**Aim**: Containerization using Docker.

**Objectives:** The objective of this task is to understand and implement containerization using Docker. This includes installing Docker, containerizing an application, updating and sharing it via Docker Hub, and running multi-container applications.

**Tools Used**: Virtual box, Ubuntu , Docker

**Concepts:** Containerization is a way to run applications in separate, lightweight environments with everything they need to work. Docker makes this process easy by packaging an app and its dependencies into a container. With multi-container setups, different parts of an app, like a database and a web server, can work together smoothly.

**Problem Statement:**

Download the app : https://docs.docker.com/get-started/02\_our\_app/

Update the app : https://docs.docker.com/get-started/03\_updating\_app/

Share the app : https://docs.docker.com/get-started/04\_sharing\_app/

Download the shared app and run it again.

Multi Container App : https://docs.docker.com/get- started/07\_multi\_container/

**Process:**

1. **Installation of Docker**

Run:

sudo apt-get install  
  
Now Install Docker Using the command:  
sudo apt-get install docker.io

then to check the docker version: docker -v



Now start the docker and enable it

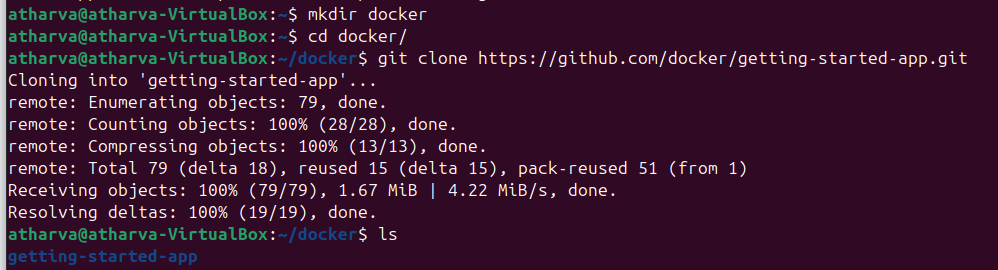
sudo systemctl start docker

sudo systemctl enable docker

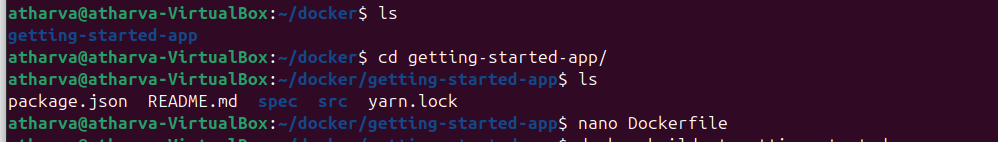


# **Containerize an application**

Make a directory and clone a repo



Now go inside the getting-started-app folder and create a Dockerfile



Inside Dockerfile write the below content

FROM node:lts-alpine

WORKDIR /app

COPY . .

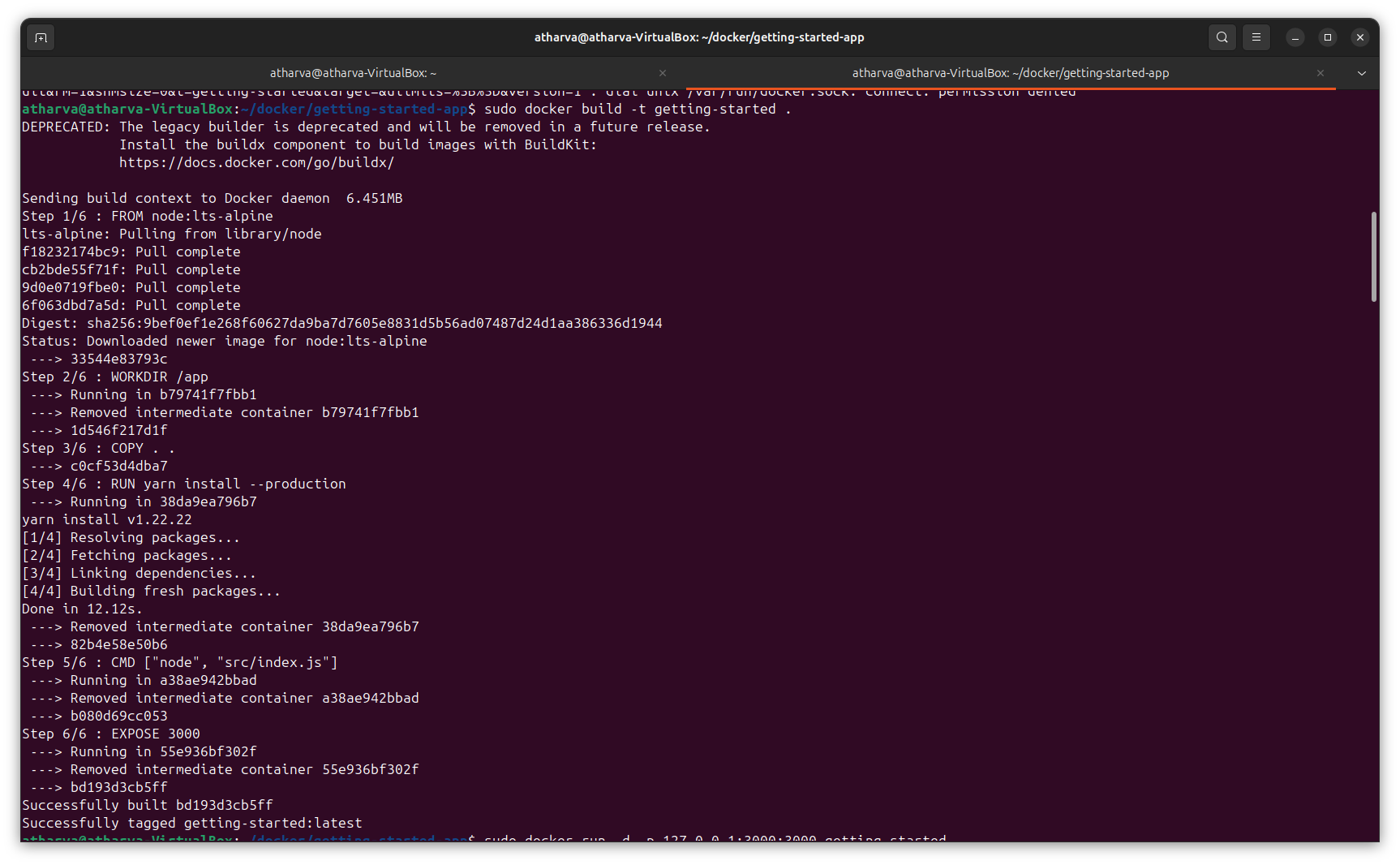
RUN yarn install --production

CMD ["node", "src/index.js"]

