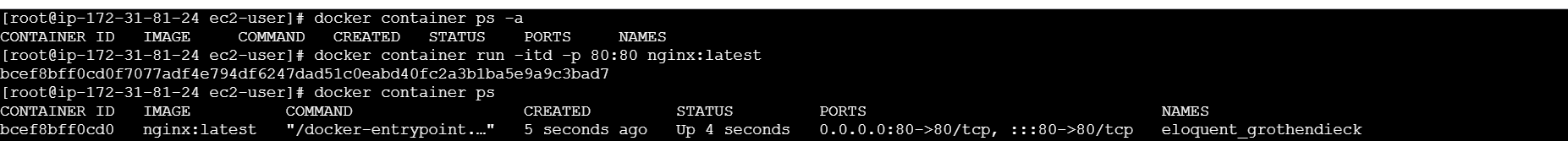
**Ansible Tasks -3:**

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

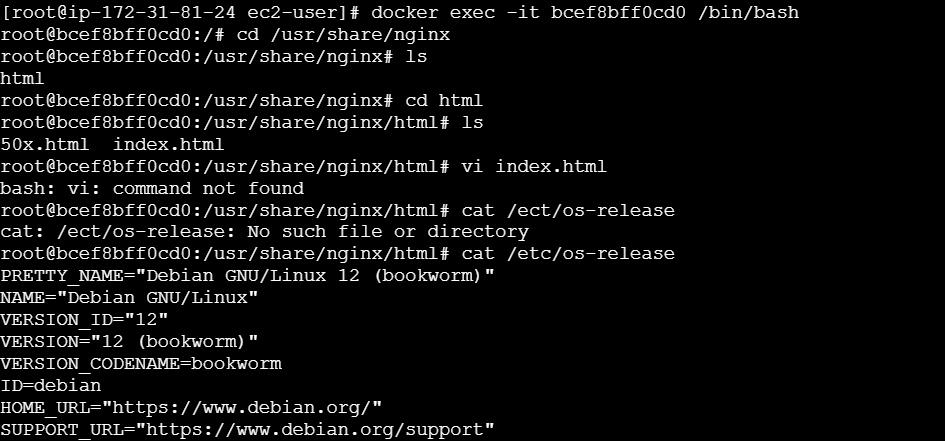
* **Create a container:**

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

****

* **Connect to the container.**

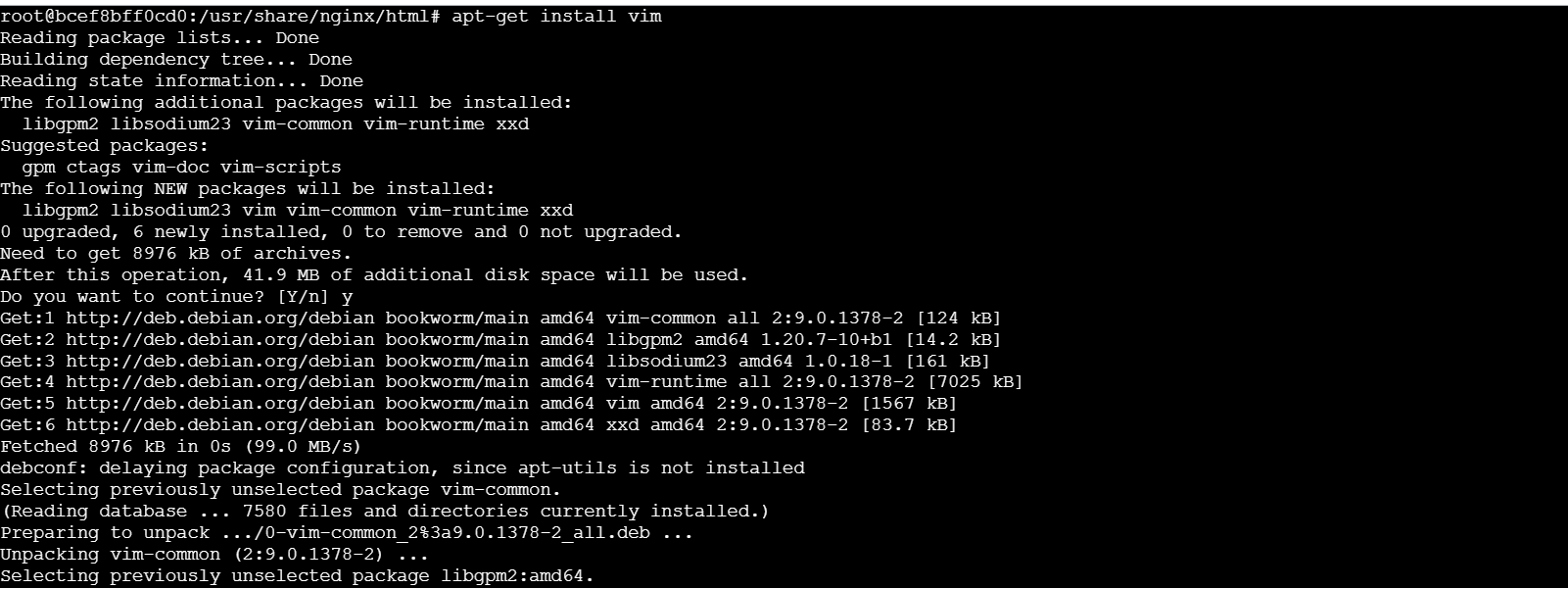
**>>> docker exec -it <container Id> /bin/bash**

****

* **Install Vim editor:**

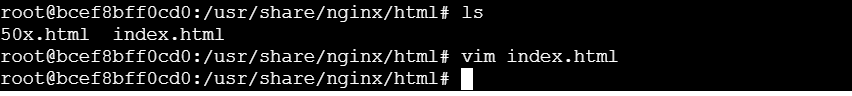
**>>> apt-get update [since it is an ubuntu os]**

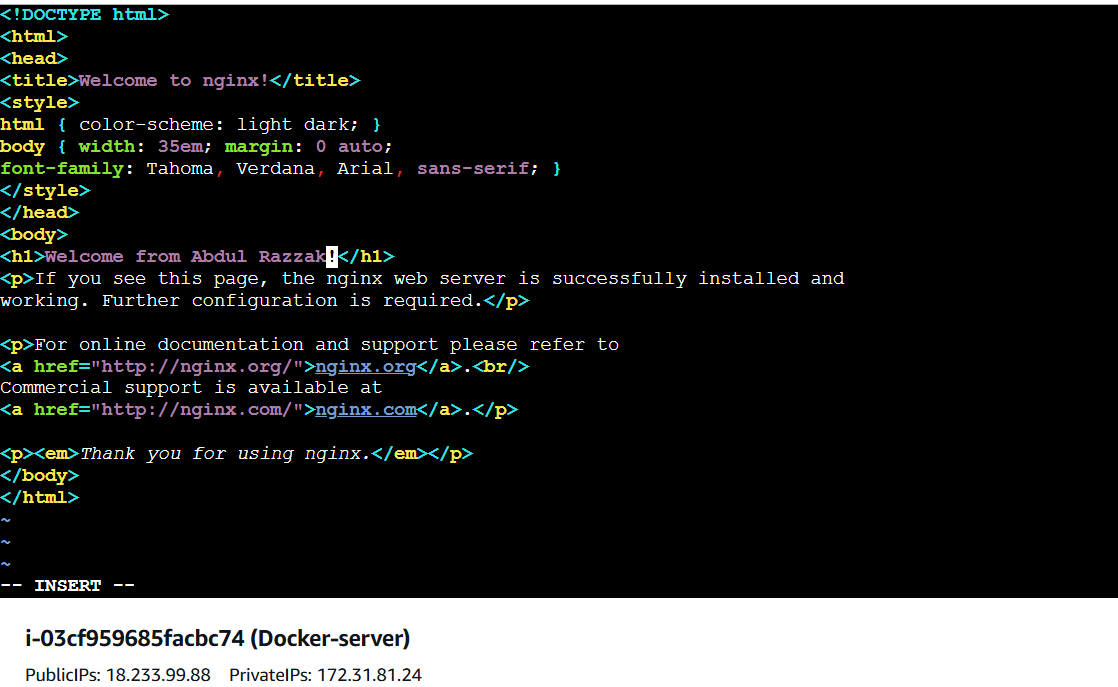
**>>> apt-get install vim**

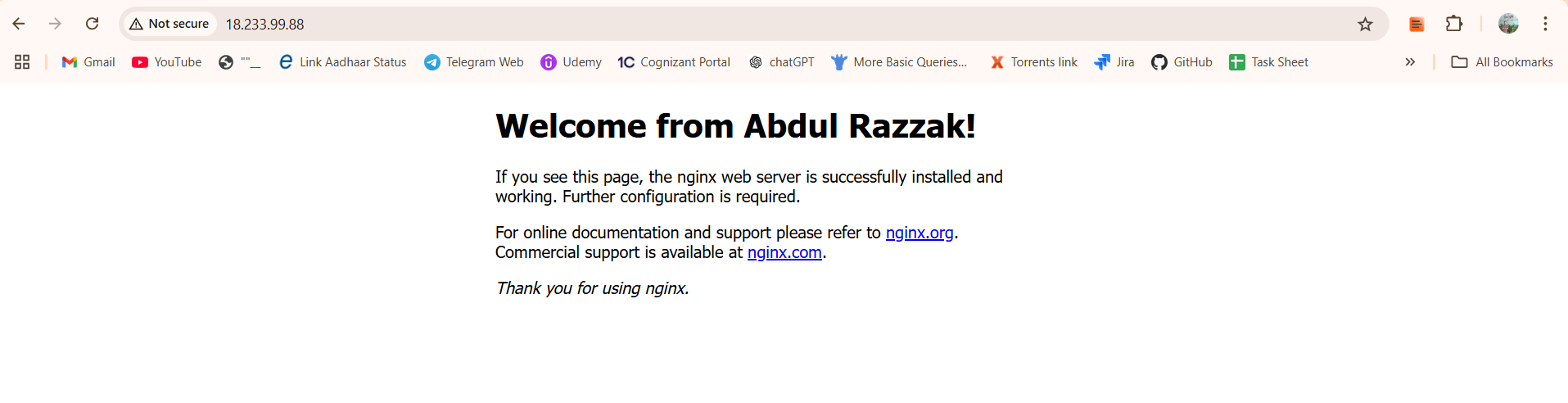


* **Edit the index.html file:**

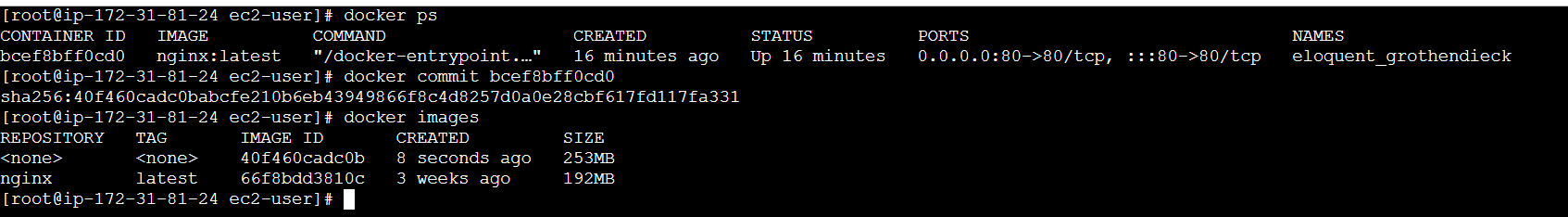
>>> vim index.html





* **Check on the browser:**
* **Create an image from this running container:**

**>>> docker commit <container Id>**

****

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

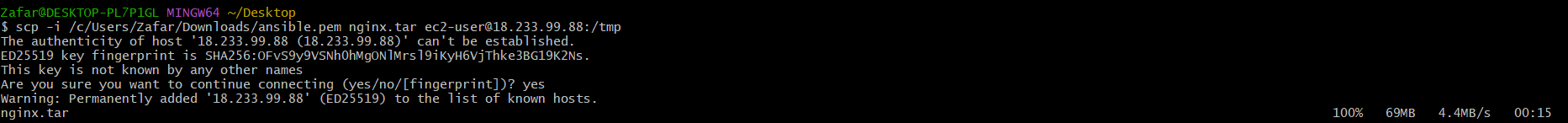
* **Pull an Image in the local machine and save it as a tar file:**

