Table of Contents

What is Docker?

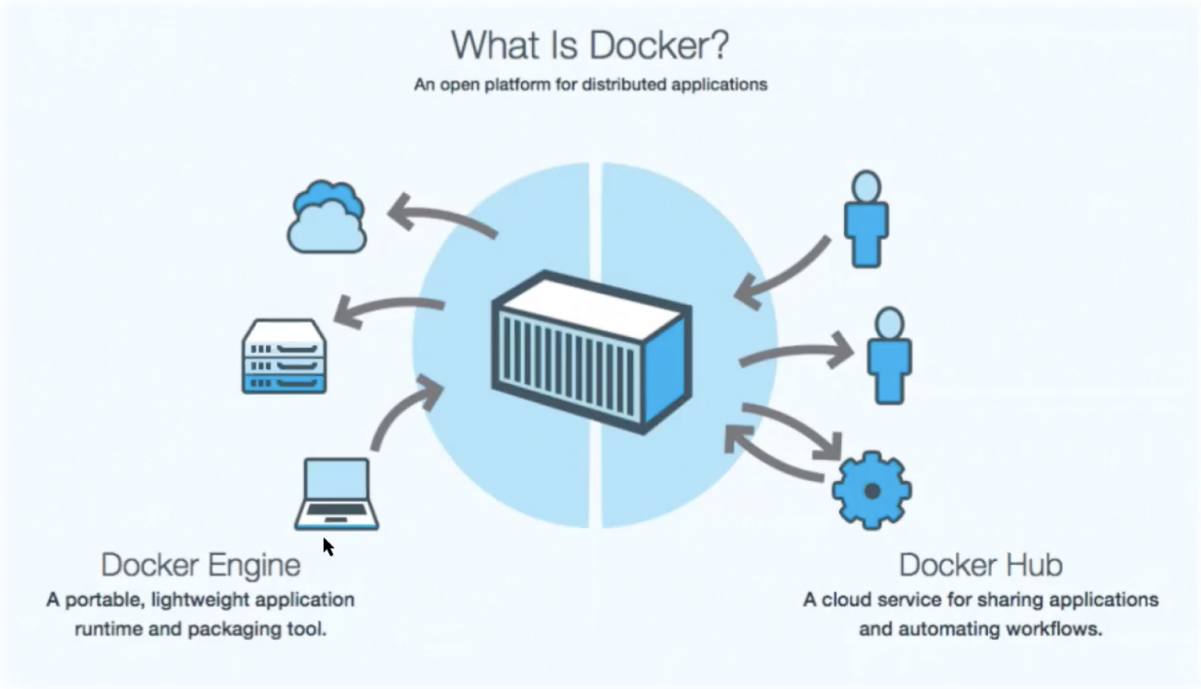
"DOCKER" refers to several things. This includes an open-source community project which started in 2013; tools from the open-source project; Docker Inc., the company that is the primary supporter of that project; and the tools that the company formally supports.

- Docker as a "Company"
- Docker as a "Product"
- Docker as a "Platform"
- Docker as a "CLI Tool"
- Docker as a "Computer Program"

```
ubuntu@clarusway: $ docker version
Client: Docker Engine - Community
 Version:           19.03.8
 API version:       1.40
 Go version:        go1.12.17
 Git commit:        afacb8b7f0
 Built:             Wed Mar 11 01:25:46 2020
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.8
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.17
  Git commit:       afacb8b7f0
  Built:            Wed Mar 11 01:24:19 2020
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.13
  GitCommit:        7ad184331fa3e55e52b890ea95e65ba581ae3429
 runc:
  Version:          1.0.0-rc10
  GitCommit:        dc9208a3303feef5b3839f4323d9beb36df0a9dd
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
ubuntu@clarusway: $
```

What is Docker?



What is Container?

