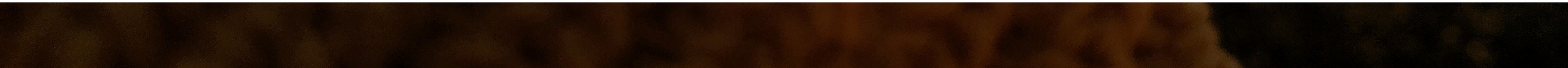
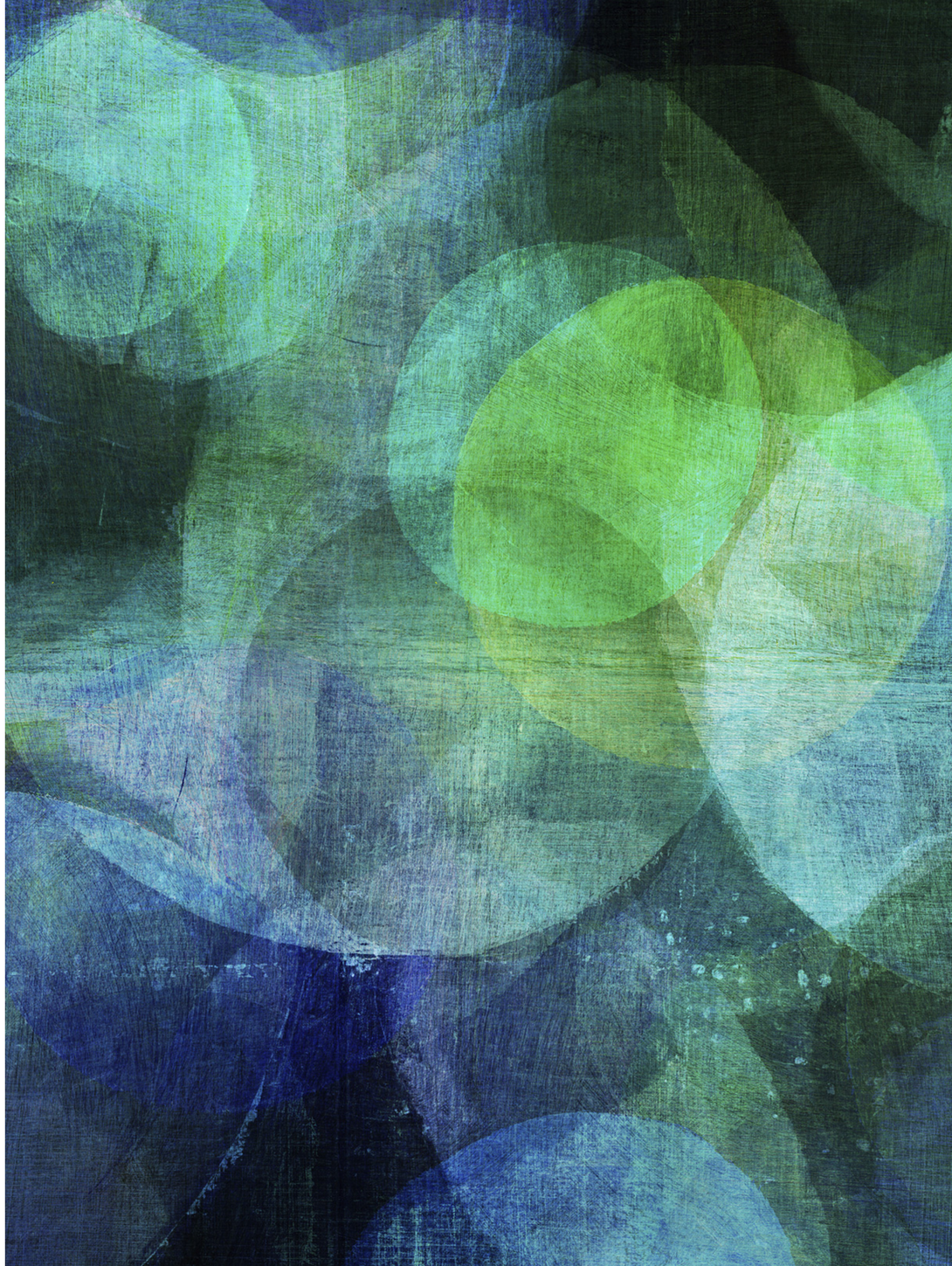
INTRODUCTION TO DOCKER

