CONTENTS

Tutorial Series: Getting Started With Cloud Computing

26/39 How To Install and Use D...

27/39 How To Use docker exec ...

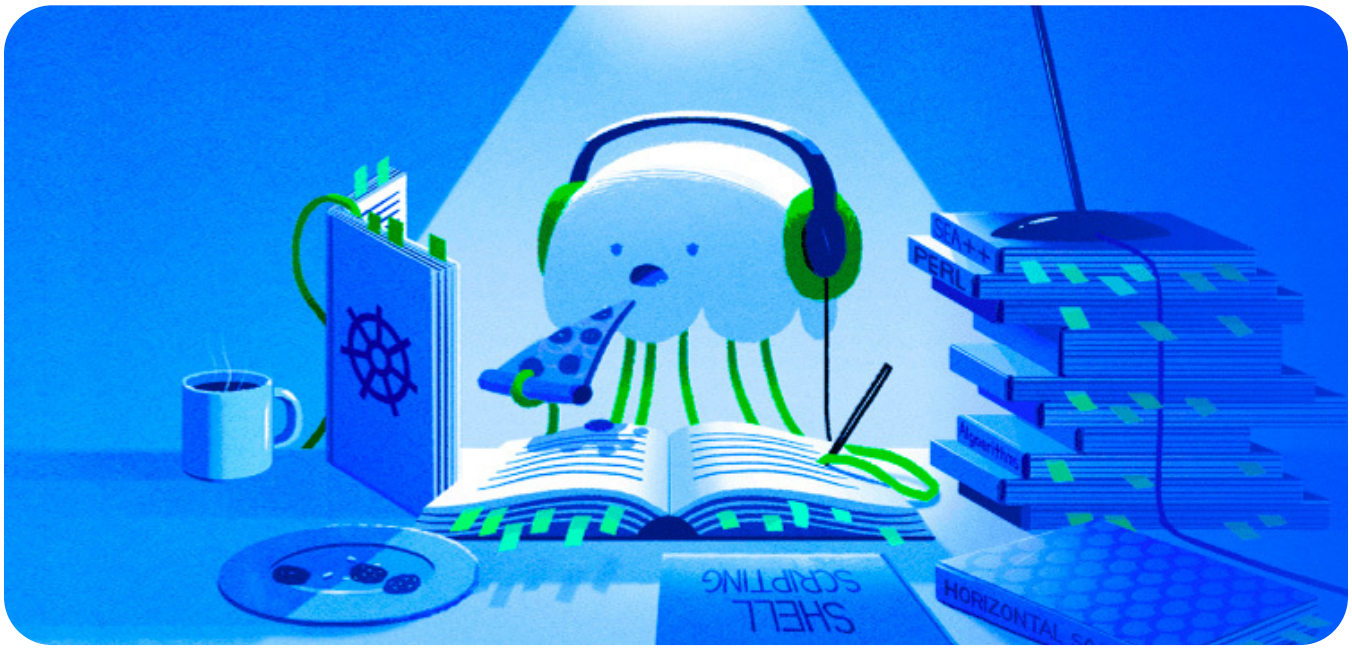
// Tutorial //

How To Install and Use Docker Compose on Ubuntu 22.04

Published on April 26, 2022

Docker Ubuntu Ubuntu 22.04

By Tony Tran and Erika Heidi



Not using Ubuntu 22.04?
Choose a different version or distribution.

Ubuntu 22.04

Introduction

Docker simplifies the process of managing application processes in containers. While containers are similar to virtual machines in certain ways, they are more lightweight and resource-friendly. This allows developers to break down an application environment into multiple isolated services.

For applications depending on several services, orchestrating all the containers to start up, communicate, and shut down together can quickly become unwieldy. Docker Compose is a tool that allows you to run multi-container application environments based on definitions set in a YAML file. It uses service definitions to build fully customizable environments with multiple containers that can share networks and data volumes.

In this guide, you'll demonstrate how to install Docker Compose on an Ubuntu 22.04 server and how to get started using this tool.