**>>> docker save -o nginx.tar nginx:latest**

****

* **Move the tar file to our Host server:**

**>>> scp -i /c/Users/Zafar/Downloads/ansible.pem nginx.tar ec2-user@18.233.99.88:/tmp**

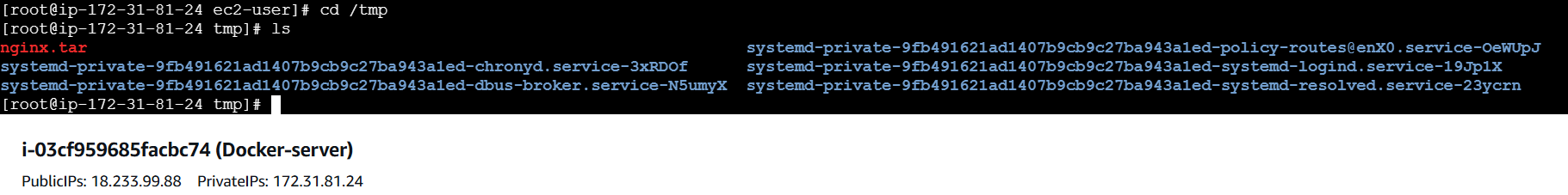
****

* **Go to our server.**

>> cd /tmp

>> ls

**Here our tar file is available.**

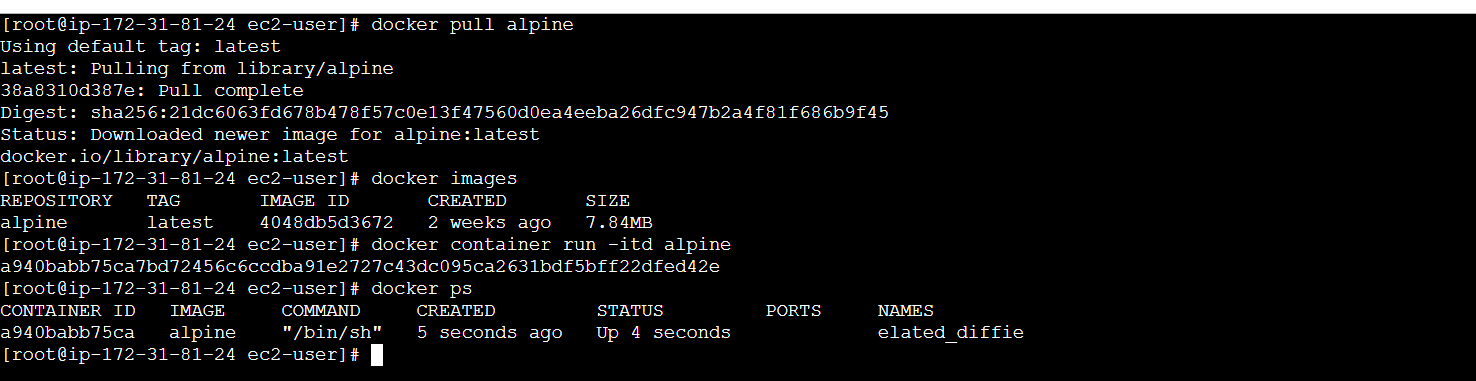
****

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

**>>> docker pull alpine**

* **Create a container using the alpine image:**

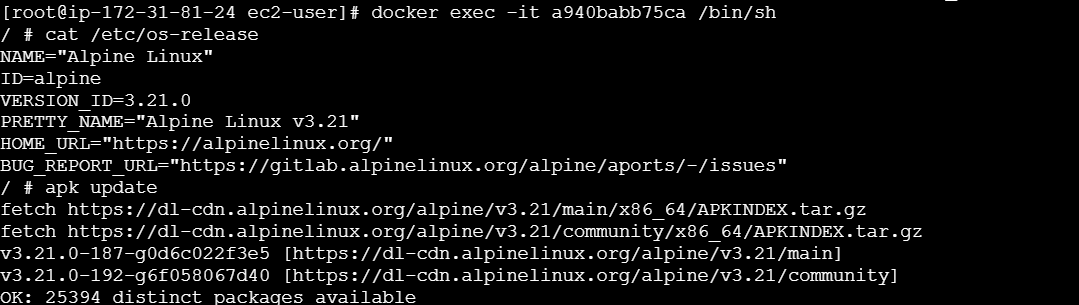
**>>> docker container run -itd alpine**

****

* **Connect to the alpine container:**

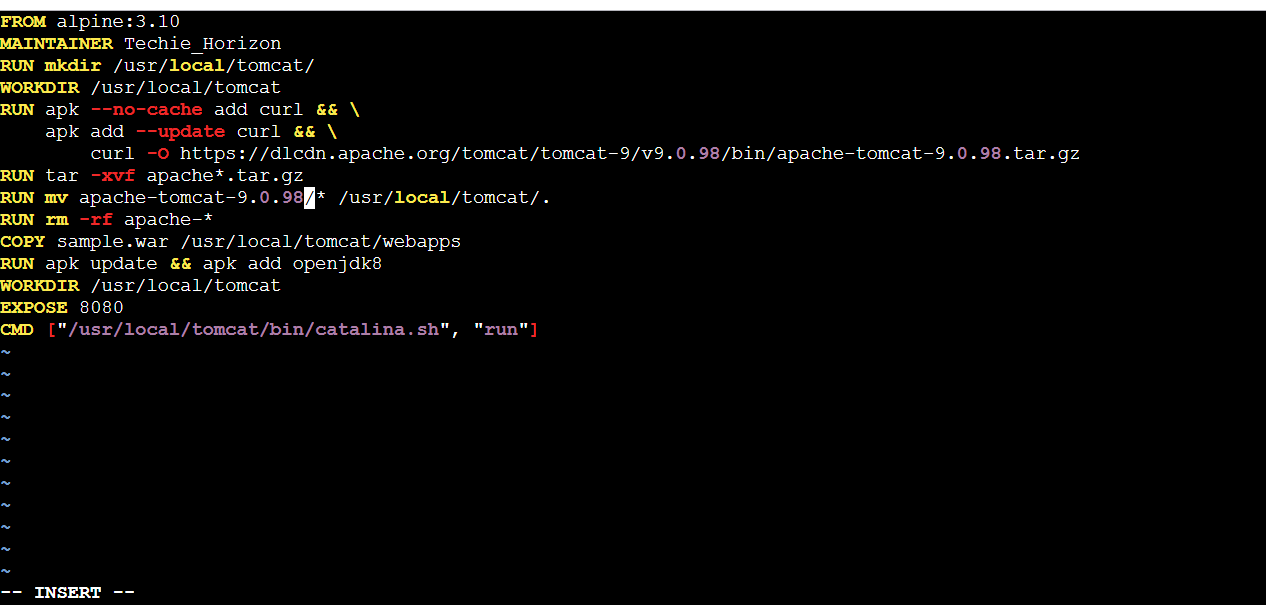
**>>> docker exec -it <cont. Id> /bin/sh**

**>>> apk update (yum update is not available)**

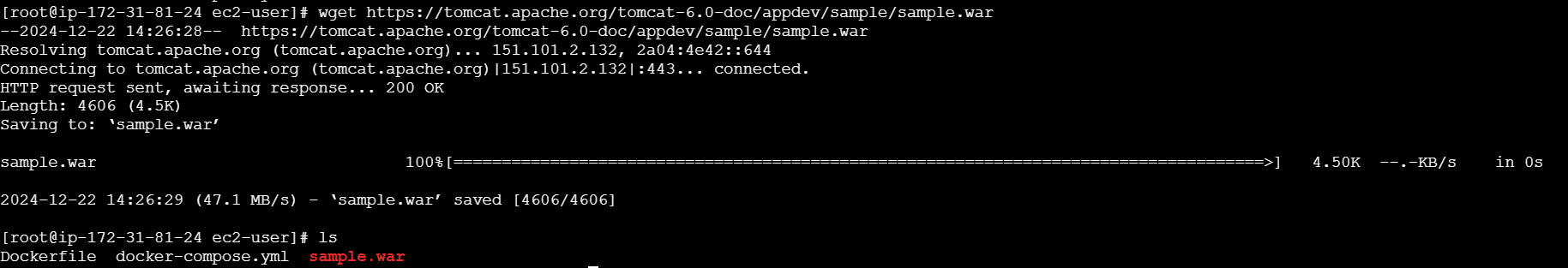
****

* **Create a Dockerfile for Tomcat:**

|  |
| --- |
| FROM alpine:3.10  MAINTAINER Techie\_Horizon  RUN mkdir /usr/local/tomcat/  WORKDIR /usr/local/tomcat  RUN apk --no-cache add curl && \  apk add --update curl && \  curl -O https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.98/bin/apache-tomcat-9.0.98.tar.gz  RUN tar -xvf apache\*.tar.gz  RUN mv apache-tomcat-9.0.98/\* /usr/local/tomcat/.  RUN rm -rf apache-\*  COPY sample.war /usr/local/tomcat/webapps  RUN apk update && apk add openjdk8  WORKDIR /usr/local/tomcat  EXPOSE 8080  CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]  ~ |

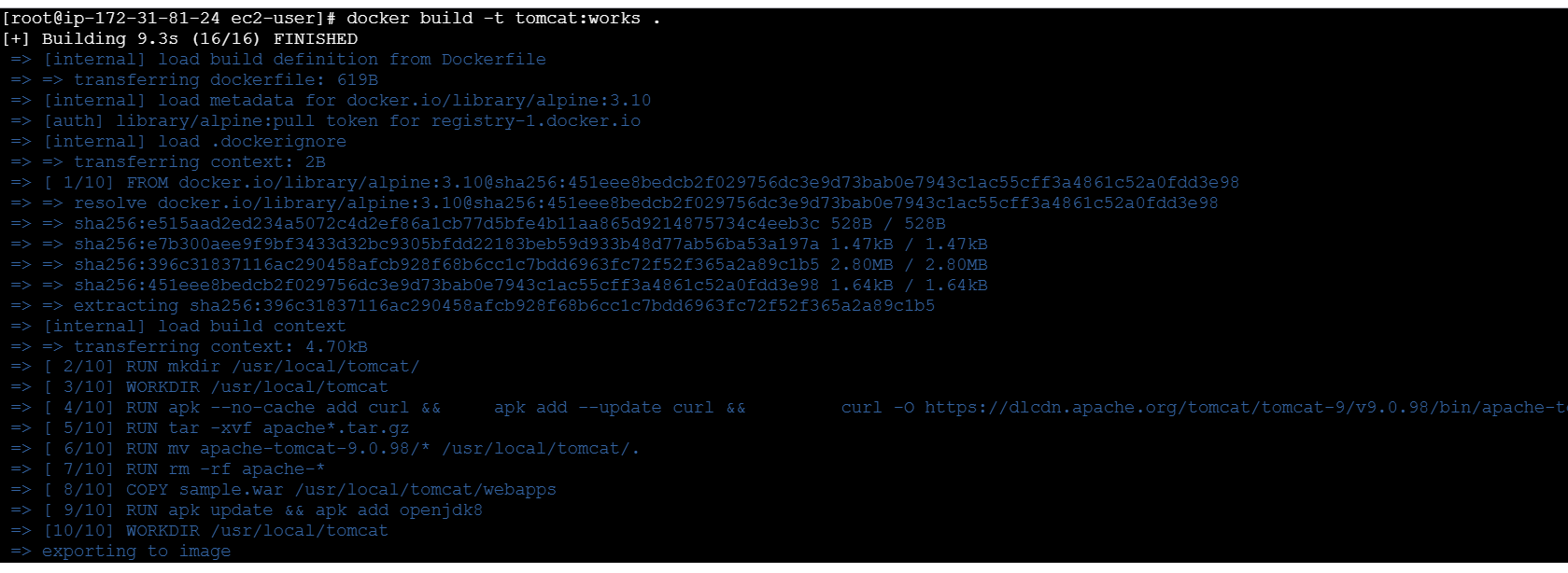
****

* **Download Sample.war file:**

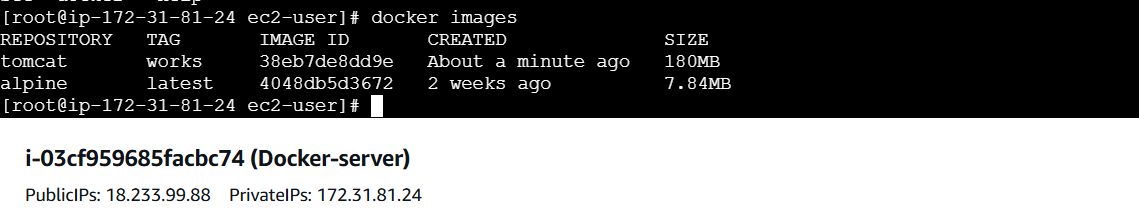
****

* **Create an image out of our Dockerfile:**

**>>> docker build -t tomcat:works .**

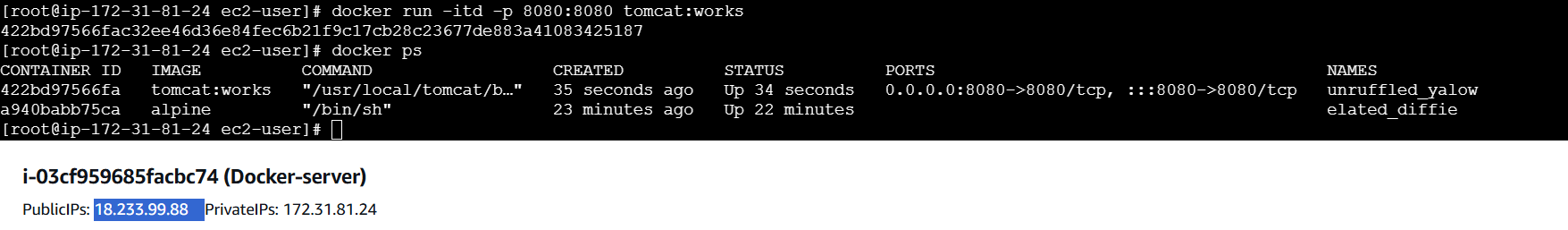
****

* **Our image is created:**

****

* **Create a container out of our image:**

**>>> docker run -itd -p 8080:8080 tomcat:works**

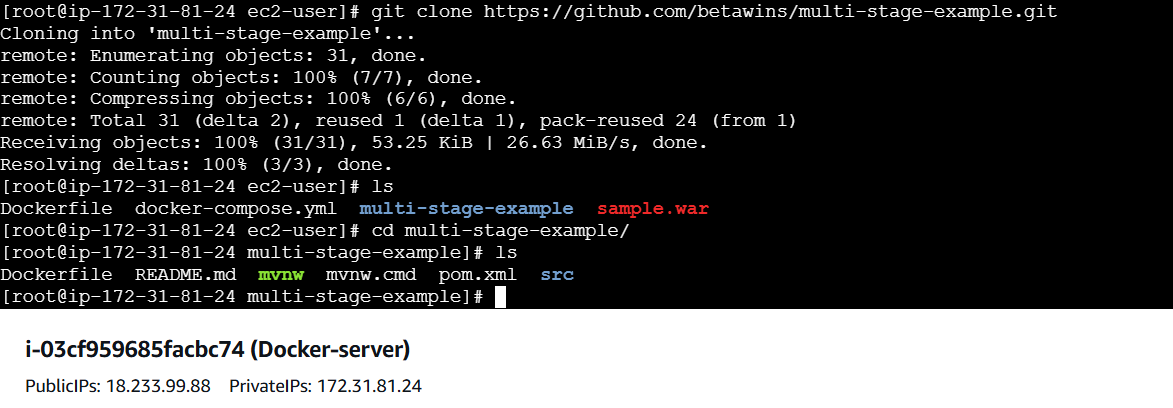
****

* **Check on the browser:**



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

* Clone the repo



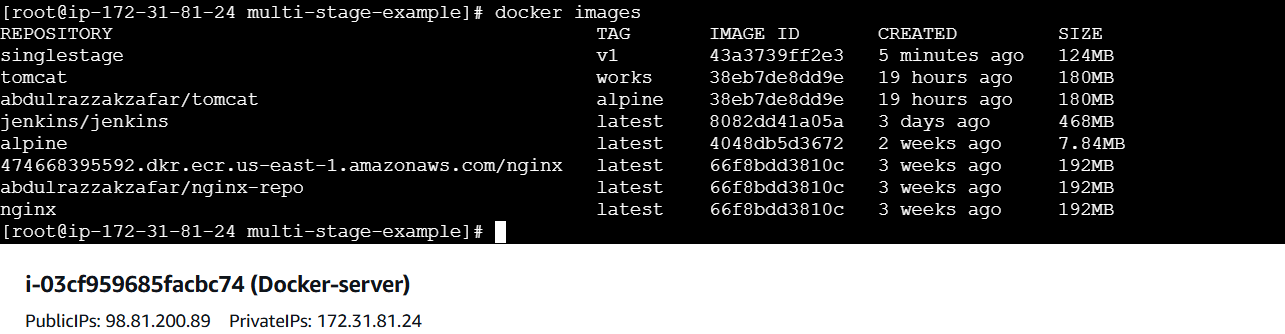
* **Create the Dockerfile (Single Stage):**

|  |
| --- |
| #Build Image FROM openjdk:17-jdk-alpine as builder RUN mkdir -p /app/source COPY . /app/source WORKDIR /app/source RUN ./mvnw clean package#Run image FROM openjdk:8-jdk-alpine WORKDIR /app COPY --from=builder /app/source/target/\*.jar /app/app.jar EXPOSE 8080 ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom", "-jar", "/app/app.jar"] |

* Create a docker image out of it:

>>> docker build -t singlestage:v1 **.** 

>>> docker images



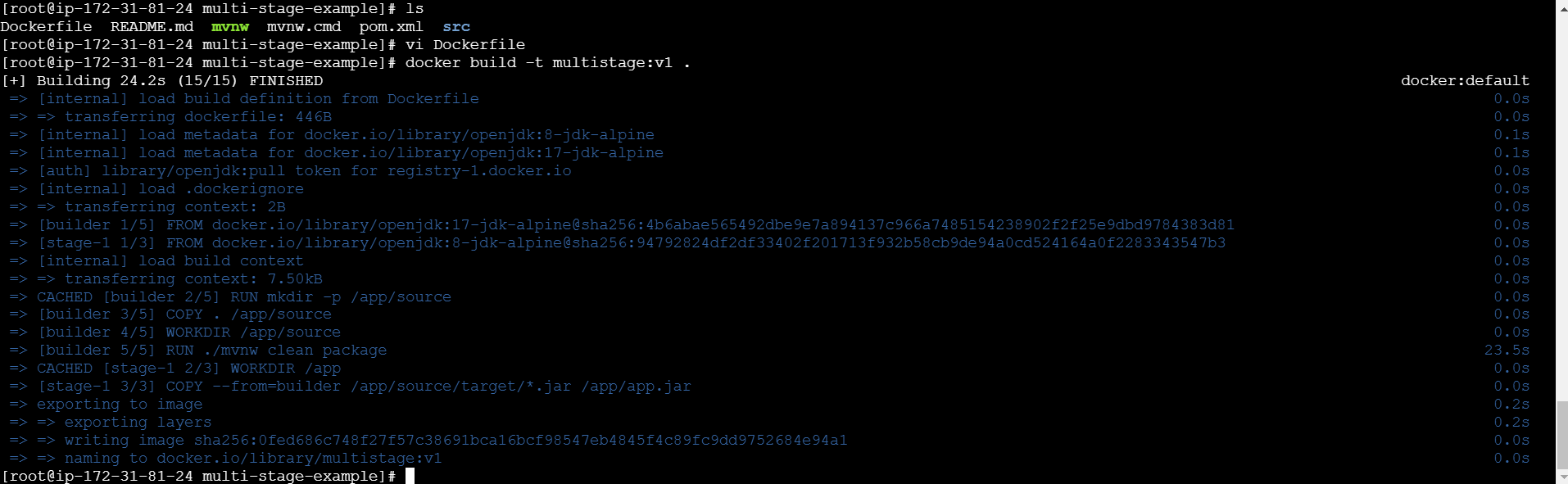
* **Create the Dockerfile (Multi Stage):**

|  |
| --- |
| #Build Image  FROM openjdk:8-jdk-alpine as builder  RUN mkdir -p /app/source  COPY . /app/source  WORKDIR /app/source  RUN ./mvnw clean package  #Run image  FROM openjdk:8-jdk-alpine  WORKDIR /app  COPY --from=builder /app/source/target/\*.jar /app/app.jar  EXPOSE 8080  ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom", "-jar", "/app/app.jar"] |

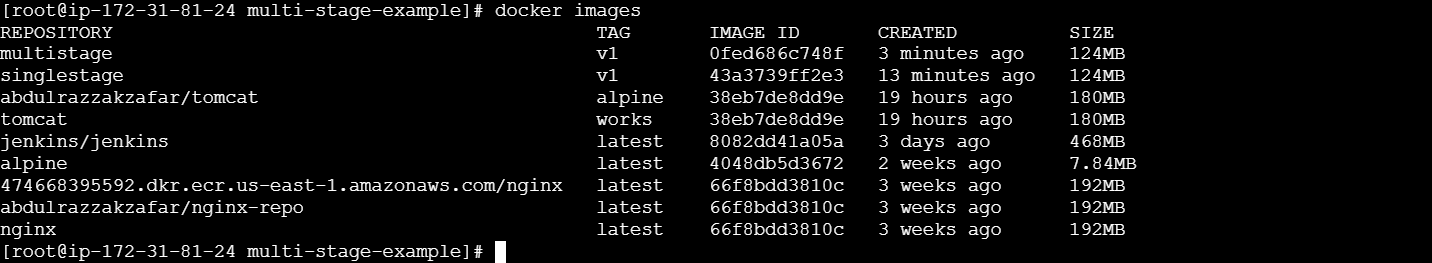


* **Create a docker image out of it:**

>>> docker build -t multistage:v1 **.**

****

**>>> docker images:**

****

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

* **To install docker-compose, run this command:**

**>>>** sudo curl -L https://github.com/docker/compose/releases/download/1.21.0/docker-compose-`uname -s`-`uname -m` | sudo tee /usr/local/bin/docker-compose > /dev/null

* **For permission:**

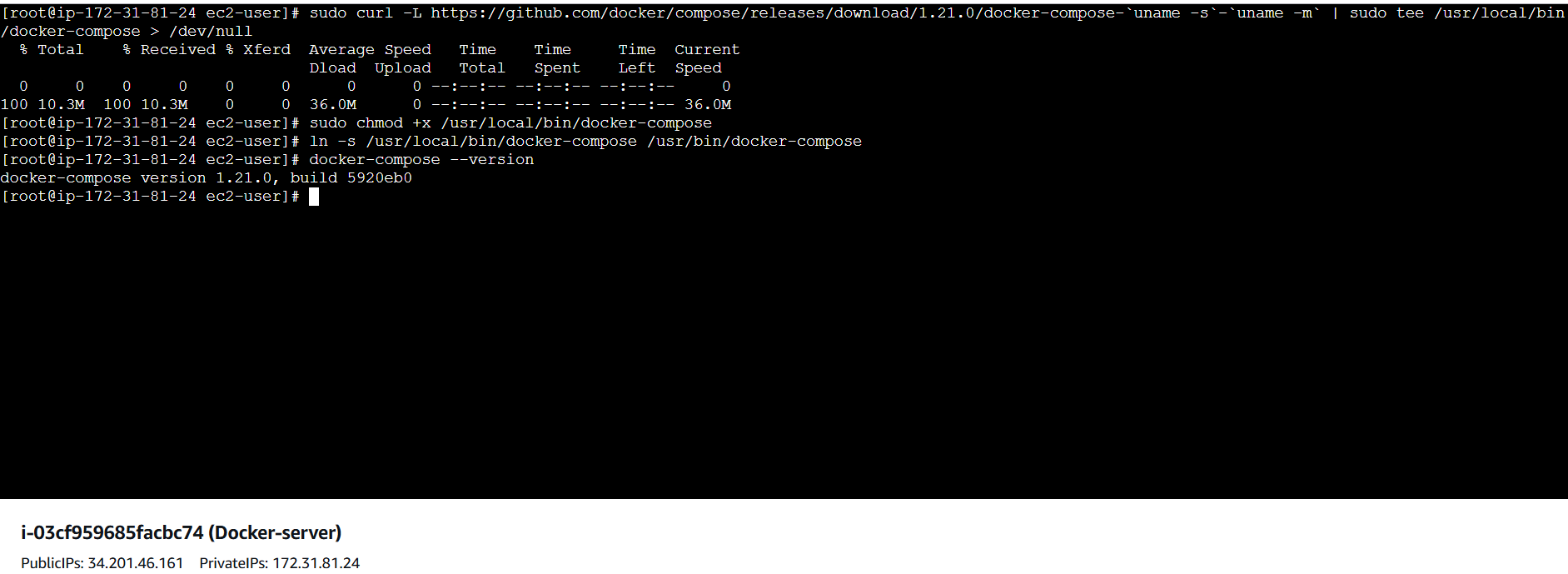
**>>>** sudo chmod +x /usr/local/bin/docker-compose

* **Create a symbolic link:**

**>>>** ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

* **Check docker-compose version**:

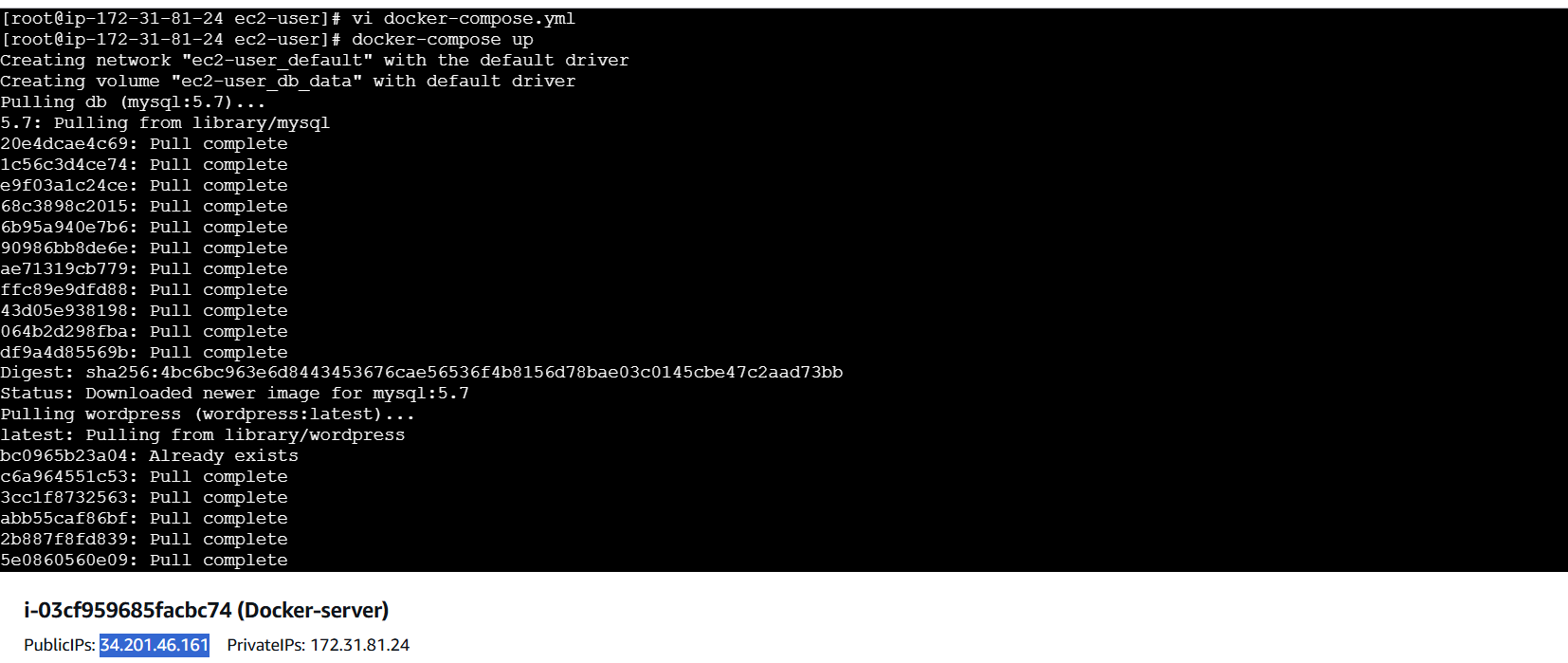
**>>>** docker-compose --version

****

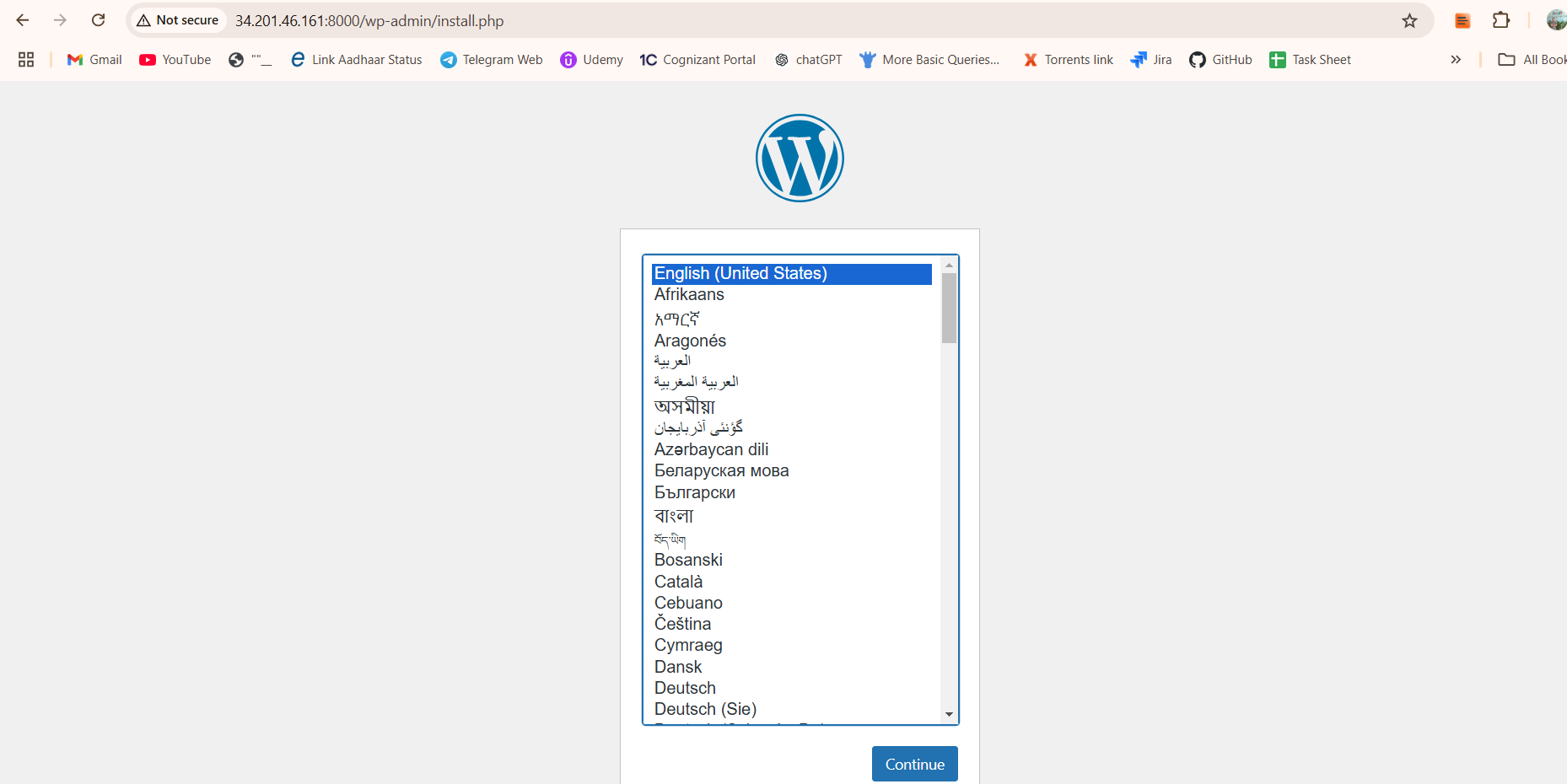
|  |
| --- |
| sample-two-tier:docker-compose file  ========================  version: '3'  services:  db:  image: mysql:5.7  volumes:  - db\_data:/var/lib/mysql  restart: always  environment:  - MYSQL\_ROOT\_PASSWORD=somewordpress  - MYSQL\_DATABASE=wordpress  - MYSQL\_USER=wordpress  - MYSQL\_PASSWORD=wordpress  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: { } |