EXPOSE 3000

Now Build the Image

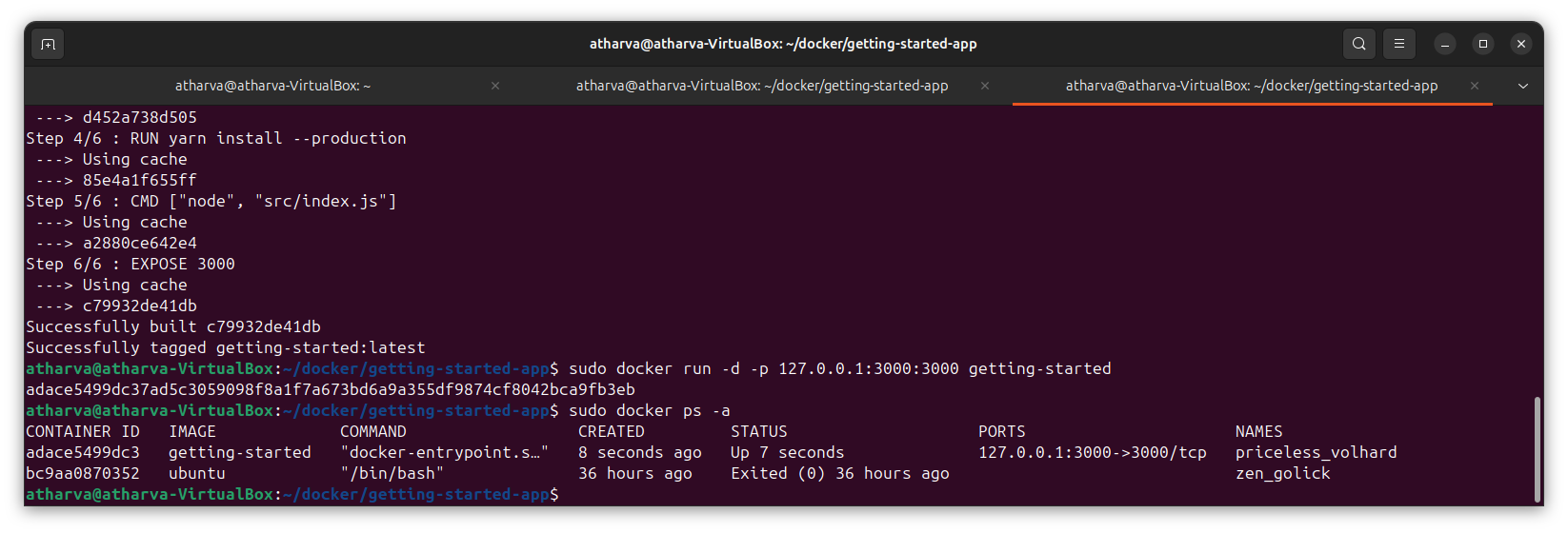
Command: sudo docker build -t getting-started .



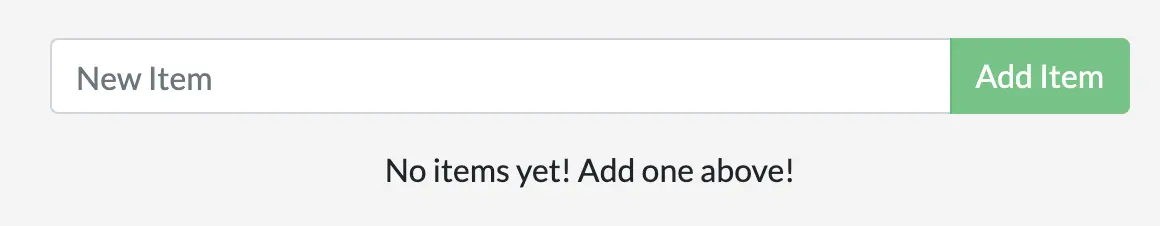
## Now Start an app container

command : sudo docker run -d -p 127.0.0.1:3000:3000 getting-started

Now to check the status run: sudo docker ps -a



After a few seconds, open your web browser to [http://localhost:3000](http://localhost:3000/). You should see your app.



# **Update the application**

Update the source code

In the src/static/js/app.js file of the project make some changes.

Now, Build your updated version of the image, using the docker build command

->docker build -t getting-started .

Now remove the old container first

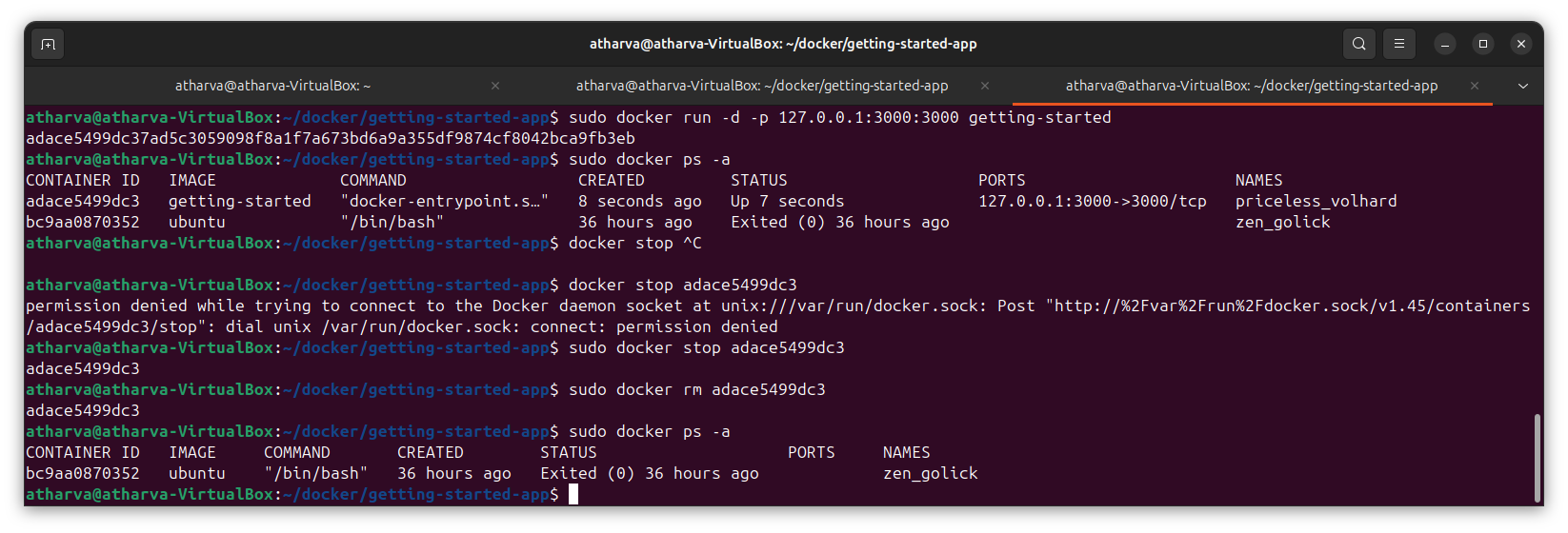
Get the ID of the container by using the docker ps -a command.

