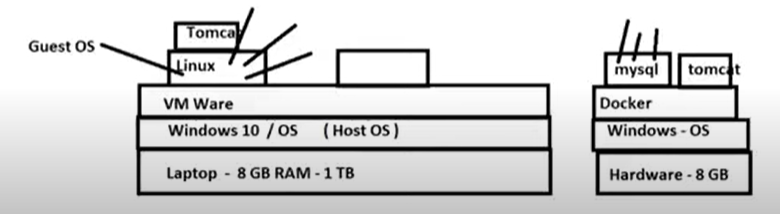
Docker : is a containerization tool



Docker Image: combination of binaries or libraries which are necessary for software application

Docker Container: when image is installed and comes into running condition is called container

Image-->run-->container

Docker Host: Machine on which docker is installed called as docker host

Docker Client: Terminal which is used to run docker commands (gitbash)

Docker commands:

Working on images:

Download image--> docker pull imagename

To see list of docker images -->docker images

To delete a docker image form docker host ->docker rmi imagename/imageid

To upload a docker image into docker hub-->docker push imagename

To build an image from customised container

Docker commit containername newimagename

To create an image from docker file

Docker build -t newimagename

Search for a docker image-->docker search imagename

Delete all images which are not attached to any container

Docker system prune -a

Working on containers:

To see list of all running containers

Docker container ls

To see list of running and stopped containers

Docker ps -a

To start a container

Docker start containername/id

To stop a container

Docker stop contatinername/id

To restart a conatiner

Docker restart containername/id

To delete stopped container

Docker rm conatinername/id

To delete a running container

Docker rm -f conatinername/id

To stop all running containers

Docker stop $(docker ps -aq)

To restart all containers

Docker restart $(docker ps -aq)

to remove all stopped containers

Docker rm $(docker ps -aq)

To remove all containers

docker rm $(docker ps -aq)

To execute any command in a container

Docker ex -it conatainername/id

Run command options:

It-->opening interactive terminal

--name--> used for giving name to container

-d -->detached mode run background

p--> used for port mapping

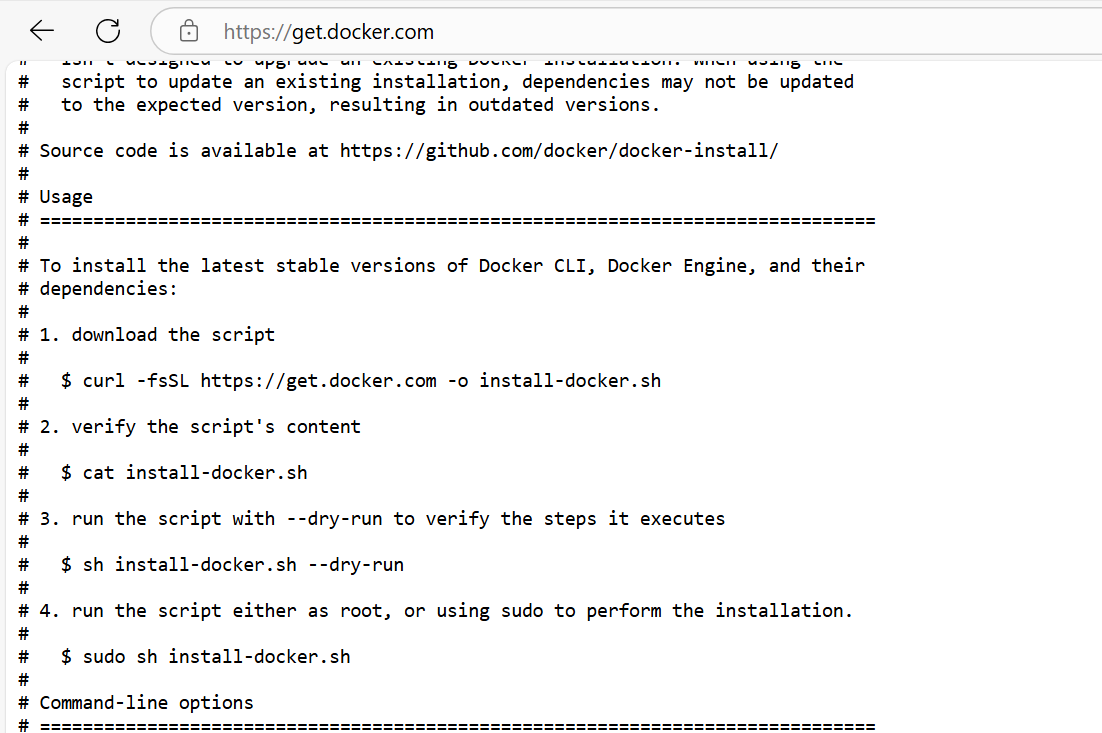
P-->automatic port mapping

--link-->link the multiple containers

PRACTICAL:

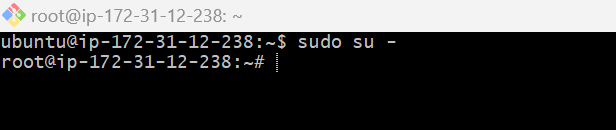
Connect to AWS Create new machine

Open browser (get.docker.com)



Always docker run on root user (#)

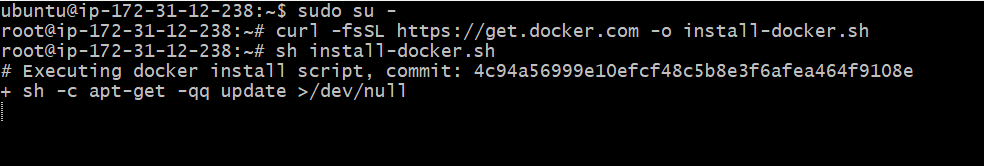
$sudo su -



Download and install docker

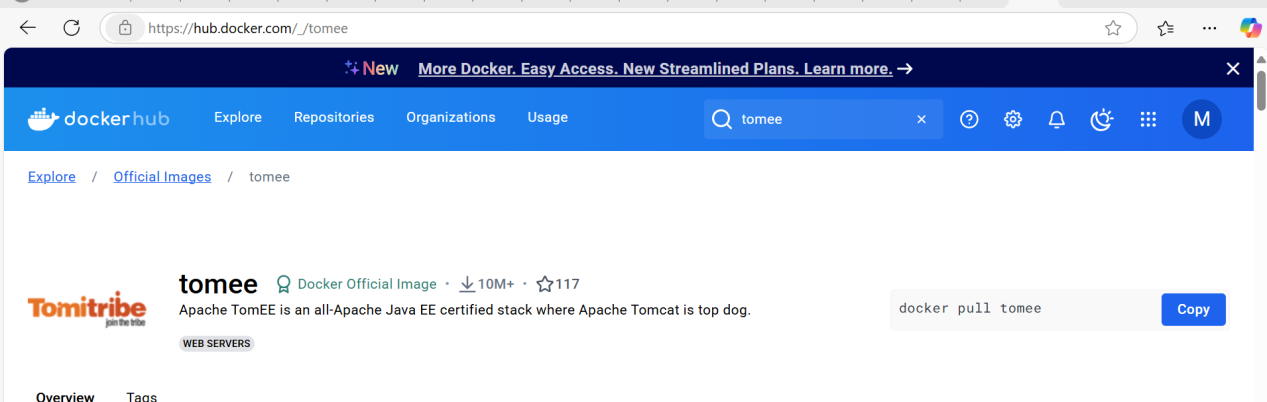
curl -fsSL https://get.docker.com -o install-docker.sh

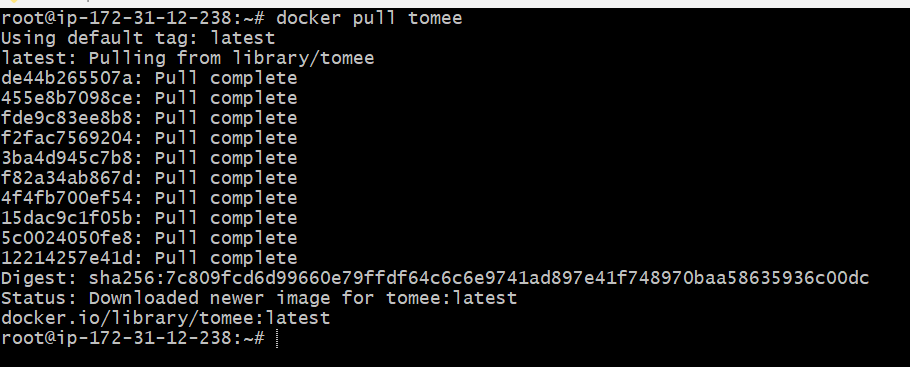
sh install-docker.sh



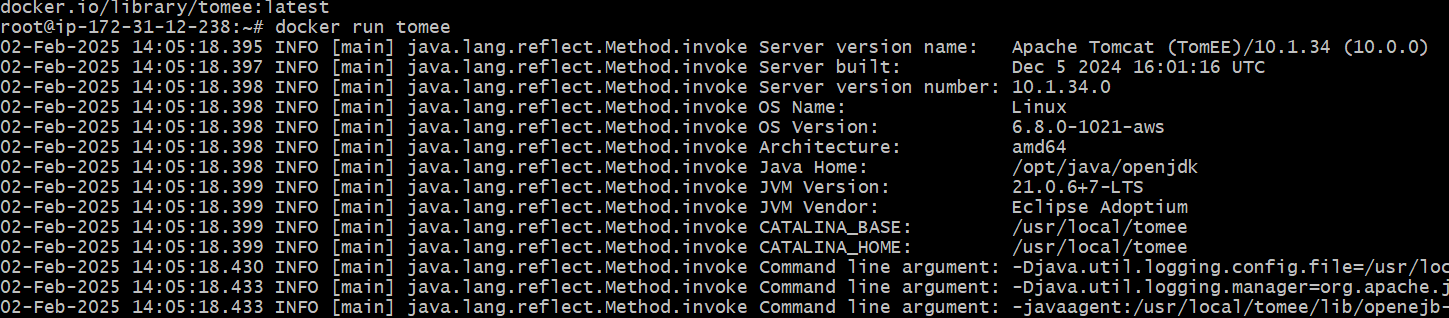
Install tomcat:

Go to the browser(hub.docker.com) search the image name





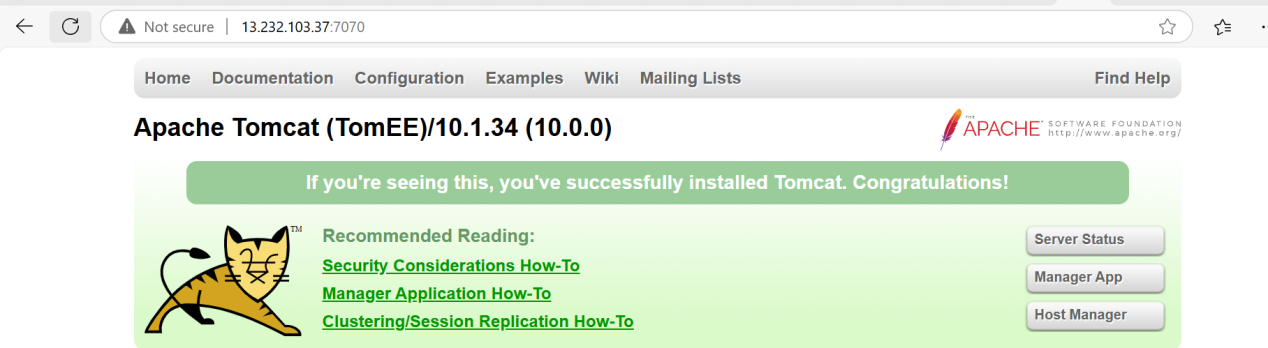
Docker run tomee

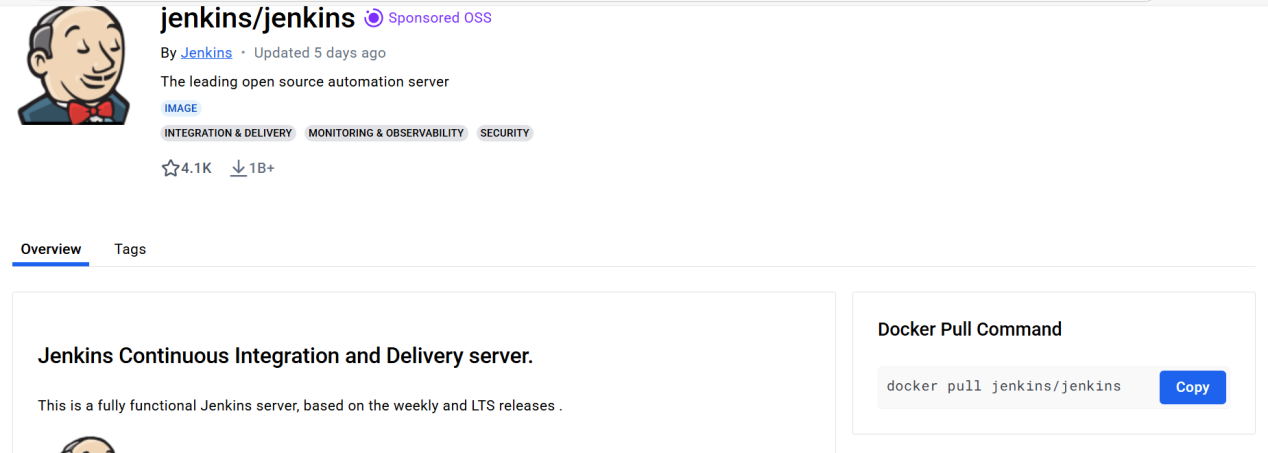


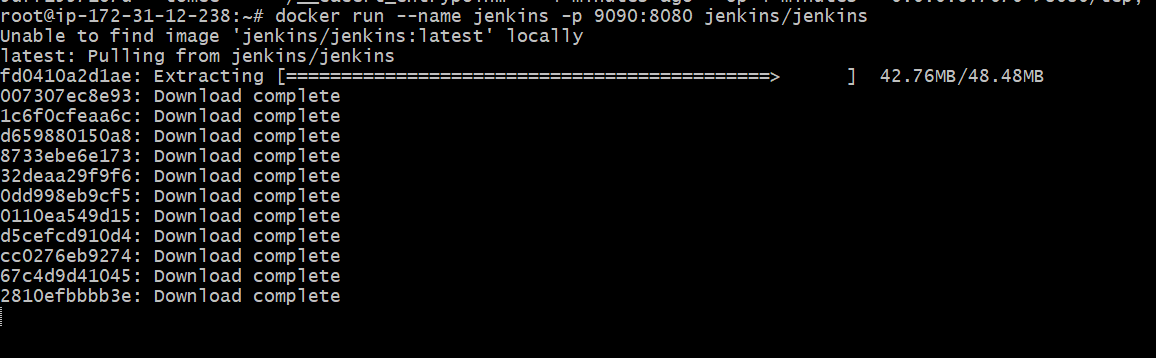
Docker run --name mytomcat -p 7070:8080 tomee

Docker run --name mytomcat -p 7070:8080 -d tomee

Take the public ip and add 7070 to access tomcat









Multi container: linking multiple conatiners

--link and docker compose

Link more than two containers use docker compose file

Docker run --name hub -d -p 4444:4444 selenium/hub

Docker run --name chrome -d -p 4445:5900 --link hub:selenium/hub selenium/node-chrome-debug

Docker volumes:

Docker containers are temporary. When a container is deleted all data will be lost. To preserve the data even after deleting the container use docker volumes

Simple docker volume and docker volume container

Customizing docker images:

Whenever docker container is deleted all the software that we have installed within the container will be deleted.

If we can save the container as an image then we can preserve the software

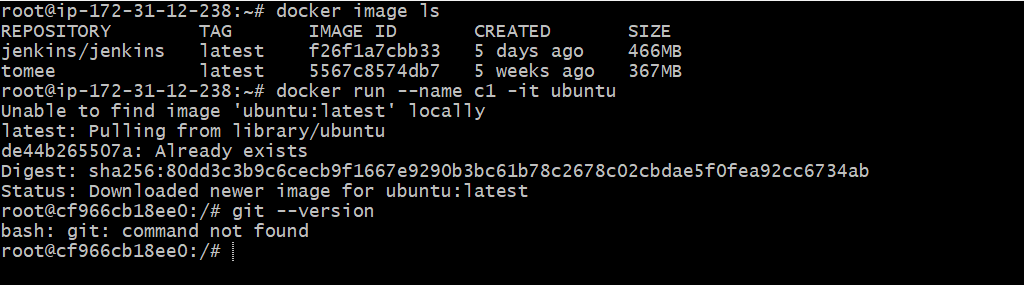
Using docker commit and using docker file