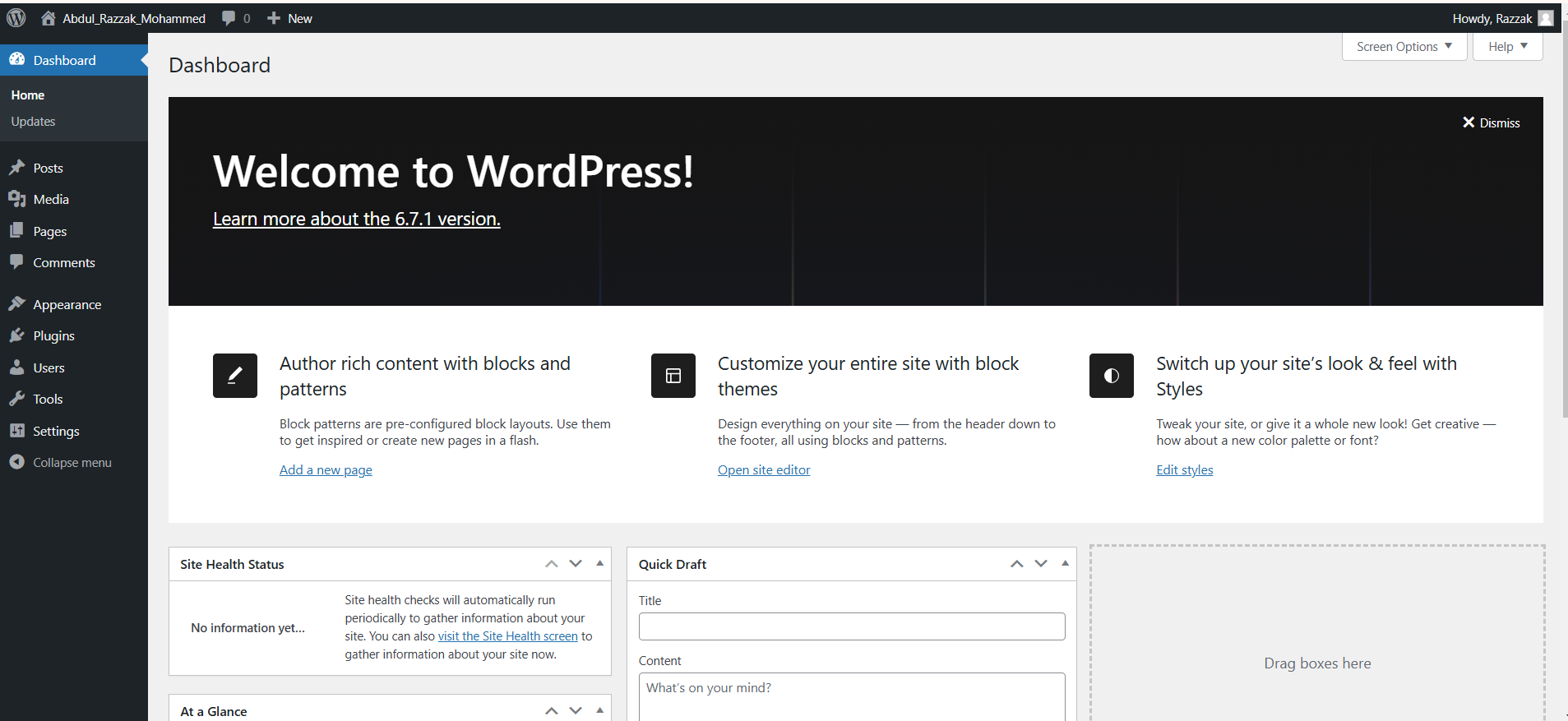
* **Run the docker-compose.yml file**

**>>>** docker-compose up

****

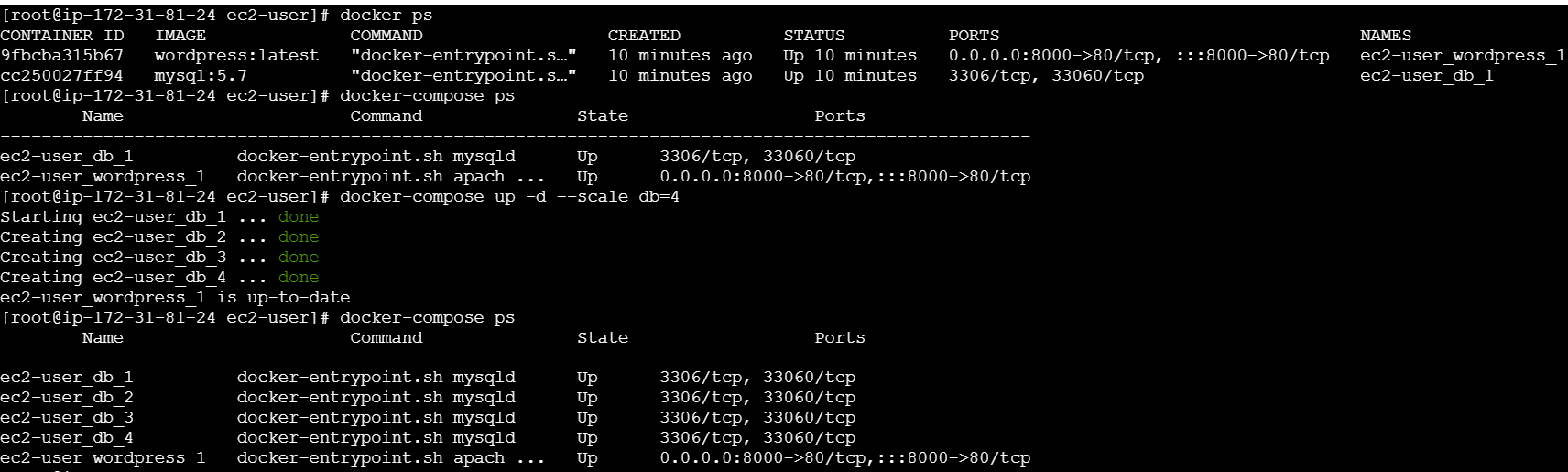
* **Copy paste the ip address along with the port number: 8000 on the browser:**

****

****

* **We can scale up the containers as per our requirement:**

**>>> docker-compose up -d –scale db=4**

****

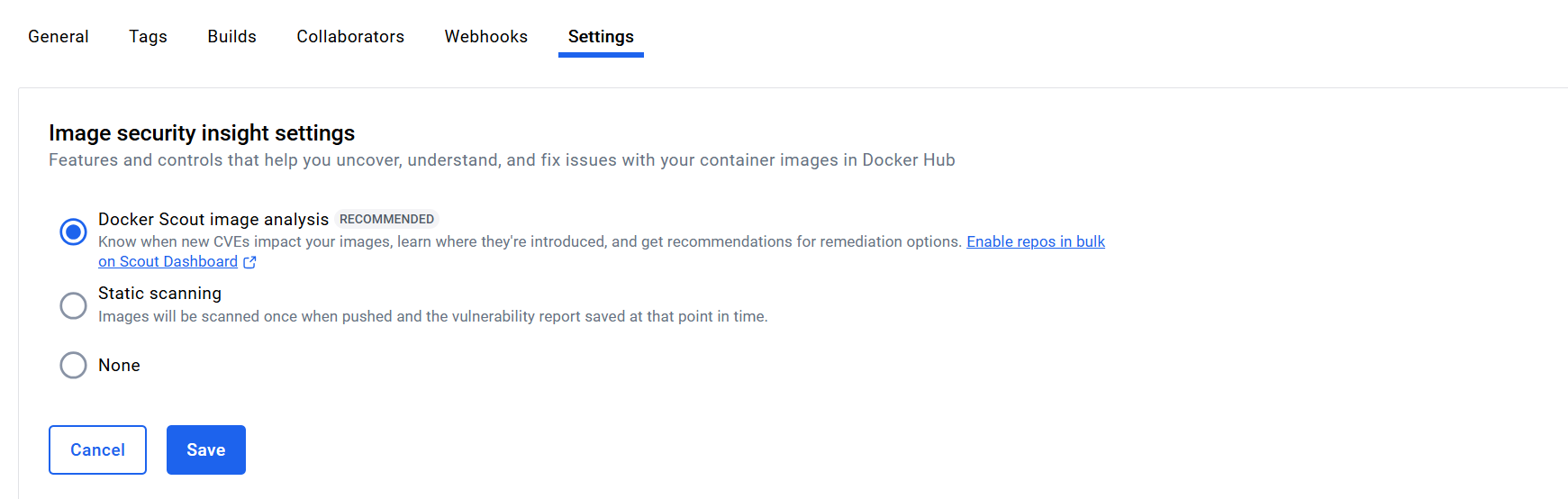
* **Shut down all the containers using command:**

**>>> docker-compose down**

****

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

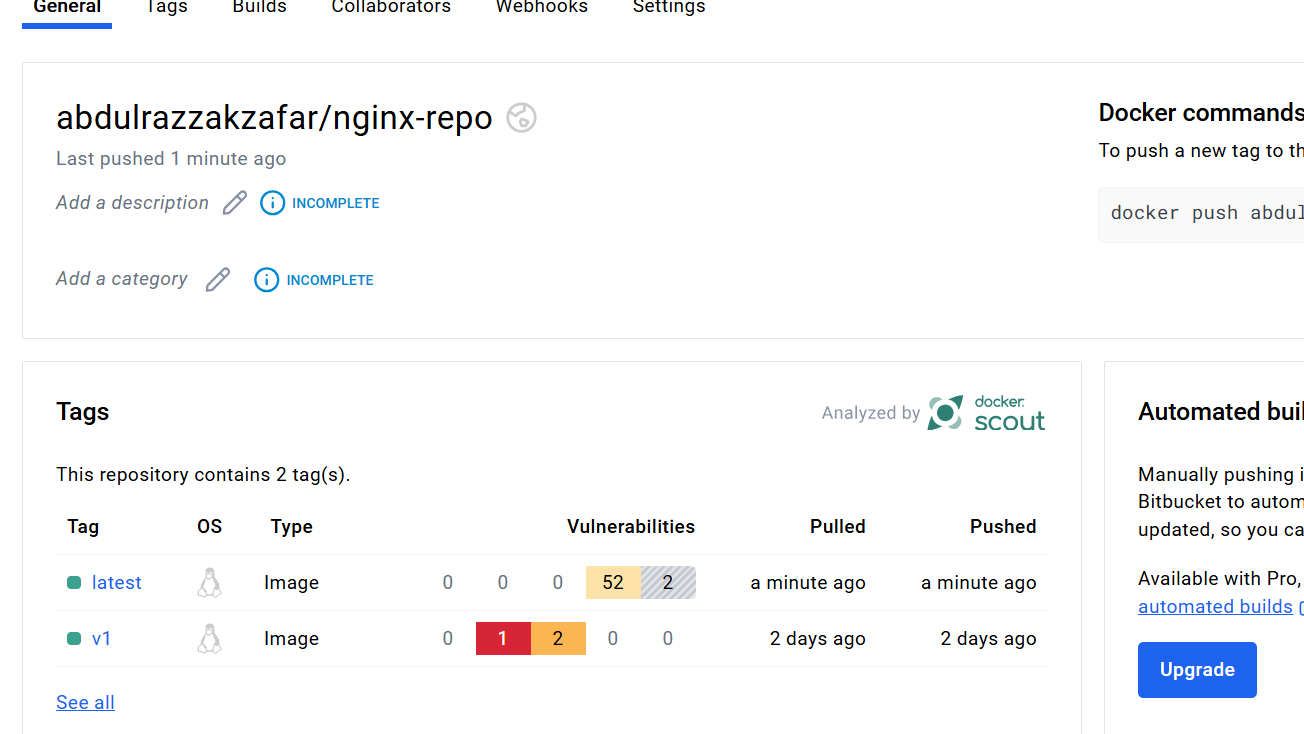
* **Go to DockerHub --- Sign-in --- repository --- settings ---- Select “Docker Scout Images analysis”**