Use the docker stop command to stop the container. Replace <the-container-id> with the ID from docker ps.

docker stop <the-container-id>

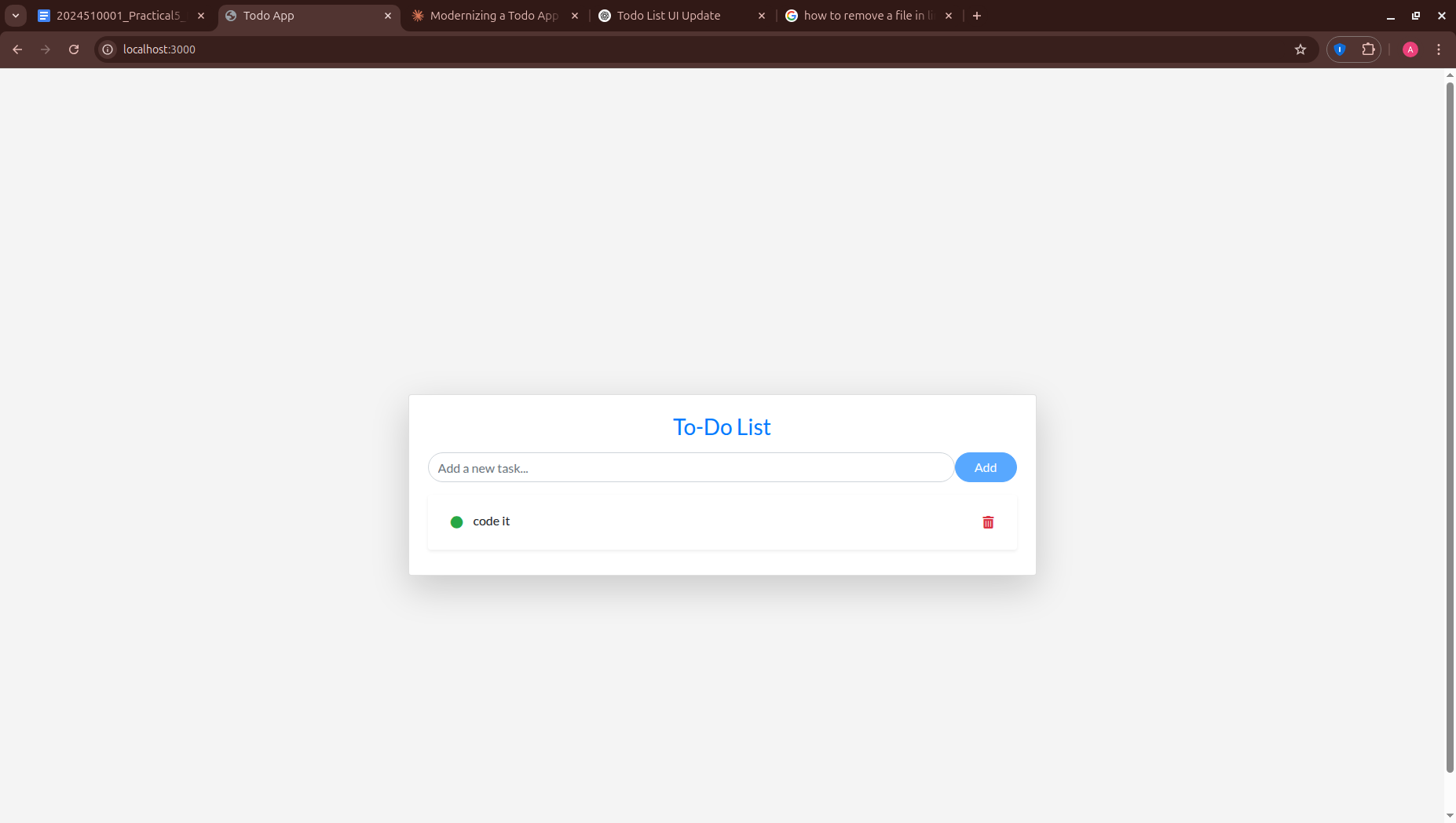
Once the container has stopped, you can remove it by using the docker rm command.

docker rm <the-container-id>



Now, start your updated app using the docker run command.

docker run -dp 127.0.0.1:3000:3000 getting-started



1. **Share the application**

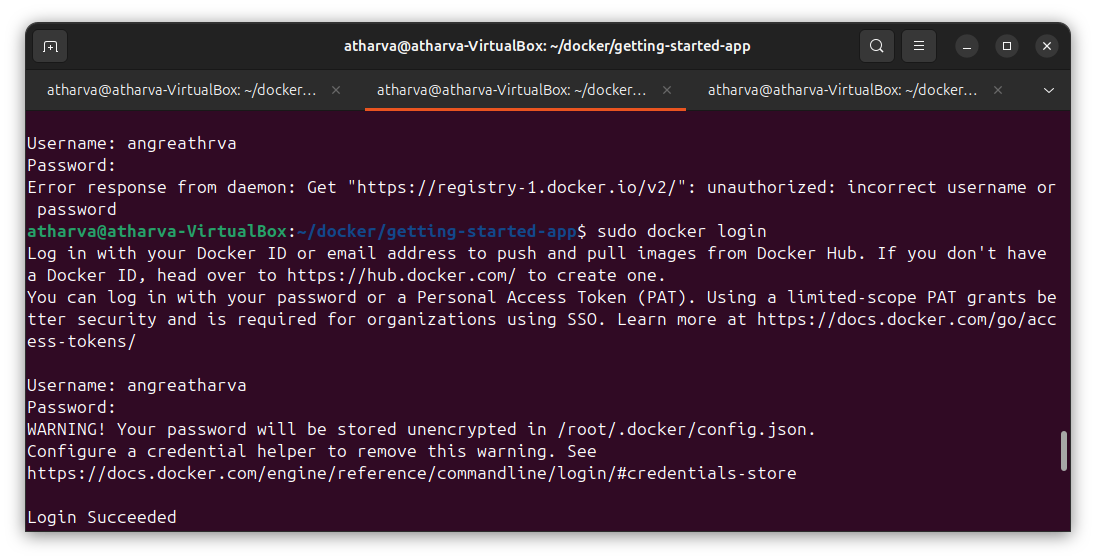
Create a repository

1. Sign up or Sign in to Docker Hub.
2. Select the Create Repository button.
3. For the repository name, use getting-started. Make sure the Visibility is Public.
4. Select Create.