Docker image ls

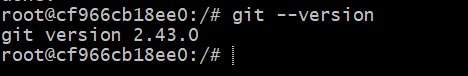
Docker run --name c1 -it ubuntu



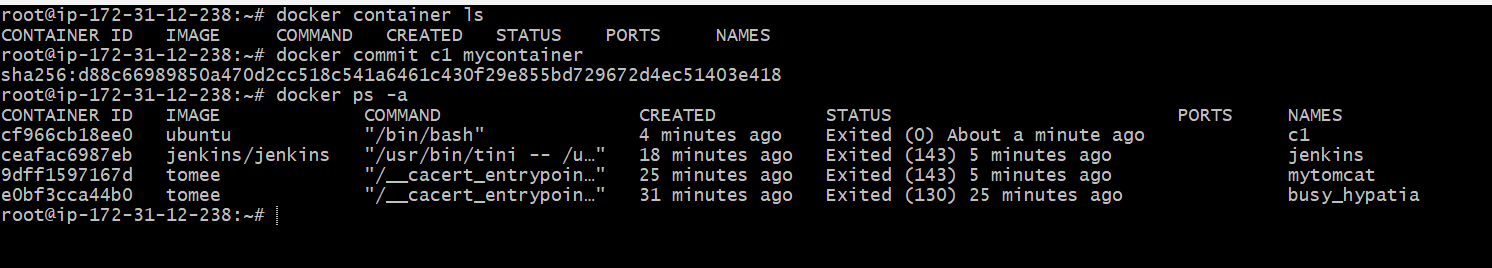
Apt-get update

Apt-get install git

Exit

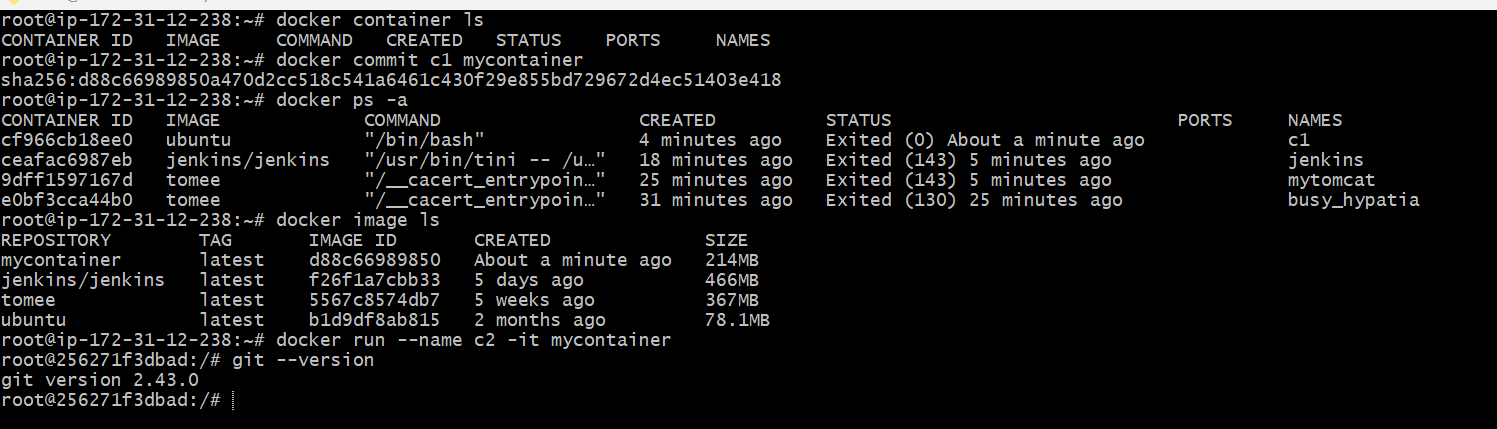


Docker commit c1 mycontainer

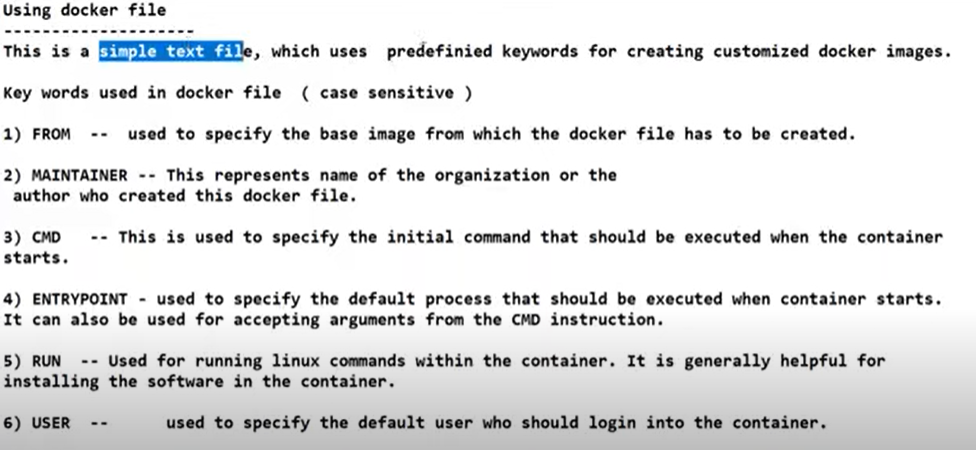


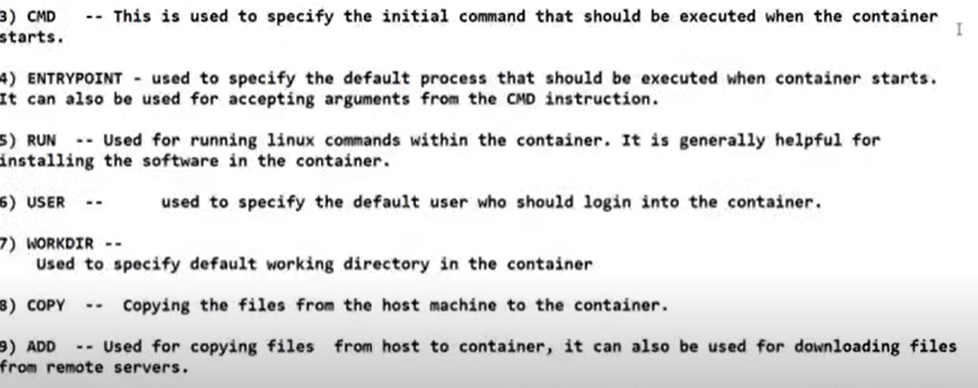
Docker image ls

Docker run --name c2 -it mycontainer



Using docker file:





Create dockerfile: vim dockerfile

FROM openjdk

WORKDIR /app

COPY . /app

RUN javac sample.java

CMD ["java","sample"]

Create sample.java(calculator program)

import java.util.Scanner;

class sample{

public static void main(String[] args) {

char operator;

Double number1, number2, result;

// create an object of Scanner class

Scanner input = new Scanner(System.in);

// ask users to enter operator

System.out.println("Choose an operator: +, -, \*, or /");

operator = input.next().charAt(0);

// ask users to enter numbers

System.out.println("Enter first number");

number1 = input.nextDouble();

System.out.println("Enter second number");

number2 = input.nextDouble();

switch (operator) {

// performs addition between numbers

case '+':

result = number1 + number2;

System.out.println(number1 + " + " + number2 + " = " + result);

break;

// performs subtraction between numbers

case '-':

result = number1 - number2;

System.out.println(number1 + " - " + number2 + " = " + result);

break;

// performs multiplication between numbers

case '\*':

result = number1 \* number2;

System.out.println(number1 + " \* " + number2 + " = " + result);

break;

// performs division between numbers

case '/':

result = number1 / number2;

System.out.println(number1 + " / " + number2 + " = " + result);

break;

default:

System.out.println("Invalid operator!");

break;

}

input.close();

