How to Install Docker Compose on Ubuntu 24.04 [Step-by-Step]

by Winnie Ondara May 24th, 2024 DOCKER

In this tutorial, you will learn how to install Docker Compose on Ubuntu 24.04. In addition, you will see how you can leverage Docker Compose to create multiple applications running in separate containers.

What is Docker?

Docker is an open-source containerization platform that lets you seamlessly build, share, deploy, and orchestrate applications anywhere - be it on Linux, Windows, Mac or any other computing environment.

What is Docker Compose?

Built on top of Docker Engine, Docker Compose is a tool for defining and running multi-container applications.

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

What does Docker Compose do?

Docker Compose creates containers and other resources defined within the Compose file written in YAML format. With the compose file in place, you can create and launch your multi-container application using the `docker compose up` command, as we shall see later.

How to use Docker Compose?

Docker Compose is used for providing simplified control when managing multi-container applications. It simplifies the complexities involved in running multiple containers with interlinked services. In addition, it also makes it easier to collect logs and debug issues in multiple containers compared to Docker.

Ready to supercharge your Docker infrastructure? Scale effortlessly and enjoy flexible storage with Cherry Servers bare metal or virtual servers. Eliminate infrastructure headaches with free 24/7 technical support, pay-as-you-go pricing, and global availability.

Get Started Today

Prerequisites

Before you set sail, ensure you have the following in place:

- An instance of Ubuntu 24.04 with SSH access;
- A sudo user configured on the server;
- Docker installed on your Linux instance. Check out our guide on how to install Docker on Ubuntu 22.04. or how to install Docker on Ubuntu 24.04 if you are using Ubuntu 24.04.

How to install Docker Compose on Ubuntu 24.04

The below five steps will show you how to install Docker Compose on Ubuntu 24.04. I've included how to update package lists, install Docker Compose, deploy LAMP stack using Docker Compose, run Docker Compose, and learn how to pause, stop, and remove containers.

Step 1: Update Package Lists

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

First, log into your server and update the package lists as follows.

```
sudo apt update
```

This command downloads and updates package information defined in the `/etc/apt/sources.list` file and `/etc/apt/source.list.d` directory.

Step 2: Install Docker Compose

Docker Compose is hosted on Ubuntu repositories and can easily be installed by running the command:

```
sudo apt install docker-compose-plugin -y
```

However, the version installed from the repositories does not give you the latest version of Docker Compose. To get the most updated version of Docker Compose, download it from its official GitHub repository. Currently, the latest release is Docker Compose 2.2.27.

Before downloading the latest Docker Compose plugin binary, create a `cli-plugins` directory in your home folder.

```
$ mkdir -p ~/.docker/cli-plugins/
```

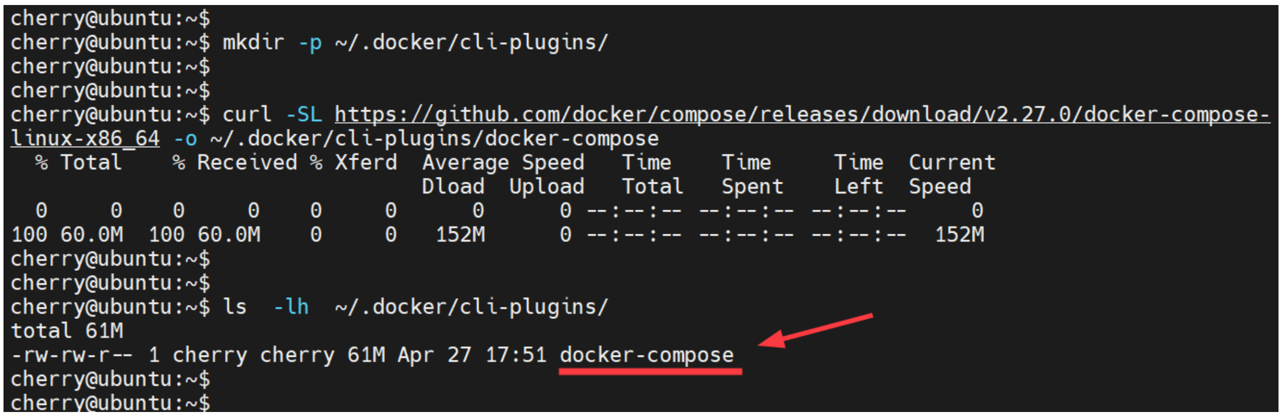
Next, download the Docker compose plugin file to the `~/.docker/cli-plugins/` directory using the `curl` command as shown.

```
 curl -SL https://github.com/docker/compose/releases/download/v2.2.27/dock
```

Run the `ls` command to verify the presence of the `docker-compose` file.

```
 ls -lh ~/.docker/cli-plugins
```

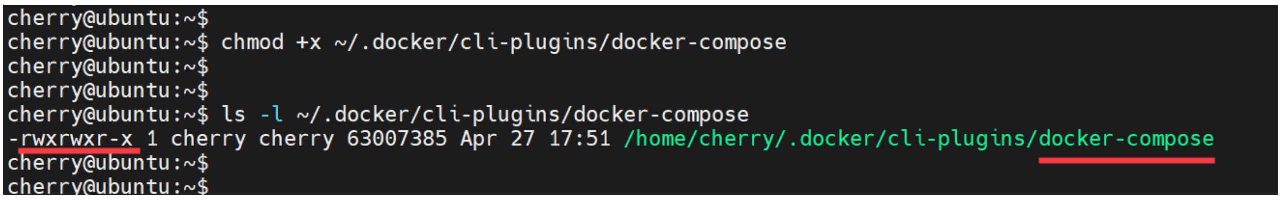
Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF



```
cherry@ubuntu:~$
cherry@ubuntu:~$ mkdir -p ~/.docker/cli-plugins/
cherry@ubuntu:~$
cherry@ubuntu:~$
cherry@ubuntu:~$ curl -SL https://github.com/docker/compose/releases/download/v2.27.0/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 60.0M  100 60.0M    0     0   152M      0 --:--:-- --:--:-- --:--:--  152M
cherry@ubuntu:~$
cherry@ubuntu:~$
cherry@ubuntu:~$ ls  -lh  ~/.docker/cli-plugins/
total 61M
-rw-rw-r-- 1 cherry cherry 61M Apr 27 17:51 docker-compose
cherry@ubuntu:~$
cherry@ubuntu:~$
```

Next, assign execute permissions to the file using the `chmod` command.

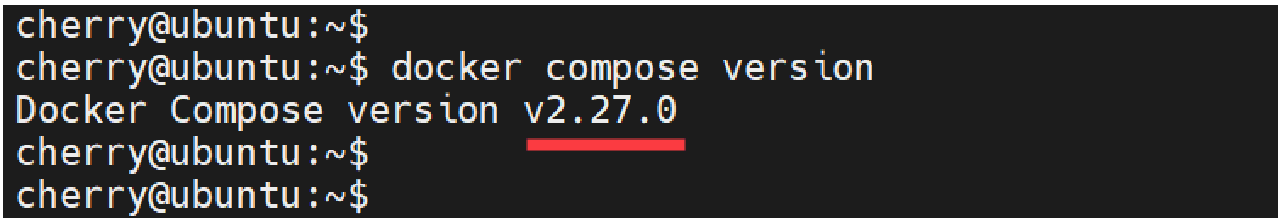
```
chmod +x  ~/.docker/cli-plugins/docker-compose
```



```
cherry@ubuntu:~$
cherry@ubuntu:~$ chmod +x ~/.docker/cli-plugins/docker-compose
cherry@ubuntu:~$
cherry@ubuntu:~$
cherry@ubuntu:~$ ls -l ~/.docker/cli-plugins/docker-compose
-rwxrwxr-x 1 cherry cherry 63007385 Apr 27 17:51 /home/cherry/.docker/cli-plugins/docker-compose
cherry@ubuntu:~$
cherry@ubuntu:~$
```

To confirm the installation was successful, check the Docker compose version.

```
docker compose version
```



```
cherry@ubuntu:~$
cherry@ubuntu:~$ docker compose version
Docker Compose version v2.27.0
cherry@ubuntu:~$
cherry@ubuntu:~$
```

Also read: How to uninstall Docker

Step 3: Deploy LAMP stack using Docker Compose

With Docker Compose already installed, we will test it by deploying a LAMP stack application. LAMP (Linux Apache Mysql PHP) is a popular web hosting stack used to host websites.

To get started, create the project directory and navigate into it. In this case, our project directory is `my-demo-app` .

