**DOCKER**

**What is Docker?**

Is a virtualization software that makes developing and deploying applications much easier. Docker package application with all the necessary dependencies, configuration, system tools and runtime.

**Differences between Docker and Virtual Machine**

|  |  |
| --- | --- |
| **Docker** | **Virtual Machine** |
| Docker images, couple of MB | Virtual Machine images, couple of GB |
| Containers takes seconds to start.   * Reuses the host kernel and you just start de application layer on top of it | VM takes minutes to start.   * Put up a kernel every time it starts |
| Compatible only with Linux distros | Compatible with all OS |

[**Download Docker Desktop**](https://www.docker.com/)

**Docker Desktop includes:**

* **Docker Engine**
  + A server with a long-running daemon process “dockerd”
  + Manage images & containers.
* **Docker CLI – Client**
  + Command Line Interface (“docker”) to interact with Docker Server
  + Execute Docker commands to start/stop .. containers
* **GUI Client**
  + To manage your containers and images with a graphical user interface.

**Differences between Docker Images and Docker Containers**

|  |  |  |
| --- | --- | --- |
|  | **Docker Images** | **Docker Containers** |
| **Definition** | Read-only templates that contain everything needed to run an application. | Running instances of Docker Images |
| **Purpose** | Used for creating and distributing applications and complete reproducible environments. | Used to run isolated applications with their own resources and environment. |
| **Content** | Include the operating system, libraries, dependencies, and application code. | Docker images running in an isolated environment, containing the application code and resources. |
| **Size** | Can have variable sizes, usually larger than containers. | Lighter than images as they share the operating system and only add additional layers for container-specific changes. |
| **Mutability** | Immutable and cannot be modified once created, only new images can be generated from them. | Ephemeral and can be modified at runtime, but any changes made will be lost once the container is stopped. |
| **Usage** | Used for building, sharing, and distributing consistent applications and environments. | Used to run applications in an isolated environment, enabling portability and resource management. |
| **Data Persistence** | Can include static data that is part of the image, but changes to the data are not reflected in the original image. | Containers can have mounted data volumes, allowing data persistence even after the container is stopped and restarted. |
| **Scalability** | Do not scale directly. Multiple containers can be generated from the same image. | Can be horizontally scaled by generating multiple instances of identical containers. |
| **Relationship** | Containers are based on images and run from them. | Images are used to create and generate containers, which are running instances of the images. |

**Docker Official Images**

* A dedicated team responsible for reviewing and publishing all content in the Docker Official Images Repositories
* Works in collaboration with software maintainers, security experts.
* However, anyone can participate as collaboration takes places openly on Github.

**Pull new Docker Image with the latest version:**

$ docker pull <nombre\_de\_lo\_que\_te\_quieres\_descargar>

**Pull new Docker Image with another version (not the latest one):**

**$** docker pull <nombre\_de\_lo\_que\_te\_quieres\_descargar>:<versión>

**To see which Docker Images you have installed:**

$ docker images

**Remove Docker Image:**

* **If the Docker Image is not used by any container.**

$ docker rmi <ID\_de\_la\_imagen>

* **If you want to delete an image regardless of whether it is in use or not, add the -f option to the command.**

$ docker rmi -f <ID\_de\_la\_imagen>

* **If you want to remove an Image that have the same ID, but each Image has different version.**

$ docker rmi <nombre\_del\_repositorio>:<tag>

**Run an Image as a Container:**

$ docker run <nombre\_del\_repositorio>:<tag>

**To run a container in the background without blocking the terminal (-d):**

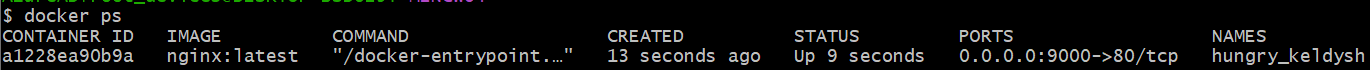
$ docker run -d <nombre\_del\_repositorio>:<tag>

*It returns the container ID.*

**To run a container in the background especifying the host port and the container port name:**

$ docker run -d -p HOST\_PORT:CONTAINER\_PORT <nombre\_del\_repositorio>:<tag>

****

****

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

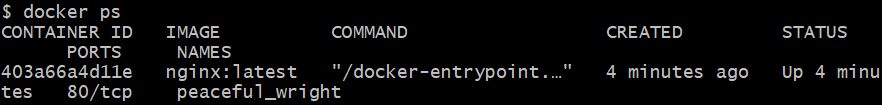
**Each container runs in different port. Redis don’t run as the same port as nginx**

**To assign a name to a container:**

$ docker run --name <nombre> -d -p HOST\_PORT:CONTAINER\_PORT <nombre\_del\_repositorio>:<tag>

**To see the actives container:**

$ docker ps



*Docker generates a random name for the container automatically if you don’t specify one.*

**To see all the containers (stopped and running):**

$ docker ps -a

**To stop an image:**

$ docker stop <id o **nombre**\_del\_contenedor (última columna al hacer *docker ps*>

**To stop all the containers that are running:**

$ docker stop $(docker ps -q)

**To resume a container that are stopped:**

$ docker start <nombre\_del\_contenedor>

or

$ docker start <container\_id>

**Dockerfile – Create own Images**

**Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media**

Docker file is a text document that contains commands to assemble an image.

Docker can then build an image by reading those instructions.

**Structure of Dockerfile**

* Dockerfiles start from a parent image of “base image”.
* It’s a Docker Image that your image is based on.
  + You choose the base image, depending on which tools you need to have available

FROM

* Build this image from the specified image.

WORKDIR

* Sets the working directory for all following commands.
* Like changing into a directory: cd ..

RUN

* Will execute any command in a shell inside the container environment.

CMD

* The instruction that is to be executed when a Docker container starts.
* There can only be one “CMD” instruction in a Dockerfile.

**Let’s create an example:**

[**Click here to go to the example repository**](https://github.com/GomezPMario/First-Dockerfile.git)

* We will write a Dockerfile for Node.js application.
* Then, build a Docker image from it.

**Texto

Descripción generada automáticamente**

**To create a new Docker Image from a Dockerfile:**

$ docker build -t node-app:1.0 .

* **docker build**: start the process of building a new Docker Image.
* **-t node-app:1.0** : the -t option is used to assign a label (name and version) to the image being built. In this case, the tag node-app:1.0 is set, where node-app is the name of the image and 1.0 is the version.
* **.** : the dot represents the build context. It specifies that the Dockerfile and related files are located in the current directory. Docker will use this context to build the image.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Run the following command:

$ docker run -d -p 3000:3000 node-app:1.0

We go to the address localhost:3000 because that is where we have told it to run.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

**To see the logs generated by a specific container:**

$ docker logs c84fc9fda90e



[Click here to go to the example repository](https://github.com/GomezPMario/First-Dockerfile.git)

**Visual example of how Docker works:**

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente