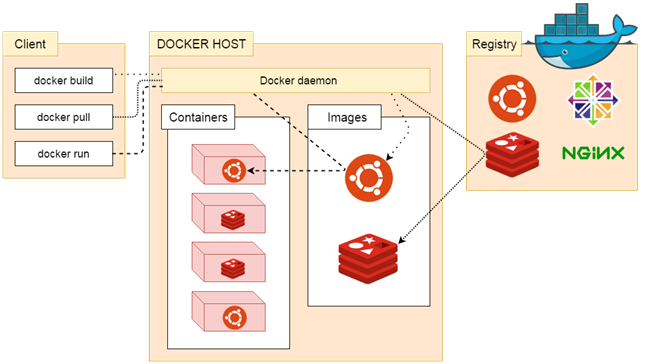
Docker Architecture

Docker follows client-server architecture. Its architecture consists mainly three parts.

1) **Client:** Docker provides Command Line Interface (CLI) tools to client to interact with Docker daemon. Client can build, run and stop application. Client can also interact to Docker\_Host remotely.

2) **Docker\_Host:** It contains Containers, Images, and Docker daemon. It provides complete environment to execute and run your application.

3) **Registry:** It is global repository of images. You can access and use these images to run your application in Docker environment.



The Docker daemon

It is a process which is used to listen for Docker API requests. It also manages Docker objects like: images, container, network etc. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

The Docker client is the primary way that many Docker users interact with Docker. When we use commands such as docker run, the client sends these commands to docker d, which carries them out. The docker command uses the Docker API.

Docker Registries

Docker registry is used to store Docker images. Docker provides the Docker Hub and the Docker Cloud which are public registries that anyone can use. Docker is configured to look for images on Docker Hub by default.

When we use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry.

Docker Dockerfile

A Dockerfile is a text document that contains commands that are used to assemble an image. We can use any command that call on the command line. Docker builds images automatically by reading the instructions from the Dockerfile.

The docker build command is used to build an image from the Dockerfile. You can use the -f flag with docker build to point to a Dockerfile anywhere in your file system.

1. $ docker build -f /path/to/a/Dockerfile .

Dockerfile Instructions

The instructions are not case-sensitive but you must follow conventions which recommend to use uppercase.

Docker runs instructions of Dockerfile in top to bottom order. The first instruction must be **FROM** in order to specify the Base Image.

A statement begin with # treated as a comment. You can use RUN, CMD, FROM, EXPOSE, ENV etc instructions in your Dockerfile.

Here, we are listing some commonly used instructions.

FROM

This instruction is used to set the Base Image for the subsequent instructions. A valid Dockerfile must have FROM as its first instruction.

Ex.

1. FROM ubuntu

LABEL

We can add labels to an image to organize images of our project. We need to use LABEL instruction to set label for the image.

Ex.

1. LABEL vendorl = "JavaTpoint"

RUN

This instruction is used to execute any command of the current image.

Ex.

1. RUN /bin/bash -c 'source $HOME/.bashrc; echo $HOME'

CMD

This is used to execute application by the image. We should use CMD always in the following form

1. CMD ["executable", "param1", "param2"?]

This is preferred way to use CMD. There can be only one CMD in a Dockerfile. If we use more than one CMD, only last one will execute.

COPY

This instruction is used to copy new files or directories from source to the filesystem of the container at the destination.

Ex.

1. COPY abc/ /xyz

**Rules**

* The source path must be inside the context of the build. We cannot COPY ../something /something because the first step of a docker build is to send the context directory (and subdirectories) to the docker daemon.
* If source is a directory, the entire contents of the directory are copied including filesystem metadata.

WORKDIR

The WORKDIR is used to set the working directory for any RUN, CMD and COPY instruction that follows it in the Dockerfile. If work directory does not exist, it will be created by default.

We can use WORKDIR multiple times in a Dockerfile.

Ex.

1. WORKDIR /var/www/html