Migrating to Docker Compose

Even with a simple image, we've already been dealing with plenty of command line options in both building, pushing and running the image.

Next we will switch to a tool called Docker Compose to manage these. Docker Compose used to be a separate tool but now it is integrated into Docker and can be used like the rest of the Docker commands.

Docker Compose is designed to simplify running multi-container applications using a single command.

Assume that we are in the folder where we have our Dockerfile with the following content:

```
FROM ubuntu:22.04

WORKDIR /mydir

RUN apt-get update && apt-get install -y curl python3
RUN curl -L https://github.com/yt-dlp/yt-dlp/releases/latest/download/yt-dlp -o /usr/local/bin/yt-dlp
RUN chmod a+x /usr/local/bin/yt-dlp

ENTRYPOINT ["/usr/local/bin/yt-dlp"]
```

Let us now create a file called `docker-compose.yml`:

```yaml
version: '3.8'

services:
  yt-dlp-ubuntu:
    image: <username>/<repositoryname>
    build: .
```

The version setting is not very strict, it just needs to be above 2 because otherwise the syntax is significantly different. See https://docs.docker.com/compose/compose-file/ for more info.

The value of the key `build` can be a file system path (in the example it is the current directory `.`) or an object with keys `context` and `dockerfile`, see the documentation for more

Now we can build and push with just these commands:

```
$ docker compose build
$ docker compose push
```

Volumes in Docker Compose

To run the image as we did previously, we will need to add the volume bind mounts. Volumes in Docker Compose are defined with the following syntax `location-in-host:location-in-container`. Compose can work without an absolute path:

```yaml
version: '3.8'

services:

  yt-dlp-ubuntu:
    image: <username>/<repositoryname>
    build: .
    volumes:
      - .:/mydir
    container_name: yt-dlp
```

We can also give the container a name it will use when running with container_name. The service name can be used to run it:

```
$ docker compose run yt-dlp-ubuntu https://imgur.com/JY5tHqr
```

Exercise 2.1

EXERCISE 2.1

Let us now leverage the Docker Compose with the simple webservice that we used in the Exercise 1.3

Without a command `devopsdockeruh/simple-web-service` will create logs into its `/usr/src/app/text.log`.

Create a docker-compose.yml file that starts `devopsdockeruh/simple-web-service` and saves the logs into your filesystem.

Submit the docker-compose.yml, and make sure that it works simply by running `docker compose up` if the log file exists.

Web services in Docker Compose

Compose is really meant for running web services, so let's move from simple binary wrappers to running a HTTP service.

https://github.com/jwilder/whoami is a simple service that prints the current container id (hostname).

```
$ docker container run -d -p 8000:8000 jwilder/whoami
  736ab83847bb12dddd8b09969433f3a02d64d5b0be48f7a5c59a594e3a6a3541
```

Navigate with a browser or curl to localhost:8000, they both will answer with the id.

Take down the container so that it's not blocking port 8000.

```
$ docker container stop 736ab83847bb
$ docker container rm 736ab83847bb
```

Let's create a new folder and a Docker Compose file `whoami/docker-compose.yml` from the command line options.

```yaml
version: '3.8'

services:
  whoami:
    image: jwilder/whoami
    ports:
      - 8000:8000
```

Test it:

```
$ docker compose up -d
$ curl localhost:8000
```

Environment variables can also be given to the containers in Docker Compose as follows:

```yaml
version: '3.8'

services:
  backend:
    image:
    environment:
      - VARIABLE=VALUE
      - VARIABLE2=VALUE2
```

Note that there are also other, perhaps more elegant ways to define the environment variables in Docker compose.

Exercises 2.2 - 2.3

EXERCISE 2.2

Read about how to add the command to docker-compose.yml from the documentation.

The familiar image `devopsdockeruh/simple-web-service` can be used to start a web service, see the exercise 1.10.

Create a docker-compose.yml, and use it to start the service so that you can use it with your browser.

Submit the docker-compose.yml, and make sure that it works simply by running `docker compose up`

MANDATORY EXERCISE 2.3

As we saw previously, starting an application with two programs was not trivial and the commands got a bit long.

In the previous part we created Dockerfiles for both frontend and backend of the example application. Next, simplify the usage into one docker-compose.yml.

Configure the backend and frontend from part 1 to work in Docker Compose.

Submit the docker-compose.yml

Edit this page

Previous
« Introduction to Part 2

Next
Docker networking »