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

```
mkdir ~/my-demo-app

cd ~/my-demo-app
```

Next, create a `docker-compose.yml` file.

```
$ nano docker-compose.yml
```

Add the following lines of code:

```
version: '2'

services:
  apache:
    image: php:apache
    container_name: apache-app
    ports:
      - "8080:80"
    volumes:
      - ./src:/var/www/html
    depends_on:
      - mysql
    links:
      - mysql


  mysql:
    image: mysql:8.0
    container_name: mysql-app
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: mysql_root_password
      MYSQL_DATABASE: mydatabase
      MYSQL_USER: myuser
      MYSQL_PASSWORD: myuser_password
    ports:
      - "3306:3306"
    volumes:
      - ./mysql-data:/var/lib/mysql
```

The configuration above sets up two services:

Explore our developer-friendly HTML to PDF API   Printed using PDFCrowd   HTML to PDF

apache: This is the Apache web service with PHP support. It runs as a container called `apache-app` created from the `php:apache` image. The webserver will serve files from the `.src` directory which will be created in the project folder. The webserver will run on port 80 on the container which will be mapped on port 8080 on the host system.

mysql: This is the MySQL database service with a specified root password, database name, MySQL username, and password. The service runs as a container called `mysql-app` on port 3306 which is mapped on port 3306 on the host system.

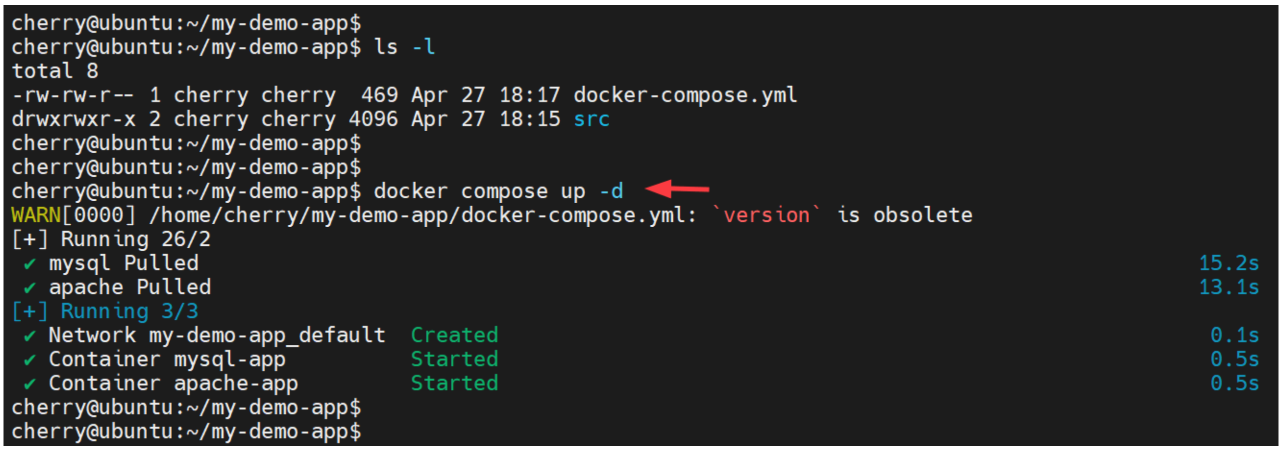
Save the changes in the YAML file and exit.

Step 4: Run Docker compose

Having defined the services to be created, the next step is to start the LAMP stack using Docker Compose. To do so, run the following `docker compose` command.

```
$ docker compose up -d
```

The command downloads the specified Docker images, in this case `php:apache` and `mysql:8.0` . Afterward, it creates the `mysql-app` and `apache-app` containers and runs them in detached mode using the `-d` flag. Additionally, it creates a default network for communication between the services, the `mysql-data` persistent directory for the database, and the `src` web directory for storing website files.

