Working with Multiple Dockerfiles

A guide for using multiple Dockerfiles

We'll cover the following

Multiple Dockerfile use cases

There might be a situation where we need to have multiple Dockerfiles for different services.

Examples could be:

- Creating a microservices app
- Dockerfile for different environments such as development, production

In these cases, you have to tell docker-compose the Dockerfile it should consider for creating that specific service.

> By default, docker-compose looks for a file named Dockerfile. Dockerfiles can have any name. It's just a file without any extension. We can override the default behaviour using `dockerfile:'custom-name'` directive inside the build section.

Implementation

Let's see this in action. We will add a new file called Dockerfile-db in the DB folder and make some changes in the docker-compose.yml file.

In the build section, we have

```
context: ./db
dockerfile: Dockerfile-db
```

`context`: Use this to specify the directory of Dockerfile or alternate Dockerfile relative to docker-compose.yml file

`dockerfile`: As mentioned in the note above, specify the name of alternate Dockerfile if it is not named as Dockerfile

```
FROM mysql/mysql-server:5.7
```

We have added one more keyword in the ‘web’ service.

`depends_on`: This is used to inform docker-compose about all the dependencies of a service. Docker-compose will then start dependencies first and the main service after.

When building a large microservices architecture, you need to use multiple Dockerfiles and this lesson will surely help you with that.

In the next lesson, we will look at how to troubleshoot most frequently-occurring issues in this area.