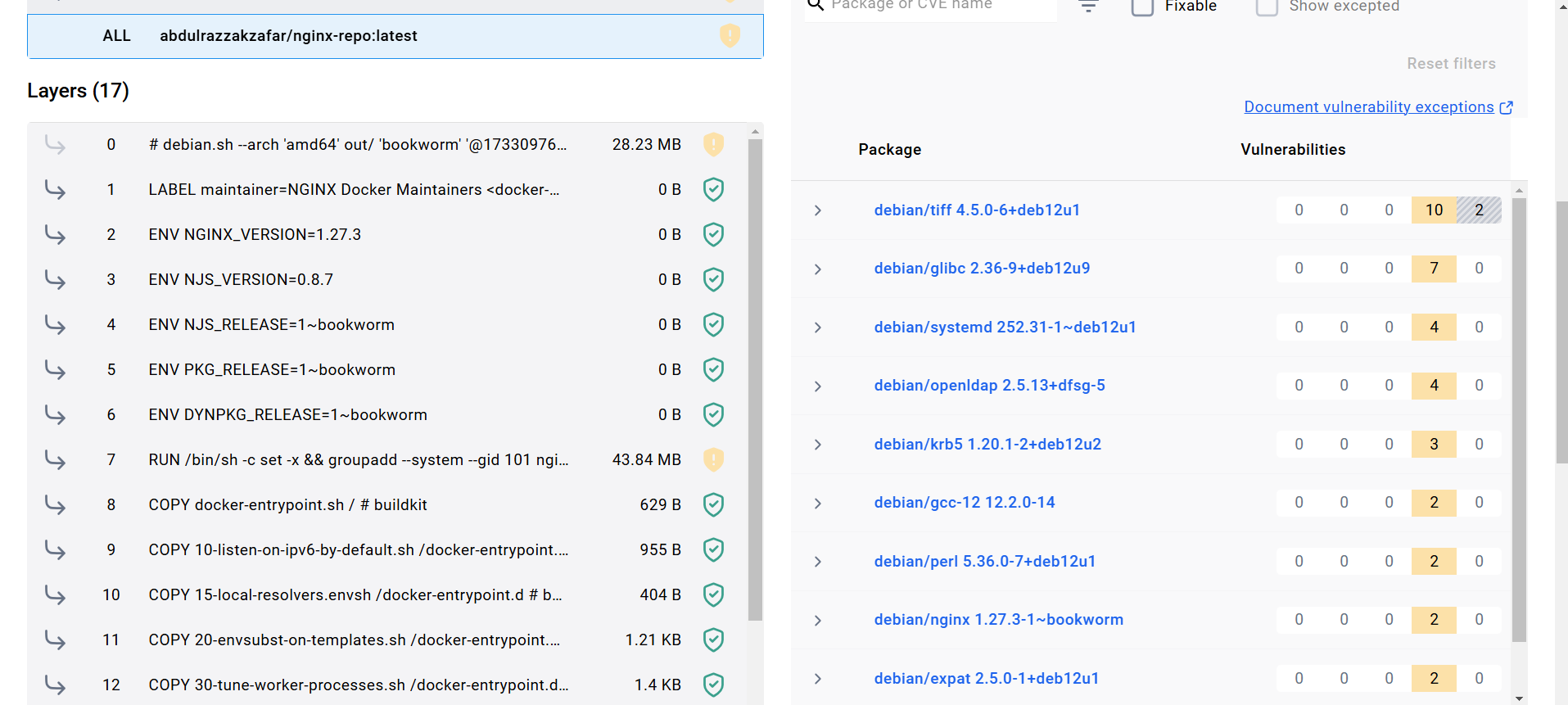
****

* **Push one image to the DockerHub**

****

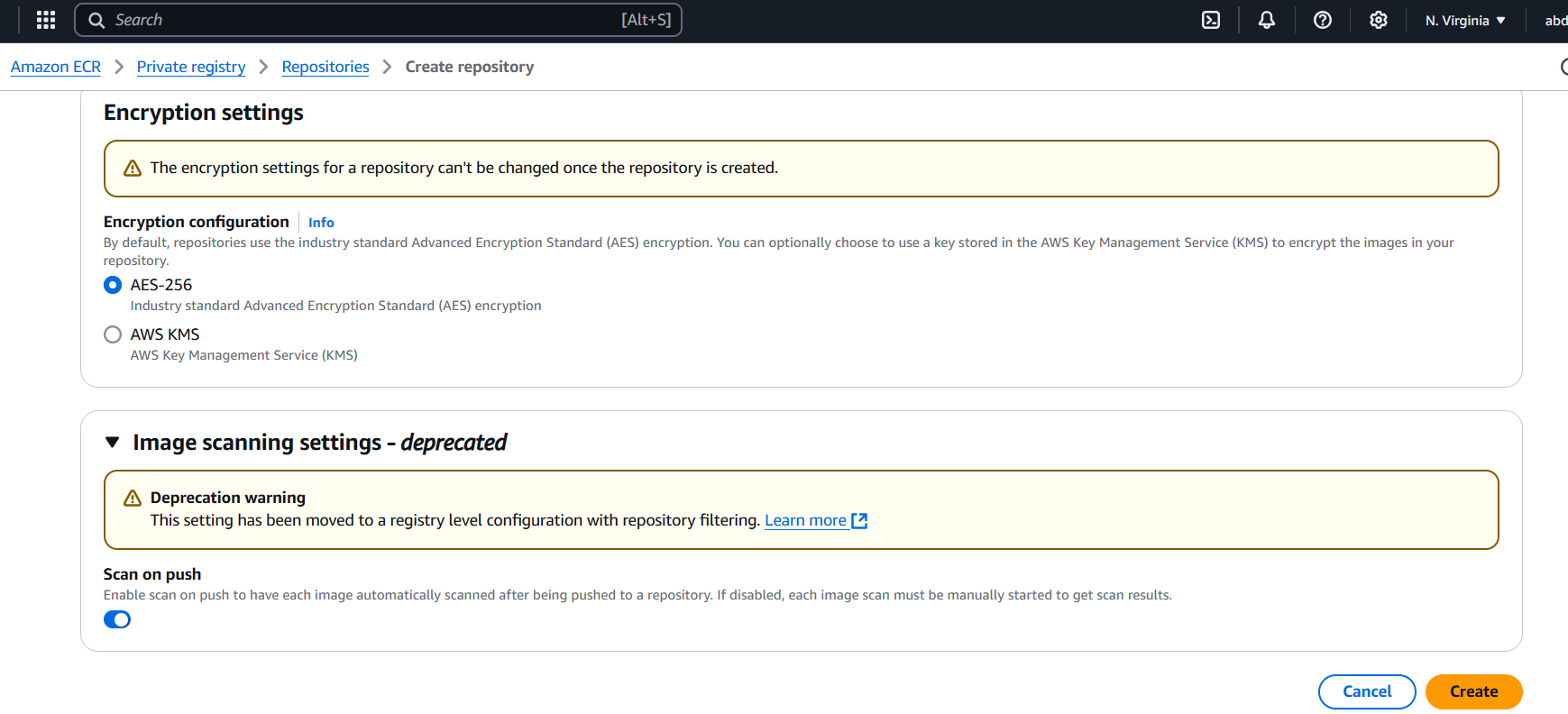
* **Scout analysis is active:**

****

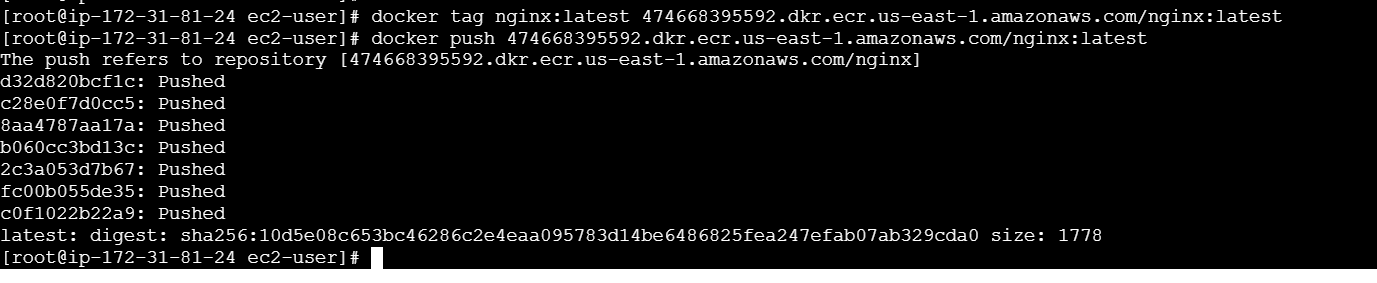
****

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

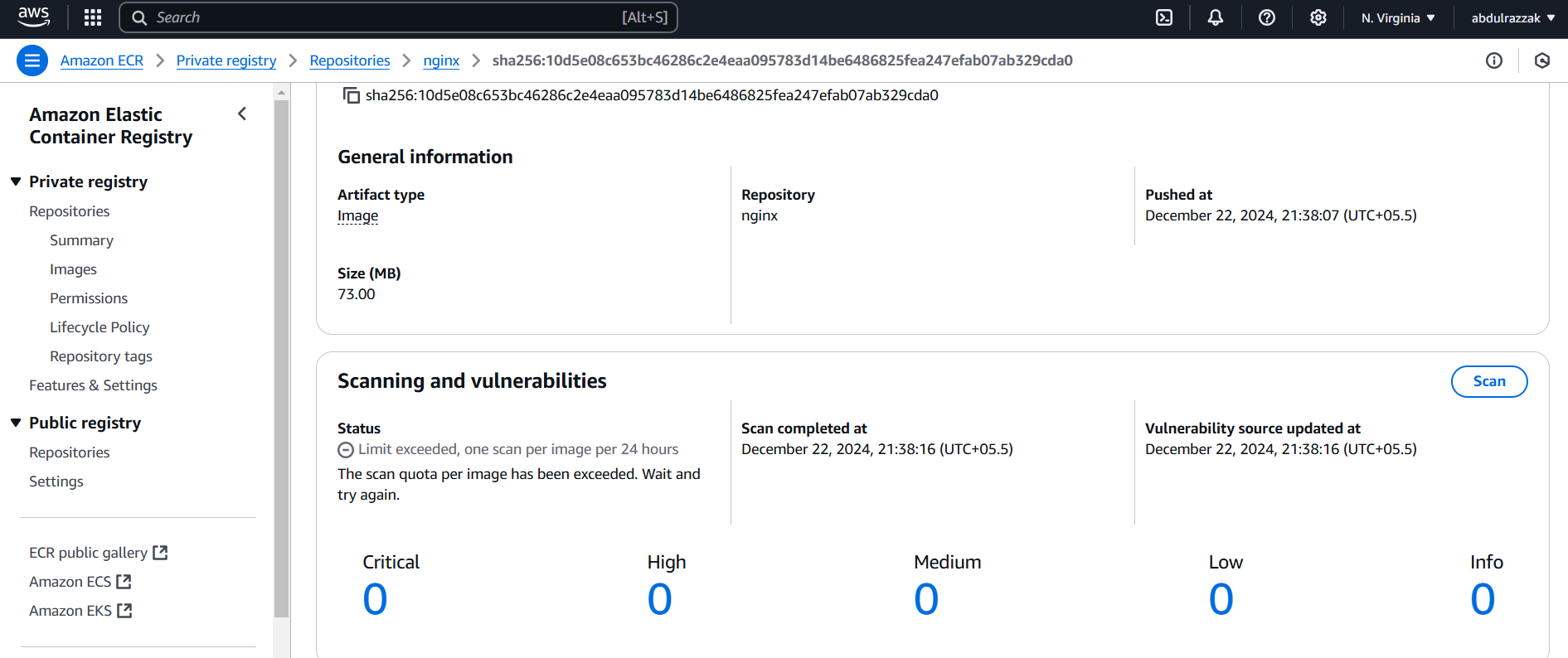
* **Go to ecr enable scan on push feature:**