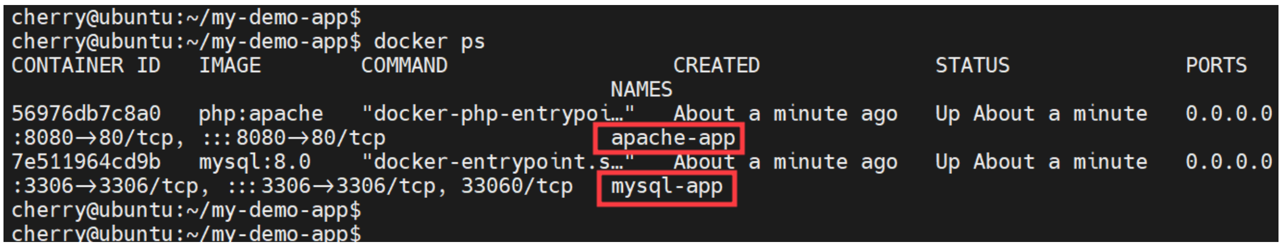


```
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ ls -l
total 8
-rw-rw-r-- 1 cherry cherry  469 Apr 27 18:17 docker-compose.yml
drwxrwxr-x 2 cherry cherry 4096 Apr 27 18:15 src
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ docker compose up -d
WARN[0000] /home/cherry/my-demo-app/docker-compose.yml: `version` is obsolete
[+] Running 26/2
 ✔ mysql Pulled                                                          15.2s
 ✔ apache Pulled                                                         13.1s
[+] Running 3/3
 ✔ Network my-demo-app_default  Created                                   0.1s
 ✔ Container mysql-app          Started                                   0.5s
 ✔ Container apache-app         Started                                   0.5s
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
```

To verify currently running containers, run the `docker ps` command:

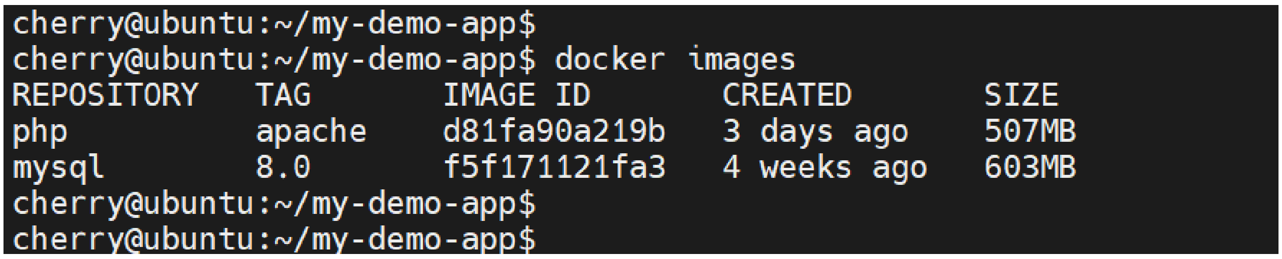
```
docker ps
```

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF


```
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED              STATUS              PORTS
                                                NAMES
56976db7c8a0   php:apache   "docker-php-entrypoi…"   About a minute ago   Up About a minute   0.0.0.0
:8080->80/tcp, :::8080->80/tcp                  apache-app
7e511964cd9b   mysql:8.0    "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0
:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql-app
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
```

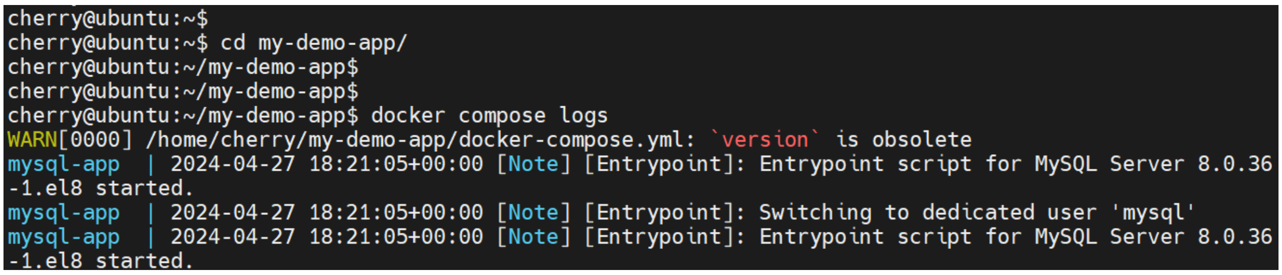
You can also list the existing images as shown.

```
docker images
```


```
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ docker images
REPOSITORY   TAG      IMAGE ID       CREATED       SIZE
php          apache   d81fa90a219b   3 days ago    507MB
mysql        8.0      f5f171121fa3   4 weeks ago   603MB
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
```

You can check the logs generated by the running containers using the `docker compose logs` command.

```
docker compose logs
```


```
cherry@ubuntu:~$
cherry@ubuntu:~$ cd my-demo-app/
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ docker compose logs
WARN[0000] /home/cherry/my-demo-app/docker-compose.yml: `version` is obsolete
mysql-app  | 2024-04-27 18:21:05+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.36
-1.el8 started.
mysql-app  | 2024-04-27 18:21:05+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-app  | 2024-04-27 18:21:05+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.36
-1.el8 started.
```

At this point, we will create a simple webpage to test the web server. So, navigate to the web directory called `src`, which is located in the project folder.

```
cd src
```

Create an `index.html` file.

```
nano index.html
```

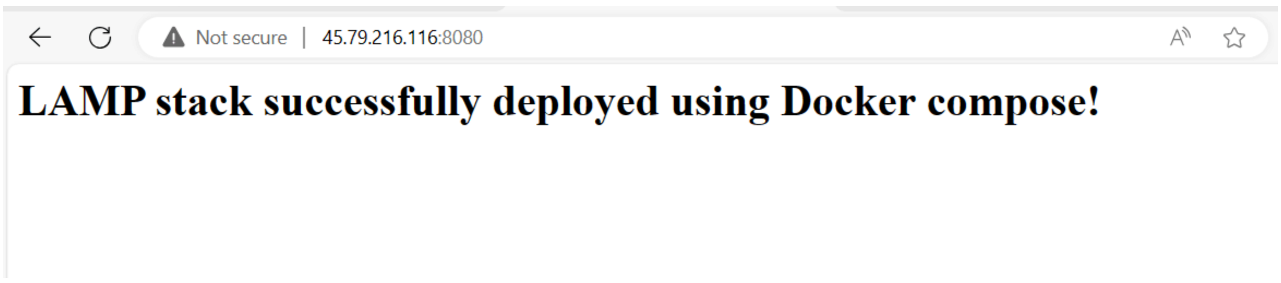
Here's some sample code you can use to define the welcome page.

```
<!DOCTYPE html>
<html>
<body>
     <h1>LAMP stack successfully deployed using Docker compose!</h1>
</body>
</html>
```

Save the changes and exit the file. Launch your browser and head over to the server's URL as shown:

```
http://server-ip:8080
```

You should get the webpage below, proof that the webserver is running as expected.



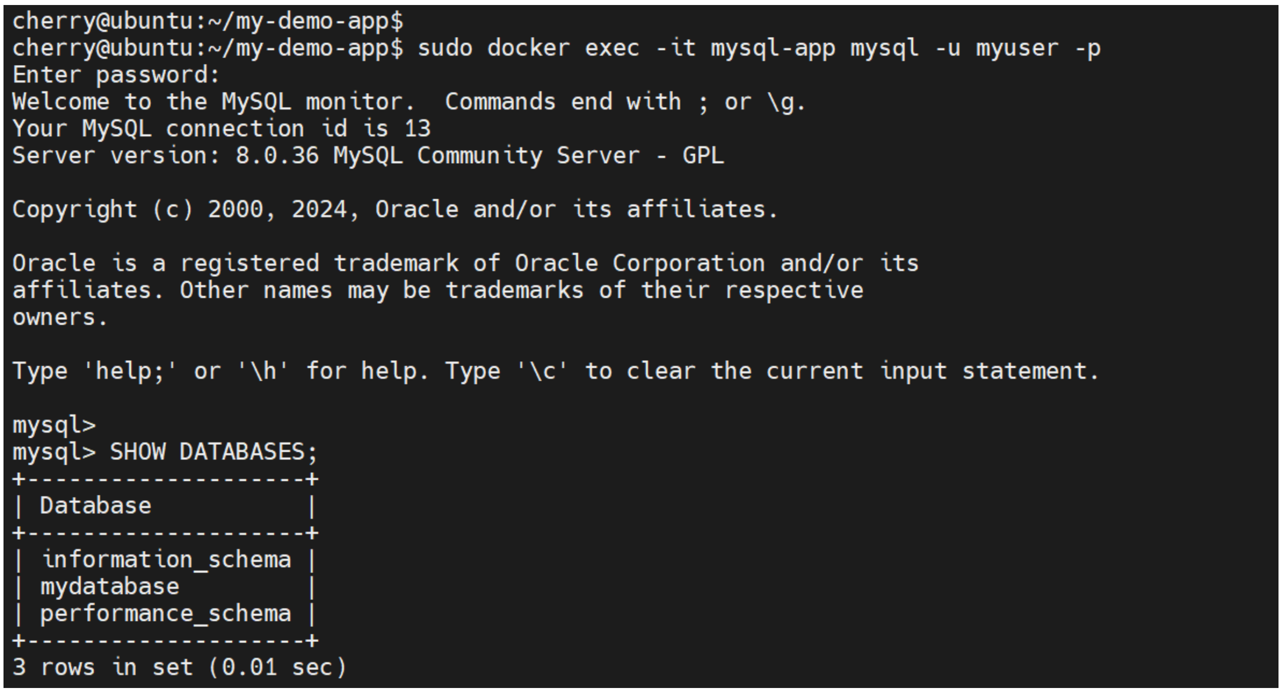
LAMP stack successfully deployed using Docker compose!

