Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker Engine

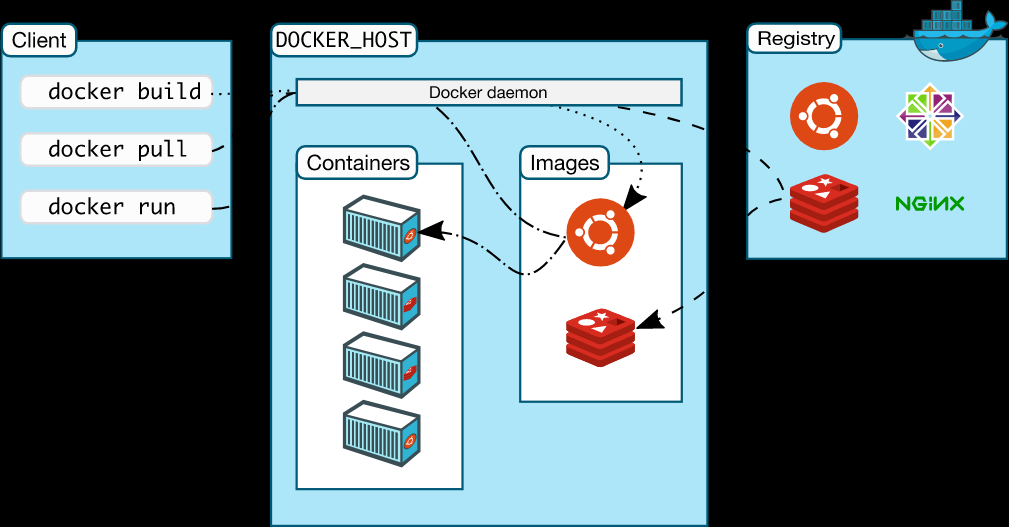
*Docker Engine* is a client-server application with these major components:

* A server which is a type of long-running program called a daemon process (the dockerd command).
* A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
* A command line interface (CLI) client (the docker command).



## Docker architecture

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.



Docker information: <https://docs.docker.com/get-started/overview/>

Docker installation: <https://docs.docker.com/desktop/>

Docker documentation: <https://docs.docker.com/docker-for-windows/>

Basic Docker Commands

Build

* Build an image from the Dockerfile in the current directory and tag the image **docker build -t myimage:1.0 .**
* List all images that are locally stored with the Docker Engine **docker image ls**
* Delete an image from the local image store **docker image rm alpine:3.4**

Share

* Pull an image from a registry **docker pull myimage:1.0**
* Retag a local image with a new image name and tag **docker tag myimage:1.0 myrepo/ myimage:2.0**
* Push an image to a registry **docker push myrepo/myimage:2.0**

Run

Run a container from the Alpine version 3.9 image, name the running container “web” and expose port 5000 externally, mapped to port 80 inside the container. **docker container run --name web -p 5000:80 alpine:3.9**

Stop a running container through SIGTERM **docker container stop web**

Stop a running container through SIGKILL **docker container kill web**

List the networks **docker network ls**

List the running containers (add --all to include stopped containers) **docker container ls**

Delete all running and stopped containers **docker container rm -f $(docker ps -aq)**

Print the last 100 lines of a container’s logs **docker container logs --tail 100 web**