}

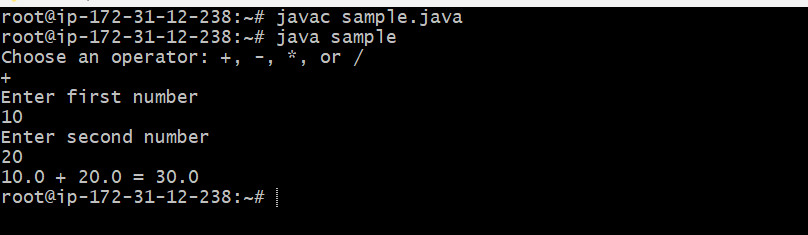
}

Install java

sudo apt-get update

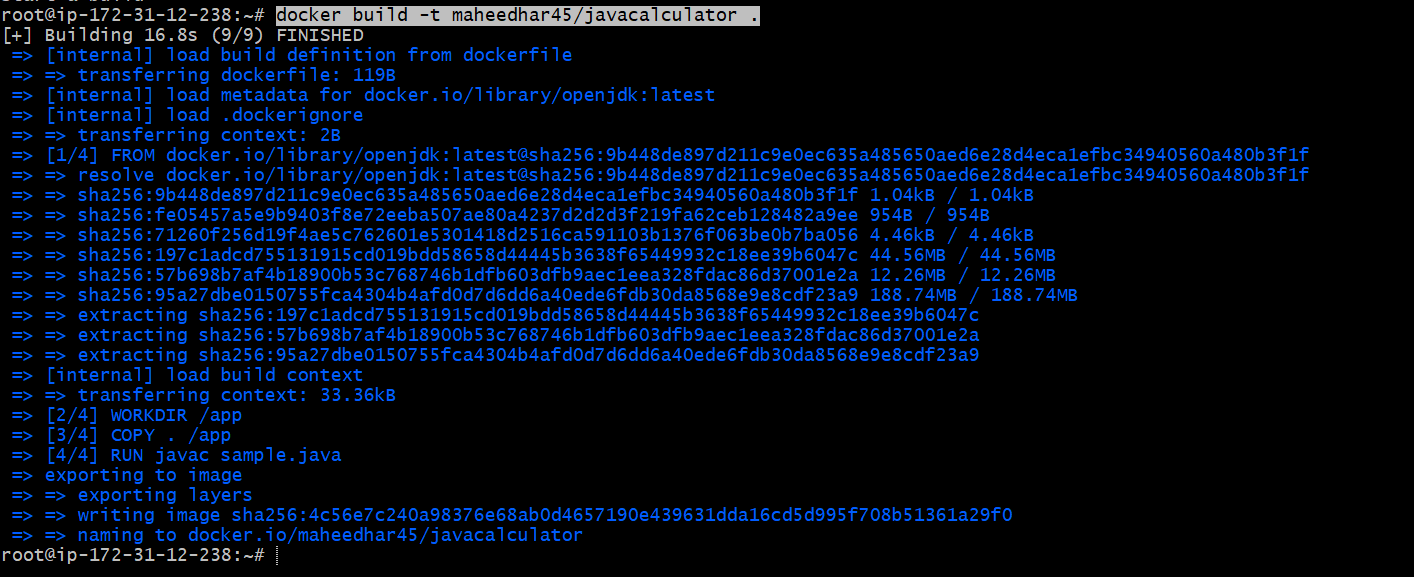
apt-get install openjdk-21-jdk -y

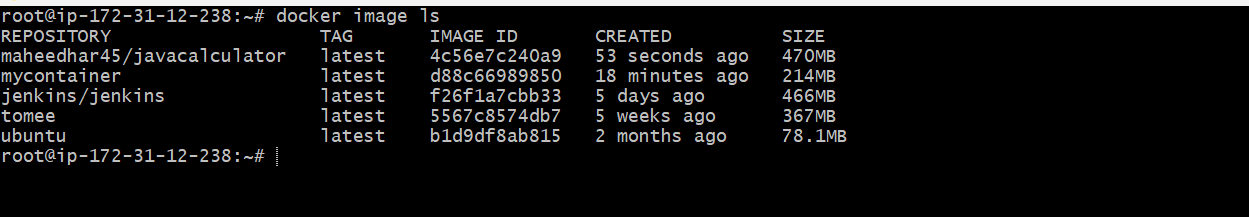
Compile and run the java program



Create a docker image for the java application

docker build -t maheedhar45/javacalculator .





