Lab # 11 Docker Lab

Background

In recent years, containerization has emerged as a transformative technology in the field of software development and deployment. With its ability to streamline application packaging, isolation, and deployment, containers have become an essential tool for modern software development teams. In this comprehensive guide, we will delve into the theory behind containerization, explore its key concepts, and demonstrate practical examples through a hands-on lab session.

1.1 What are Containers?
Containers are lightweight, portable, and self-contained units that package applications and their dependencies. Unlike traditional virtual machines, which require a separate operating system instance for each application, containers share the host operating system's kernel while providing isolated environments for applications.

1.2 Historical Context
The concept of containerization traces its roots back to UNIX chroot jails and FreeBSD jails. However, it gained widespread adoption with the development of Linux Containers (LXC) and later, Docker, which revolutionized container technology by providing user-friendly tools for container management.

Key Concepts of Containerization**

2.1 Namespaces
Namespaces provide process isolation within a container by creating separate namespaces for various system resources such as processes, networks, filesystems, and more. This isolation ensures that processes within a container are unaware of processes outside the container, enhancing security and resource management.

2.2 Control Groups (cgroups)
Control groups, or cgroups, allow fine-grained control over resource allocation and usage within containers. By limiting CPU, memory, disk I/O, and other resources, cgroups ensure that containers operate within specified resource constraints, preventing resource contention and ensuring consistent performance.

2.3 Docker and Container Orchestration
Docker, initially released in 2013, played a significant role in popularizing containerization by providing a user-friendly interface for building, running, and managing containers. Container orchestration tools such as Kubernetes further streamline container deployment and management at scale, automating tasks such as scaling, load balancing, and service discovery.

Practical Lab Session: Deploying Containers with Docker

Installation of docker Linux

step-by-step guide to install Docker on Linux:

1. Update Package Index:
Before installing Docker, it's a good practice to ensure that your system's package index is up-to-date. Run the following command to update the package index:

```
sudo apt update
```

2. Install Dependencies:
Docker requires a few dependencies to be installed. Install them using the following command:

```
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
```

3. Add Docker Repository Key:
To ensure that the Docker packages are from the official repository, add Docker's official GPG key:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. Add Docker Repository:
Add the Docker repository to your system's APT sources list:

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

5. Update Package Index (Again):
After adding the Docker repository, update the package index again:

```
sudo apt update
```

6. Install Docker CE (Community Edition):
Now, you can install Docker Community Edition (CE) using the following command:

```
sudo apt install -y docker-ce
```

7. Verify Docker Installation:
After the installation is complete, verify that Docker is installed correctly by running the following command:

```
docker --version
```

You should see the installed Docker version printed to the console.

8. Manage Docker as a Non-Root User (Optional):
If you want to use Docker as a non-root user, you can add your user to the `docker` group. This allows you to run Docker commands without using `sudo`.

```
sudo usermod -aG docker ${USER}
```

After running this command, you'll need to log out and log back in for the changes to take effect.

Note: If you didn't use the above command you will need to add **sudo** whenever you use the docker command

9. Start and Enable Docker Service:
If Docker is not started automatically after installation, you can start and enable the Docker service using the following commands:

```
sudo systemctl start docker
sudo systemctl enable docker
```

10. Test Docker Installation:
You can test Docker installation by running a simple Docker command:

```
docker run hello-world
```

This command will download a test image and run a container, which should print a "Hello from Docker!" message if Docker is installed correctly.

That's it! Docker is now successfully installed on your Linux system. You can start using Docker to create, deploy, and manage containers.

Lab Session

In this lab session students will gain practical experience deploying MySQL and Nginx containers using Docker commands.

lab session outline that covers Docker commands for deploying MySQL and Nginx containers:

Lab Session: Docker Container Deployment with MySQL and Nginx

Objective:
To familiarize students with Docker commands for deploying and managing containers, using MySQL and Nginx as examples.

Prerequisites:
- Basic understanding of Docker concepts and commands.
- Docker installed on each student's machine.

