Docker: Write a Docker File to pull the Ubuntu with open jdk and write any java application.give me step by step procedure:

Certainly! Below is an example Dockerfile that pulls the Ubuntu image, installs OpenJDK, and then copies a simple Java application into the container. This example assumes you have a Java application JAR file named MyApp.jar in the same directory as the Dockerfile.

Dockerfile code:

# Use the official OpenJDK base image

FROM openjdk:11

# Set the working directory in the container

WORKDIR /usr/src/app

# Copy the Java application JAR file into the container

COPY MyApp.jar .

# Define the command to run the Java application

CMD ["java", "-jar", "MyApp.jar"]

Java code:

// HelloWorld.java

public class HelloWorld {

public static void main(String[] args) {

System.out.println("Hello, World!");

}

}

Commands:

# Compile the Java program

javac HelloWorld.java

# Create a JAR file

jar cfe MyApp.jar HelloWorld HelloWorld.class

Now, you should have the MyApp.jar file in the same directory as your Dockerfile.

Here's a quick explanation of the steps:

* javac HelloWorld.java: Compiles the Java source code and produces the HelloWorld.class file.
* jar cfe MyApp.jar HelloWorld HelloWorld.class: Creates a JAR file named MyApp.jar with the HelloWorld class as the entry point.
* Save the Dockerfile in a directory along with your MyApp.jar file.
* Open a terminal and navigate to the directory containing the Dockerfile.
* Build the Docker image using the following command:

docker build -t my-java-app .

This command tells Docker to build an image using the Dockerfile in the current directory (.) and tags it with the name my-java-app.

* Once the image is built, you can run a container based on it using:

docker run my-java-app

Install and configure NFS server:

<https://youtu.be/px0A_P1V06k?si=oUXb0RfMH24W1olc>

NFS stands for "Network File System." It is a distributed file system protocol that enables a user to access files and directories on a remote server as if they were local. NFS allows for file sharing between computers in a network, providing a mechanism for efficient and transparent access to shared files and resources.

NFS operates on the client-server model, where the machine hosting the shared files is the server, and the machine accessing those files is the client

Terminal1:(server)

sudo apt update

sudo apt install nfs-kernel-server

sudo mkdir -p /mnt/nfs\_share

sudo chown -R nobody:nogroup /mnt/nfs\_share/

sudo chmod 777 /mnt/nfs\_share/

sudo vim /etc/exports:

Add following line to the file-

/mnt/nfs\_share 10.0.2.15(rw,sync,no\_subtree\_check)

sudo exportfs -a

sudo systemctl restart nfs-kernel-server

sudo ufw allow from 10.0.2.15 to any port nfs

sudo ufw enable

sudo ufw status:

Status: active

To Action From

-- ------ ----

2049 ALLOW 192.168.0.127

2049 ALLOW 10.0.2.15

touch file1.txt file2.txt file3.txt

terminal2:(client)

sudo apt install nfs-common

sudo mkdir -p /mnt/nfs\_clientshare

sudo mount 10.0.2.15:/mnt/nfs\_share /mnt/nfs\_clientshare

ls -l /mnt/nfs\_clientshare/ -

total 0

-rw-rw-r-- 1 sonali sonali 0 Dec 14 22:15 file1.txt

-rw-rw-r-- 1 sonali sonali 0 Dec 14 22:15 file2.txt

-rw-rw-r-- 1 sonali sonali 0 Dec 14 22:15 file3.txt

1. Joomla-

<https://youtu.be/Eak9TmaaLII?si=Y626thk149CN46Ai>

apt install apache2 -y

systemctl start apache2

systemctl enable apache2

systemctl status apache2

apt install php libapache2-mod-php php-dev php-bcmath php-intl php-soap php-zip php-curl php-mbstring php-mysql php-gd php-xml -y