Push the image

First Login to docker in CLI

command : sudo docker login



Now first

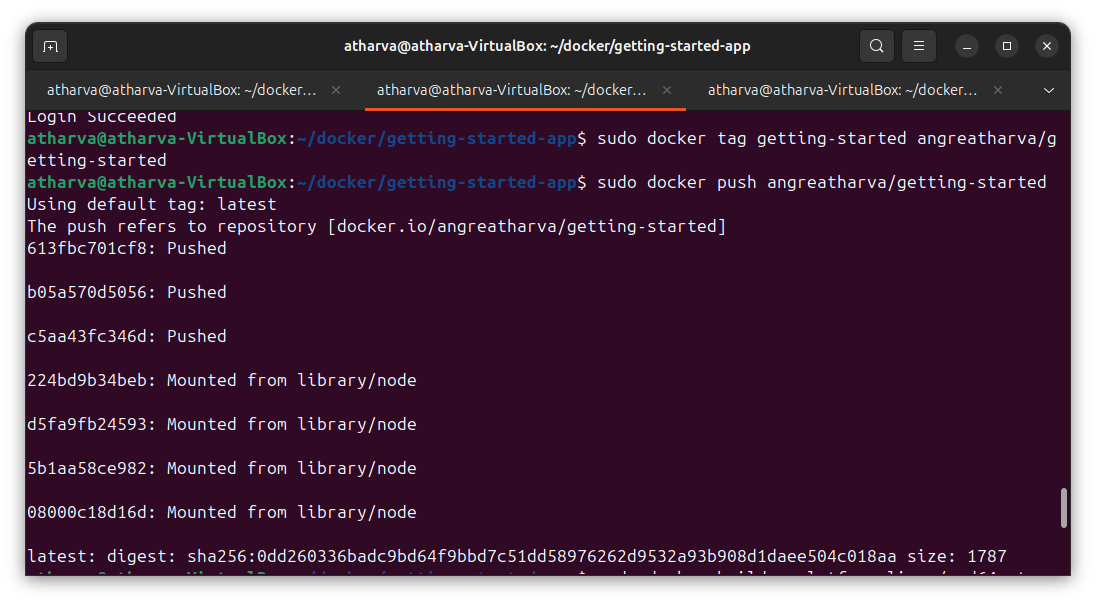
Use the docker tag command to give the getting-started image a new name. Replace YOUR-USER-NAME with your Docker ID.

Command: docker tag getting-started YOUR-USER-NAME/getting-started

And then

In the command line, run the docker push command that you see on Docker Hub. Note that your command will have your Docker ID.

For example, docker push YOUR-USER-NAME/getting-started.



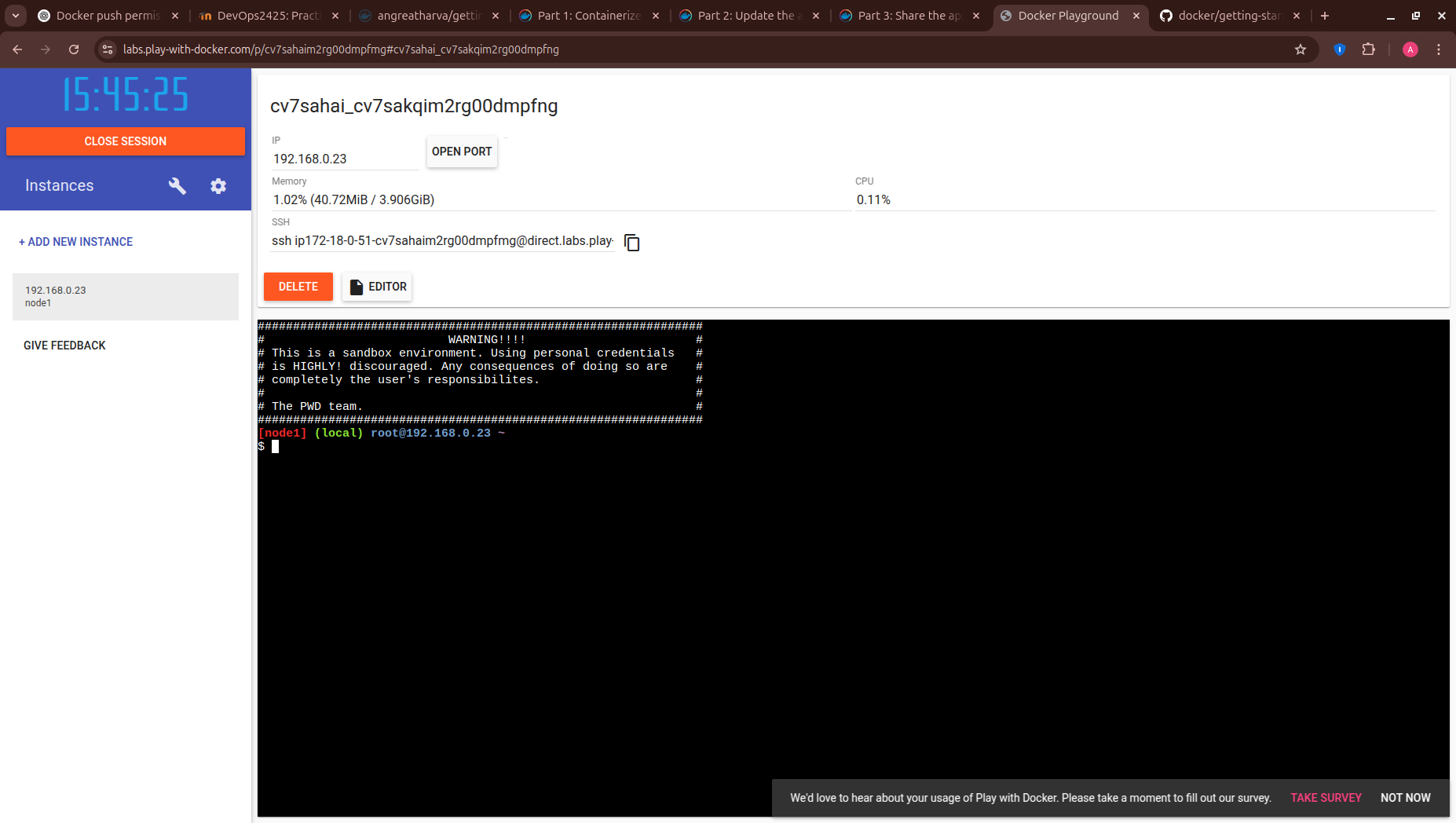
Now Run the image on a new instance

To build an image for the amd64 platform, use the --platform flag.