INTRODUCTION TO DOCKER

What is Docker

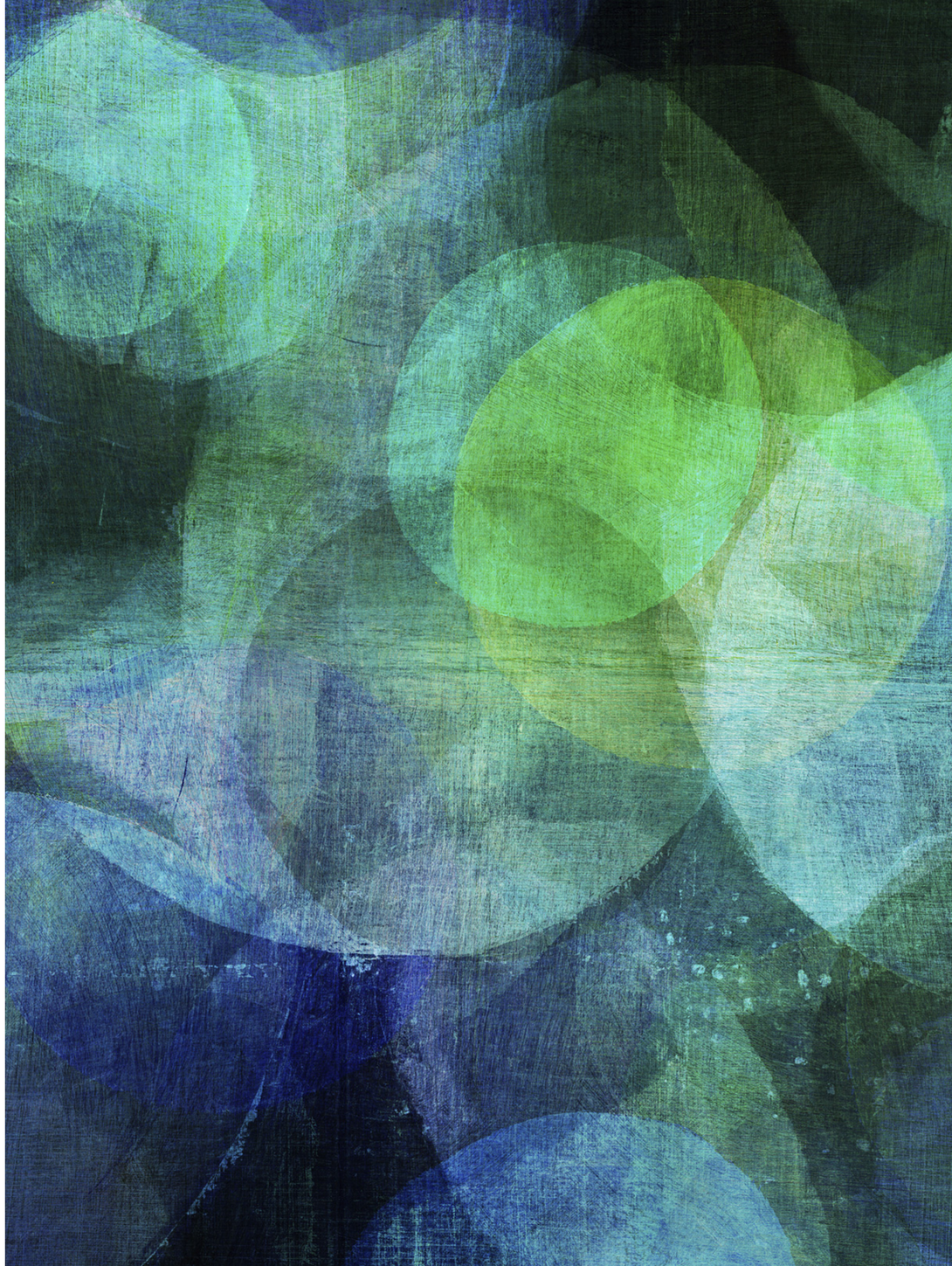


AGENDA



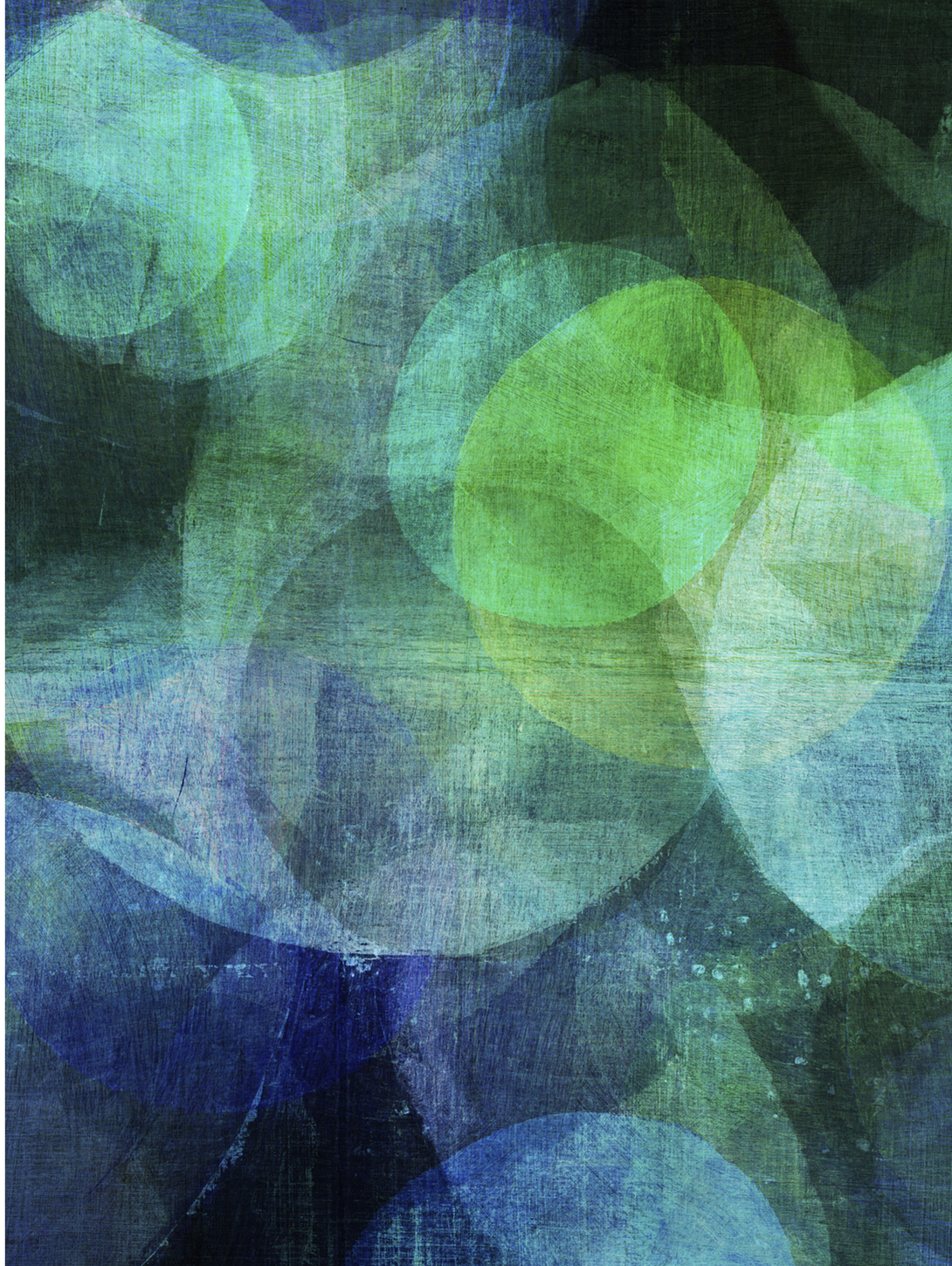
AGENDA

- What is Docker ?



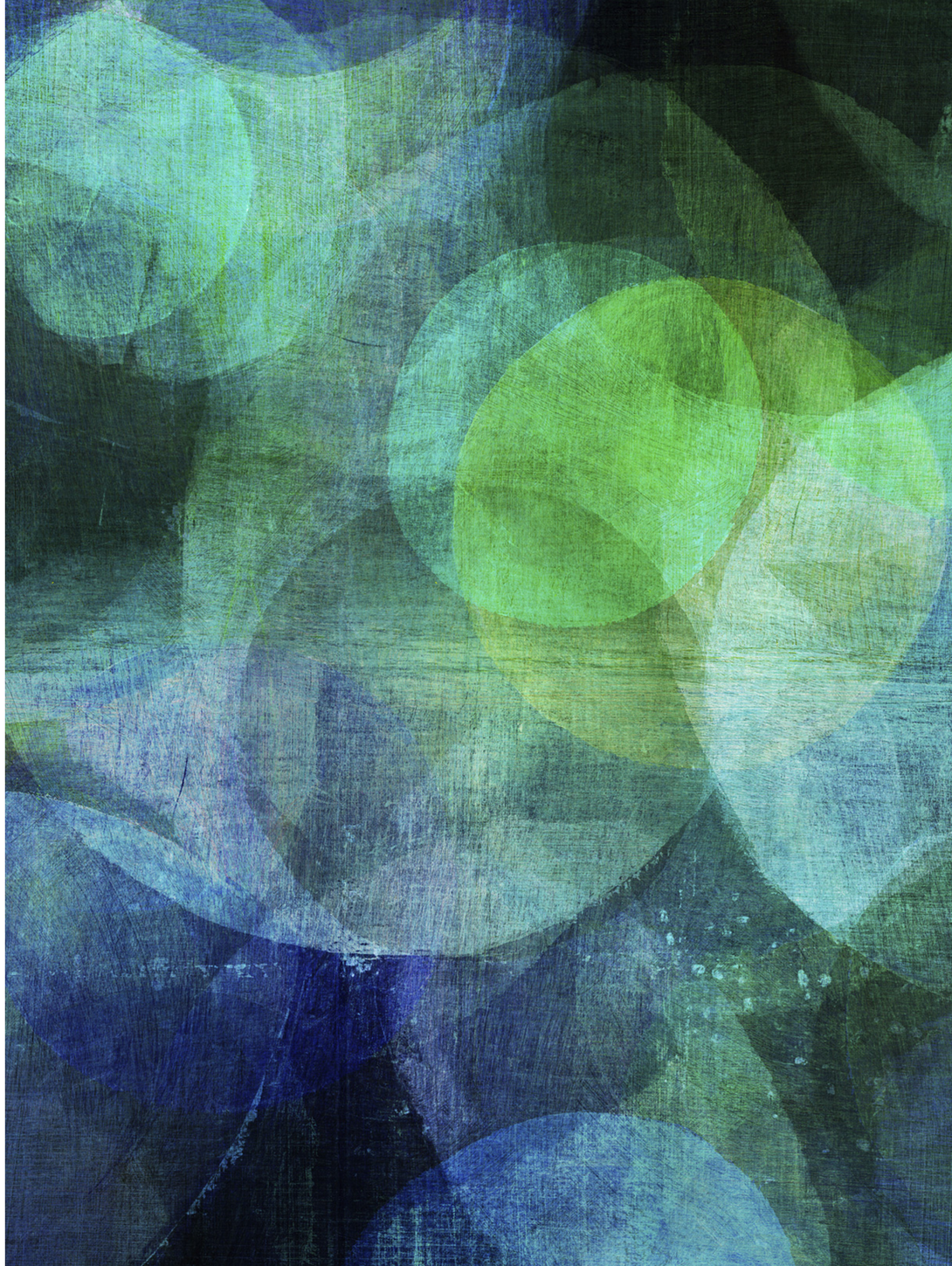
AGENDA

- What is Docker ?
- Docker Architecture



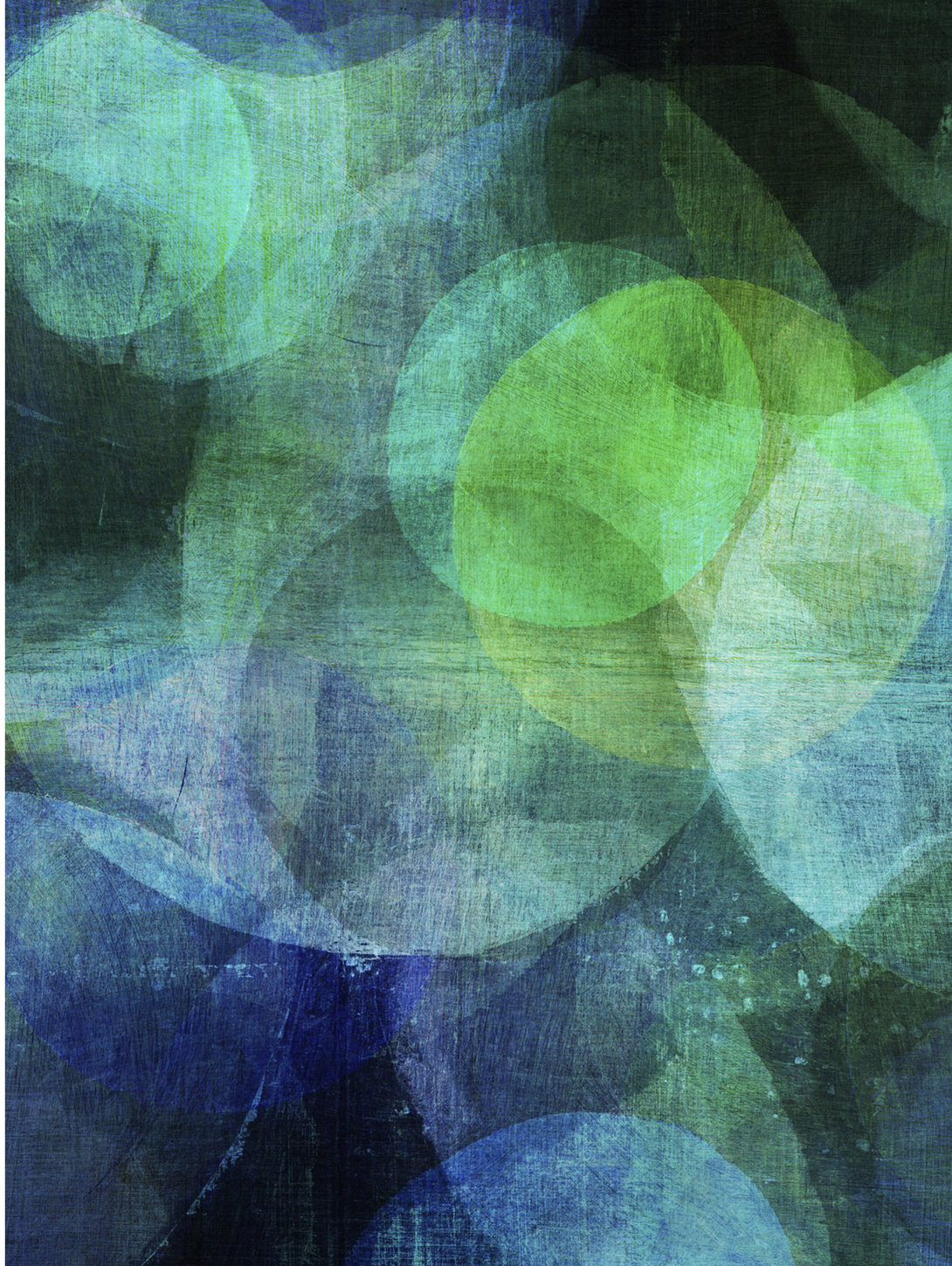
AGENDA

- What is Docker ?
- Docker Architecture
- Install Docker



AGENDA

- What is Docker ?
- Docker Architecture
- Install Docker
- Run your first container



WHAT IS DOCKER ?