Lab Content:

Part 1: Deploying MySQL Container

1. **Introduction (5 minutes)**
~~- Briefly review Docker basics and containerization concepts.~~

2. **Download MySQL Image (5 minutes)**
   - Instruct students to pull the MySQL Docker image from Docker Hub:

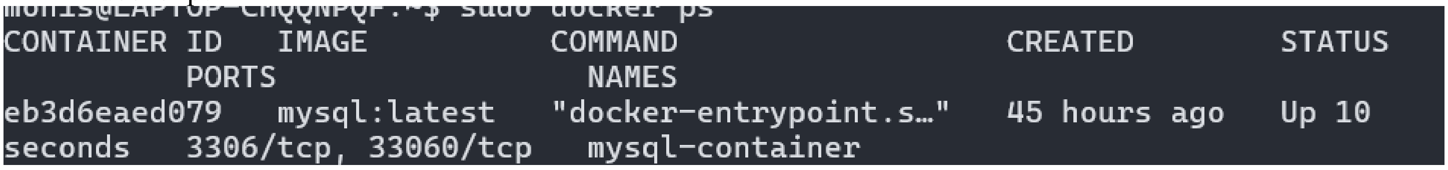
```
docker pull mysql:latest
```

3. **Run MySQL Container (10 minutes)**
   - Guide students to run a MySQL container with the following command:

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=password -d mysql:latest
```

   - Explain the significance of each parameter, such as `--name` for container naming and `-e` for environment variables.

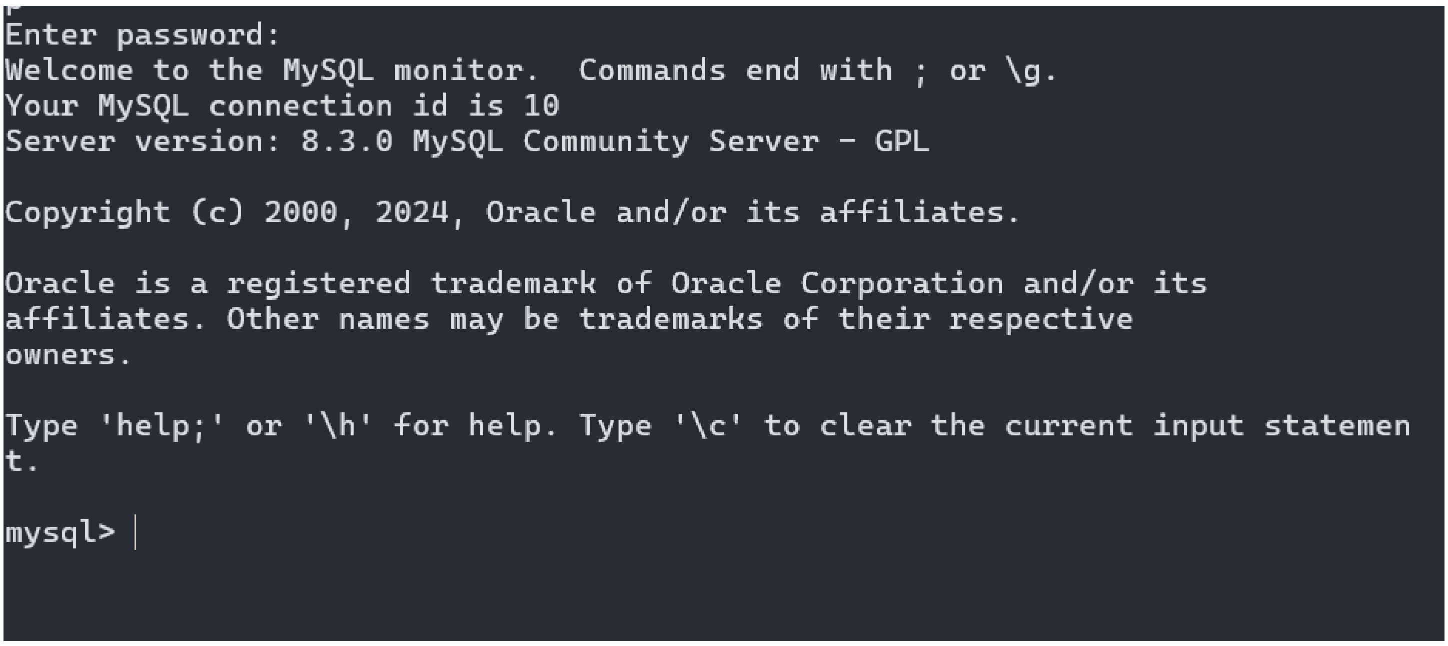
docker ps



```
monis@LAPTOP-CMQQNPQF:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS
          PORTS                 NAMES
eb3d6eaed079   mysql:latest   "docker-entrypoint.s…"   45 hours ago    Up 10
seconds   3306/tcp, 33060/tcp   mysql-container
```

4. **Access MySQL Container (5 minutes)**

docker exec -it mysql-container mysql -uroot -p



```
p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statemen
t.