Command: docker build --platform linux/amd64 -t YOUR-USER-NAME/getting-started .

Now

1. Open your browser to Play with Docker.
2. Select Login and then select docker from the drop-down list.
3. Sign in with your Docker Hub account and then select Start.
4. Select the ADD NEW INSTANCE option on the left side bar. If you don't see it, make your browser a little wider. After a few seconds, a terminal window opens in your browser.



1. In the terminal, start your freshly pushed app.

-> In the terminal, start your freshly pushed app.

# **5) Multi container apps**

Create the network.

command : sudo docker network create todo-app

Start a MySQL container and attach it to the network.

Command:

docker run -d \

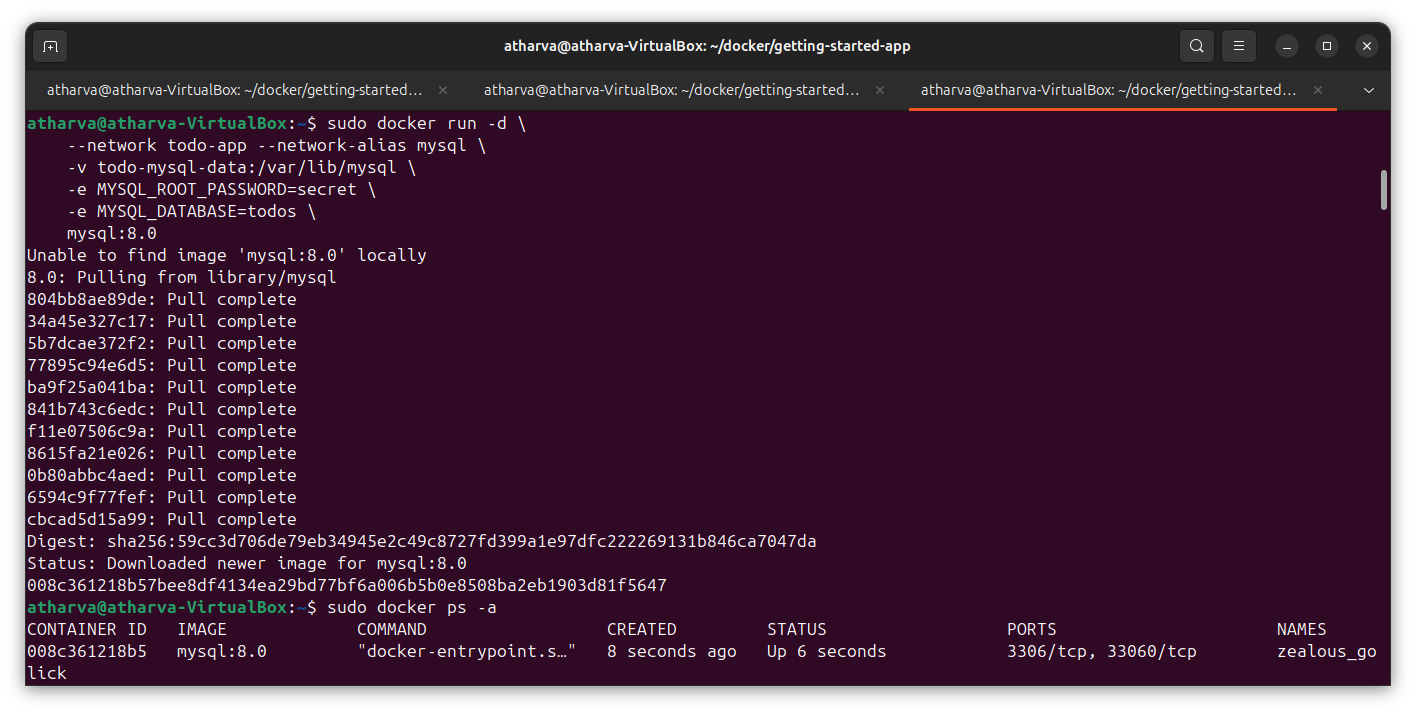
--network todo-app --network-alias mysql \

