Docker Architecture - Client-server

Let's look farther behind the scenes of Docker!

We'll cover the following

Till now, we have used the term, 'Docker' very broadly when describing its functionalities, but Docker has different components as part of its ecosystem. From now, we will be specific about the component we are referring to from the ecosystem. Let's look at the different components of the Docker ecosystem.

Docker ecosystem

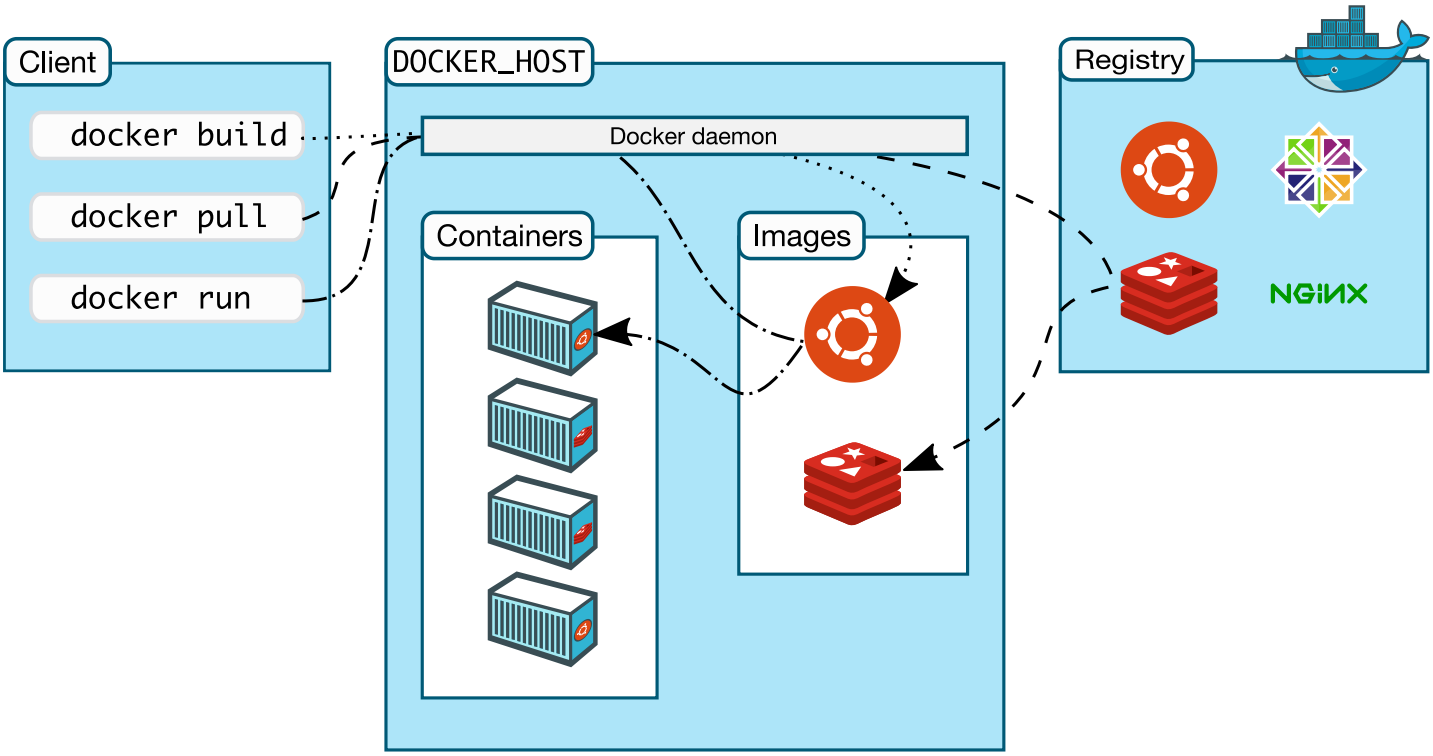
- **Docker Registry**: Docker maintains all the images in the registry and they can be pulled from the registry too
- **Docker Hub**: This is the repository for all your custom-built images. Images can be pushed and accessed from the Hub
- **Docker Client**: The CLI tool used to interact with the Docker server
- **Docker Daemon**: The Docker server process responsible for pulling, pushing, and building the images. It is also used for running the container

Since we are now clear on the terminologies, let's see how to run the containers in the next chapter.

Communication between a Docker client and the Docker daemon

A container is an instance of an image. Whenever we need to run a particular image, we need to have that image on the system. Using the Docker client, you can ask the Docker daemon to run a particular image. The daemon will go ahead and look for the image on a system. If it finds the image, it will run the container forked from that image. However, if the image is not present on the system, it will

pull the image from the Docker registry and create a container from the image.



In the diagram above, you can see that the Docker client asks the daemon to pull a Redis image. Also, it asks the daemon to run the Ubuntu image. The Docker client can be used to communicate with the Docker daemon present on the host machine or any remote daemon as well. The communication happens through APIs provided by the Docker server.

Throughout this course, we will be using a local Docker client and daemon only. There are only a few things you need to keep in mind while working with Docker:

1. Images
2. Containers
3. Networks - which we will be cover in the Docker Compose section

The short list above is all you need. Docker takes care of the rest. There are so many abstractions in the implementation of Docker for every platform other than Linux, but the whole idea remains the same: providing the native Linux kernel to containers. So, let's focus on the core functionality we need to do development and let the Docker daemon do the rest.

In the next section, we will install Docker for different operating systems. If you have a Linux or Mac system, that will be a huge plus.