This tutorial shows you how to use some basic commands with Docker Compose.

### Prerequisites

* 1 VM spun up with your cloud provider of choice
  + Running Ubuntu 18.04 LTS
  + Docker installed
  + Docker Compose installed
  + Allow incoming network traffic on port 5000

### Clone the Repository

docker-compose commands require you to be in the working directory of a docker-compose.yaml file. This is how Compose knows which configuration of containers it's meant to be managing. You should therefore only have one Compose file per directory.

We're going to use some Compose commands with an existing pair of containers. Clone down [this repository](https://gitlab.com/qacdevops/python-front-and-back) and change directory into it with the following commands:

git clone https://gitlab.com/qacdevops/python-front-and-back

cd python-front-and-back

This repository contains a simple pair of Python applications that communicate with one another. The exact functionality of this application is not important for this tutorial, but feel free to study the code.

### Running Containers

Run the following command to run the containers in our configuration:

docker-compose up

If the images don't already exist, docker-compose up will build them for you (assuming a build context is specified).

Once the images have been successfully built, you should see the following output:

Creating frontend ... done

Creating backend ... done

Attaching to backend, frontend

backend | \* Serving Flask app "app" (lazy loading)

backend | \* Environment: production

backend | WARNING: This is a development server. Do not use it in a production deployment.

backend | Use a production WSGI server instead.

backend | \* Debug mode: on

backend | \* Running on http://0.0.0.0:5001/ (Press CTRL+C to quit)

backend | \* Restarting with stat

frontend | \* Serving Flask app "app" (lazy loading)

frontend | \* Environment: production

frontend | WARNING: This is a development server. Do not use it in a production deployment.

frontend | Use a production WSGI server instead.

frontend | \* Debug mode: on

frontend | \* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

frontend | \* Restarting with stat

backend | \* Debugger is active!

backend | \* Debugger PIN: 188-950-633

frontend | \* Debugger is active!

frontend | \* Debugger PIN: 149-867-548

Compose handily allows us to see the logs for multiple containers at once. This is particularly handy for troubleshooting applications with multiple services sending HTTP requests to one another.

Enter Ctrl+C to stop the containers. Let's run them in detached mode (i.e. in the background) so we can continue using our terminal session:

docker-compose up -d

### Logs

To view the logs of a Compose configuration, enter the following command:

docker-compose logs

The output should be identical to the logs you saw when you first spun the containers up in attached mode.

We can view the commands in realtime with the -f option. Run the following command:

docker-compose logs -f

While viewing these logs, navigate to the public IP address of your virtual machine on port 5000 (make sure your firewall rules have allowed incoming traffic on that port). Refresh the page a few times.

In realtime, you should see the following logs appear each time you refresh the page:

backend | 172.18.0.3 - - [09/Sep/2020 14:17:48] "GET /hostname HTTP/1.1" 200 -

backend | 172.18.0.3 - - [09/Sep/2020 14:17:48] "GET /random HTTP/1.1" 200 -

frontend | 87.75.101.48 - - [09/Sep/2020 14:17:48] "GET / HTTP/1.1" 200 -

The frontend container is receiving your HTTP requests on port 5000 and the backend is receiving two requests from the frontend. This is a great way to pinpoint where networking issues between your containers are occurring.

Enter Ctrl+C to stop viewing the logs.

### Building Images

Let's change the functionality of the frontend container. The repo has another version of the application on a separate branch called red-background. Change to that branch by entering:

git checkout red-background

Run the following command to rebuild our new image with our new functionality without bringing the current containers down:

docker-compose build

### Redeploying Containers

Compose can detect new versions of images and recreate running services with the new version without bringing the running versions down. Simply run the following command again:

docker-compose up -d

Refresh the page on your browser. The app should now display a red background!

### Cleaning Up

We can easily stop our containers with docker-compose down. If we want to bring the containers down and delete their associated images, we can run:

docker-compose down --rmi all

Do this now.