Quick Reminder About Docker

What we gonna use throughout this section

What is Docker?

Docker is a containerization platform that packages your application and all its dependencies together in the form of a **docker container** to ensure that your application works seamlessly in any environment.
source: https://www.edureka.co/blog/what-is-docker-containe

In other words, you can think of Docker as being a way to run your application anywhere, whatever the environment and the operating system your application is running on. Your application will always produce the same result.

What is a Docker Container?

A **Docker container** is an instance of a **Docker image**.

A **Docker image** is a single file with a **Dockerfile**, dependencies and code your application needs to run.

A **Docker container** is a program (instance of a docker image) with its own isolated set of hardware resources. It has its own space of networking, memory and hard drive space.

The difference between a container and a VM is that a VM has an entire operating system running in it whereas a container shares the operating system where it is running on. A Docker container is way more lighter than a VM and allows to quickly start and stop resources as we need. For example, we could easily add airflow workers to scale out our Apache Airflow installation.

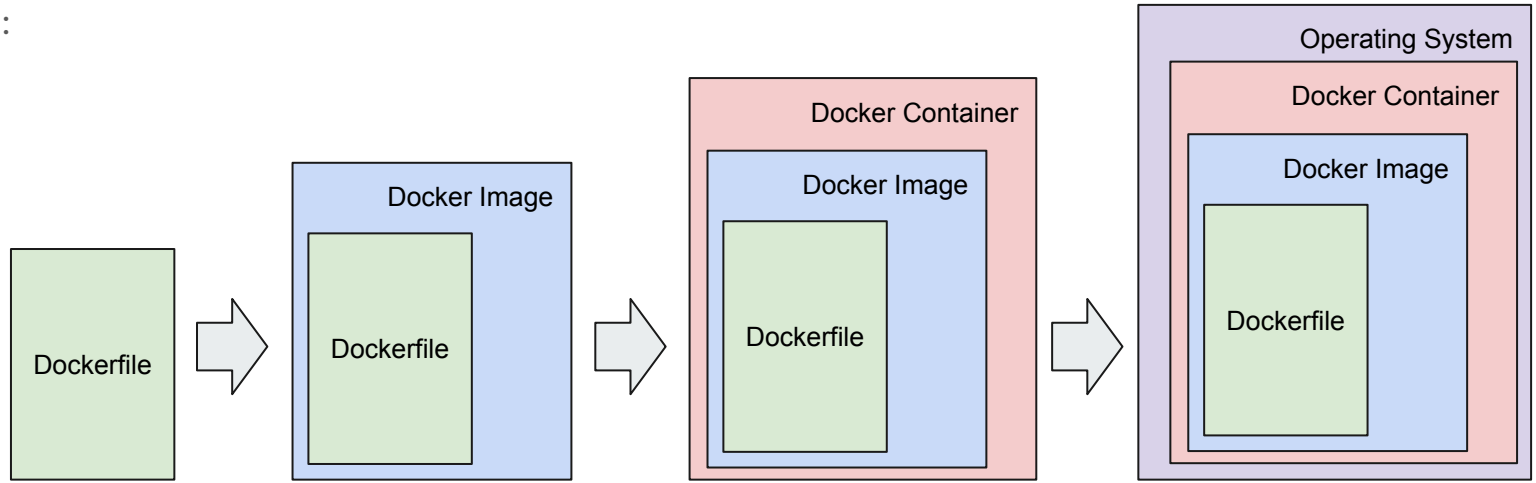
What is a Dockerfile?

A **Dockerfile** is nothing more than a file containing the instructions for telling to Docker how it should build your Docker image which contains everything your application needs to run .

We gonna see an example of a Dockerfile in the practical section of this course.

Schema

Just to sum up how everything work together in a very simplified way, we could have the following schema:



The Ecosystem of Docker

Docker has many components so I'm gonna focus only on those we gonna use during this section.

- **Docker Client**, which is the primary way to interact with Docker. Each time you use the Docker Command Line Interface you refer to the Docker Client.
- **Docker Daemon**, which is the Docker server listening for Docker API requests. It manages images, containers, volumes and networks.
- **Docker Volume**, which is a way to store the persistent data your applications create and use.
- **Docker Compose**, which is the tool used to make easier the fact of running an application using multiple Docker containers. Docker Compose uses a .yml file that defines how Docker containers should behave in production.

Basic Commands to Know

- `docker info`
  - Gives information about your Docker installation (number of running, paused, stopped containers, number of images).
- `docker run image`
  - Runs an instance of the `image` into a Docker container.
- `docker image ls`
  - Lists the images available on your machine.
- `docker container ls --all`
  - Lists the containers running or not on your machine. If you only want to see your running containers, don't use the --all option.

Basic Commands to Know

- `docker container stop container-id`
  - Stops a running Docker container by specifying its container-id (you can find it by using the command `docker container ls`)
- `docker build --tag=name-of-your-docker-image`
  - Builds a Docker image with the specified name passed into the --tag parameter.
- `docker exec -it container-name bash`
  - Allows to interact with the container named `container-named` using a Bash shell.
- `docker logs container-id`
  - Allows to read the logs produced by your running container associated with the `container-id`
- `docker-compose -f docker-compose.yml up -d`
  - Runs multiple Docker containers as defined in to docker-compose.yml file

A Final Word About Docker Compose

Docker Compose is used to define and run multi-container Docker applications. By using a Compose file (with the .yml extension) you can configure as many containers as you want and specify how they should be built and connected as well as where the data should be stored.

Then with a single command, you can build, run and configure all of the containers.

We will use docker-compose to run Airflow along with its web server, scheduler and one or multiple workers.

Time to Practice!

Enough talking, time to practice!