WHAT IS DOCKER ?

WHAT IS 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 OBJECTS

DOCKER OBJECTS

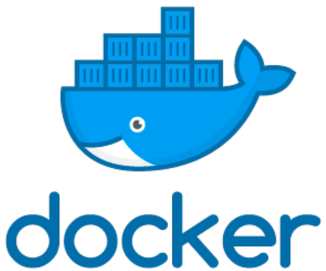
CONTAINERS

DOCKER OBJECTS

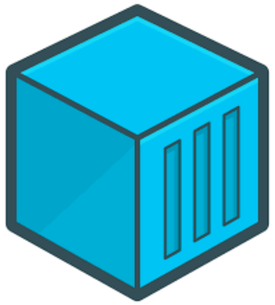
IMAGES

CONTAINERS

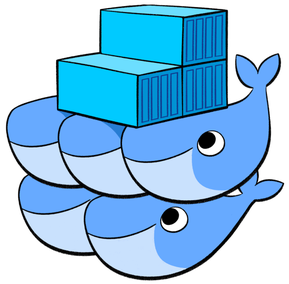
DOCKER OBJECTS



IMAGES



CONTAINERS



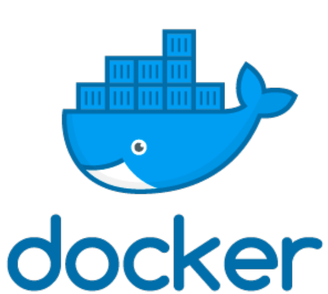
SERVICES

DOCKER IMAGE – CONTAINERS

DOCKER IMAGE - CONTAINERS

DOCKER IMAGE - CONTAINERS

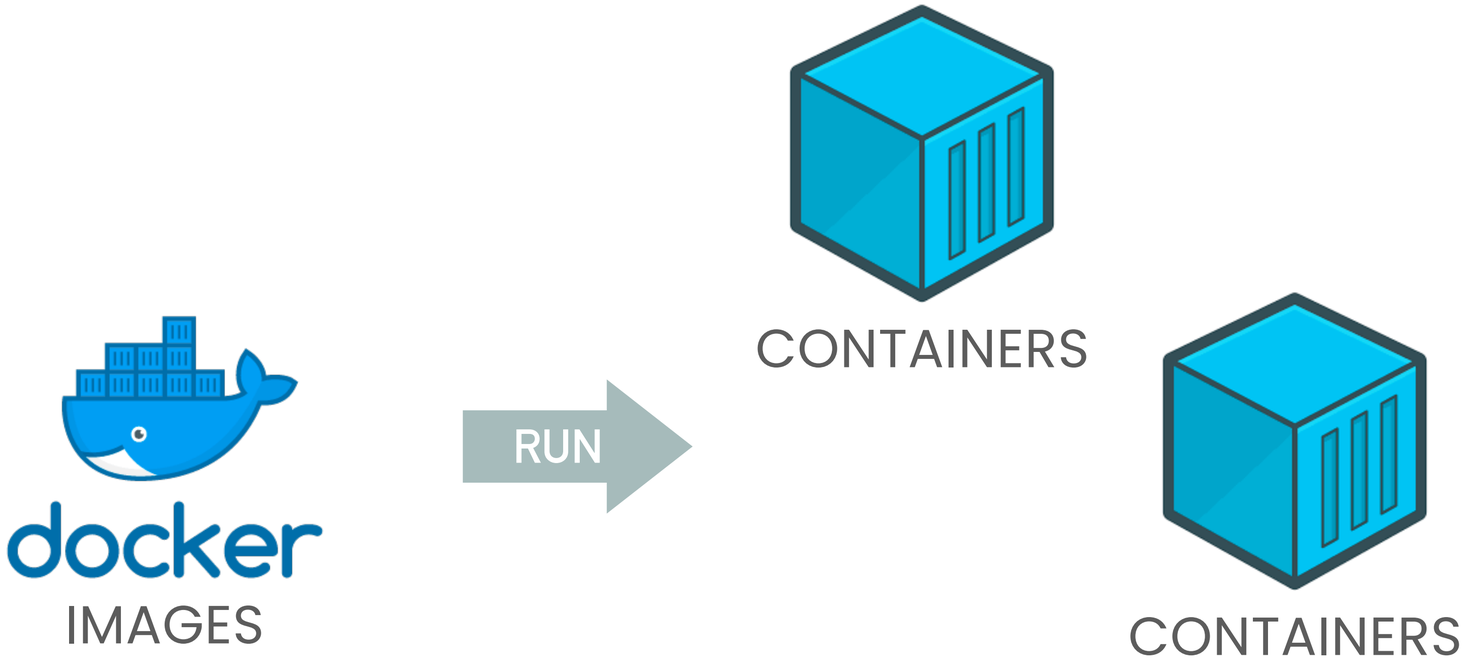
DOCKER IMAGE - CONTAINERS



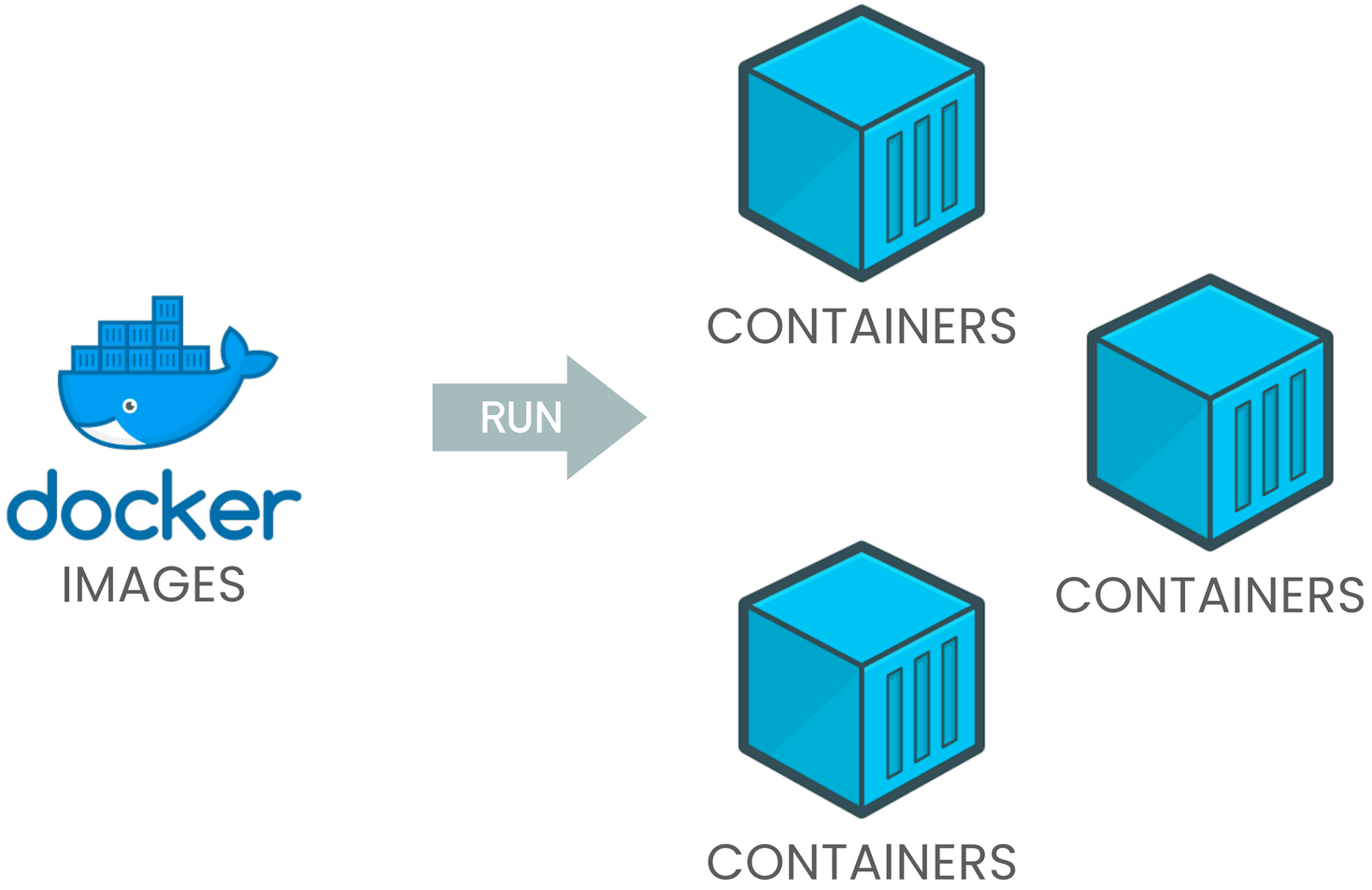
IMAGES

CONTAINERS

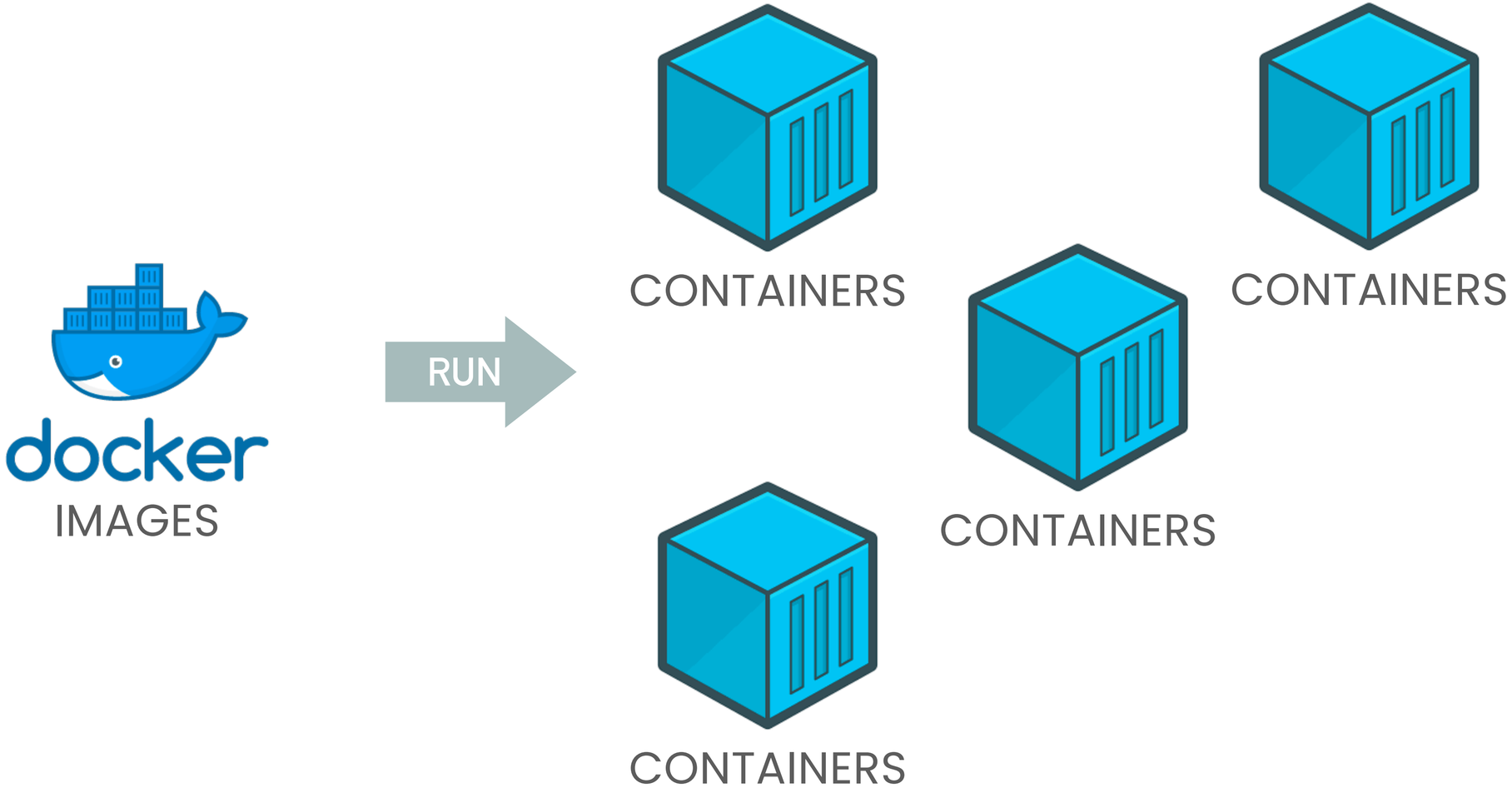
DOCKER IMAGE - CONTAINERS



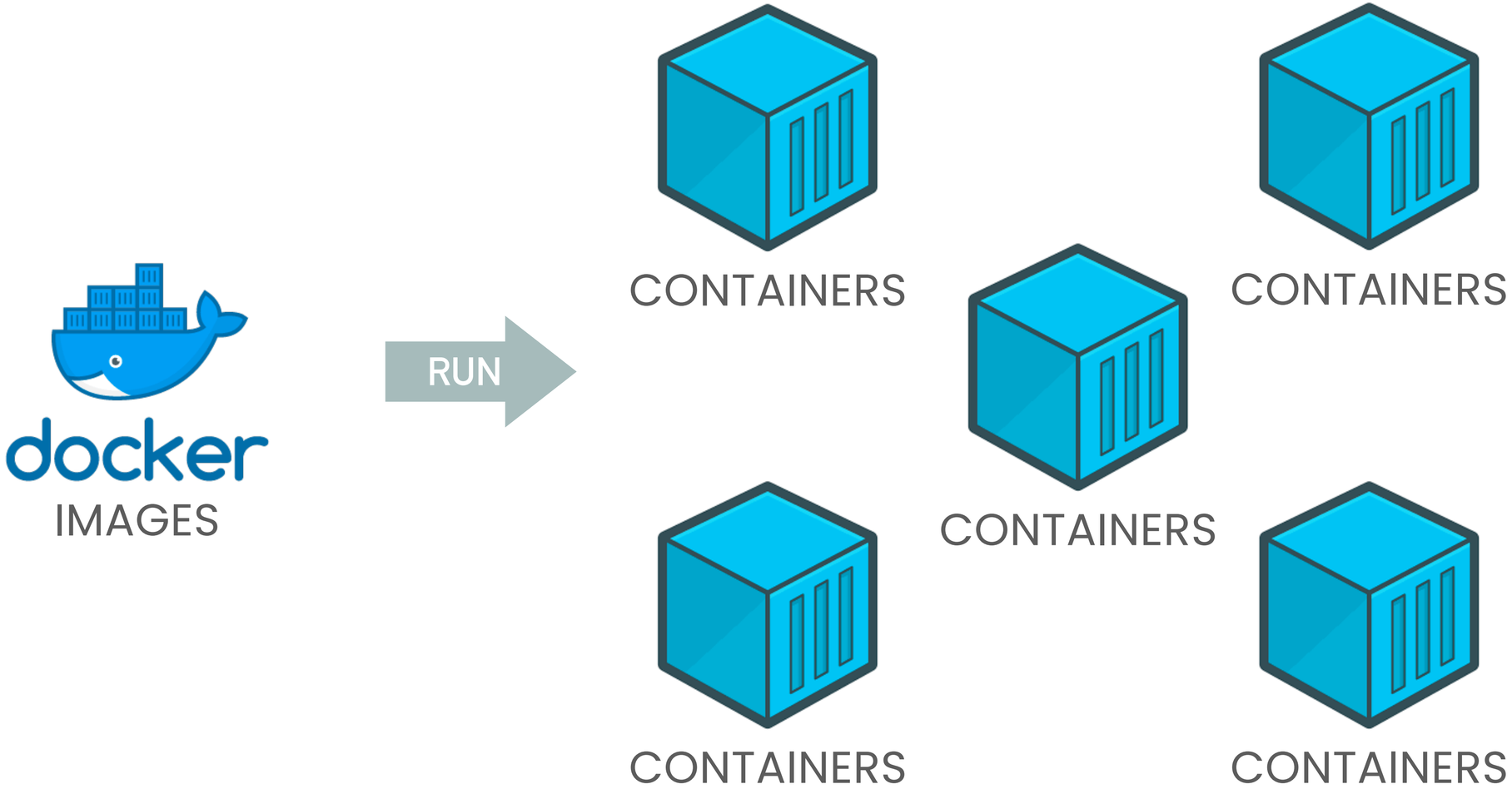
DOCKER IMAGE - CONTAINERS