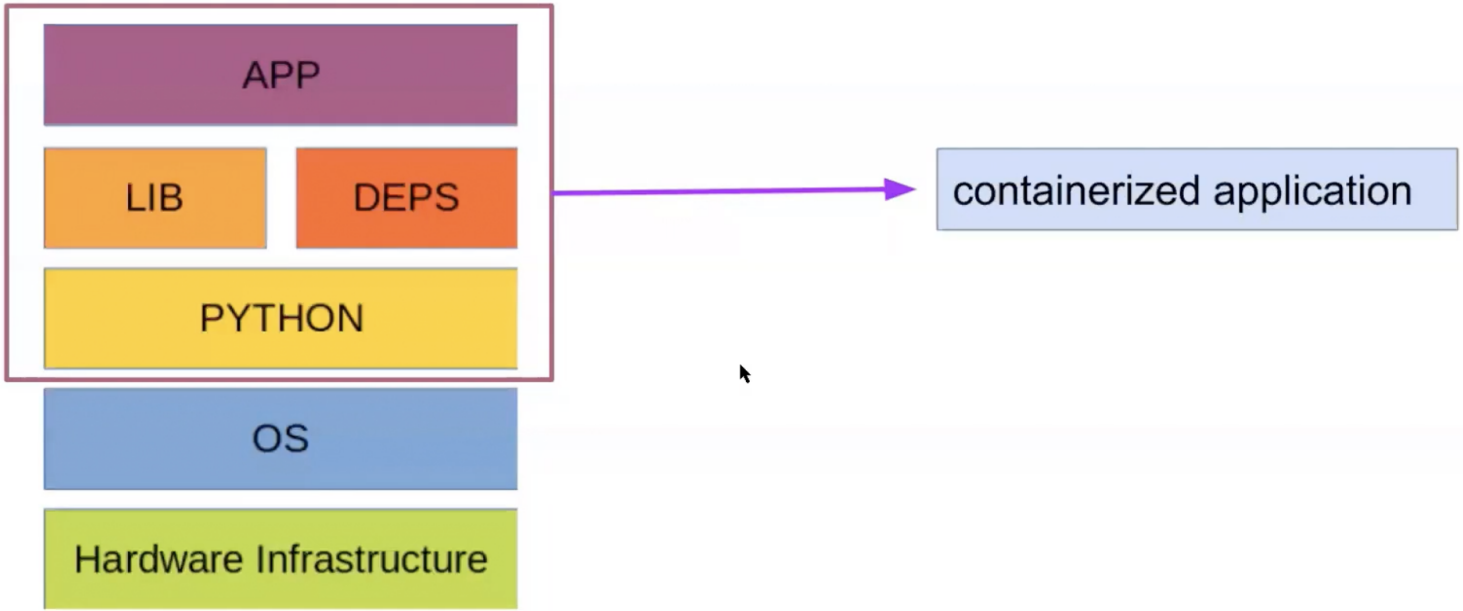
Imagine you're developing an python application. In order to do so you will setup an environment with python installed in it. You do your work on a laptop and your environment has a specific configuration. The application you're developing relies on that configuration and is dependent on specific libraries, dependencies, and files.

Once the application is developed, it needs to be tested by the tester. Now the tester will again set up same environment.

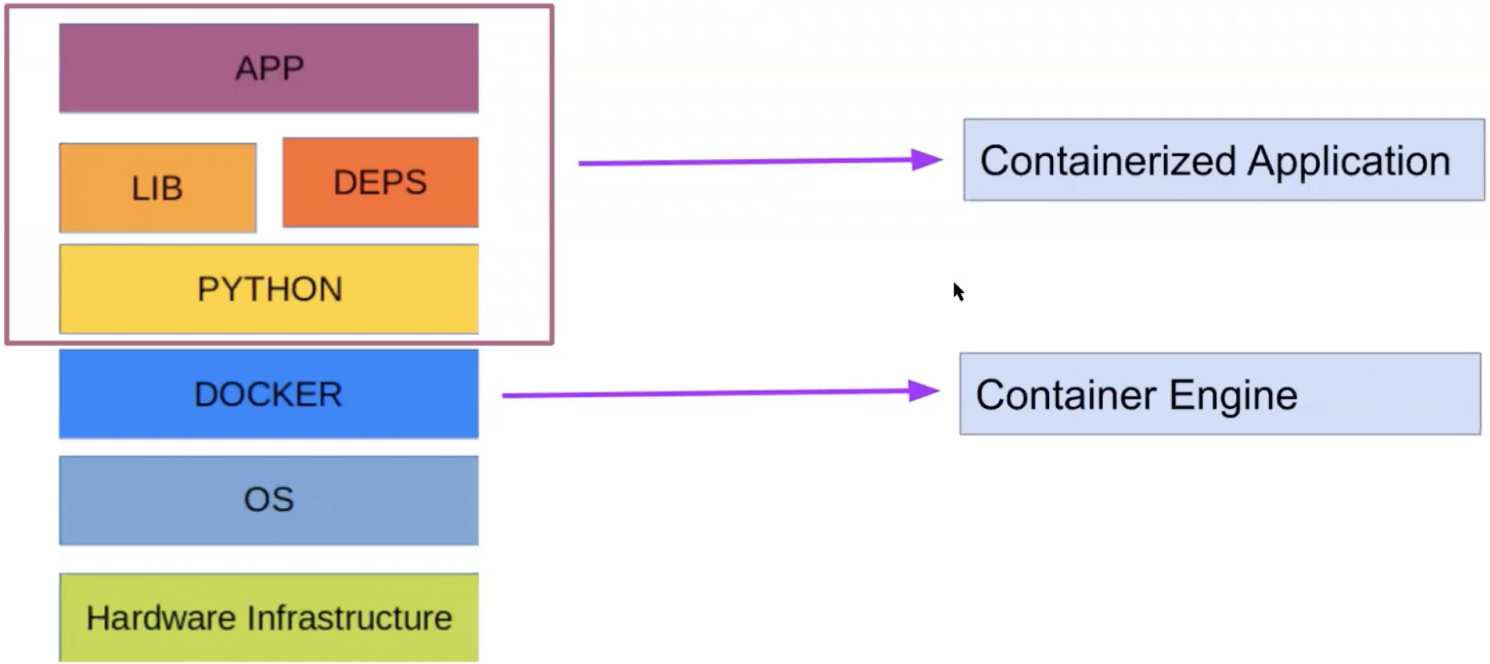
Once the application testing is done, it will be deployed on the production server. Again the production needs an environment with libraries, dependencies, files and python installed on it.

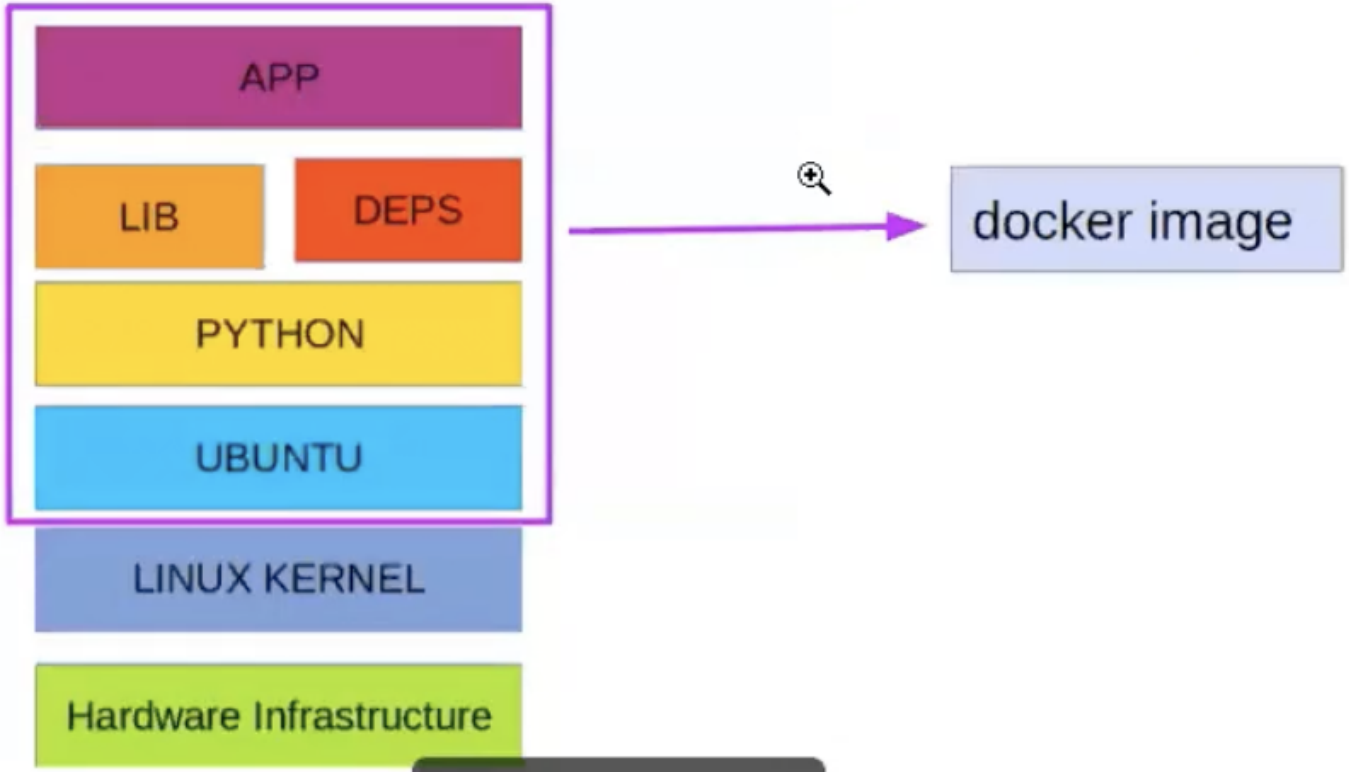
How do you make your app work across these environments, pass quality assurance, and get your app deployed without massive headaches, rewriting, and break-fixing?

What is Container?



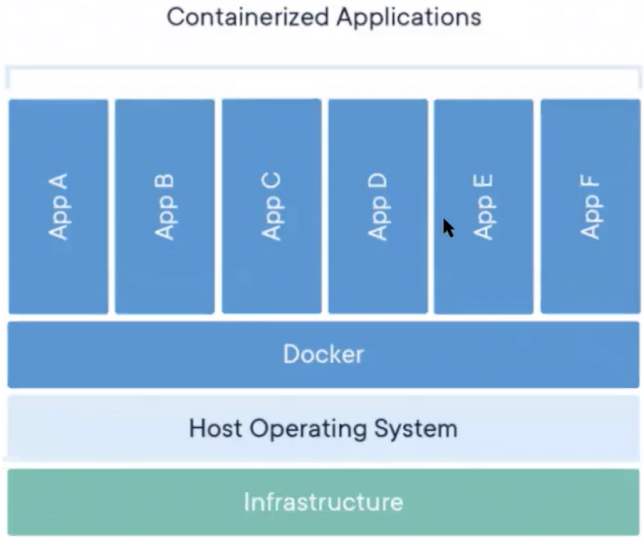
What is Container?




APP
LIB
DEPS
PYTHON
UBUNTU
LINUX KERNEL
Hardware Infrastructure
docker image

What is Container?

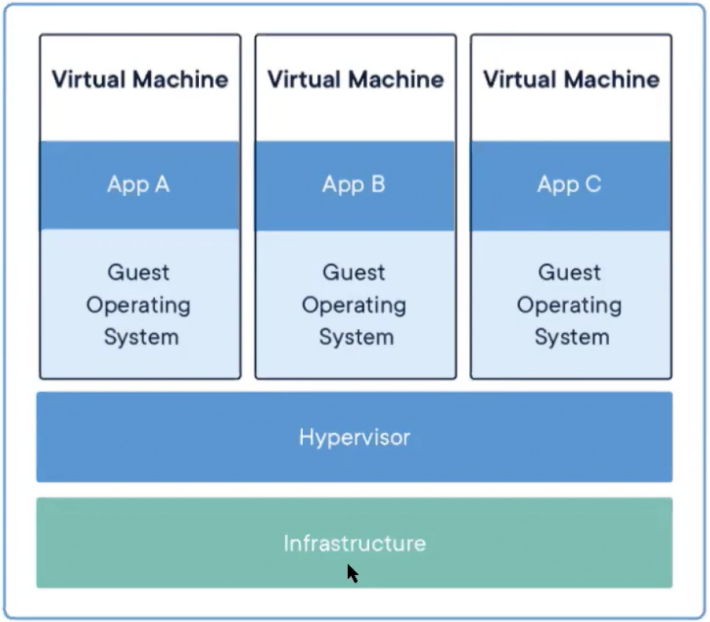
A **container** is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.



Docker vs. VMs

A virtual machine (VM) is software that runs programs or applications without being tied to a physical machine.

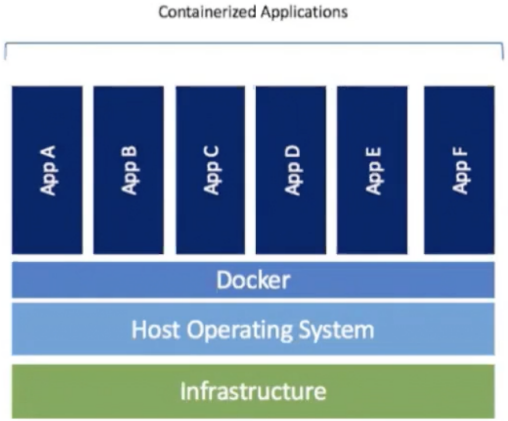
Virtual Machines are built over the physical hardware, there is a hypervisor layer which sits between physical hardware and operating systems.

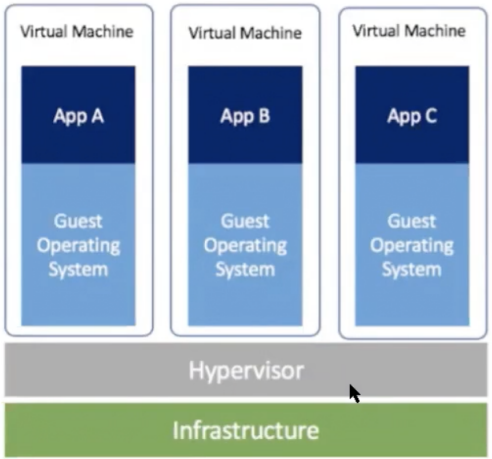


Docker vs. VMs

Unlike virtual machines where hypervisor divides physical hardware into parts, Containers are like normal operating system processes.

A virtual machine (VM) virtualize the hardware. But the containers virtualize the operating system.





Docker vs. VMs

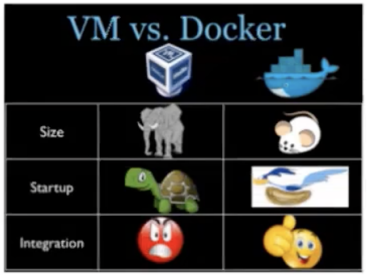
Virtual Machine

Containers

Docker containers are executed with the Docker engine rather than the hypervisor. Containers are therefore smaller than Virtual Machines and enable faster startup with better performance, less isolation and greater compatibility possible due to sharing of the host's kernel. Hence, it looks very similar to the residential flats system where **we share resources of the building.**

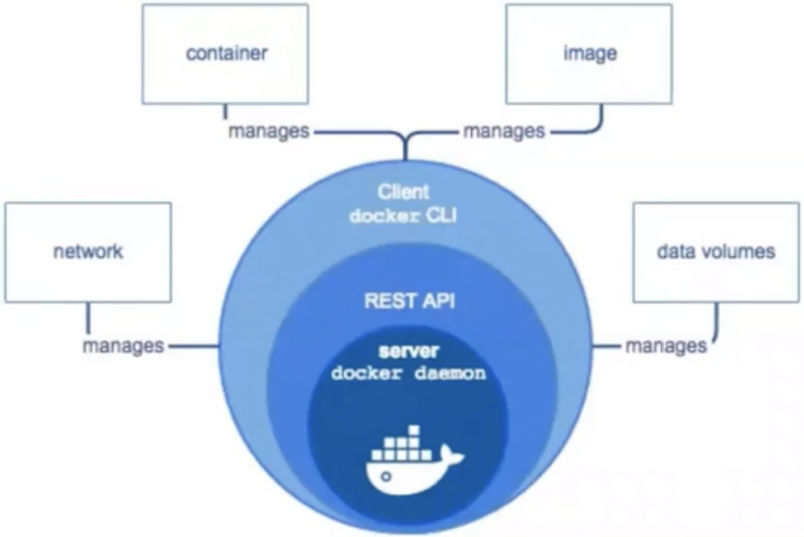
Docker vs. VMs

Docker	Virtual Machines
All containers share the same kernel of the host	Each VM runs its own OS
Containers instantiate in seconds	Boots uptime is in minutes
Images are built incrementally on top of another like layers. Lots of images/snapshots	VMs snapshots are used sparingly
Images can be diffed and can be version controlled. Dockerhub is like GitHub	Not effective diffs. Not version controlled
Can run many Docker containers on a laptop.	Cannot run more than a couple of VMS on an average laptop
Multiple Docker containers can be started from one Docker image	Only one VM can be started from one set of VMX and VMDK files



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.



Terminology

Docker Editions

- **Docker Community Edition (CE)** is ideal for Developers who are looking for experimenting with docker and creating container-based applications. It's free.

- **Docker Enterprise Edition (EE)** is a Containers-as-a-Service (CaaS) platform. Enterprise Edition Subscription packages include an integrated Docker platform and tooling for container management and security.

Terminology

Registry

- A Docker registry stores Docker images.

- **Docker Hub (Like GitHub)** is a cloud-based registry service that allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker Cloud so you can deploy images to your hosts.

- **Docker Cloud** uses the hosted Docker Cloud Registry, which allows you to publish Dockerized images on the internet either publicly or privately. Docker Cloud can also store pre-built images, or link to your source code so it can build the code into Docker images, and optionally test the resulting images before pushing them to a repository.

Terminology

Docker Client: The command-line tool that allows the user to interact with the daemon. It is the primary user interface to Docker. Accepts commands from the user and communicates back and forth with a Docker daemon.

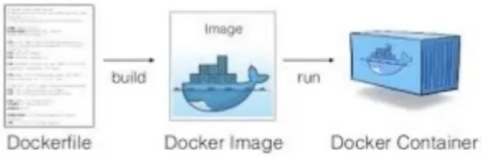
Docker Daemon: The background service running on the host that manages the building, running and distributing Docker containers. Runs on a host machine.

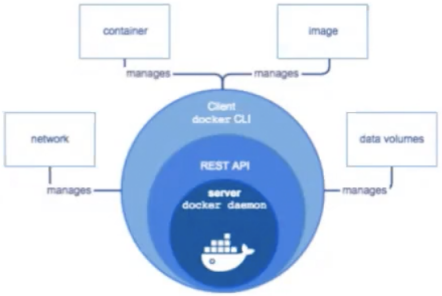
Dockerfile: Dockerfile is a text document that contains all commands a user could call on the command line to create an image.

Docker Image: Docker images are read-only templates with instructions for creating a Docker container.

Docker Container: Created from Docker images and run the actual application. It is a runnable instance of an image.

Docker Compose: Compose is a tool for defining and running multi-container Docker applications.

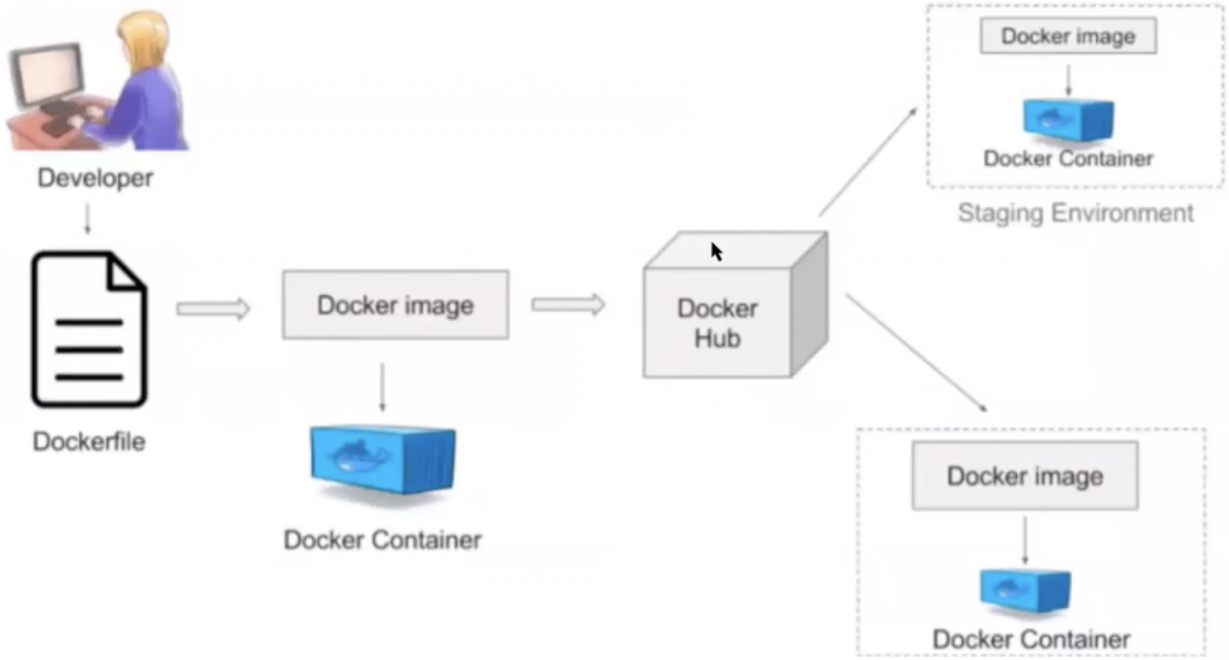




Images and Containers

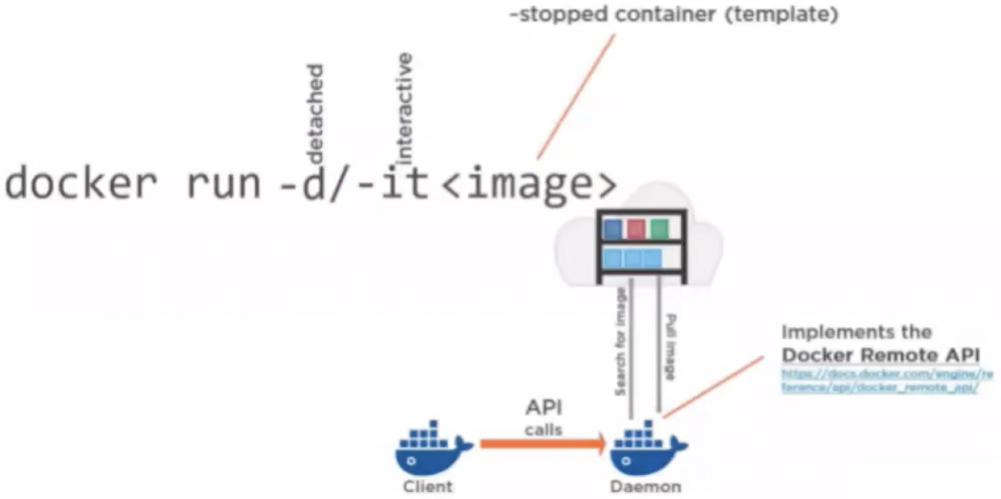
- An image is a read-only template with instructions for creating a Docker container.
- A container is a runnable instance of an image.

Images and Containers



docker run command

docker run command is used to create a container. The docker run command provides all of the ”launch” capabilities for Docker.



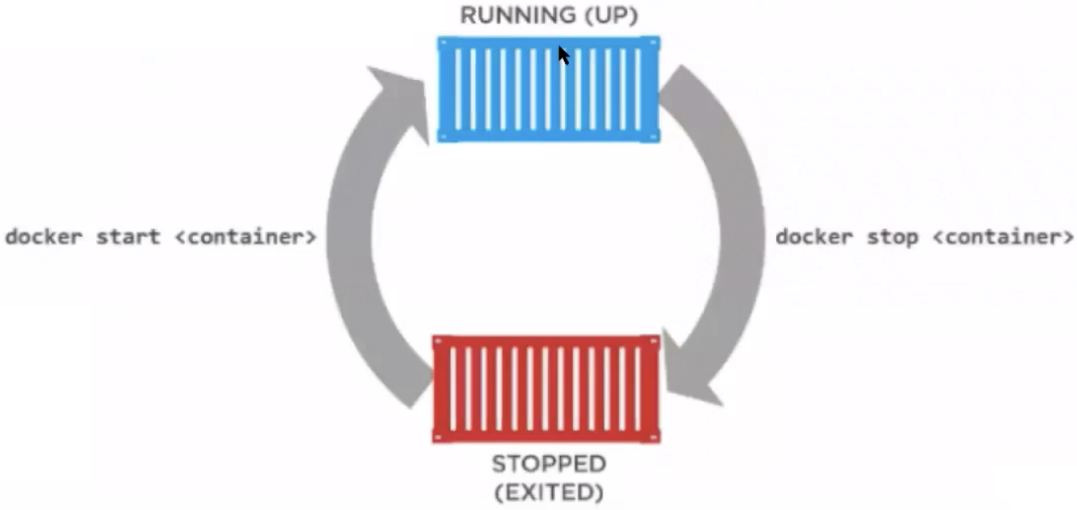
docker run command

```
$ docker run -i -t ubuntu /bin/bash
```

When we run this command, the following happens.

- If you do not have the ubuntu image locally, Docker pulls it from your configured registry, as though you had run docker pull ubuntu manually.
- Docker creates a new container, as though you had run a docker container create command manually.
- Docker starts the container and executes /bin/bash. Because the container is running interactively and attached to your terminal (due to the -i and -t flags), you can provide input using your keyboard while the output is logged to your terminal.
- When you type exit to terminate the /bin/bash command, the container stops but is not removed. You can start it again or remove it.

Starting a stopped container



Container naming

```
$ sudo docker run --name clarusway -i -t ubuntu /bin/bash
```

- Docker will automatically generate a name at random for each container we create.
- If we want to specify a particular container name in place of the automatically generated name, we can do so using the --name flag.

docker container Commands

Command	Description
docker container attach	Attach local standard input, output, and error streams to a running container
docker container create	Create a new container
docker container exec	Run a command in a running container
docker container inspect	Display detailed information on one or more containers
docker container ls	List containers
docker container prune	Remove all stopped containers
docker container rename	Rename a container
docker container rm	Remove one or more containers