Prerequisites

To follow this article, you will need:

- Access to an Ubuntu 22.04 local machine or development server as a non-root user with sudo privileges. If you're using a remote server, it's advisable to have an active firewall installed. To set these up, please refer to our Initial Server Setup Guide for Ubuntu 22.04.
- Docker installed on your server or local machine, following **Steps 1 and 2** of How To Install and Use Docker on Ubuntu 22.04.

Note: Starting with Docker Compose v2, Docker has migrated towards using the `compose` CLI plugin command, and away from the original `docker-compose` as documented in our previous Ubuntu 20.04 version of this tutorial. While the installation differs, in general the actual usage involves dropping the hyphen from `docker-compose` calls to become `docker compose`. For full compatibility details, check the official Docker documentation on command compatibility between the new `compose` and the old `docker-compose`.

Step 1 – Installing Docker Compose

To make sure you obtain the most updated stable version of Docker Compose, you'll download this software from its official Github repository.

First, confirm the latest version available in their releases page. At the time of this writing, the most current stable version is `2.3.3`.

Use the following command to download:

```
$ mkdir -p ~/.docker/cli-plugins/
$ curl -SL https://github.com/docker/compose/releases/download/v 2.3.3 /docke
```

Copy

Next, set the correct permissions so that the `docker compose` command is executable:

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

Copy

To verify that the installation was successful, you can run:

```
$ docker compose version
```

Copy

You'll see output similar to this:

```
Output
Docker Compose version v 2.3.3
```

Docker Compose is now successfully installed on your system. In the next section, you'll see how to set up a `docker-compose.yml` file and get a containerized environment up and running with this tool.

Step 2 – Setting Up a File

To demonstrate how to set up a `docker-compose.yml` file and work with Docker Compose, you'll create a web server environment using the official Nginx image from Docker Hub, the public Docker registry. This containerized environment will serve a single static HTML file.

Start off by creating a new directory in your home folder, and then moving into it:

```
$ mkdir ~/compose-demo
$ cd ~/compose-demo
```

Copy

In this directory, set up an application folder to serve as the document root for your Nginx environment:

```
$ mkdir app
```

Copy

Using your preferred text editor, create a new `index.html` file within the `app` folder:

```
$ nano app/index.html
```

Copy

Place the following content into this file:

~/compose-demo/app/index.html

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Docker Compose Demo</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/kognise/water.cs
</head>
<body>

    <h1>This is a Docker Compose Demo Page.</h1>
    <p>This content is being served by an Nginx container.</p>

</body>
</html>
```

Copy

Save and close the file when you're done. If you are using `nano`, you can do that by typing `CTRL+X`, then `Y` and `ENTER` to confirm.

Next, create the `docker-compose.yml` file:

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

Copy

Insert the following content in your `docker-compose.yml` file:

docker-compose.yml

```
version: '3.7'
services:
  web:
    image: nginx:alpine
    ports:
      - "8000:80"
    volumes:
      - ./app:/usr/share/nginx/html
```

Copy

The `docker-compose.yml` file typically starts off with the `version` definition. This will tell Docker Compose which configuration version you're using.

You then have the `services` block, where you set up the services that are part of this environment. In your case, you have a single service called `web`. This service uses the `nginx:alpine` image and sets up a port redirection with the `ports` directive. All requests on port `8000` of the **host** machine (the system from where you're running Docker Compose) will be redirected to the `web` container on port `80`, where Nginx will be running.

The `volumes` directive will create a shared volume between the host machine and the container. This will share the local `app` folder with the container, and the volume will be located at `/usr/share/nginx/html` inside the container, which will then overwrite the default document root for Nginx.

Save and close the file.

You have set up a demo page and a `docker-compose.yml` file to create a containerized web server environment that will serve it. In the next step, you'll bring this environment up with Docker Compose.

Step 3 – Running Docker Compose

With the `docker-compose.yml` file in place, you can now execute Docker Compose to bring your environment up. The following command will download the necessary Docker images, create a container for the `web` service, and run the containerized environment in background mode:

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

Copy

Docker Compose will first look for the defined image on your local system, and if it can't locate the image it will download the image from Docker Hub. You'll see output like this:

```
Output
Creating network " compose-demo_default " with the default driver
Pulling web (nginx:alpine)...
alpine: Pulling from library/nginx
cbdbe7a5bc2a: Pull complete
10c113fb0c77: Pull complete
9ba64393807b: Pull complete
c829a9c40ab2: Pull complete
61d685417b2f: Pull complete
Digest: sha256:57254039c6313fe8c53f1acbf15657ec9616a813397b74b063e32443427c55
Status: Downloaded newer image for nginx:alpine
Creating  compose-demo_web_1  ... done
```

Note: If you run into a permission error regarding the Docker socket, this means you skipped Step 2 of How To Install and Use Docker on Ubuntu 22.04. Going back and completing that step will enable permissions to run docker commands without `sudo`.

Your environment is now up and running in the background. To verify that the container is active, you can run:

```
$ docker compose ps
```

Copy

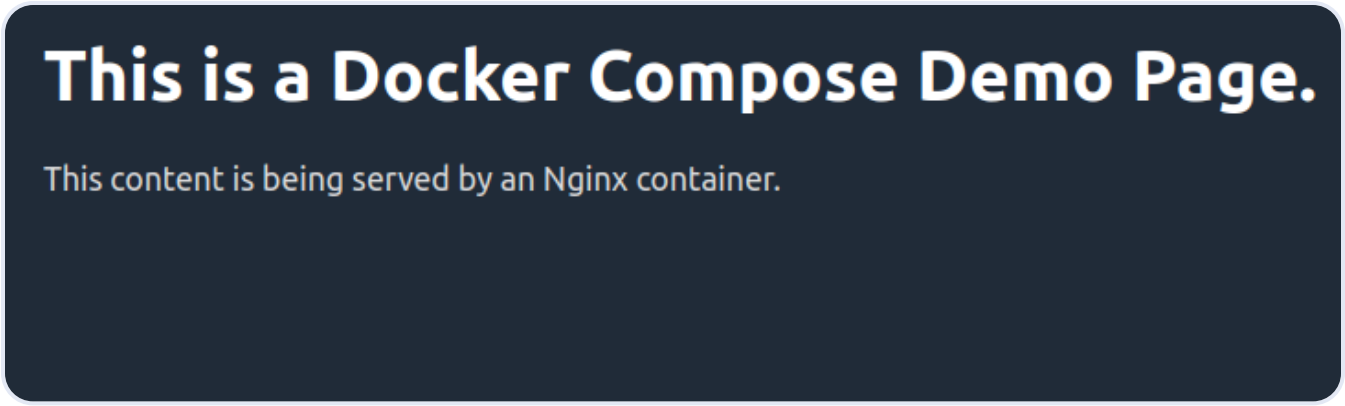
This command will show you information about the running containers and their state, as well as any port redirections currently in place:

```
Output
       Name                      Command               State          Ports
-------------------------------------------------------------------------------
 compose-demo_web_1    /docker-entrypoint.sh ngin ...   Up      0.0.0.0:8000->
```

You can now access the demo application by pointing your browser to either `localhost:8000` if you are running this demo on your local machine, or

`your_server_domain_or_IP :8000` if you are running this demo on a remote server.

You'll see a page like this:



The shared volume you've set up within the `docker-compose.yml` file keeps your `app` folder files in sync with the container's document root. If you make any changes to the `index.html` file, they will be automatically picked up by the container and thus reflected on your browser when you reload the page.

In the next step, you'll see how to manage your containerized environment with Docker Compose commands.

Step 4 – Getting Familiar with Docker Compose Commands

You've seen how to set up a `docker-compose.yml` file and bring your environment up with `docker compose up`. You'll now see how to use Docker Compose commands to manage and interact with your containerized environment.

To check the logs produced by your Nginx container, you can use the `logs` command:

```
$ docker compose logs
```

Copy

You'll see output similar to this:

```
Output
Attaching to  compose-demo_web_1
web_1  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will att
web_1  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entry
```

```
web_1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-i
web_1  | 10-listen-on-ipv6-by-default.sh: Getting the checksum of /etc/nginx/
web_1  | 10-listen-on-ipv6-by-default.sh: Enabled listen on IPv6 in /etc/ngin
web_1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on
web_1  | /docker-entrypoint.sh: Configuration complete; ready for start up
web_1  | 172.22.0.1 - - [02/Jun/2020:10:47:13 +0000] "GET / HTTP/1.1" 200 353
```

If you want to pause the environment execution without changing the current state of your containers, you can use:

```
$ docker compose pause
```

Copy

```
Output
Pausing  compose-demo_web_1  ... done
```

To resume execution after issuing a pause:

```
$ docker compose unpause
```

Copy

```
Output
Unpausing  compose-demo_web_1  ... done
```

The `stop` command will terminate the container execution, but it won't destroy any data associated with your containers:

```
$ docker compose stop
```

Copy

```
Output
Stopping  compose-demo_web_1  ... done
```

If you want to remove the containers, networks, and volumes associated with this

containerized environment, use the `down` command:

```
$ docker compose down
```

Copy

```
Output
Removing  compose-demo_web_1  ... done
Removing network  compose-demo_default
```

Notice that this won't remove the base image used by Docker Compose to spin up your environment (in your case, `nginx:alpine`). This way, whenever you bring your environment up again with a `docker compose up`, the process will be much faster since the image is already on your system.

In case you want to also remove the base image from your system, you can use:

```
$ docker image rm nginx:alpine
```

Copy

```
Output
Untagged: nginx:alpine
Untagged: nginx@sha256:b89a6ccbda39576ad23fd079978c967cecc6b170db6e7ff8a769bf
Deleted: sha256:7d0cdcc60a96a5124763fddf5d534d058ad7d0d8d4c3b8be2aefedf4267d0
Deleted: sha256:05a0eaca15d731e0029a7604ef54f0dda3b736d4e987e6ac87b91ac7aac03
Deleted: sha256:c6bbc4bdac396583641cb44cd35126b2c195be8fe1ac5e6c577c14752bbe9
Deleted: sha256:35789b1e1a362b0da8392ca7d5759ef08b9a6b7141cc1521570f984dc7905
Deleted: sha256:a3efaa65ec344c882fe5d543a392a54c4ceacd1efd91662d06964211b1be4
Deleted: sha256:3e207b409db364b595ba862cdc12be96dcdad8e36c59a03b7b3b61c946a57
```

Note: Please refer to our guide on How to Install and Use Docker for a more detailed reference on Docker commands.

Conclusion

In this guide, you've seen how to install Docker Compose and set up a containerized

Next in environment based on an image. You've also seen how to manage this environment using Compose commands.

For a complete reference of all available `docker compose` commands, check the official documentation.

Tutorial Series: Getting Started With Cloud Computing

This curriculum introduces open-source cloud computing to a general audience along with the skills necessary to deploy applications and websites securely to the cloud.

Subscribe

Docker Ubuntu Ubuntu 22.04

Browse Series: 39 articles

Expand to view all

About the authors

Tony Tran Author

Erika Heidi Author
Developer Advocate

Dev/Ops passionate about open source, PHP, and Linux.

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No

Comments

1 Comments

Leave a comment...

This textbox defaults to using `Markdown` to format your answer.

You can type `!ref` in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

Cava Pip • May 23, 2022

Wonderful tutorial! I followed this tutorial for installing. Then I found this one that has a good tip on using bash aliases.

Is there any difference in installing docker compose in .docker/cli-plugins/ vs installing it in /usr/local/bin/ ? The latter is what the linked guide says to do. This is what your older guide (20.04) and the one linked above is asking me to

do.

Show replies Reply

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

MySQL

Docker

Kubernetes

All tutorials →

Talk to an expert →

Featured on Community

Kubernetes Course Learn Python 3 Machine Learning in Python

Getting started with Go Intro to Kubernetes

DigitalOcean Products

Cloudways Virtual Machines Managed Databases Managed Kubernetes

Block Storage Object Storage Marketplace VPC Load Balancers