Docker

source: Docker for Developers Stage 1

What is it

Docker's purpose is to encapsulate an application **together with** its execution environment (*containerize* it), making it practical to **consistently deploy** it.

It can be used for:

- having an isolated, protected and portable environment for the execution (and development, and test) of the app
- safe and easier portability of the whole project
- containers that execute one task and exit, useful to do some configuration anywhere

Main concepts

host : your computer

image : it's the actual software + its environment, wrapped together in a ready-to-run (=already compiled) bundle

container : a running instance of an image, when docker runs it, "the container process is isolated in that it has its own file system, its own networking, and its own isolated process tree separate from the host"

layer : images are built by adding layers to a base

dockerfile : instructions to build the image

→ it is possible to commit a container to make an image, but it should be avoided in favour of building it through a dockerfile

volume : a "physical" storage place, if not specified it is created within the container after launching and destroyed when it is stopped

→ it is possible to specify a binding to a local repo on your machine, in order to actively refer to your files while the container is running

Hands-on

- check
  - `docker version`
  - `docker info`
- `docker container ls`: lists all runningcontainers
- `docker container ls --all`: lists all containers
- `docker container run <NAME> <PARAMETERS>`: executes a container, if docker doesn't find it locally it will try to pull it from Docker Hub

  → `docker container run alpine hostname`

  - `docker container run --interactive --tty --rm <NAME> <PROCESS>`: executes `<PROCESS>` in `<NAME>` container
    * `--interactive`: self-explicative :D
      this can be useful for example while defining the dockerfile, to test the steps needed to deploy the app
    * `--tty`: allocates a pseudo –tty (**T**ele**TY**pewriter, the file name of the terminal)
    * `--rm`: removes the container after its execution
    * `<NAME>`: container
    * `<PROCESS>`: executes this as the container's main process (so type `exit` to... *exit* the bash and, consequently, the container)

    → `docker container run --interactive --tty --rm ubuntu bash`
  - `docker container run --detach --name <NEWNAME> -e <ENV-VAR=value> <NAME>`: executes container `<NAME>` in background renaming and setting an environment variable
    * `--detach`: executes in background
    * `--name`: renames it as `<NEWNAME>`
    * `-e`: environment variable

    → `docker container run --detach --name mydb`
      `-e MYSQL_ROOT_PASSWORD=my-secret-pw mysql:latest`
  - `docker container run --publish <HOST_PORT>:<CONTAINER_PORT>`
  - `docker container run --mount type=bind, source=..., target=... <NAME>`: mounts `source` into container `<NAME>` (within it, it's in the location `target`)

    → any changes in the source from the host are automatically reflected into the container
    * obviously, still need to rebuild the image to change it

- `docker container logs <NAME>`: shows the logs from the `<NAME>` container
- `docker container top <NAME>`: shows the processes running inside `<NAME>` container
- `docker container exec <NAME>/<ID> <COMMAND PARAMTERS/FLAG>`: executes the command inside the container

  → `docker exec -it mydb mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version` equivalent to:

    → `docker exec -it mydb sh`
    → (into the shell) `mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version`

- `docker container stop`: stop the container execution

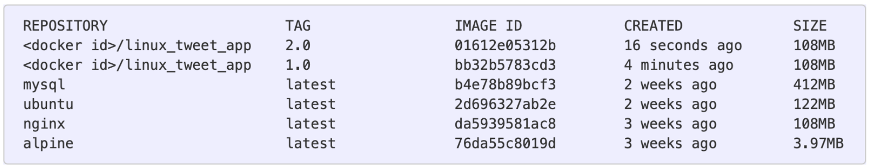
Package a custom app as an image

Dockerfile:

```
FROM nginx:latest <BASE_IMAGE>

COPY index.html /usr/share/nginx/html <FILE TO COPY INTO
THE IMAGE> <DESTINATION>
COPY linux.png /usr/share/nginx/html

EXPOSE 80 443 <PORT_NUMBER1> <PORT_NUMBER2>

CMD ["nginx", "-g", "daemon off;"] ["COMMAND",
"FLAG/PARAMETER", "FLAG/PARAMETER;"]
```

Create and delete image:

- `docker image build --tag <NEW_IMAGE_NAME> .`
  → `docker image build --tag $DOCKERID/linux_tweet_app:1.0`
  → `docker image build --tag $DOCKERID/linux_tweet_app:2.0`
- `docker image ls`: see all images on the system

  → 

```
REPOSITORY                     TAG        IMAGE ID       CREATED          SIZE
<docker id>/linux_tweet_app    2.0        01612e05312b   16 seconds ago   108MB
<docker id>/linux_tweet_app    1.0        bb32b5783cd3   4 minutes ago    108MB
mysql                          latest     b4e78b89bcf3   2 weeks ago      412MB
ubuntu                         latest     2d696327ab2e   2 weeks ago      122MB
nginx                          latest     da5939581ac8   3 weeks ago      108MB
alpine                         latest     76da55c8019d   3 weeks ago      3.97MB
```

- `docker container rm --force <NAME>`: remove container

  - `--force`: removes it even if it is running

- push the image to Docker Hub
  - `docker login`
  - `docker image push <NAME>/<ID>`

Application Containerization and Microservice Orchestration

Deploying a Multi-Service App in Docker Swarm Mode