Docker Compose, a more advanced Docker tool, can be used to simplify your workflow.

Docker Compose is used to run multiple containers as a single service.

For example, suppose you had an application which required NGNIX and MySQL, you could create one file which would start both the containers as a service without the need to start each one separately.

Docker Compose is a tool for defining and running multi-container Docker applications.

Docker Compose is a Docker tool used to define and run multi-container applications.

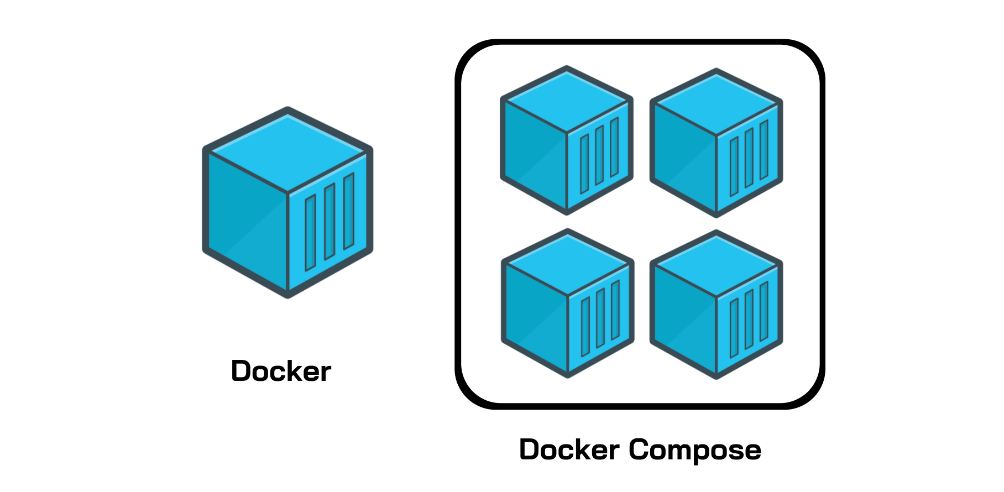
Docker Compose uses a YAML (Ain't Markup Language) file to configure your application’s services.

Docker Compose uses a single command, to create and start all the services from your configuration.

Compose works in all environments: production, staging, development, testing, as well as CI workflows.

**popular features of Docker Compose are:**

* Multiple isolated environments on a single host
* Preserve volume data when containers are created
* Only recreate containers that have changed
* Variables and moving a composition between environments
* Orchestrate multiple containers that work together



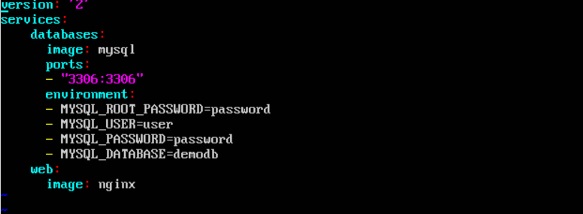
**How to use and install Docker Compose**

Compose uses the Docker Engine, so you’ll need to have the Docker Engine installed on your device. You can run Compose on Windows, Mac, and 64-bit Linux. Installing Docker Compose is actually quite easy. On desktop systems, such as Docker Desktop for Mac and Windows, Docker Compose is already included. No additional steps are needed.

Using Compose is basically a three-step process:

1. Define your app’s environment with a Dockerfile so it can be reproduced anywhere.
2. Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
3. Run docker-compose up and Compose starts and runs your entire app.

A docker-compose.yml looks like this:



* The **database** and **web** keyword are used to define two separate services. One will be running our **mysql** database and the other will be our **nginx** web server.
* The **image** keyword is used to specify the image from **dockerhub** for our **mysql** and **nginx** containers
* For the database, we are using the ports keyword to mention the ports that need to be exposed for **mysql**.
* And then, we also specify the environment variables for **mysql** which are required to run **mysql**.

version: '3.4'

services:

web:

build: .

ports:

- "5000:5000"

volumes:

- .:/code

- logvolume01:/var/log

links:

- redis

redis:

image: redis

volumes:

logvolume01: {}

Docker Compose Commands

|  |  |
| --- | --- |
| Command Name | Description |
| docker-Compose version | To check the version of the Docker-Compose |
| docker-compose --help |  |
| docker-compose build |  |
| Docker-compose images |  |
| Docker-compose stop |  |
| Docker-compose run |  |
| Docker-compose up |  |
| Docker-compose ps |  |
| Docker-compose down |  |