mysql> 
```

   - Inside the container, demonstrate connecting to MySQL using the MySQL command-line client.

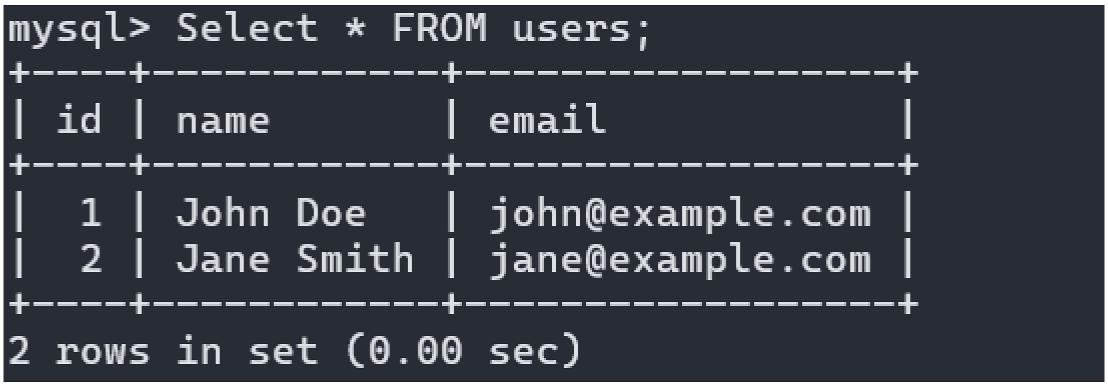
```
CREATE DATABASE test_db;

USE test_db

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(50)
);

INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com');
INSERT INTO users (name, email) VALUES ('Jane Smith', 'jane@example.com');
```

Now we can see if the data is inserted in the SQL database by running a simple command



```
mysql> Select * FROM users;
+----+------------+------------------+
| id | name       | email            |
+----+------------+------------------+
|  1 | John Doe   | john@example.com |
|  2 | Jane Smith | jane@example.com |
+----+------------+------------------+
2 rows in set (0.00 sec)
```

To exit just type the **'exit'** command

Part 2: Deploying Nginx Container

5. **Download Nginx Image (5 minutes)**
   - Instruct students to pull the Nginx Docker image from Docker Hub:

```
docker pull nginx:latest
```

6. **Run Nginx Container (10 minutes)**
   - Guide students to run an Nginx container with the following command:

```
docker run --name nginx-container -p 8080:80 -d nginx:latest
```

   - Explain the `-p` flag for port mapping, allowing access to Nginx on port 8080 of the host machine.

7. **Access Nginx Container (5 minutes)**
   - Show students how to access the Nginx container by opening a web browser and navigating to `http://localhost:8080`.

Part 3: pull and run a Django Docker image from Docker Hub:

Web Application

Here's a simple Django application code that you can deploy on the Django container to demonstrate integration between Nginx, MySQL, and Django containers:
For starters installation is necessary run the following command to get started

pip install Django (Note pip3 would also work in this case)

1. Create a Django Project:

First, create a new Django project using the following command:

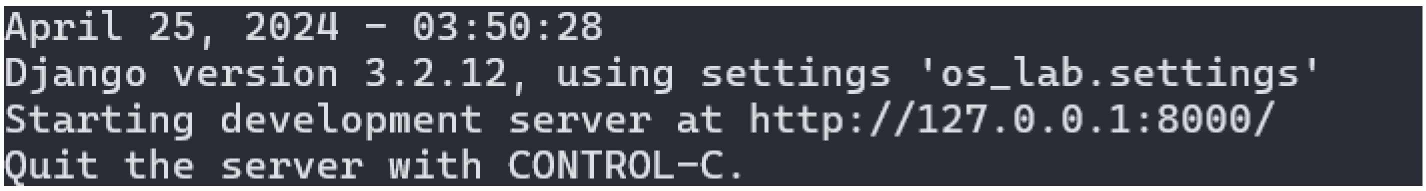
django-admin startproject {myproject}
replace the {myproject} with your project name (obviously)

2. Create a Django App:

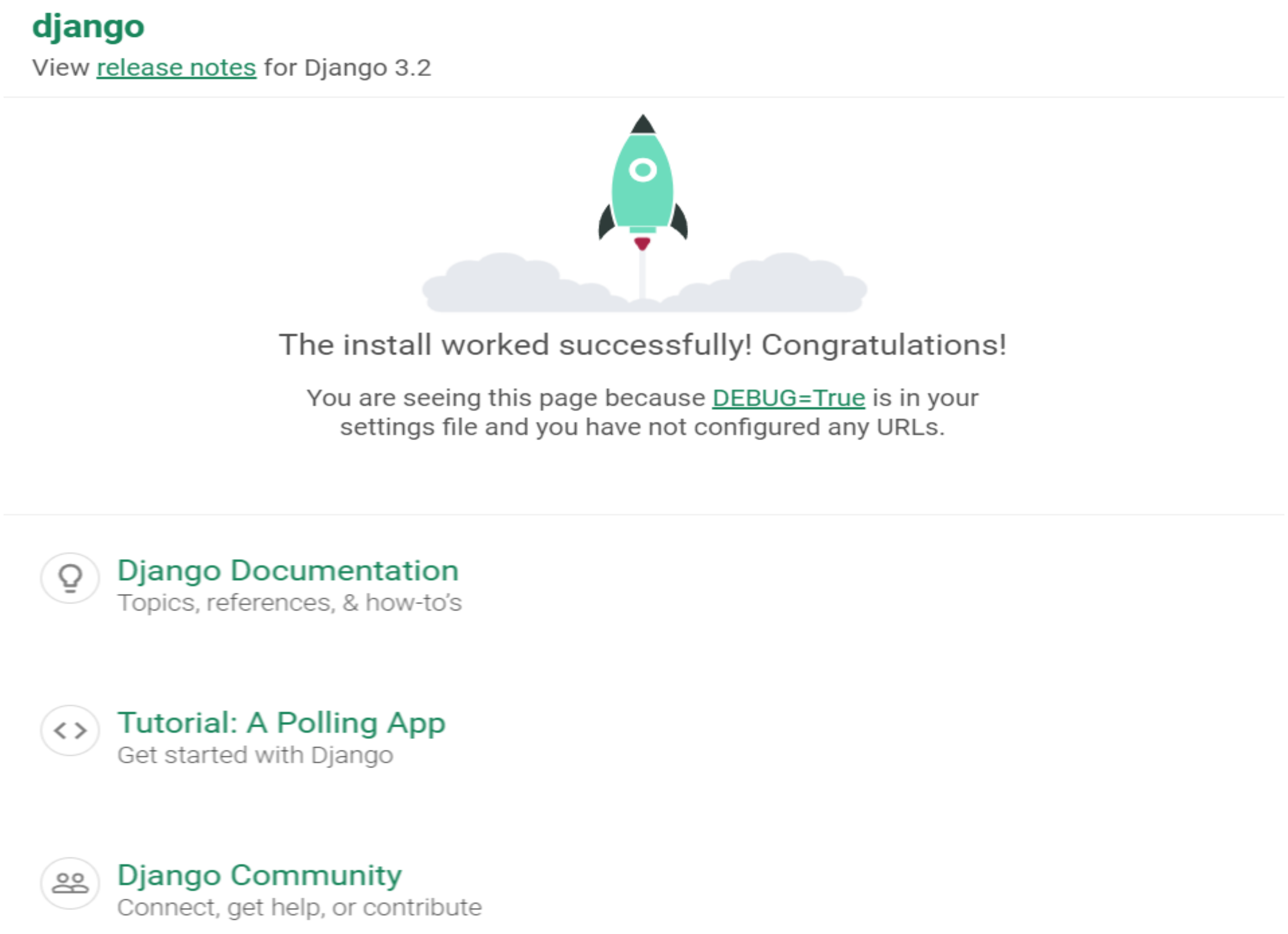
Inside your Django project directory (`myproject`), create a new Django app using the following command:

```
cd myproject
python manage.py startapp myapp
```

You should see a window something like this when you open the browser and type the ip provided by Django



```
April 25, 2024 - 03:50:28
Django version 3.2.12, using settings 'os_lab.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```



django

View release notes for Django 3.2

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation
Topics, references, & how-to's

Tutorial: A Polling App
Get started with Django

Django Community
Connect, get help, or contribute

3 Create your first polls up
You can use the already created app (link: https://github.com/monisj/First_Django_Project) its not the best as it made in a very short amount of time (15 minutes) but if you are curious this is an easy to use framework built mostly on python and will get you up to speed on web development
https://docs.djangoproject.com/en/5.0/intro/tutorial01/ Link to tutorial it's the same thing

Once cloned you can see your project via this command
python3 manage.py runserver

type the IP 127.0.0.1:8000 to see the website.
To access the admin page use this link
http://127.0.0.1:8000/admin/.
Id would be monis

And password would be dellwireless (you can change this as well)

Once you have this Django application running locally, you can deploy it on the Django container along with Nginx and MySQL containers to demonstrate the integration between the three components.

Part 4: Cleanup and Conclusion

8. **Cleanup (5 minutes)**
   - Instruct students to stop and remove the running containers:
     ```
     docker stop mysql-container nginx-container
     docker rm mysql-container nginx-container
     ```

9. **Discussion and Q&A (10 minutes)**
   - Lead a discussion on students' experiences with deploying and managing containers.
   - Address any questions or concerns raised by the students.