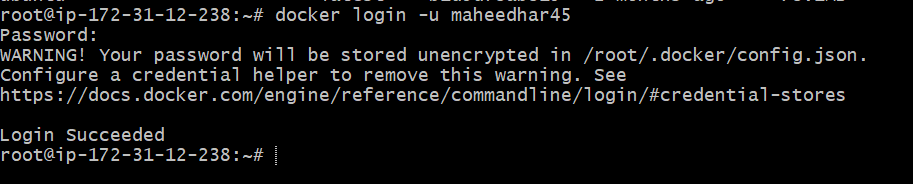
Now u push the docker image into docker hub

So create dockerhub account (hub.docker.com)

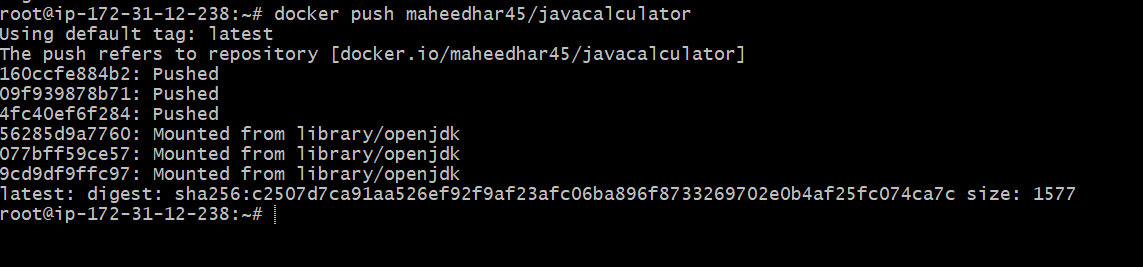
Login

docker login -u ganga20

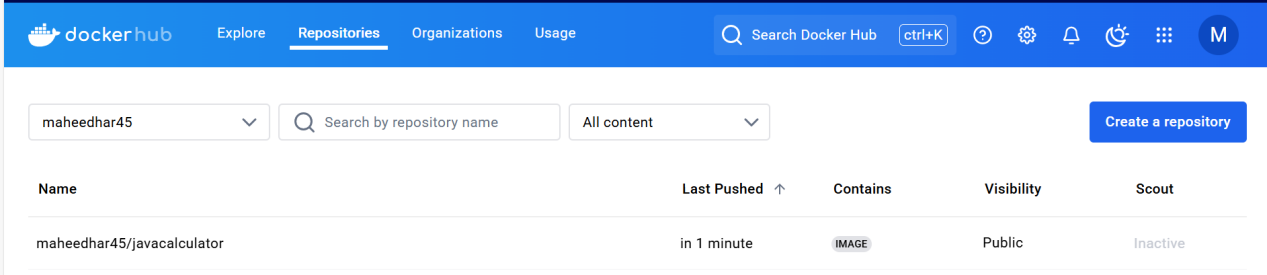
Password:



docker push ganga20/javacalculator2

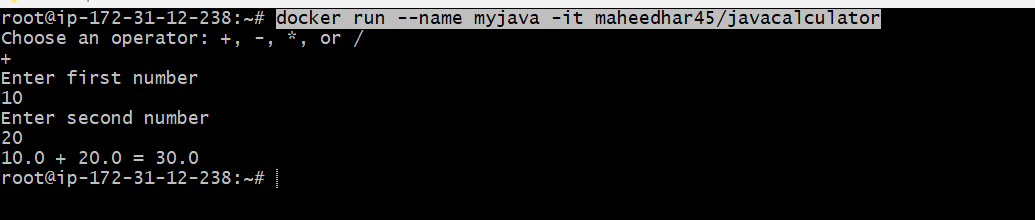


Now go and check in the docker hub



Accessing the image

docker run --name myjava -it ganga20/javacalculator2



Improvise docker image:

Docker file should be edited

FROM openjdk:alpine

WORKDIR /app

COPY . /app

RUN javac sample.java

CMD ["java","sample"]

