Section 10 - Docker Compose

1 Docker Compose - Introduction

Docker Compose - Intro (1)

- Most modern application are made of multiple smaller *services* that interact with each other to form an application
- Example:
  - front-end (html - javascript)
  - back-end (business logic - Rest endpoints)
  - DB
- Deploy and manage multiple services (multiple containers) can be difficult. This is where docker-compose comes in to play
- Instead of executing a separate `docker run` commands for each service of the application we can use a single `docker-compose up` command deploy the entire application.

Overview of Docker Compose (1)

- There are 2 separate components required to use docker-compose
  1. The docker-compose.yml (YAML) file used to define:
     - services
     - networks
     - volumes
  2. The CLI tool docker-compose used in conjunction with the *yml* file

Overview of Docker Compose (2)

- docker-compose is mainly used for local development purposes.
- The yml files can be used on a production environment with Docker Swarm.
- docker-compose.yml is the default filename but any file name can be used with `docker-compose -f`.

Docker Compose - Documentation pages

- Overview
- Install
- Reference

Docker Compose - Example (1a)

- The docker-compose.yml of this example is available in the `resources` directory

```
version: '3.6'
# same as
# docker run -p 8080:80 --name nginx nginx

services:
  nginx:
    image:
      nginx
    ports:
      - "8080:80"
```

- To start the services defined in the docker-compose.yml file execute the `docker-compose up` command

Docker Compose - Example (1b)

```
# cd resources/compose-sample-1
# docker-compose up

# docker container ls
CONTAINER ID   IMAGE   COMMAND       CREATED          STATUS          PORTS
94bd5ee203f4   nginx   "nginx -g…"   2 minutes ago   Up 2 minutes   0.0.0.0

$ docker network ls
NETWORK ID     NAME                      DRIVER    SCOPE
9f7a65b273f0   bridge                    bridge    local
8acc86631a77   composesample1_default    bridge    local
b0897e090893   host                      host      local
3163420f3967   none                      null      local
```

Docker Compose - version

- The *version* is first top level key attribute of the `docker-compose.yml` file
- It is mandatory and we should normally use the latest version
- The *version* value defines the format (basically the API)
- More information here

Docker Compose - Example 2 notes

- In this example we will see how to use Docker compose to replace all the commands used in the D_S9_L3_Persistent_Data_LAB

- The docker-compose.yml of this example is available in the `resources` directory.

Docker Compose - Example 2 part1

- Example 2 part1

```
version: '3.6'

services:
  postgres10:
    image: postgres:10
    environment:
      POSTGRES_DB: "db-test1"
      POSTGRES_USER: "db-user1"
      POSTGRES_PASSWORD: "db-pw1"
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql:ro
      - db-data:/var/lib/postgresql/data
    networks:
      - net-db
...
```

Docker Compose - Example 2 part2

- Example 2 part2

```
  pgadmin4:
    image: dpage/pgadmin4:4.6
    environment:
      PGADMIN_DEFAULT_EMAIL: "pgadmin"
      PGADMIN_DEFAULT_PASSWORD: "pgadmin"
    networks:
      - net-db
    ports:
      - "8080:80"

volumes:
  db-data:

networks:
  net-db:
```

Compose file structure (1)

- The `<key>: <value>` format is used to define single components such as the image name

```
...
image: postgres:10
...
```

Compose file structure (2)

- The value of some attributes keys such as the volumes and ports (note plural) is an array. The " - " symbol is used to define an element of an array

```
volumes:
  - ./init.sql:/docker-entrypoint-initdb.d/init.sql:ro
  - db-data:/var/lib/postgresql/data
```

- Note also that we can use the (.) symbol to define the current working directory in the volumes section

Compose file structure (3)

- Every single key value option that can be used in the `docker-compose.yml` file is described on the documentation page

- The default compose file name is `docker-compose.yml`, in this case we do not need to specify the yml file when we execute the `docker-compose` cli commands, e.g:

```
# docker-compose up
```

- Use `-f` to specify the name and path of a custom yml compose file, e.g:

```
# docker-compose -f docker-compose-custom.yml up
```


docker