****

* **Push the image to ecr:**

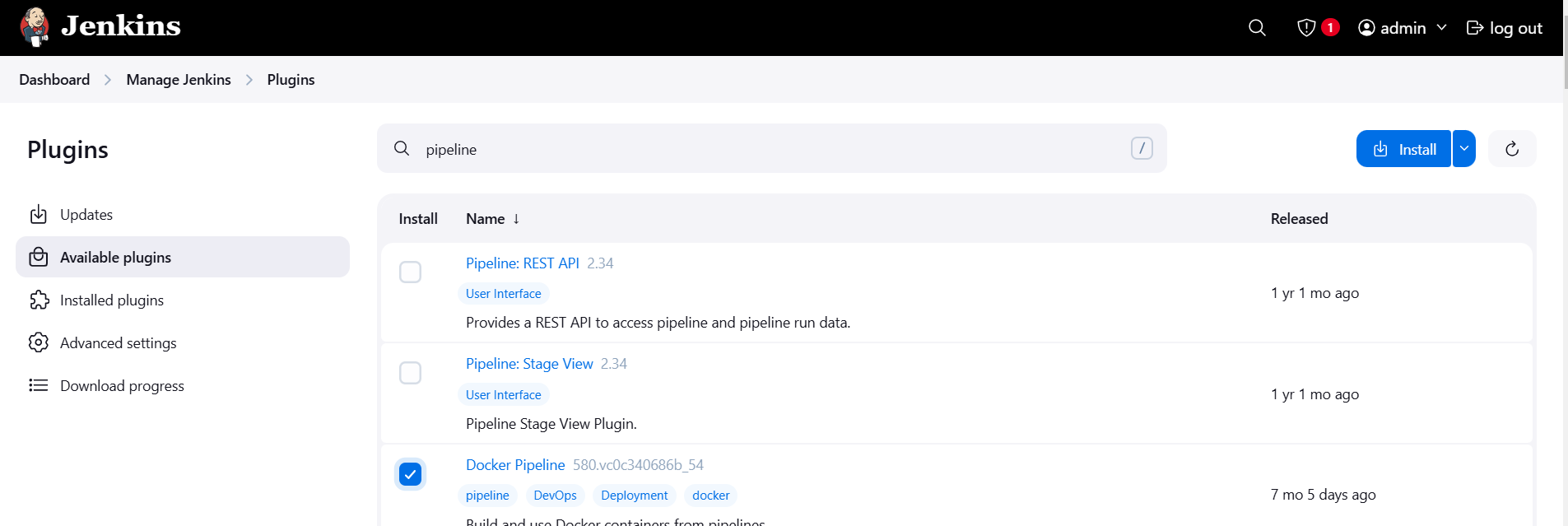
****

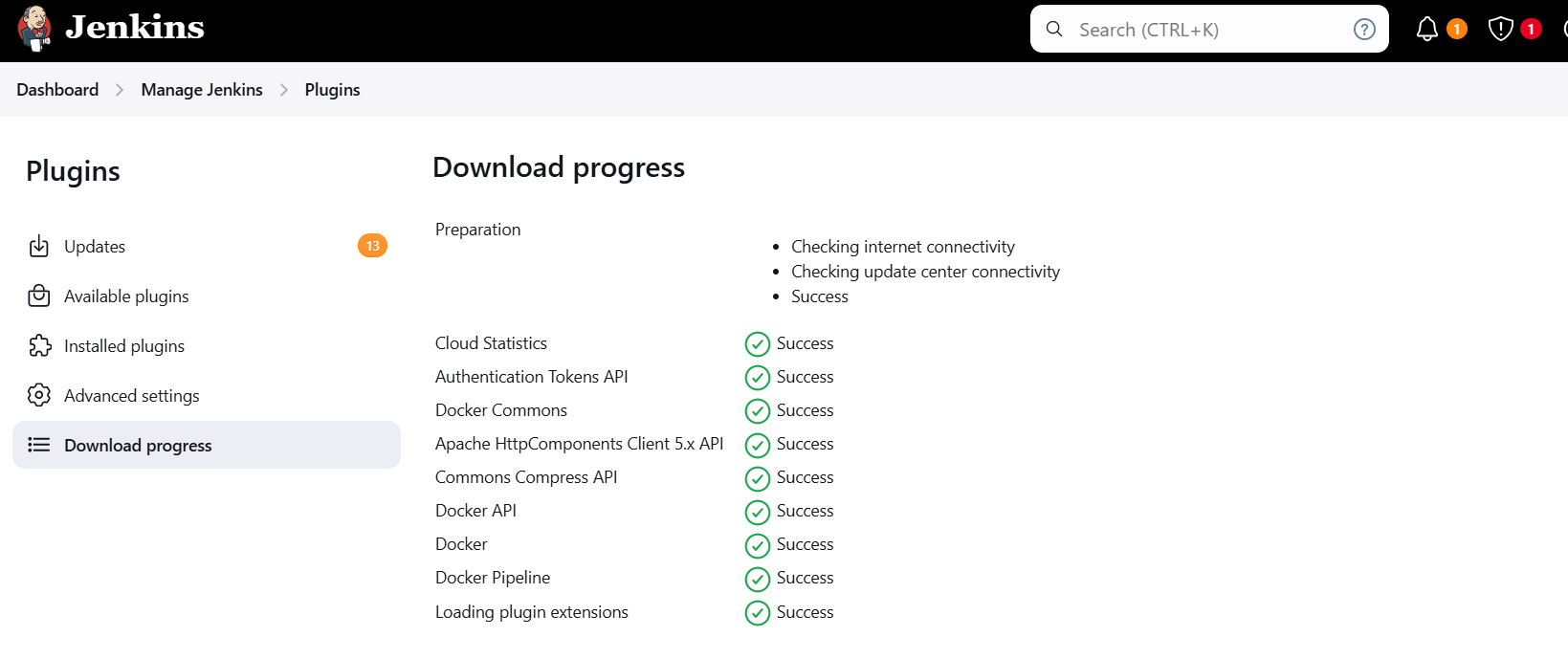
* **Scan completed:**

****

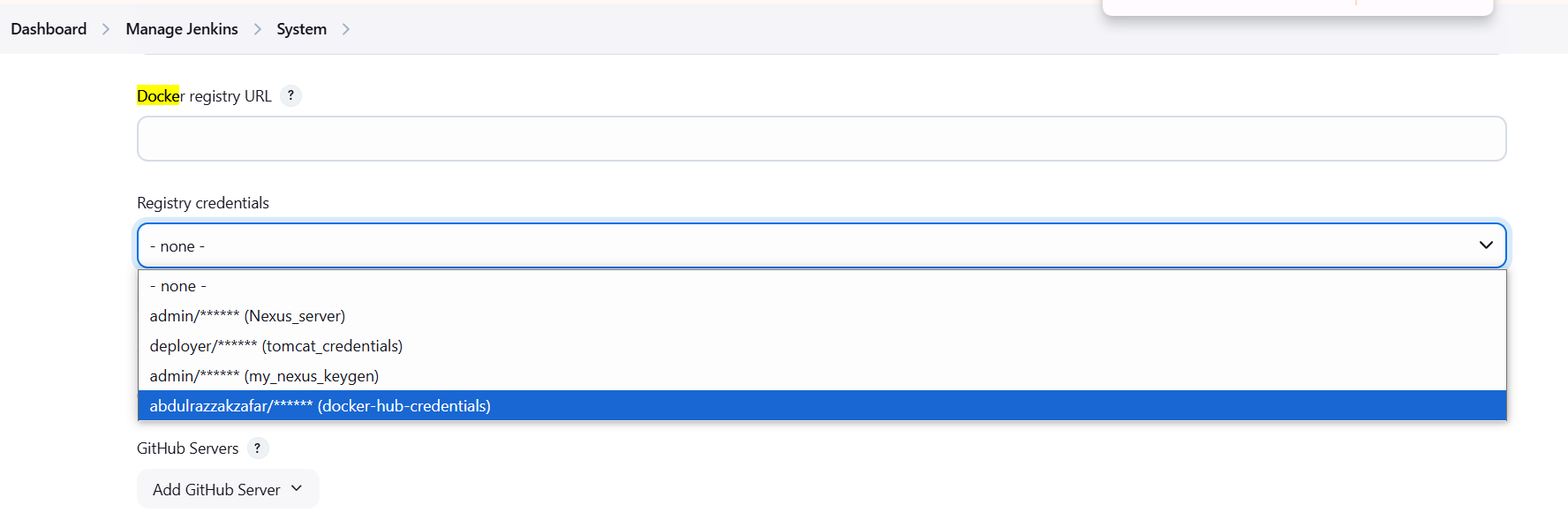
**8) Create a jenkins pipeline to create a docker image and push the image to dockerhub.**

