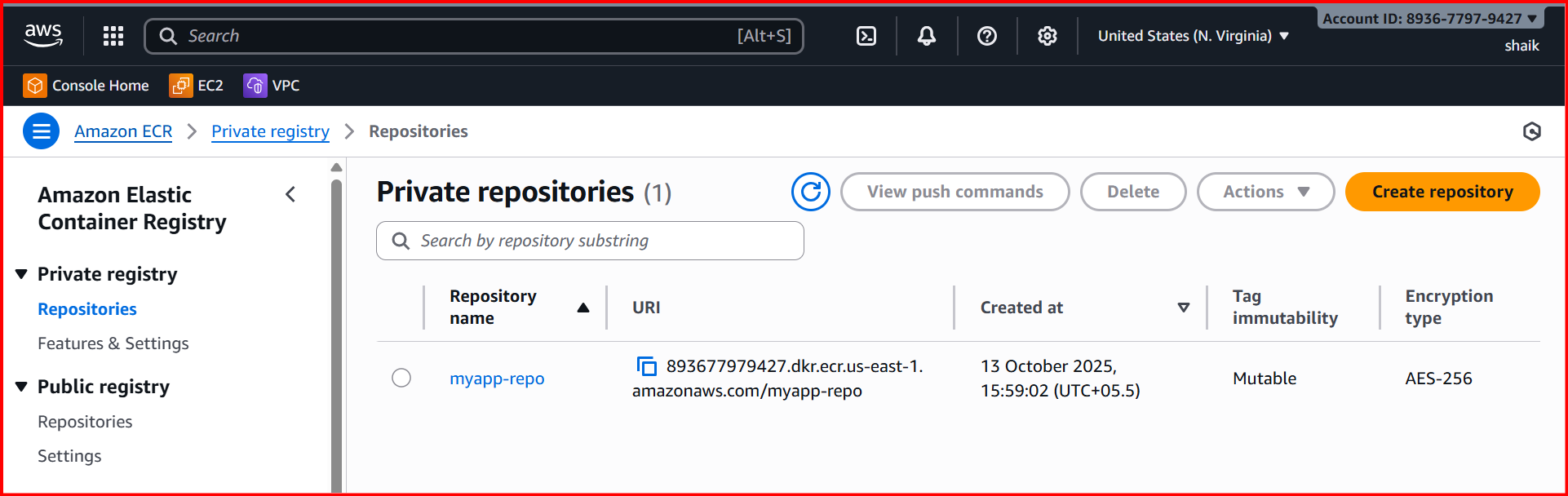
**View Scan Results**

aws ecr start-image-scan \

--repository-name myapp-repo \

--image-id imageTag=latest

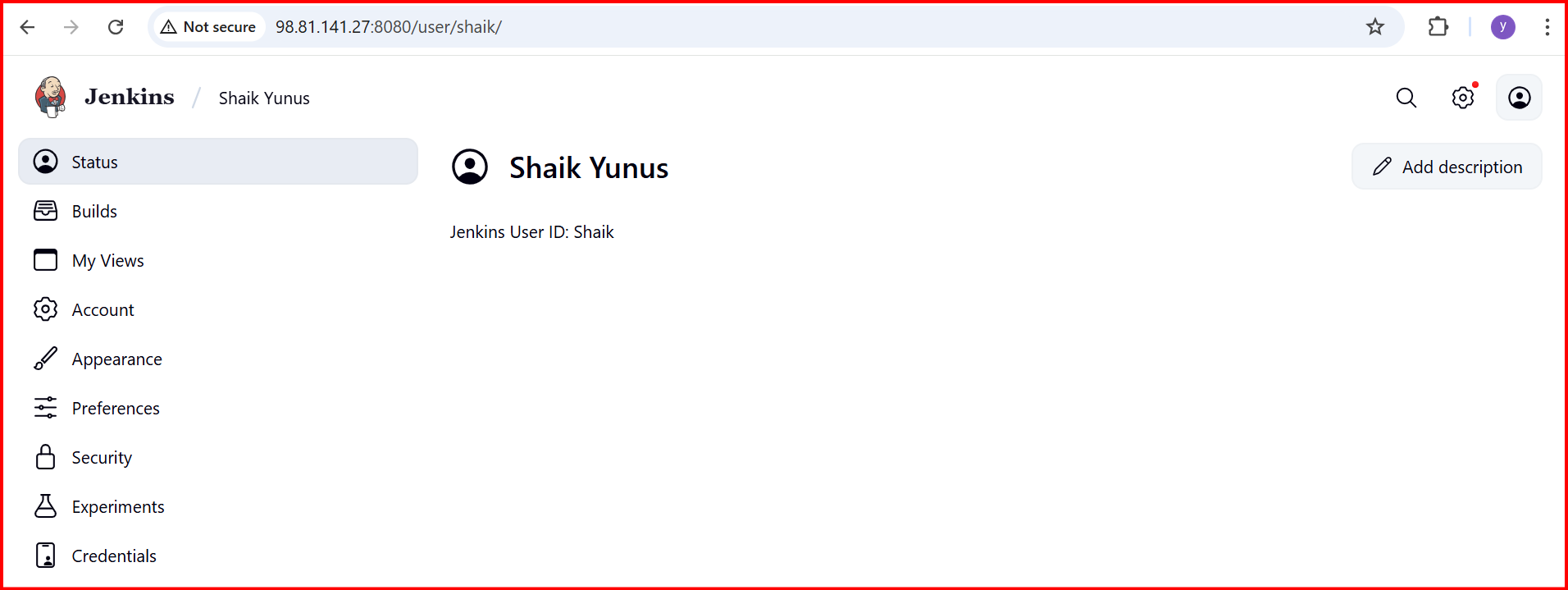




**8.Create a Jenkins pipeline to create a docker image and push the image to Docker hub.**

**Make sure you have:**

* Jenkins installed and running
* Docker installed and running
* Git installed
* Jenkins user added to Docker group:
* **sudo usermod -aG docker jenkins**
* **sudo systemctl restart jenkins**

****

**Jenkinsfile:**

**pipeline {**

**agent any**

**environment {**

**DOCKERHUB\_USER = 'aleemuddin138'**

**IMAGE\_NAME = 'app'**

**IMAGE\_TAG = 'latest'**

**}**

**stages {**

**stage('Checkout') {**

**steps {**

**git branch: 'master',**

**credentialsId: 'github',**