To connect to the `mysql-app` database container using the MySQL user specified in the YAML file, run the command:

```
sudo docker exec -it mysql-app mysql -u myuser -p
```

Provide the user's password and hit ENTER to access the MySQL shell as shown.

Explore our developer-friendly HTML to PDF API   Printed using PDFCrowd   HTML to PDF



```
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ sudo docker exec -it mysql-app mysql -u myuser -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.36 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 statement.

mysql>
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydatabase         |
| performance_schema |
+--------------------+
3 rows in set (0.01 sec)
```

To exit the shell and the container by extension, type `exit` and hit ENTER.

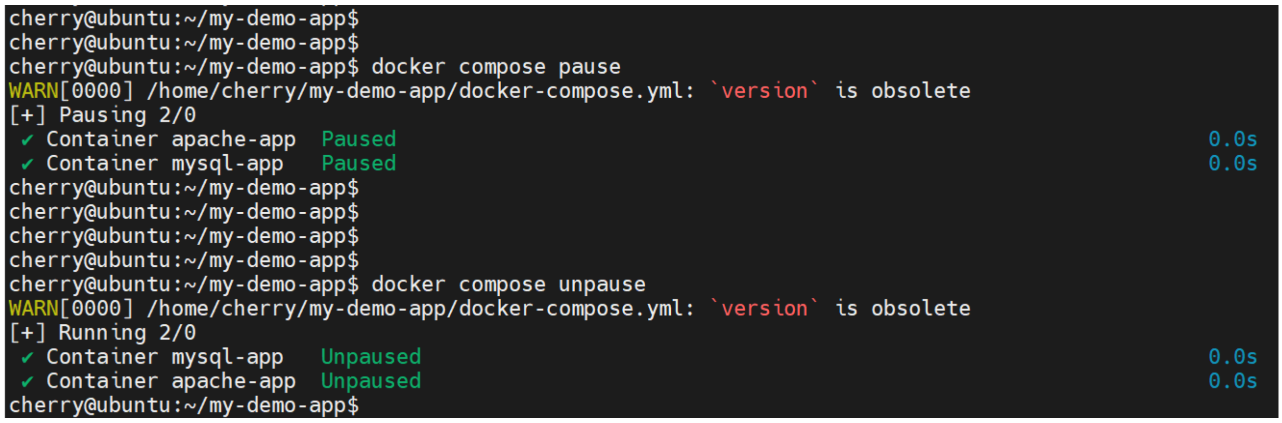
Step 5: Pause, stop, and remove containers

To pause the Docker compose environment execution without altering the state of the containers, use the command:

```
docker compose pause
```

To resume execution or unpause the environment, run the command:

```
docker compose unpause
```



```
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ docker compose pause
WARN[0000] /home/cherry/my-demo-app/docker-compose.yml: `version` is obsolete
[+] Pausing 2/0
 ✔ Container apache-app  Paused                                                   0.0s
 ✔ Container mysql-app   Paused                                                   0.0s
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ docker compose unpause
WARN[0000] /home/cherry/my-demo-app/docker-compose.yml: `version` is obsolete
[+] Running 2/0
 ✔ Container mysql-app   Unpaused                                                 0.0s
 ✔ Container apache-app  Unpaused                                                 0.0s
cherry@ubuntu:~/my-demo-app$
```