* **Installing the Docker and Docker pipeline plugin:**

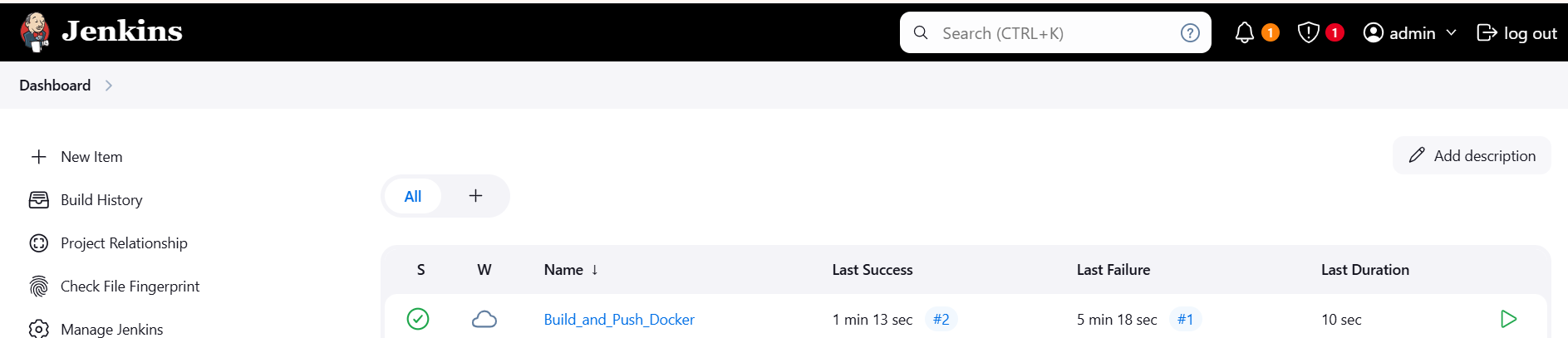
****

****

* **Adding dockerHub credentials[username & password]:**

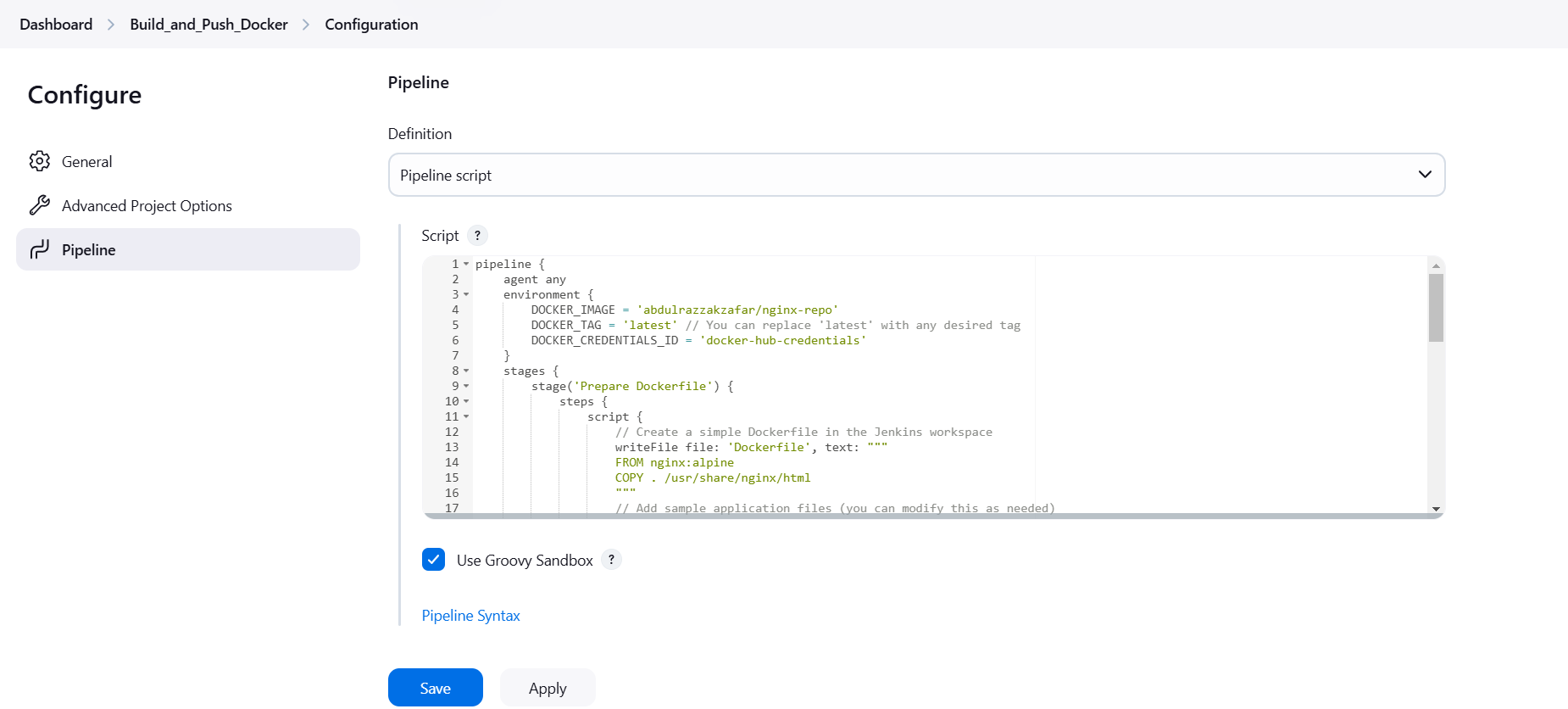
****

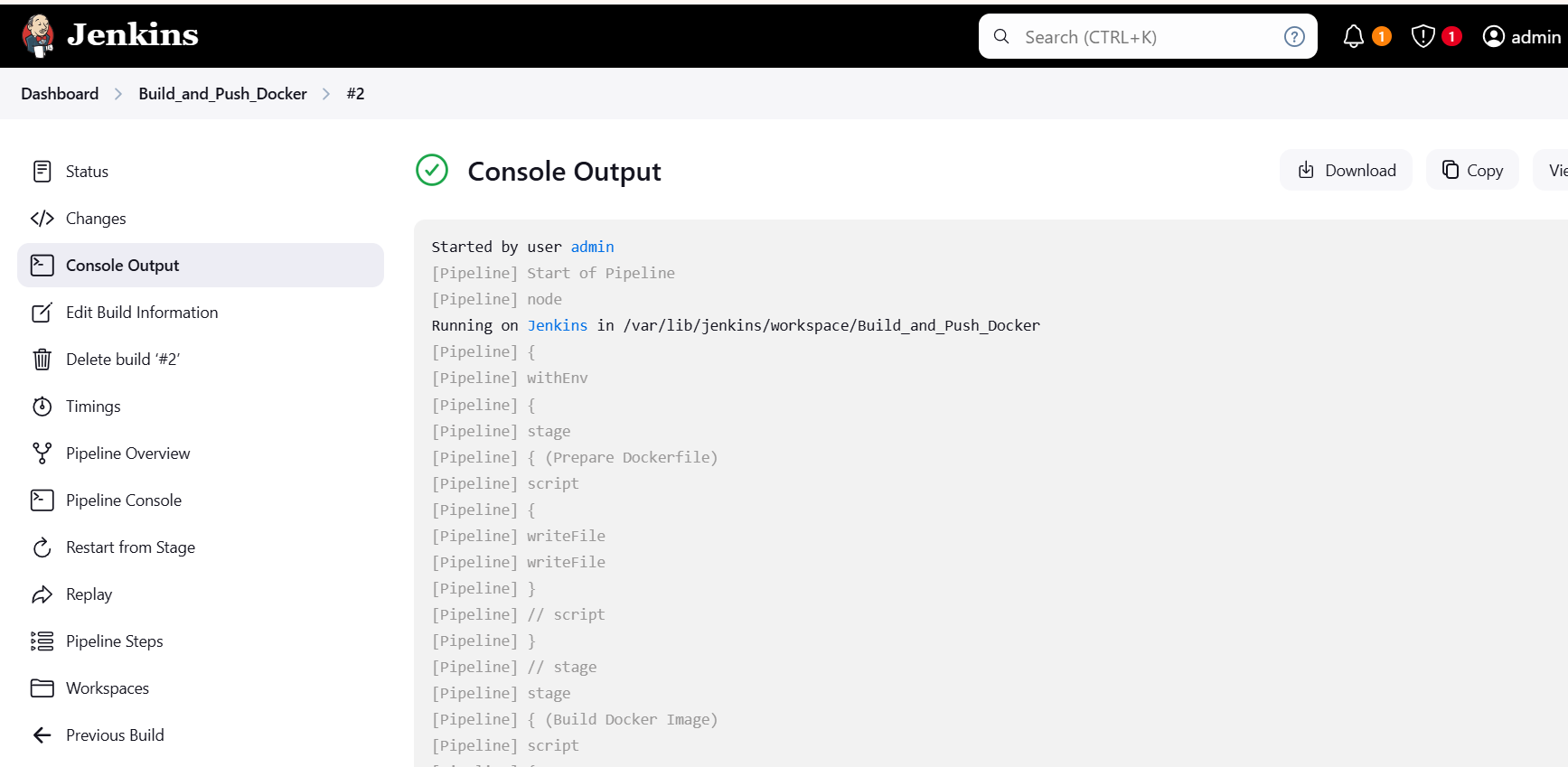
* **Select “New Item” and create a job:**

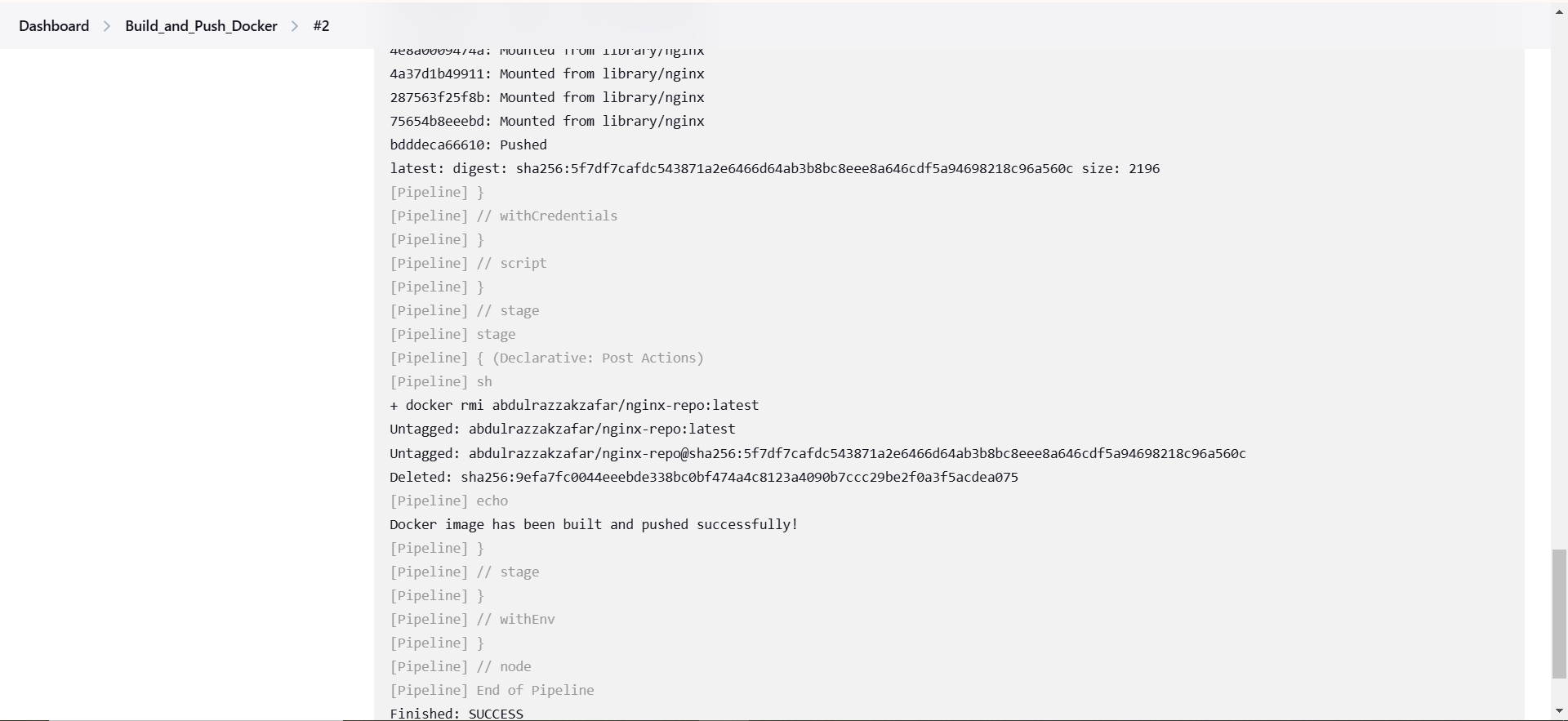
****

* **Script to create a docker image and push the image to dockerhub.**

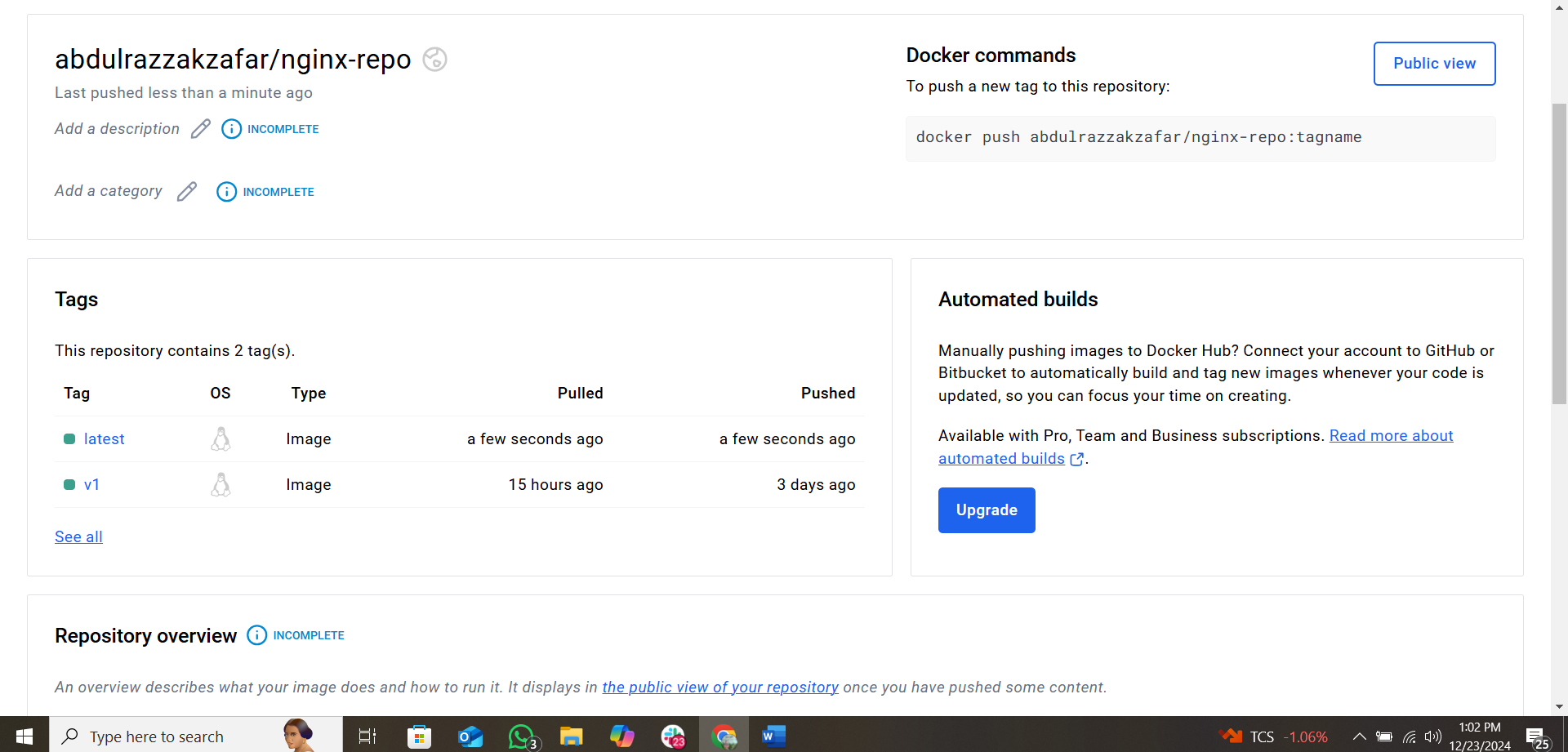
|  |
| --- |
| pipeline {  agent any  environment {  DOCKER\_IMAGE = 'abdulrazzakzafar/nginx-repo'  DOCKER\_TAG = 'latest' // You can replace 'latest' with any desired tag  DOCKER\_CREDENTIALS\_ID = 'docker-hub-credentials'  }  stages {  stage('Prepare Dockerfile') {  steps {  script {  // Create a simple Dockerfile in the Jenkins workspace  writeFile file: 'Dockerfile', text: """  FROM nginx:alpine  COPY . /usr/share/nginx/html  """  // Add sample application files (you can modify this as needed)  writeFile file: 'index.html', text: '<h1>Welcome to My App</h1>'  }  }  }  stage('Build Docker Image') {  steps {  script {  // Build the Docker image with the specified tag  sh "docker build -t ${DOCKER\_IMAGE}:${DOCKER\_TAG} ."  }  }  }  stage('Push Docker Image to Docker Hub') {  steps {  script {  // Log in to Docker Hub and push the image  withCredentials([usernamePassword(credentialsId: "${DOCKER\_CREDENTIALS\_ID}",  usernameVariable: 'DOCKER\_USERNAME', passwordVariable: 'DOCKER\_PASSWORD')]) {  sh '''  echo $DOCKER\_PASSWORD | docker login -u $DOCKER\_USERNAME --password-stdin  docker push ${DOCKER\_IMAGE}:${DOCKER\_TAG}  '''  }  }  }  }  }  post {  always {  // Cleanup local Docker resources  sh "docker rmi ${DOCKER\_IMAGE}:${DOCKER\_TAG} || true"  }  success {  echo 'Docker image has been built and pushed successfully!'  }  failure {  echo 'Pipeline execution failed.'  }  }  } |

****

****

****

* **Check the DockerHub repository. The new image created and pushed to the repo successfully!**

****