ceedcv

CENTRE ESPECÍFIC D´EDUCACIÓ A DISTÀNCIA DE LA COMUNITAT VALENCIANA

# UT 11. DOCKER

Activities

**Computer Systems**
**CFGS DAW**

Álvaro Maceda

a.macedaarranz@edu.gva.es

2022/2023

Version:230227.0719

## License

## Nomenclature

Throughout this unit different symbols will be used to distinguish important elements within the content. These symbols are:

| |
|---|
| Important |
| Attention |
| Interesting |

# UT 11. Docker
Solutions

## 1. Exercise 1

**With `docker create` and `docker start`:**

1. Create the containers

```
docker create \
     --name http1 \
     --mount type=bind,src=/my/dir1,dst=/usr/local/apache2/htdocs/ \
     -p 8080:80 \
     httpd

docker create \
     --name http2 \
     --mount type=bind,src=/my/dir1,dst=/usr/local/apache2/htdocs/ \
     -p 8080:80 \
     httpd

docker create \
     --name http3 \
     --mount type=bind,src=/my/dir1,dst=/usr/local/apache2/htdocs/ \
     -p 8080:80 \
     httpd
```

2. Start the containers:

```
docker start http1
docker start http2
docker start http3
```

3. You can open the web pages at http://localhost:8080, http://localhost:8081 and http://localhost:8082

4. Stop and destroy the containers:

```
docker stop http1
docker rm http1
```

Repeat for http2 and http3

**With `docker run`:**

To launch the containers:

```
docker run --rm -d \
     --name http1 \
     --mount type=bind,src=/my/dir1,dst=/usr/local/apache2/htdocs/ \
     -p 8080:80 \
     httpd
```

To stop and remove the containers:

```
docker stop http1
```

You can also run the containers interactively omitting the `-d` option, each one in a different terminal, and stop them by pressing `Ctrl+C` or using `docker stop` in another terminal.

## 2. Exercise 2

**With docker create:**

```
docker create --name python-bug -ti python:2.7.18
docker start python-bug
```

You need to pass the `-ti` parameters or the container will exit and you won't be able to attach to it or to exec commands there.

If you attach to the container you will see a python interpreter. You need to execute a shell in the container:

```
docker exec -ti /bin/bash python-bug
```

Once there, you can install an editor with `apt update` and `apt install nano/vim`. Then, you can launch another shell (in another terminal in your host) with the same command, and run the script.

To stop and destroy the container:

```
docker stop python-bug
docker rm python-bug
```

**With docker run:**

You will need to use docker run and pass a shell as the command. If not, the container will exit and you won't be able to attach to it or to exec commands there:

```
docker run --name python-bug -ti python:2.7.18 /bin/bash
```

This will open a shell. You can install an editor with `apt update` and `apt install nano/vim`. Once you have the file, you can open another shell with:

```
docker exec -ti /bin/bash python-bug
```

And run the script.

The container will be stopped and destroyed once you exit the shell of `docker run`. Launching the command again will generate a new container, but you won't have the editor or the script there.

## 3. Exercise 3

### Part 1

1. Create and run a MySQL server container with the official image:

```
docker run --rm \
      --name database-server \
      --env MYSQL_ROOT_PASSWORD=secret \
      -p 3306:3306 mysql
```

The `-e` or `--env` option is used for sending information to the container through environment variables. In this case, we assign a password for the root user.

You will need the -p option to connect from your computer in the second part of the exercise

2. Open a shell to the server. In another terminal:

docker exec -ti database-server /bin/bash

And then run:

```
root@310...99:/# mysql --user root --password=secret
```

This will open a mysql shell:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.28 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

Run the commands (remember to write the semicolon at the end of each one) and type `exit`;

Do the same from your computer, and stop the server with `docker stop database-server`.

You can also use `docker create` and `docker start` for the exercise. Remember to `docker rm` the container in that case.

### Part 2

Create the volume:

```
docker volume create database-volume
```

Launch the database server:

```
docker run --rm \
           --name database-server \
           --env MYSQL_ROOT_PASSWORD=secret \
           -p 3306:3306 \
           --mount type=volume,src=database-volume,dst=/var/lib/mysql \
           mysql
```

Create the database as in Part 1.

Destroy the container with `docker volume rm database-volume`.

## Part 3

Launch the database server:

```
docker run --rm \
           --name database-server \
           --env MYSQL_ROOT_PASSWORD=secret \
           -p 3306:3306 \
           --mount type=bind,src=/your-dir,dst=/var/lib/mysql \
           mysql
```

Create the database as in Part 1.

The container will create the files with the user 999, because that is the ID of the user running the database server in the container (the container was designed that way)

If you have a user with that id in your system, the files will appear with its name. You can display the id with `ls -n`.