**url: 'https://github.com/ALEEMUDDIN138/multi-stage-example.git'**

**}**

**}**

**stage('Build JAR on Jenkins Host') {**

**steps {**

**echo "Building JAR on Jenkins host"**

**sh './mvnw clean package -DskipTests'**

**}**

**}**

**stage('Build Docker Image') {**

**steps {**

**echo "Building Docker image from runtime-only Dockerfile"**

**script {**

**docker.build("${DOCKERHUB\_USER}/${IMAGE\_NAME}:${IMAGE\_TAG}")**

**}**

**}**

**}**

**stage('Push to DockerHub') {**

**steps {**

**echo "Pushing Docker image to Docker Hub"**

**script {**

**docker.withRegistry('https://index.docker.io/v1/', 'dockerhub') {**

**docker.image("${DOCKERHUB\_USER}/${IMAGE\_NAME}:${IMAGE\_TAG}").push()**

**}**

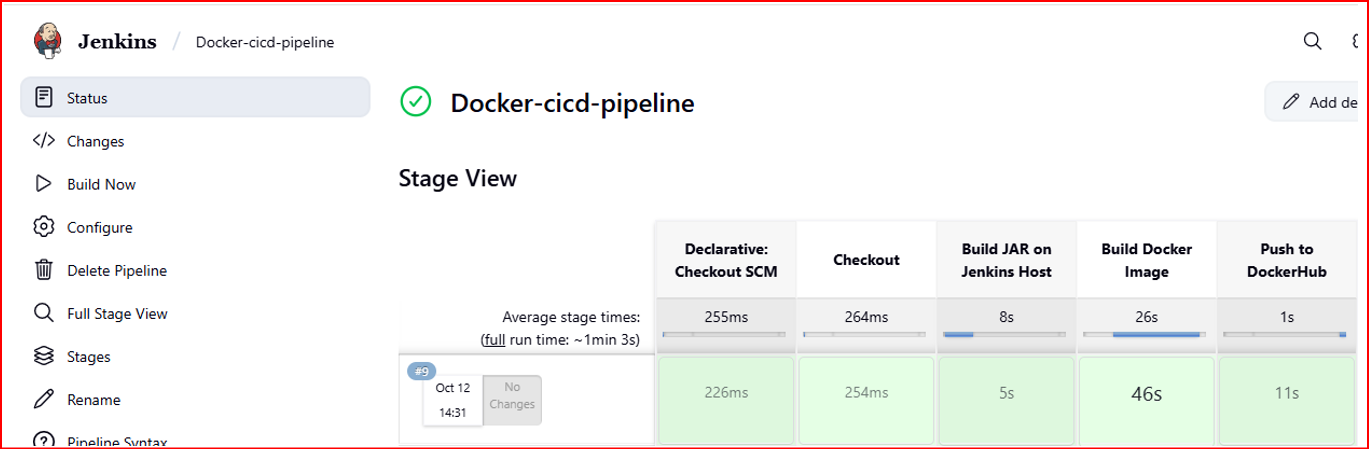
**}**

**}**

**}**

**}**

**}**

****