php -v

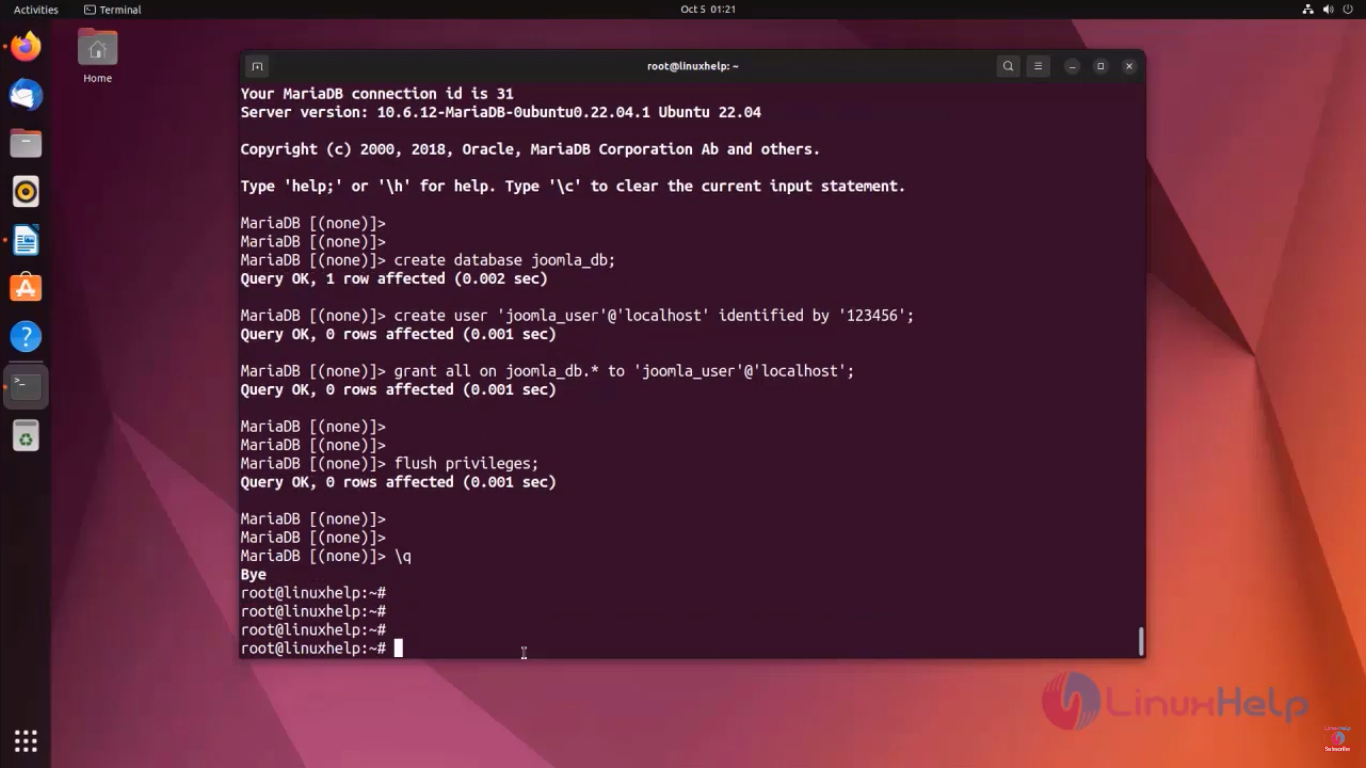
apt install mariadb-server -y

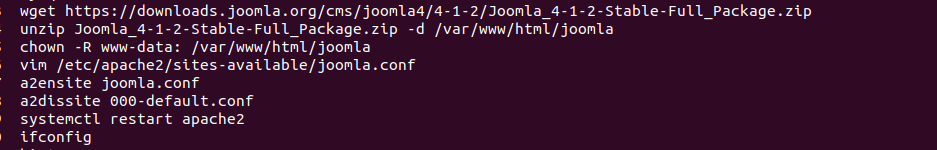
systemctl start mariadb

systemctl enable mariadb

systemctl status mariadb

mysql:

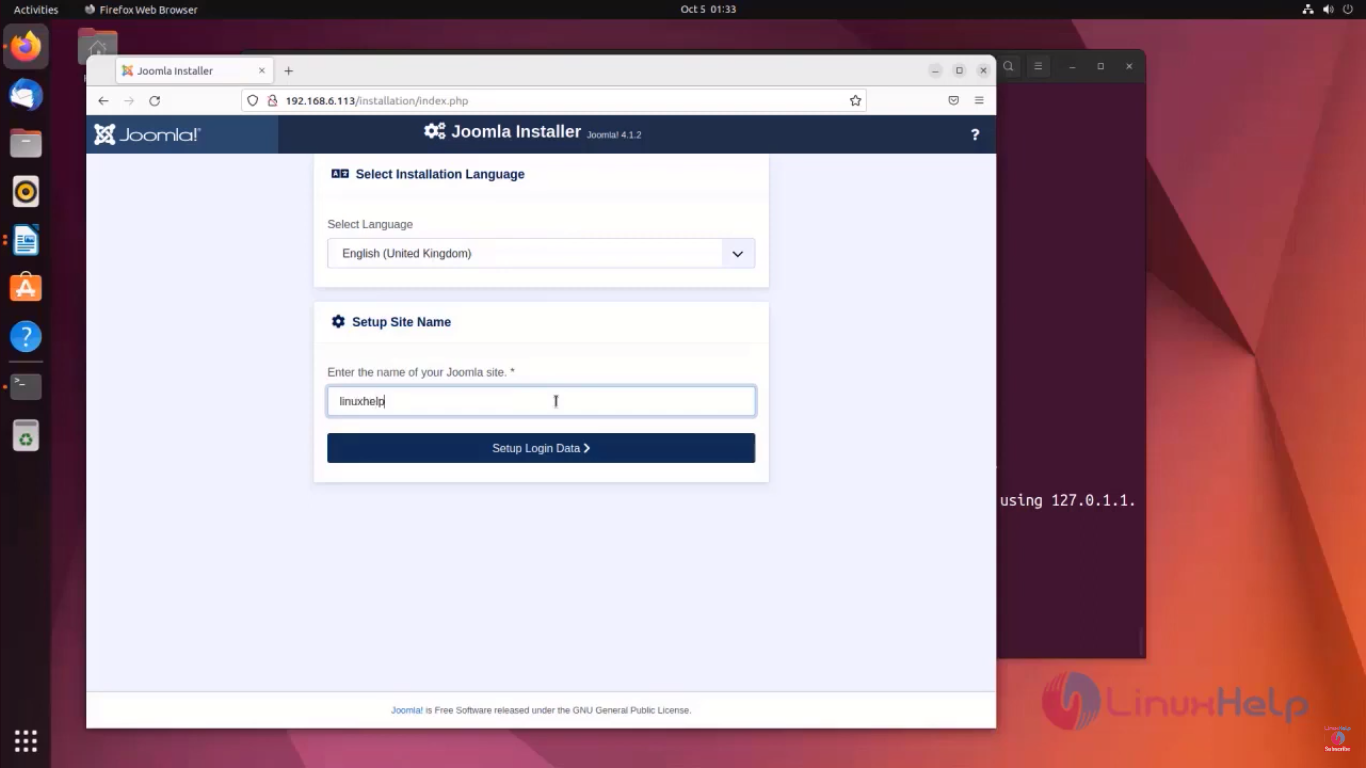


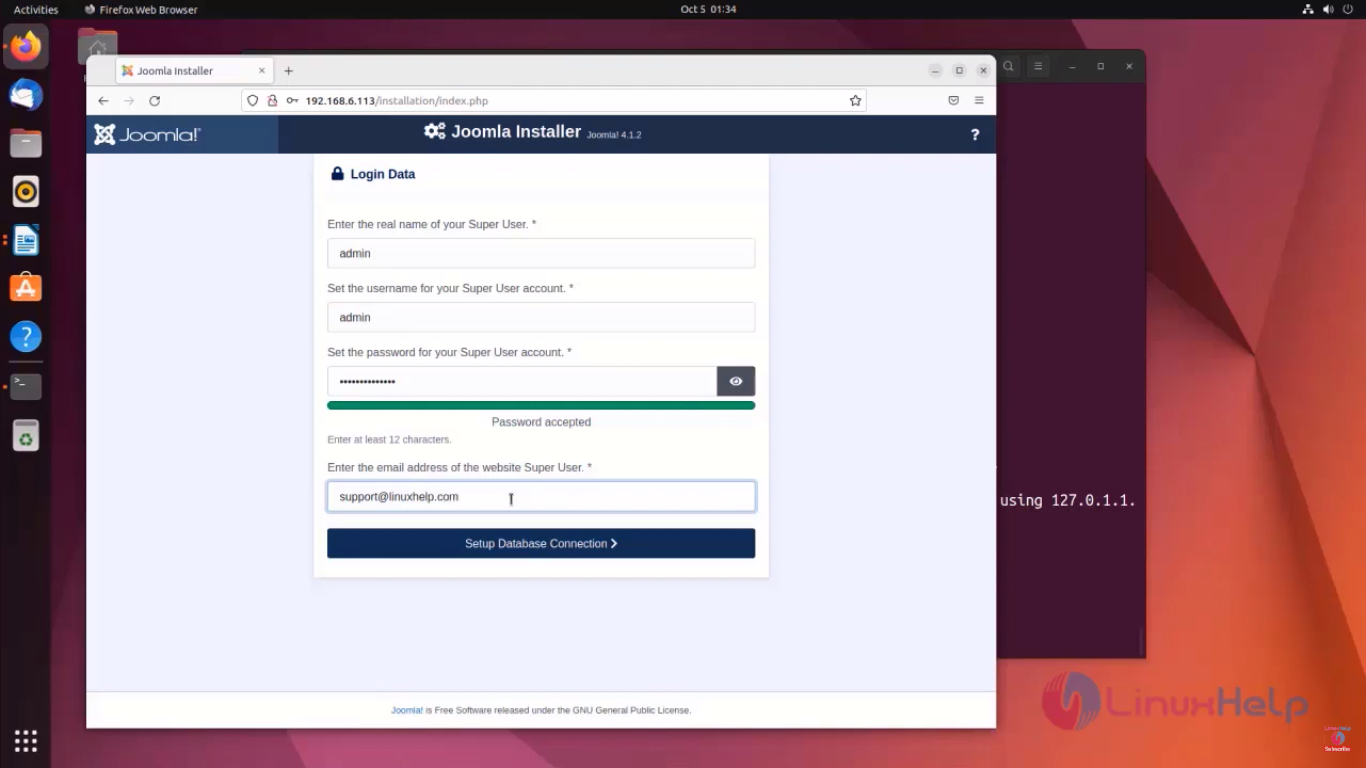


Paste below content in joomla.conf:

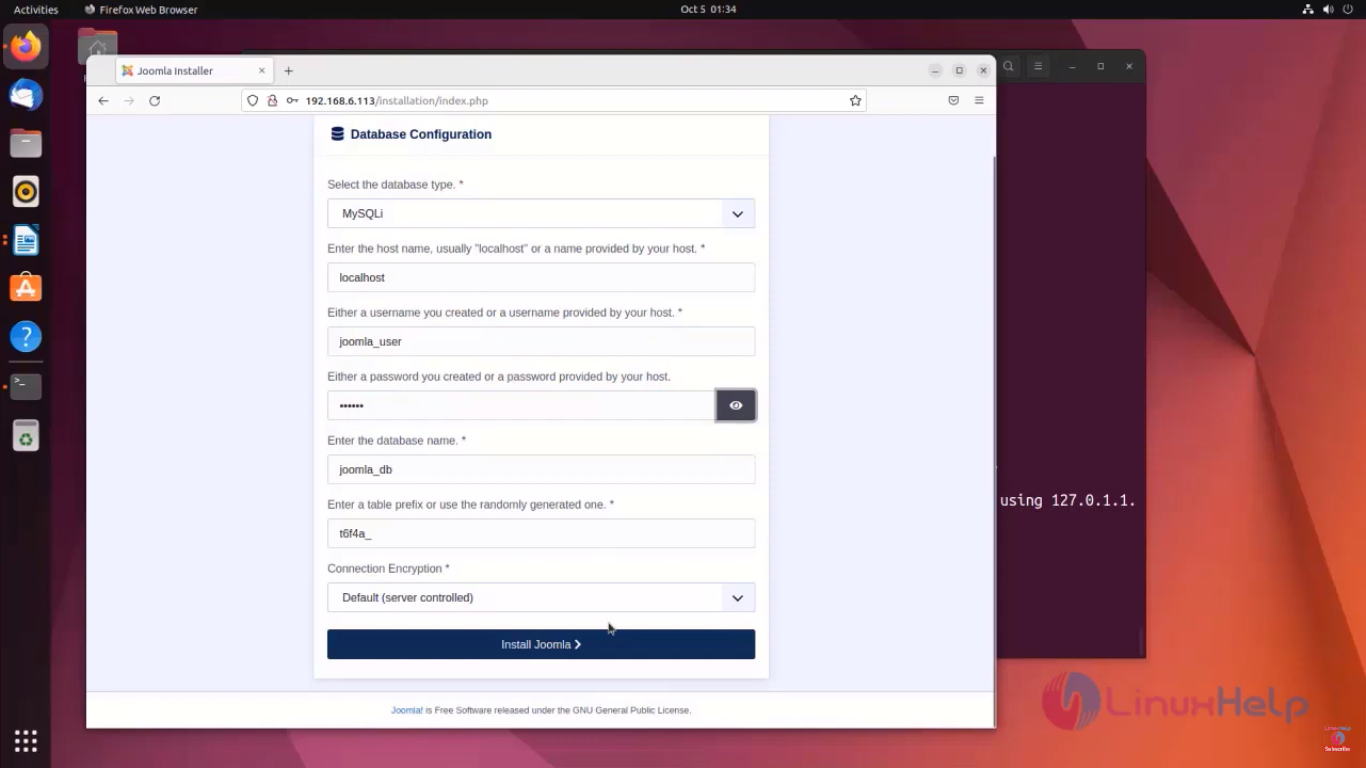


Now go to firefox browser and type your machine IP address and press enter

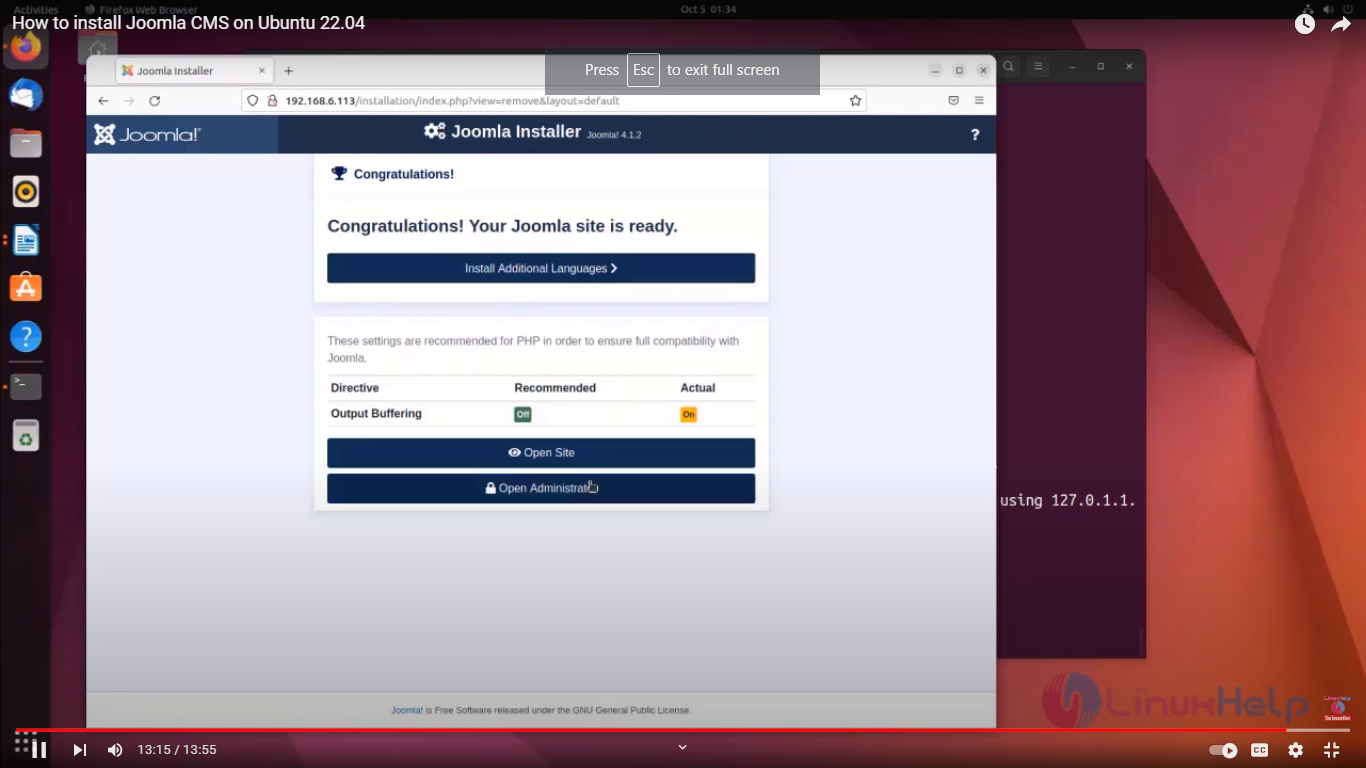


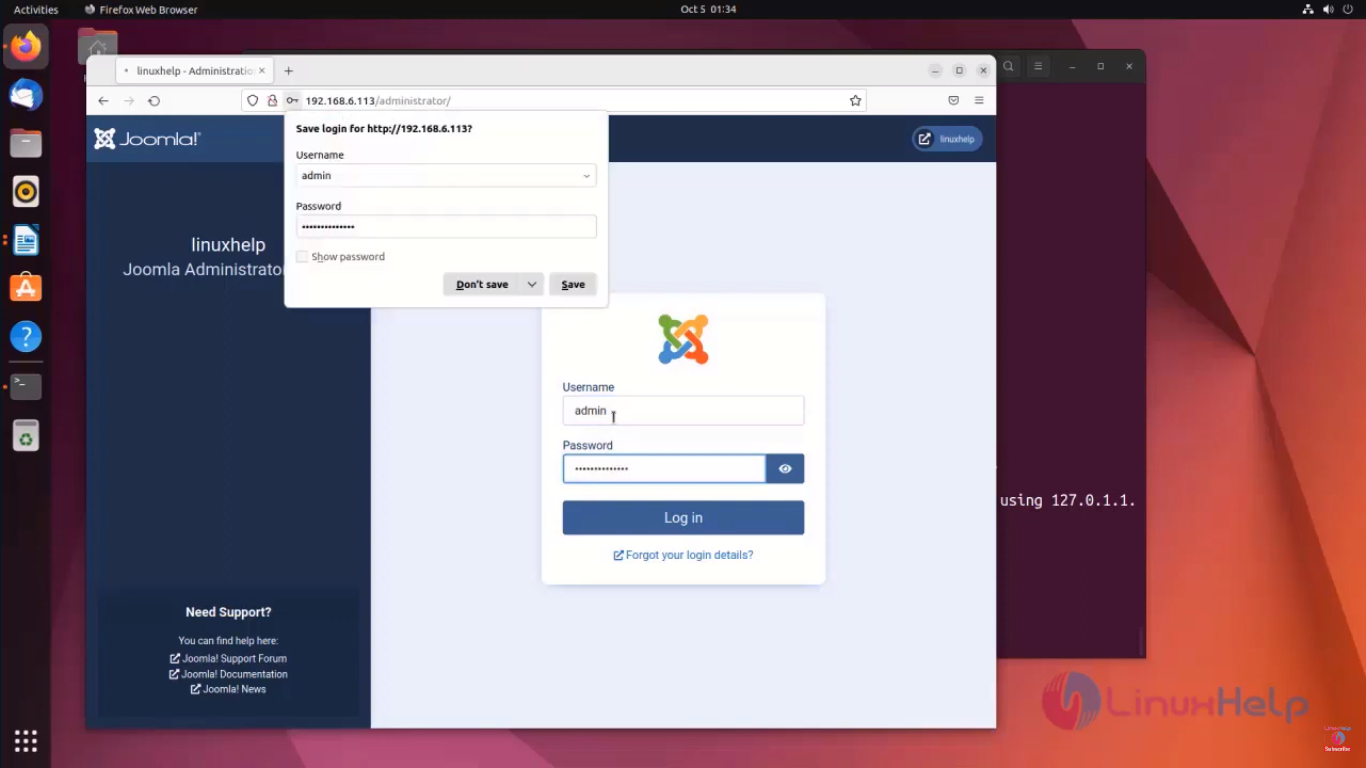


Here password is:Passed@12345



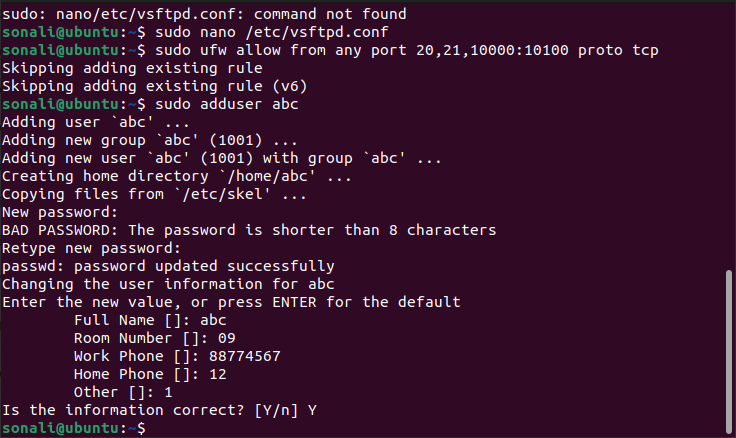
Here password for database configuration is 123456





FTP password:

12345



Docker: A. Create a simple Hello-world python flask application and create the docker image of that Flask application.

B. Run the docker container from recently created image and run that docker container to 5000 port of host system on ubuntu giev me step by step procedure with detailed explanation

app.py:

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return 'Hello, World! This is a Flask application.'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True, host='0.0.0.0')

Dockerfile:

# Use an official Python runtime as a parent image

FROM python:3.8-slim

# Set the working directory to /app

WORKDIR /app

# Copy the current directory contents into the container at /app

COPY . /app

# Install any needed packages specified in requirements.txt

RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 5000 available to the world outside this container

EXPOSE 5000

# Define environment variable

ENV NAME World

# Run app.py when the container launches

CMD ["python", "app.py"]

requirements.txt:

Flask==2.0.1

Werkzeug==2.0.2

### **Step 4: Build the Docker Image**

Open a terminal and navigate to your project directory. Run the following command:

docker build -t flask-hello-world .

docker build -t flask-hello-world .

This command will build a Docker image named flask-hello-world using the current directory as the build context.