-v todo-mysql-data:/var/lib/mysql \

-e MYSQL\_ROOT\_PASSWORD=secret \

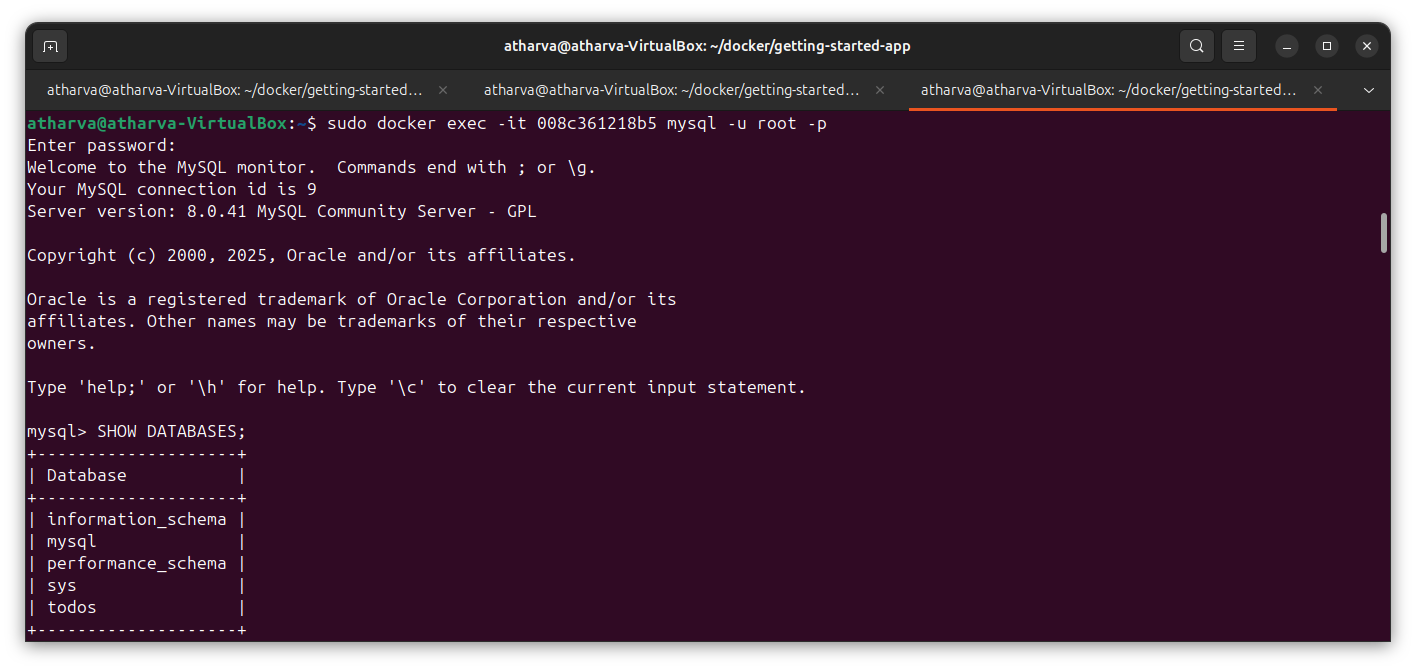
-e MYSQL\_DATABASE=todos \

mysql:8.0



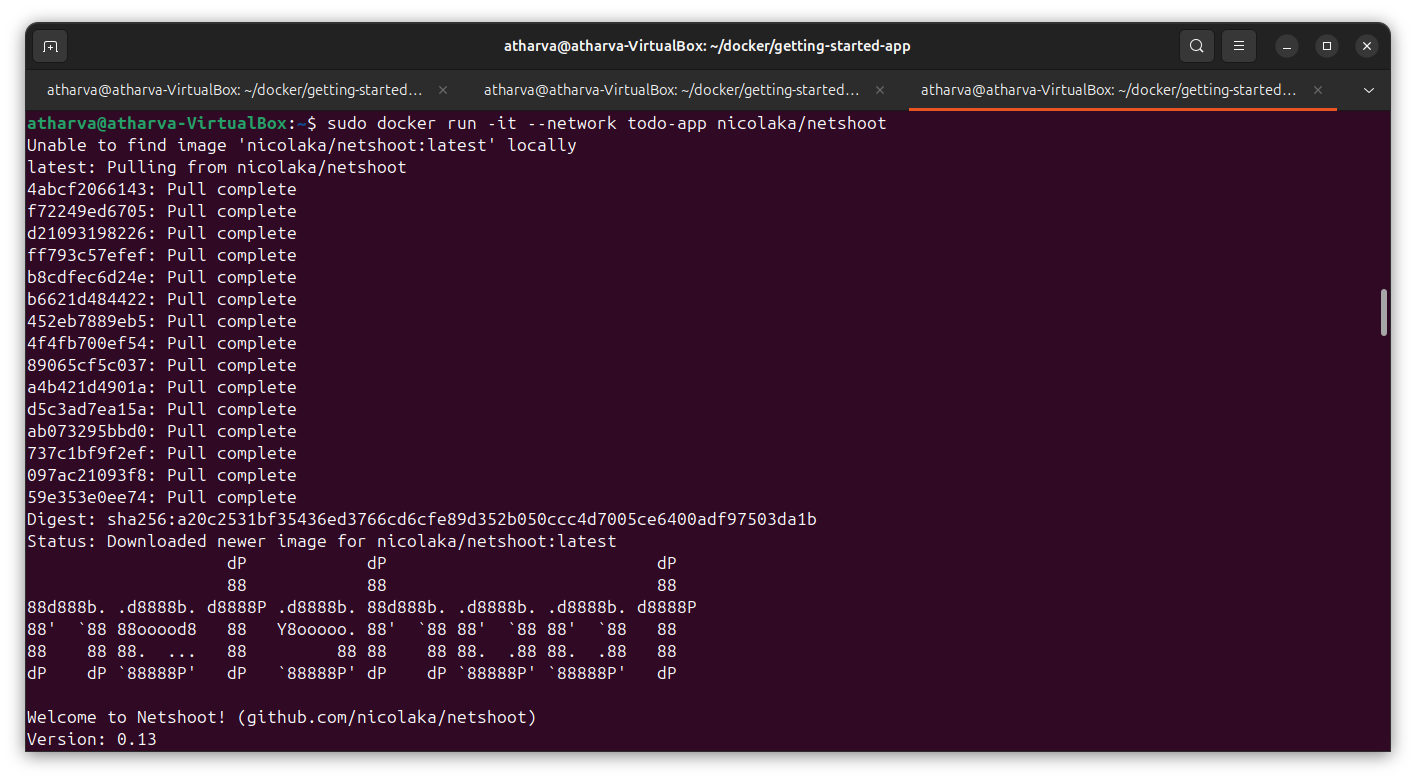
To confirm you have the database up and running, connect to the database and verify that it connects.