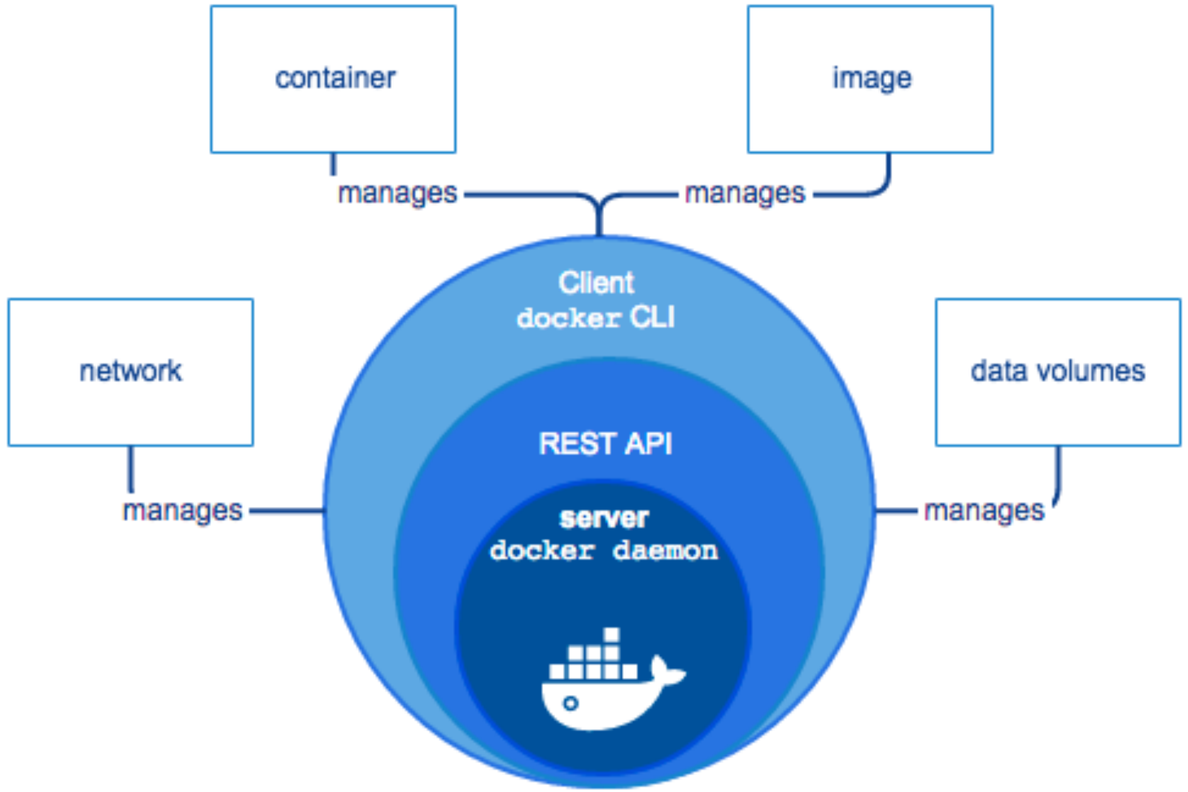
DOCKER IMAGE - CONTAINERS



DOCKER IMAGE - CONTAINERS

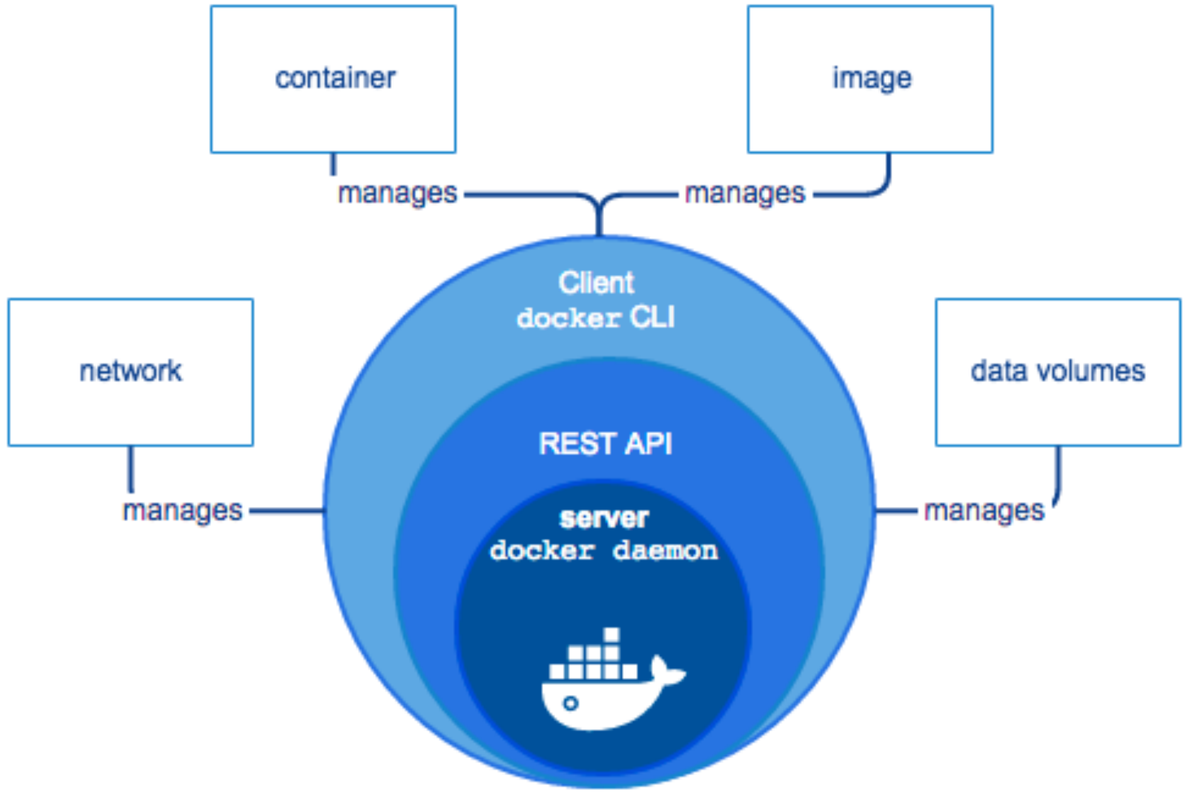


DOCKER ENGINE

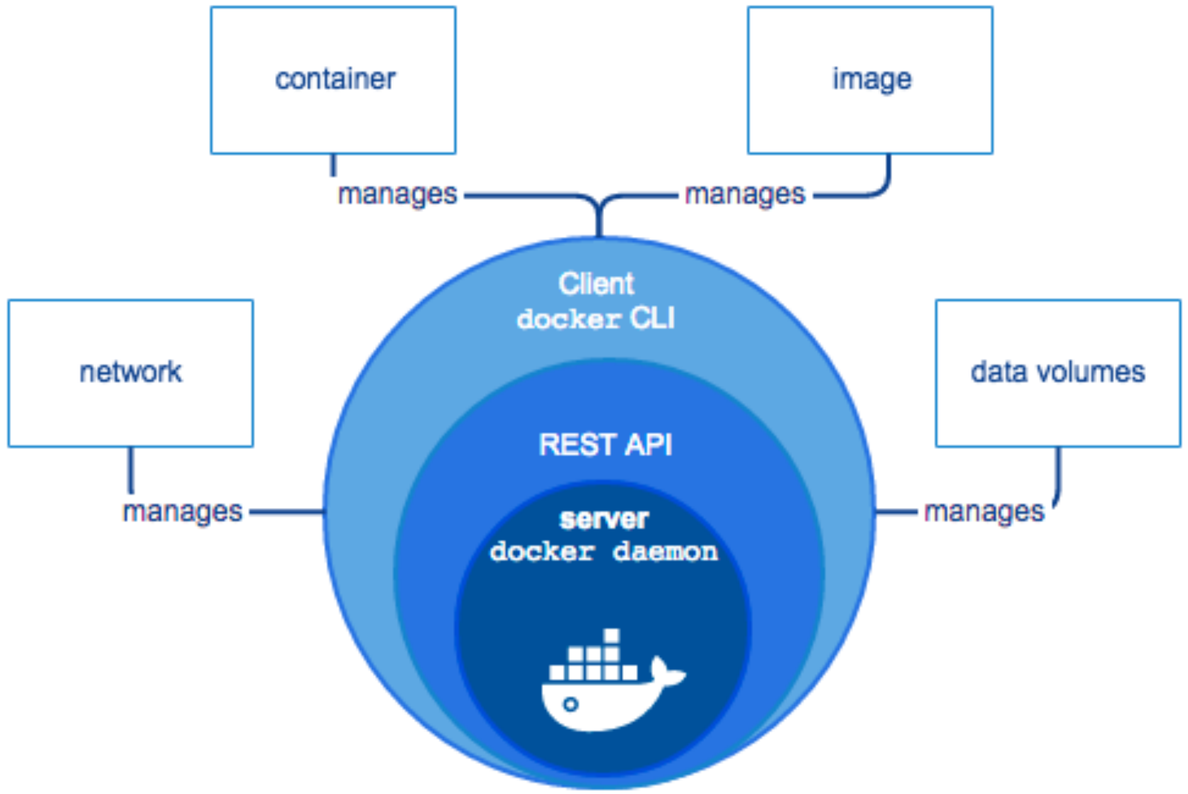


DOCKER ENGINE

Docker Engine is a client-server application with:



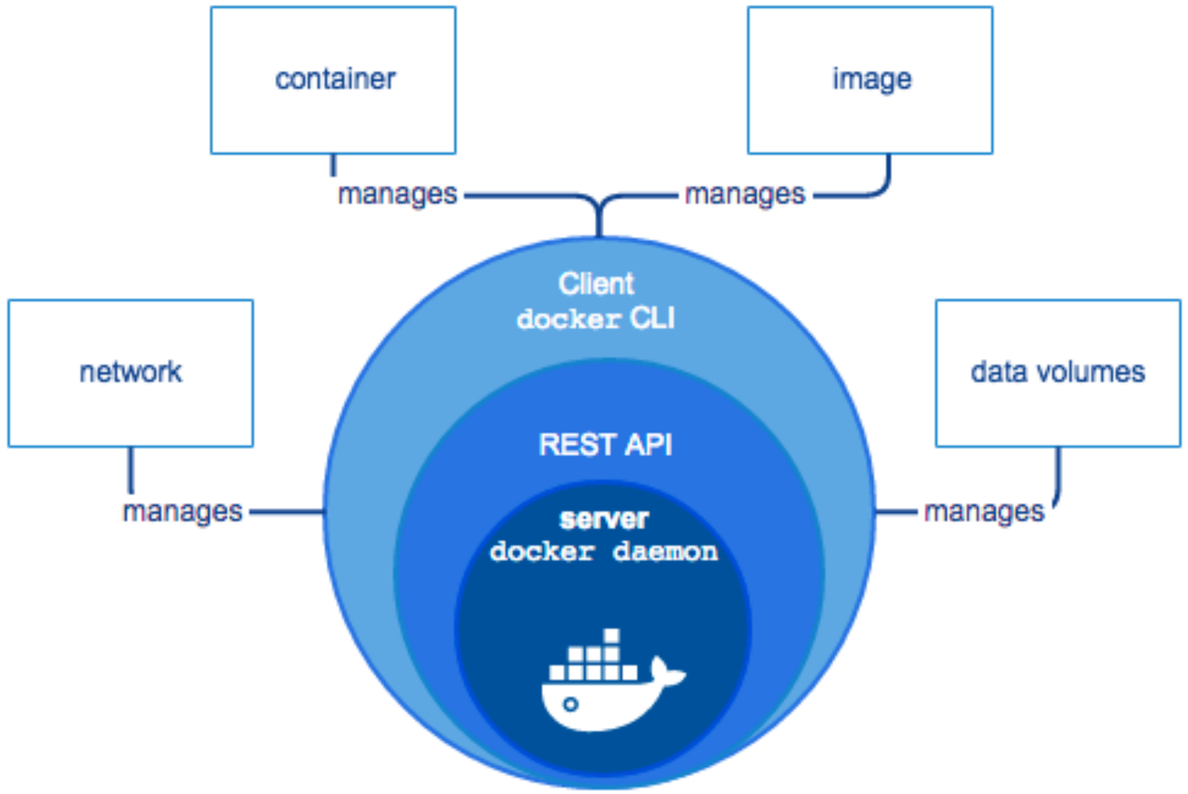
DOCKER ENGINE



Docker Engine is a client-server application with:

- *A server* which is a type of long-running program called a daemon process (the dockerd command).

DOCKER ENGINE

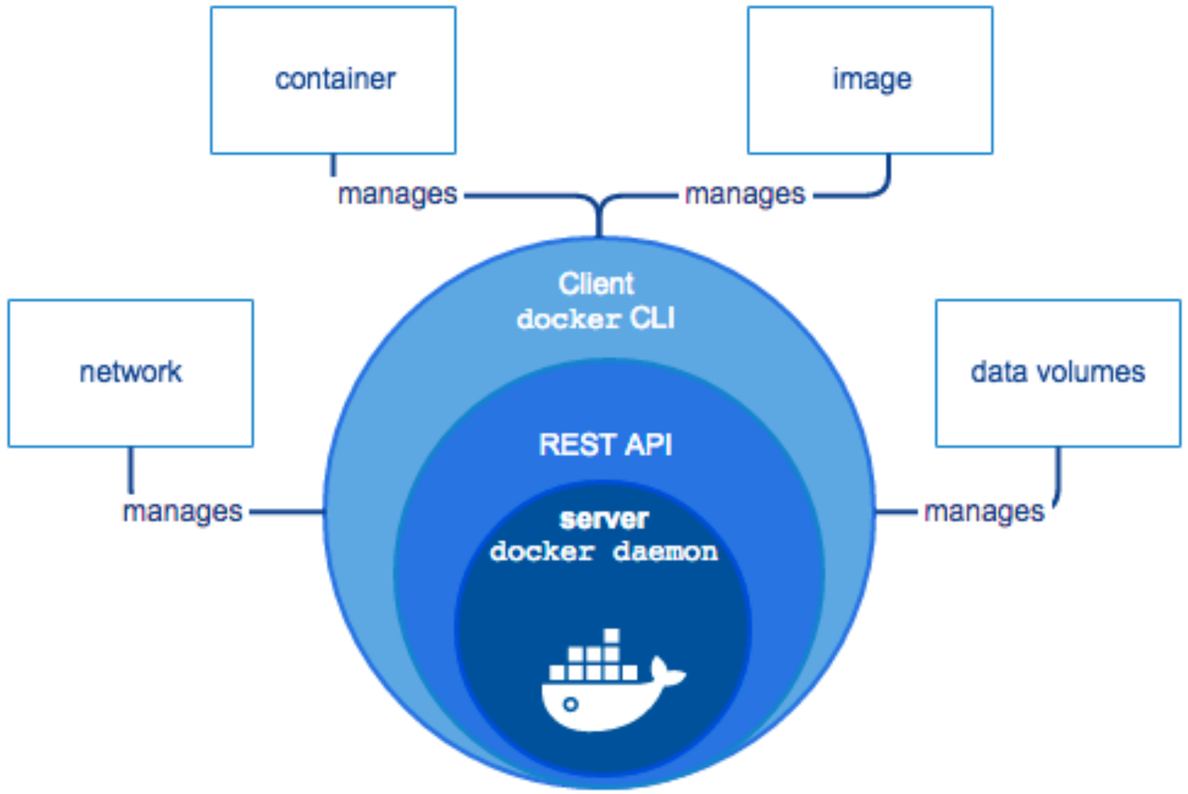


Docker Engine is a client-server application with:

- *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.

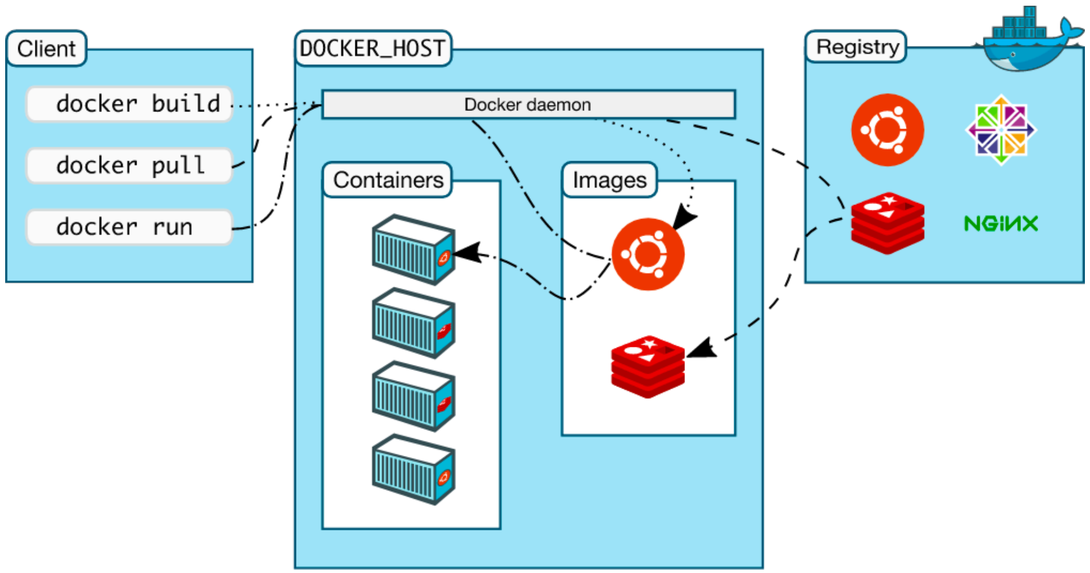
DOCKER ENGINE



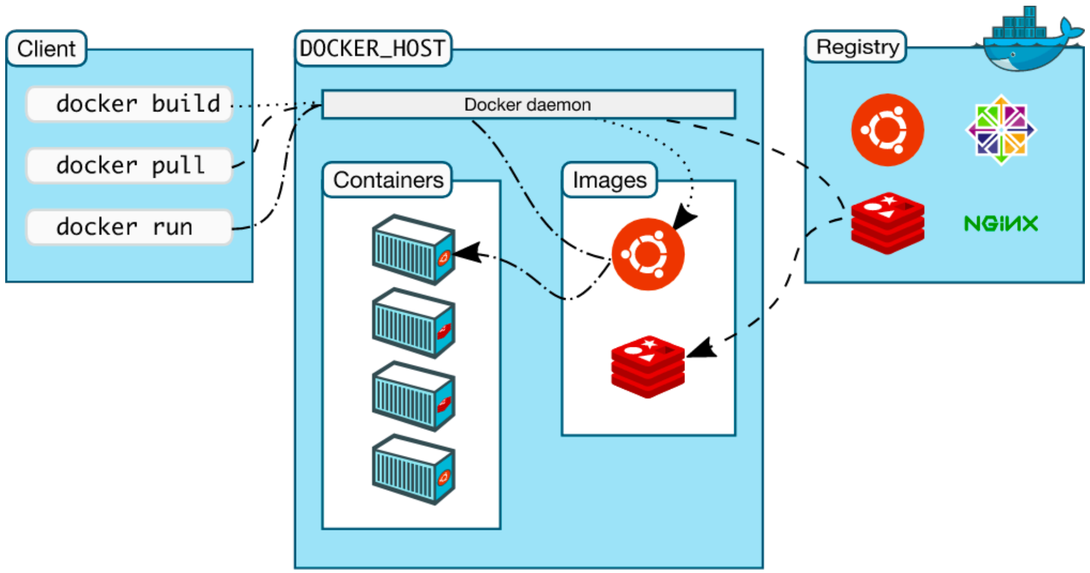
Docker Engine is a client-server application with:

- *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

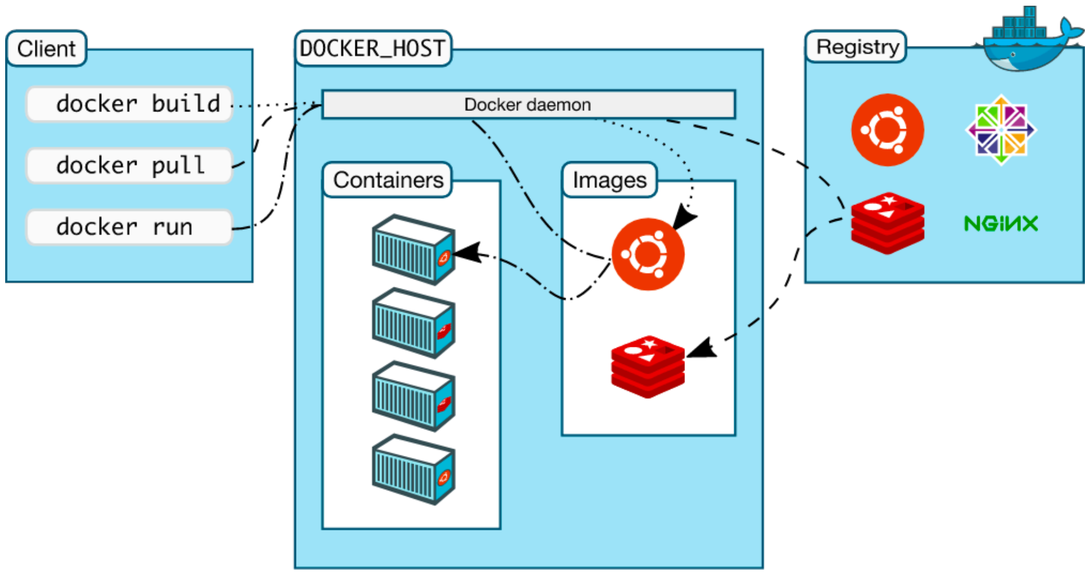


DOCKER ARCHITECTURE



The Docker daemon

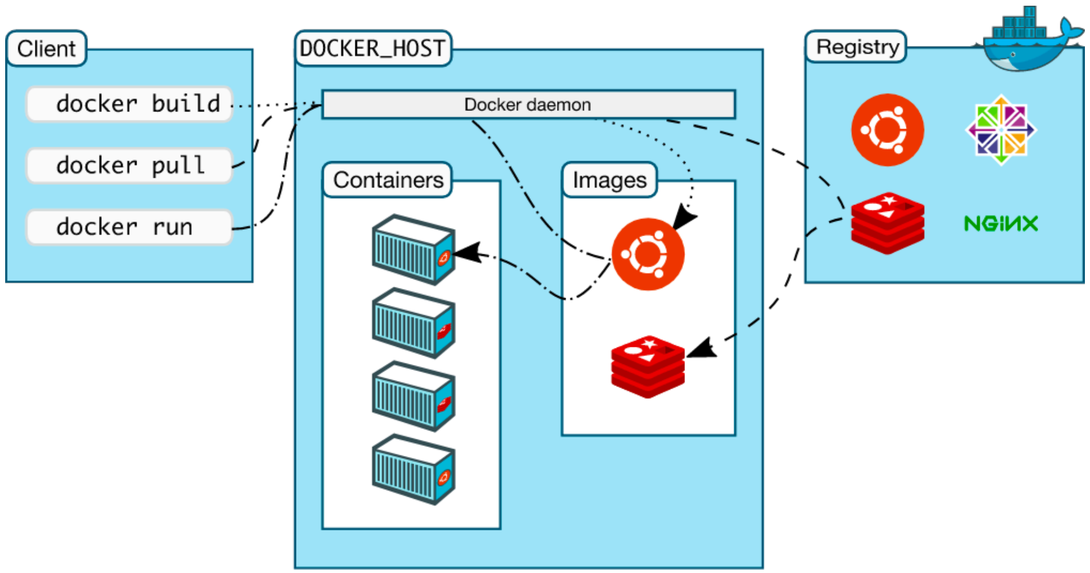
DOCKER ARCHITECTURE



The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

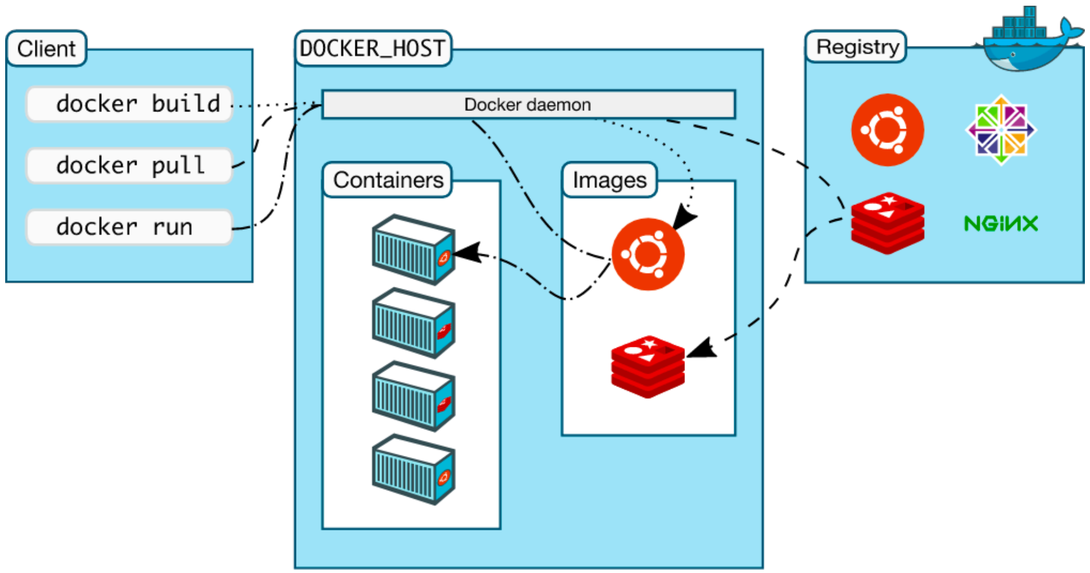
DOCKER ARCHITECTURE



The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

DOCKER ARCHITECTURE

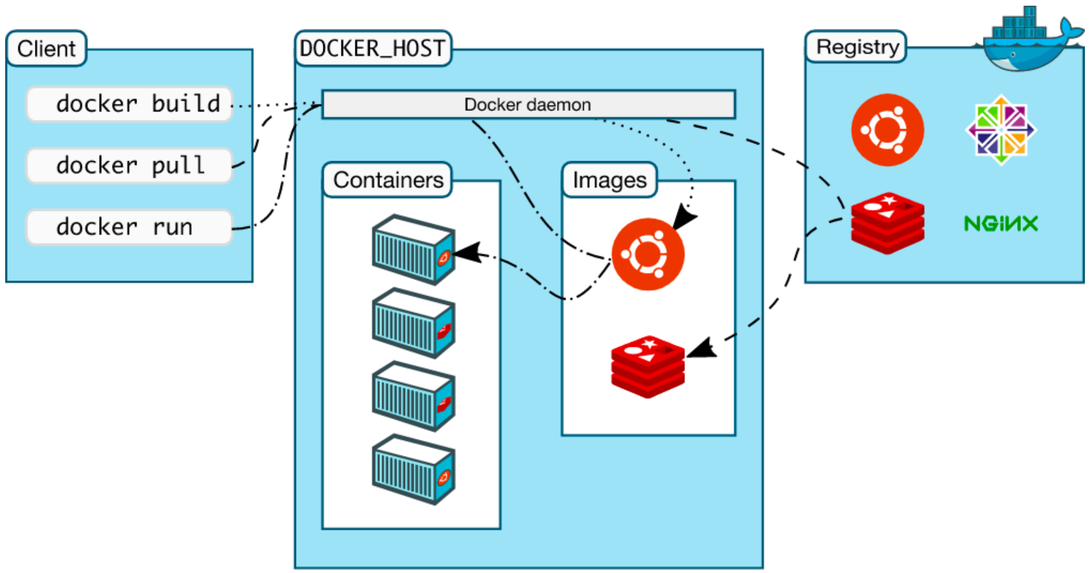


The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

DOCKER ARCHITECTURE



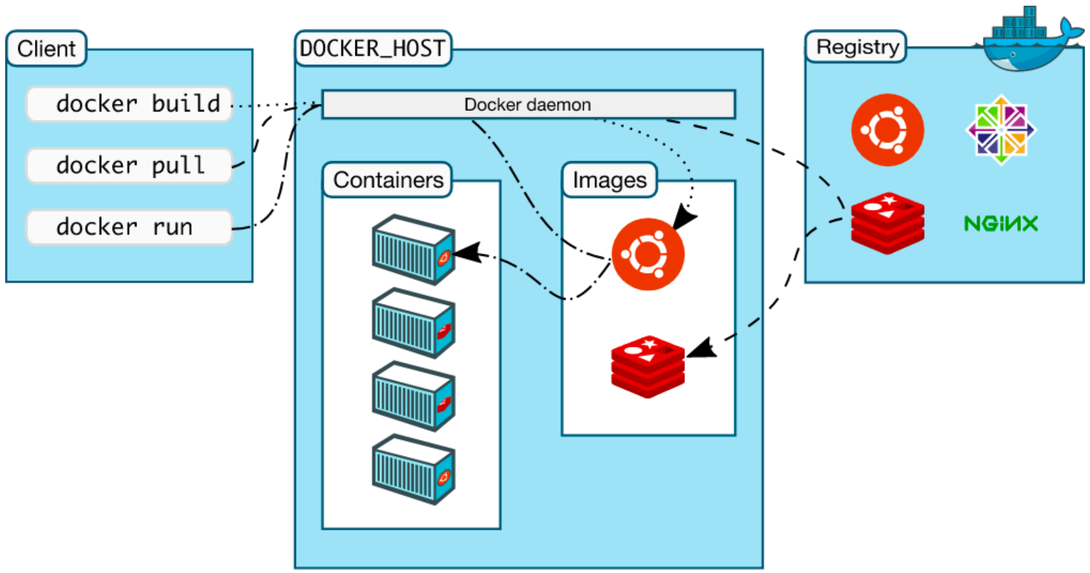
The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