### **Step 5: Run the Docker Container**

Now, let's run the Docker container and map port 5000 on the host to port 5000 on the container:

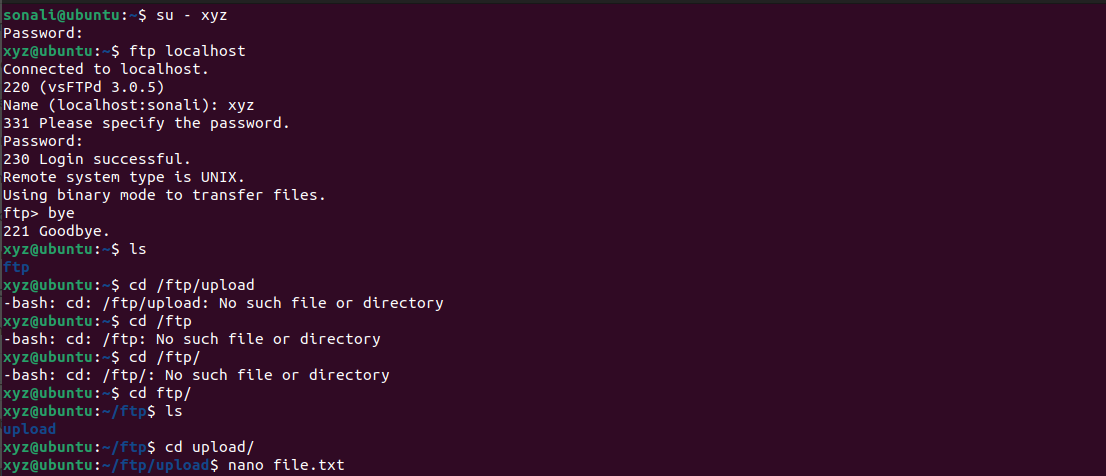
docker run -p 5000:5000 flask-hello-world

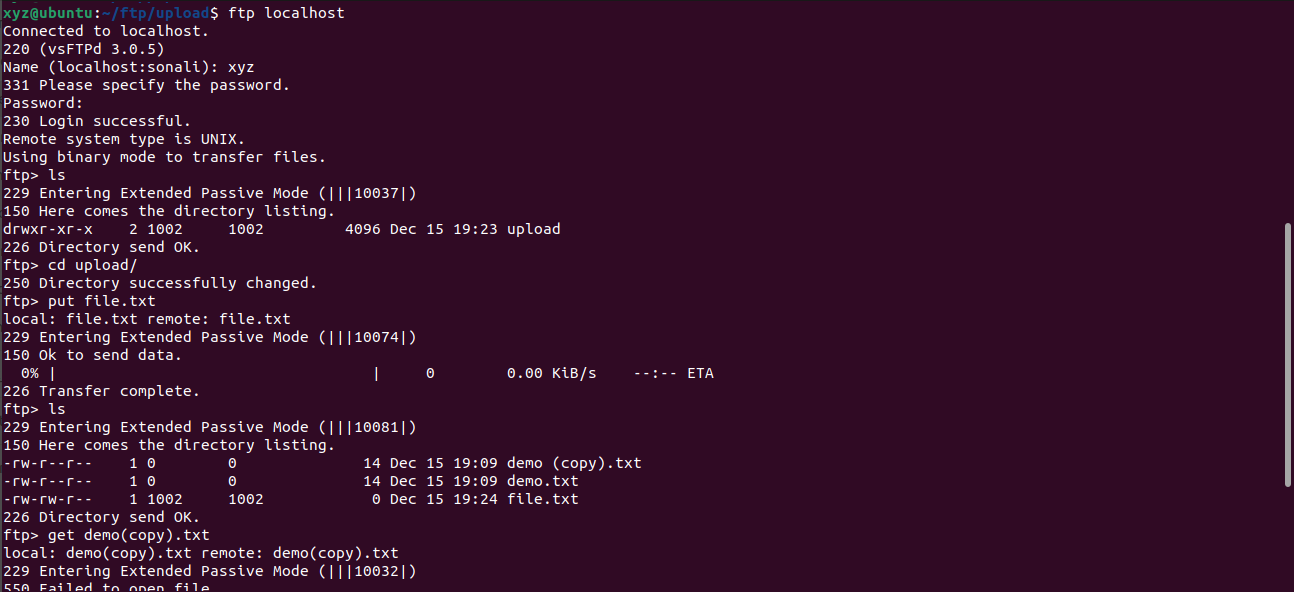
docker run -p 5000:5000 flask-hello-world

### **Step 6: Access the Flask Application**

Open your web browser and go to http://localhost:5000. You should see the "Hello, World! This is a Flask application." message.

Now you have successfully created a simple Flask application, containerized it with Docker, and accessed it from your host machine.