Command : docker exec -it <mysql-container-id> mysql -u root -p



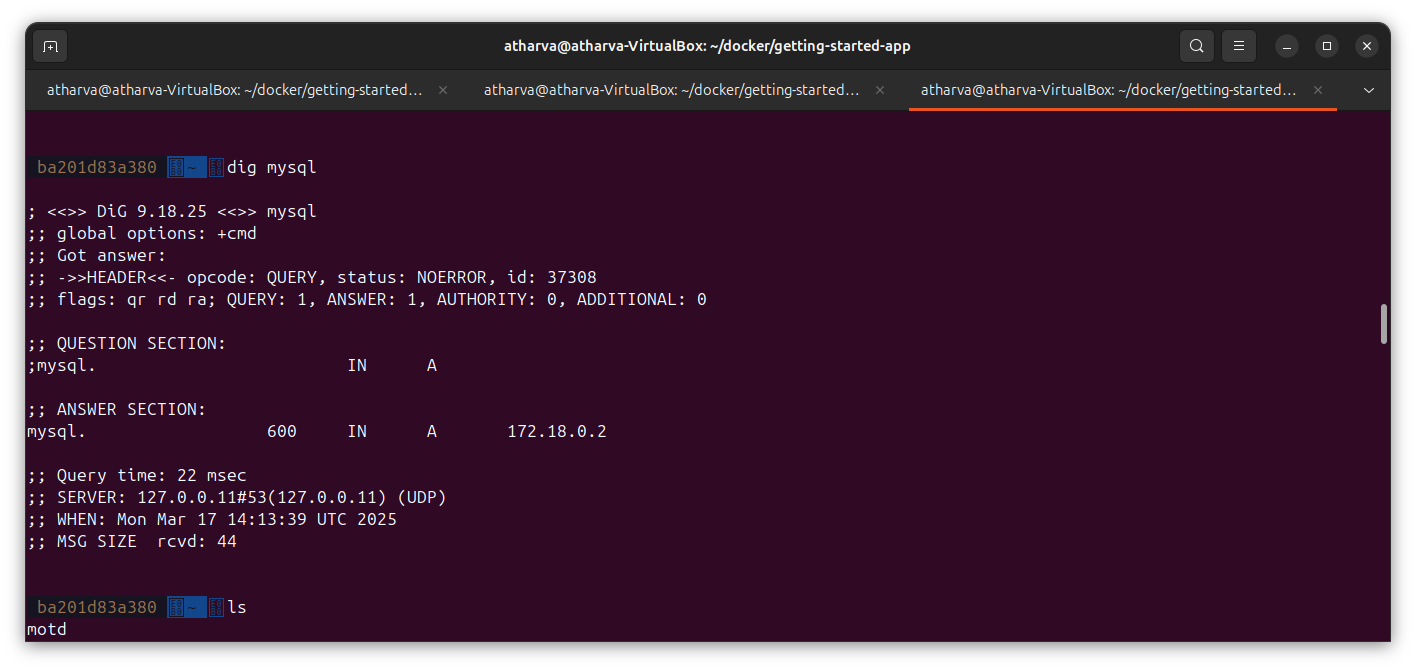
Start a new container using the nicolaka/netshoot image. Make sure to connect it to the same network.

Command: docker run -it --network todo-app nicolaka/netshoot



Inside the container, you're going to use the dig command, which is a useful DNS tool. You're going to look up the IP address for the hostname mysql.

Command : dig mysql



You can now start your dev-ready container.

1. Specify each of the previous environment variables, as well as connect the container to your app network. Make sure that you are in the getting-started-app directory when you run this command.