DOCKER ARCHITECTURE



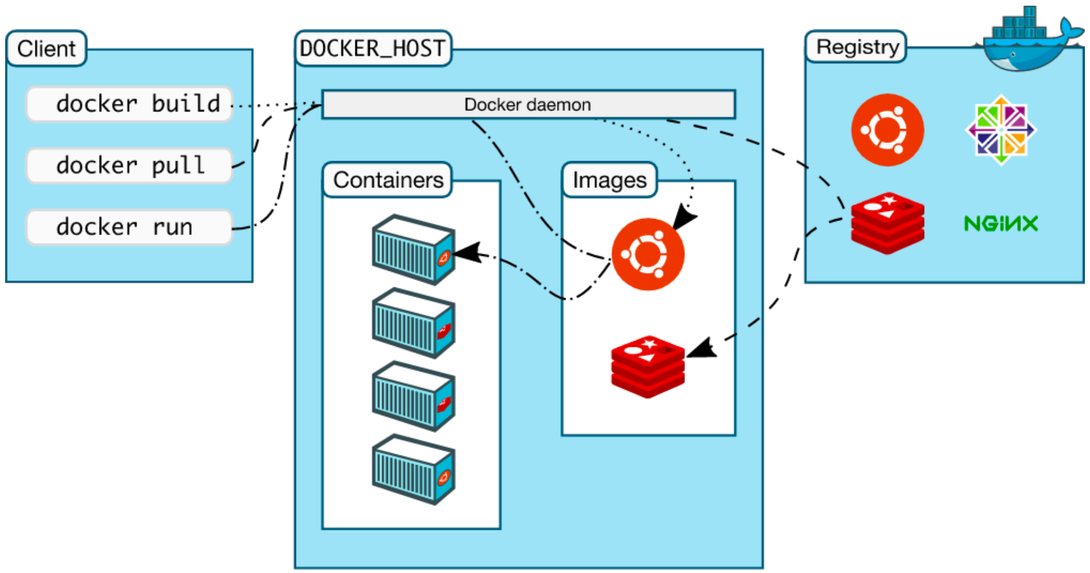
The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

DOCKER ARCHITECTURE



The Docker daemon

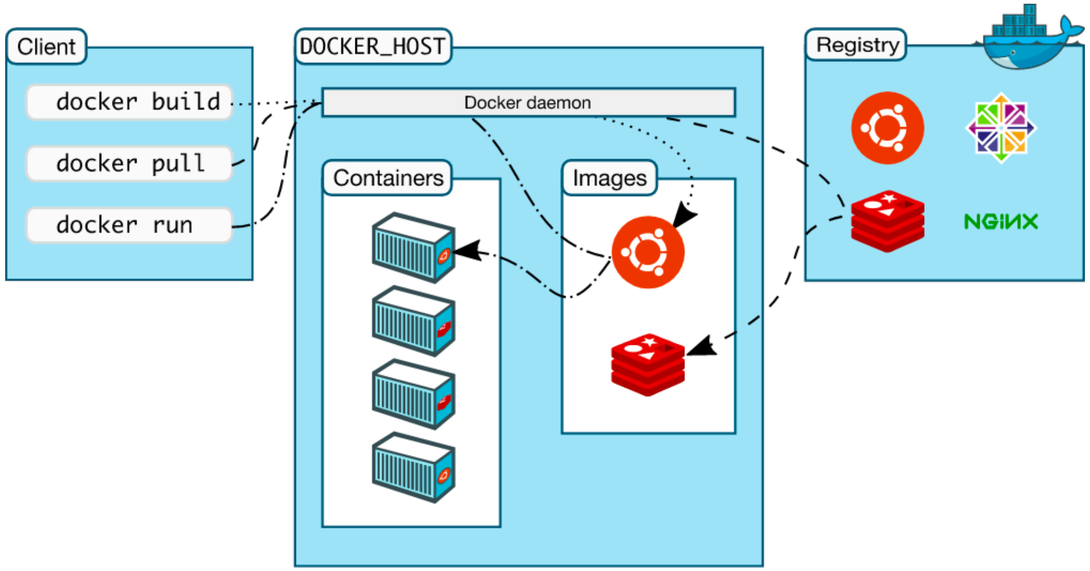
The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

The Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

Docker registries

DOCKER ARCHITECTURE



The Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

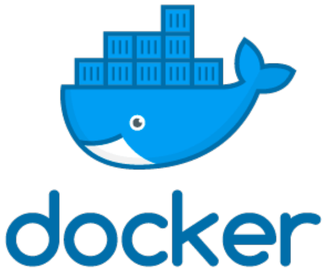
The Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

Docker registries

A Docker *registry* stores Docker images. Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry. If you use Docker Datacenter (DDC), it includes Docker Trusted Registry (DTR)

GETTING DOCKER



COMMUNITY EDITION (CE)

is ideal for **developers and small teams** looking to **get started** with Docker and **experimenting** with container-based apps.

ENTERPRISE EDITION (EE)

is designed for **enterprise development** and IT teams who build, ship, and **run business critical applications in production at scale.**

INSTALL DOCKER

Docker Desktop for Mac

A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.

Docker Desktop for Windows

A native Windows application which delivers all Docker tools to your Windows computer.

Docker for Linux

Install Docker on a computer which already has a Linux distribution installed.

INSTALL DOCKER

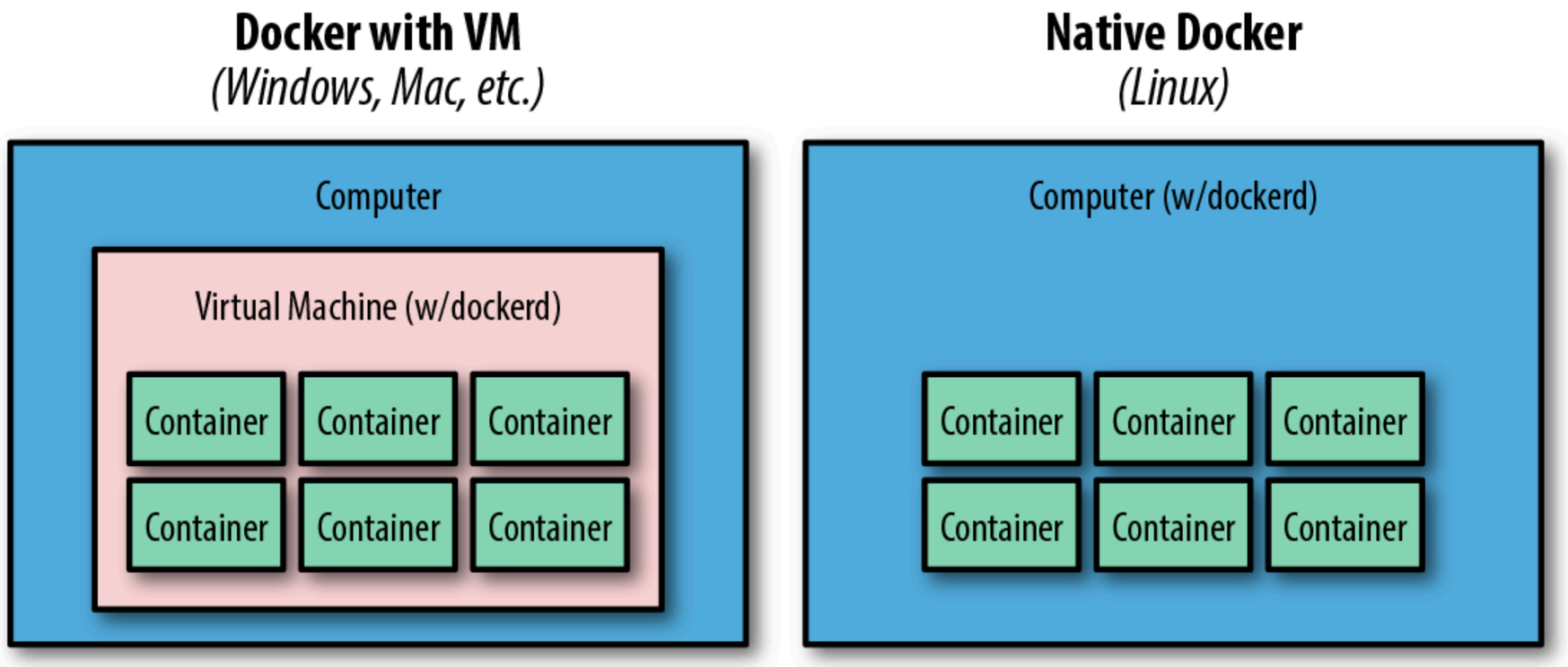
For Mac

- MacBook 2010+
- macOS 10.13+
- RAM: 4 GB+
- https://docs.docker.com/docker-for-mac/install/

For Windows

- Windows 10 64-bit: Pro, Enterprise
- Hyper-V and Containers Windows features must be enabled
- RAM: 4 GB+
- https://docs.docker.com/docker-for-windows/install/

DOCKER DESKTOP



DOCKER DESKTOP – COMPONENTS

RUN YOUR FIRST CONTAINER

```
$ docker run hello-world
```

RUN YOUR FIRST CONTAINER

IMAGES

```
$ docker run hello-world
```

RUN YOUR FIRST CONTAINER

docker

IMAGES


```
$ docker run hello-world
```

RUN YOUR FIRST CONTAINER


```
$ docker run hello-world
```

RUN YOUR FIRST CONTAINER

```
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/
```

RUN YOUR FIRST CONTAINER

Image Name

```
$ docker run hello-world
```

Or Alternative Way

```
$ docker run hello-world:latest
```

Image Tag

THANK YOU