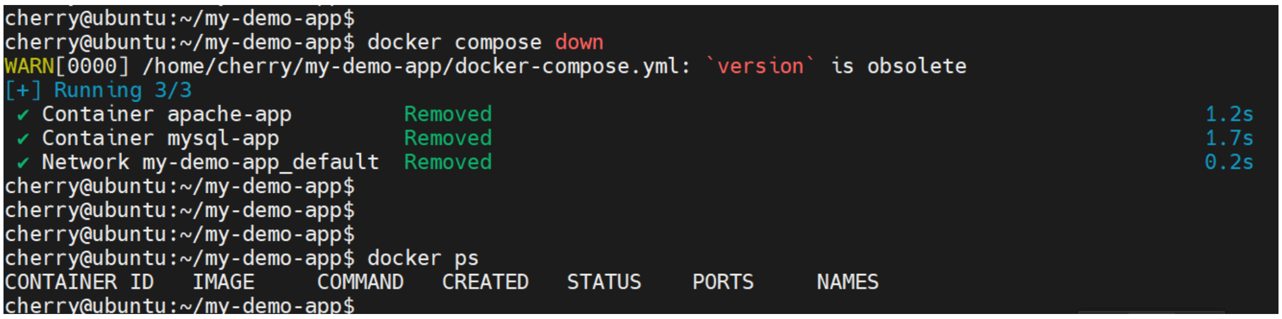
To stop the environment without removing the containers, run:

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

```
docker compose stop
```

To stop and remove containers and the default network created by Docker Compose, run the command:

```
docker compose down
```



```
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ docker compose down
WARN[0000] /home/cherry/my-demo-app/docker-compose.yml: `version` is obsolete
[+] Running 3/3
 ✔ Container apache-app         Removed                                              1.2s
 ✔ Container mysql-app          Removed                                              1.7s
 ✔ Network my-demo-app_default  Removed                                              0.2s
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$
cherry@ubuntu:~/my-demo-app$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
cherry@ubuntu:~/my-demo-app$
```

Explore how web hosting service provider Debesis improved its service quality, performance, and reliability by migrating to Cherry Servers' bare-metal servers.

"Cherry Servers engineers always help when we need them, while their customer service quality is a blast!"

Read Their Success Story Here

Also read: How to start Docker Daemon

Conclusion

In this guide, you have learned how to install Docker Compose on Ubuntu 24.04. You have also seen how to start a multi-container application, and stop and delete all the containers and resources associated with the application.

Check out the official documentation for a comprehensive coverage of all the Docker Compose commands.

Explore our developer-friendly HTML to PDF API   Printed using PDFCrowd   HTML to PDF

Winnie Ondara

Linux System Administrator

Winnie is a seasoned Linux Systems administrator, currently specializing in writing technical Linux tutorials. With over seven years of experience in deploying and working with major Linux distributions such as Ubuntu, Debian, RHEL, OpenSUSE, and ArchLinux, she has written detailed and well-written "How to" Linux guides and tutorials. Winnie holds a Bachelor's Degree in Computer Science from Masinde Muliro University, Kenya and resides in Nairobi, Kenya. She is an expert in authoring Linux and DevOps topics involving Docker, Ansible, and Kubernetes. She currently works as a freelance technical writer and consultant. In her previous roles, she worked in the capacity of an IT support specialist and Linux administrator. Her key roles included offering level 1 and 2 support to both in-house and remote staff and managing and monitoring Linux servers.

Share this article

Twitter Facebook LinkedIn

Start Building Now

Deploy your new Cloud VPS server in 5 minutes starting from $5.83 / month.

Check Available Servers

Related Articles

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

How to Install Nextcloud AIO on Docker | Step-by-Step

Apr 10, 2024 • by Goodness Chris-Ugari

This tutorial will show you how to install Nextcloud using Nextcloud AIO Docker image to help significantly reduce configuration steps.

Portainer Tutorial: How to Update [and Restart] Portainer

May 15, 2024 • by Winnie Ondara

This tutorial demonstrates how to manage Portainer: how to update Portainer, how to restart Portainer, and update existing containers.

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF

Docker Swarm vs. Kubernetes – What are the Subtle Differences?

Apr 14, 2020 • by Marius Rimkus

Kubernetes and Docker Swarms are essential tools used to deploy containers inside a cluster. Learn how they differ and when to use them

Explore our developer-friendly HTML to PDF API Printed using PDFCrowd HTML to PDF