Command : docker run -dp 127.0.0.1:3000:3000 \

-w /app -v "$(pwd):/app" \

--network todo-app \

-e MYSQL\_HOST=mysql \

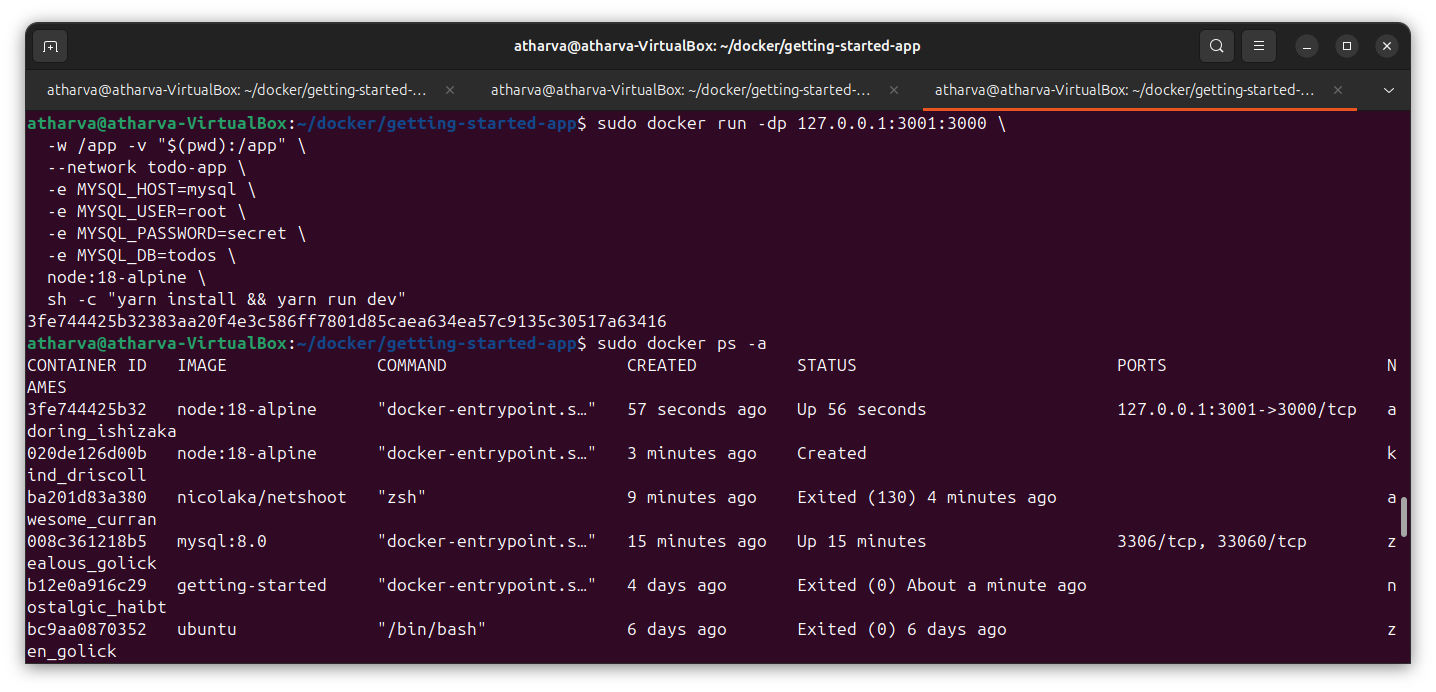
-e MYSQL\_USER=root \

-e MYSQL\_PASSWORD=secret \

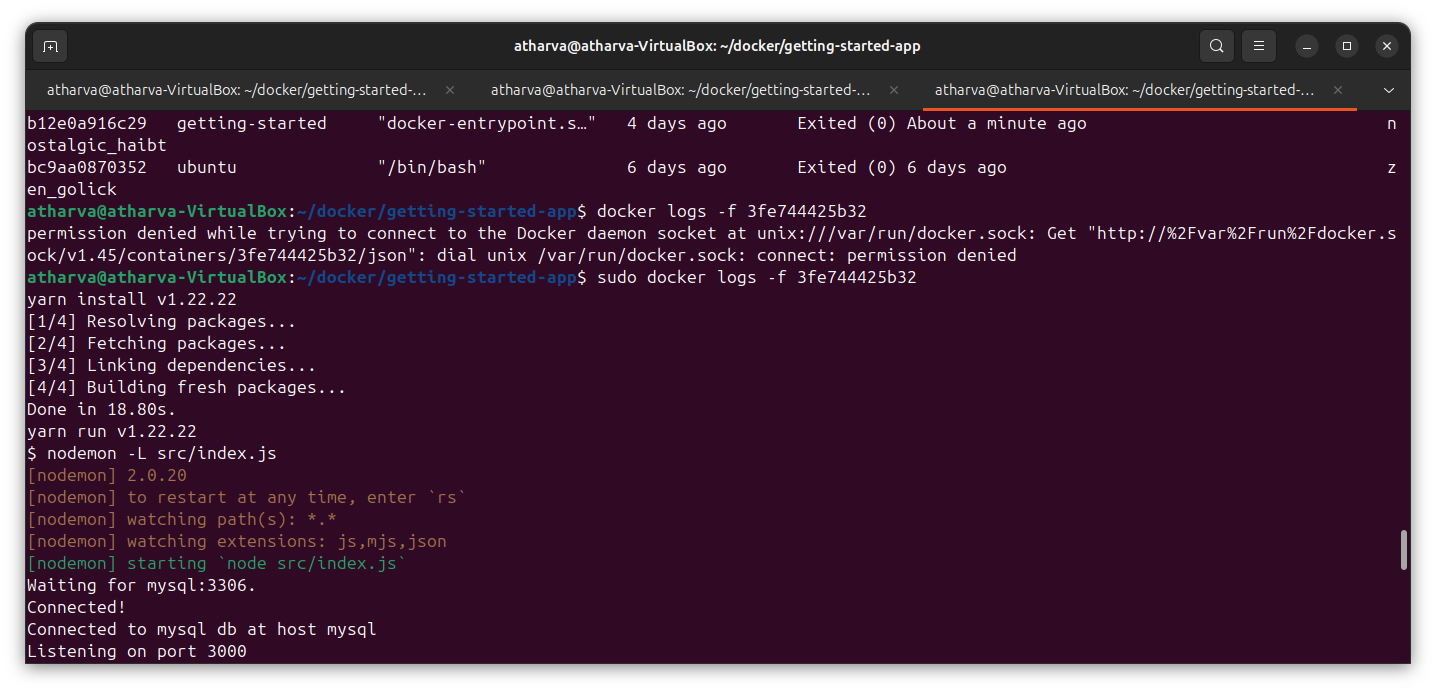
-e MYSQL\_DB=todos \

node:18-alpine \

sh -c "yarn install && yarn run dev"



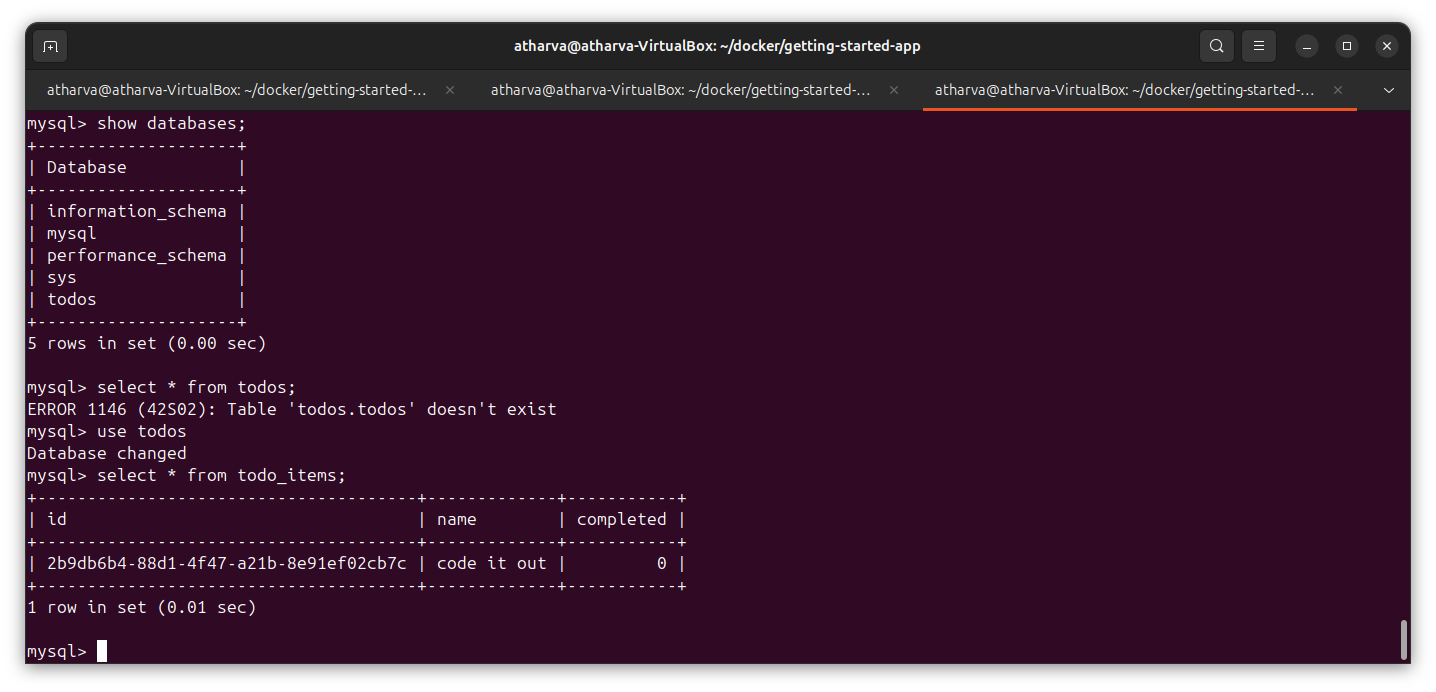
2. If you look at the logs for the container (docker logs -f <container-id>), you should see a message similar to the following, which indicates it's using the mysql database.



3. Open the app in your browser and add a few items to your todo list.

4. Connect to the mysql database and prove that the items are being written to the database. Remember, the password is secret.

$ docker exec -it <mysql-container-id> mysql -p todos



**Observation:** Docker made it simple to package, run, and update the app. Sharing it through Docker Hub was easy, and using multiple containers helped organize the app better. This method makes applications more portable, scalable, and easy to manage.