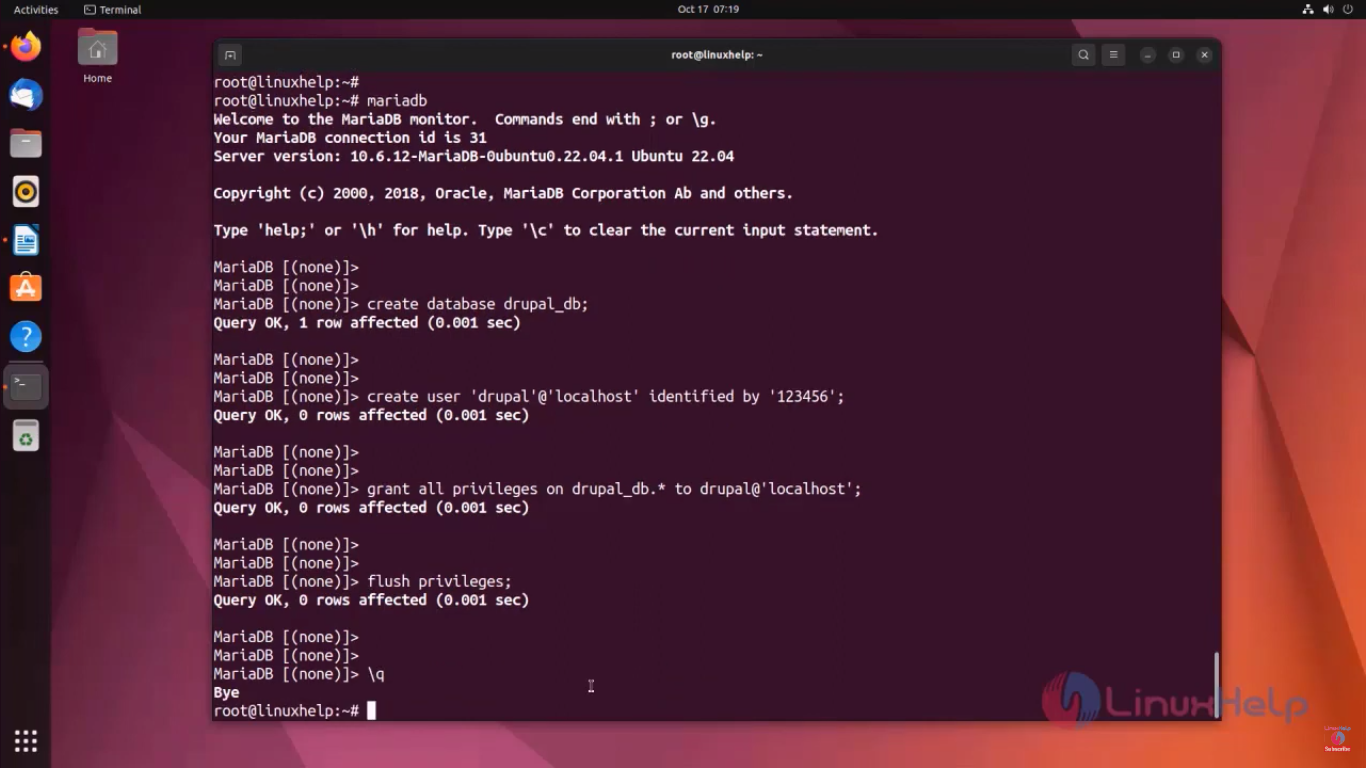




Drual:

Sudo apt install -y mariadb-server mariadb-client

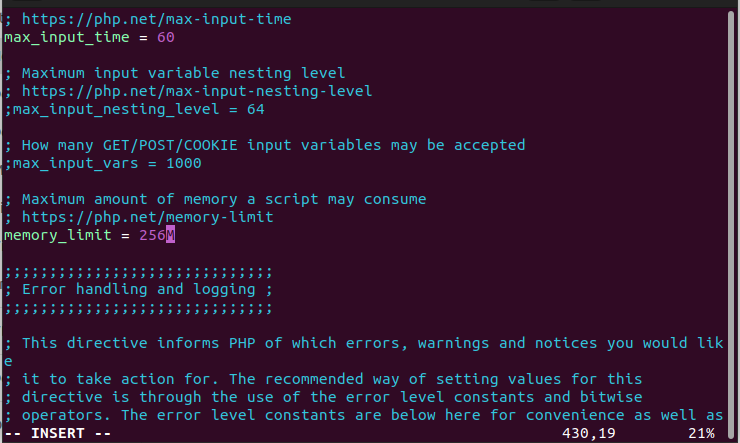
systemctl status mariadb



apt install php php-{cli,fpm,json,common,mysql,zip,gd,intl,mbstring,curl,xml,pear,tidy,soap,bcmath,xmlrpc}

apt install apache2 libapache2-mod-php -y

vim /etc/php/8.1/apache2/php.ini



Change memory\_limit to 256M

wget https://www.drupal.org/download-latest/tar.gz -O drupal.tar.gz

tar xvf drupal.tar.gz

mv drupal-\*/ /var/www/html/drupal

sudo chmod -R 755 /var/www/html/