Introduction to Docker

Drafted by - Aadesh Shrivastava
aadesh.shrivastava@infosys.com

Agenda

- Why we need Docker?
- What is Docker?
- Docker Platform
- Docker File

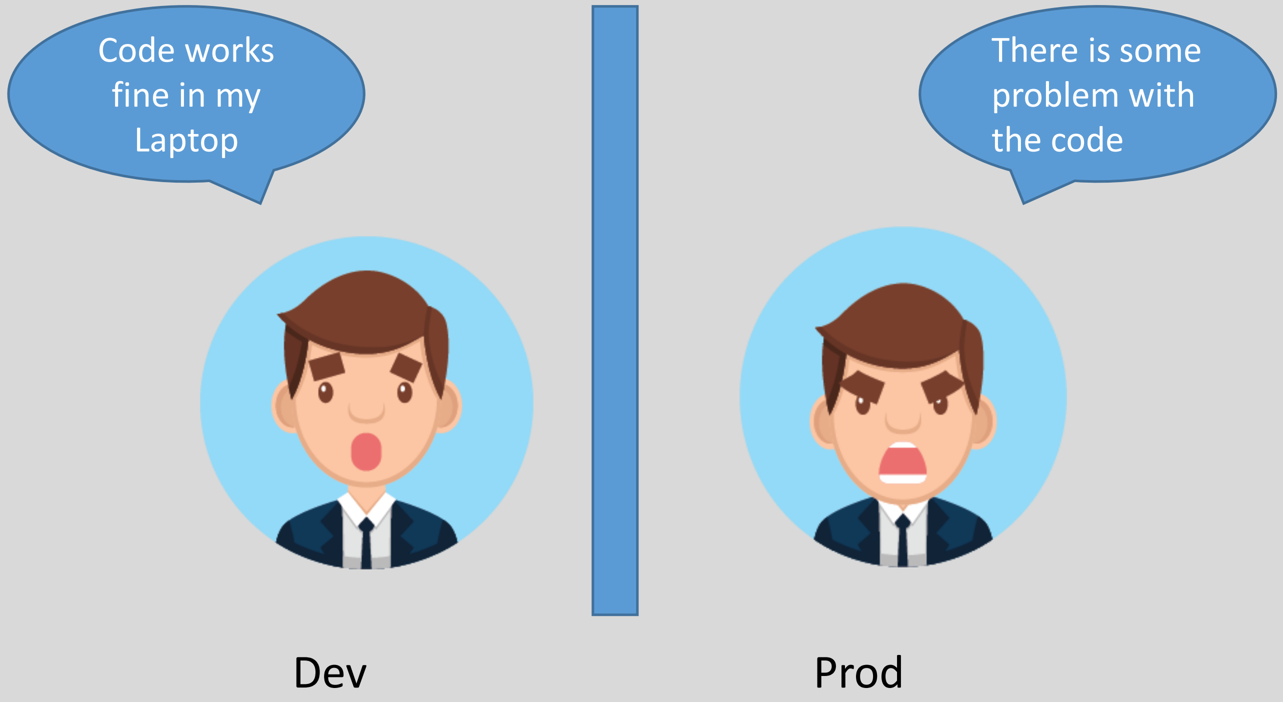
Agenda

- How containers are created?
- Docker Command Cheat sheet

Why we need Docker?

Problems Before Docker

An application works in developer’s laptop but not in testing or production. This is due to difference in environments between Dev, Test and Prod.

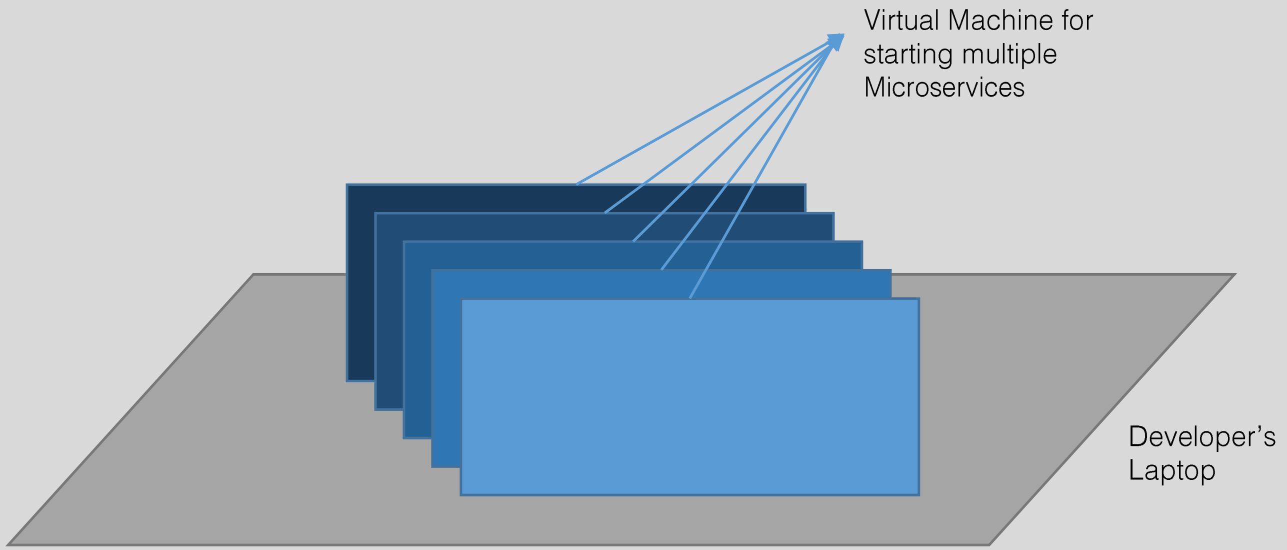

Code works fine in my Laptop
There is some problem with the code
Dev
Prod

In Dev there can be a software that is upgraded and in Prod the old version of software might be present.

Problems Before Docker

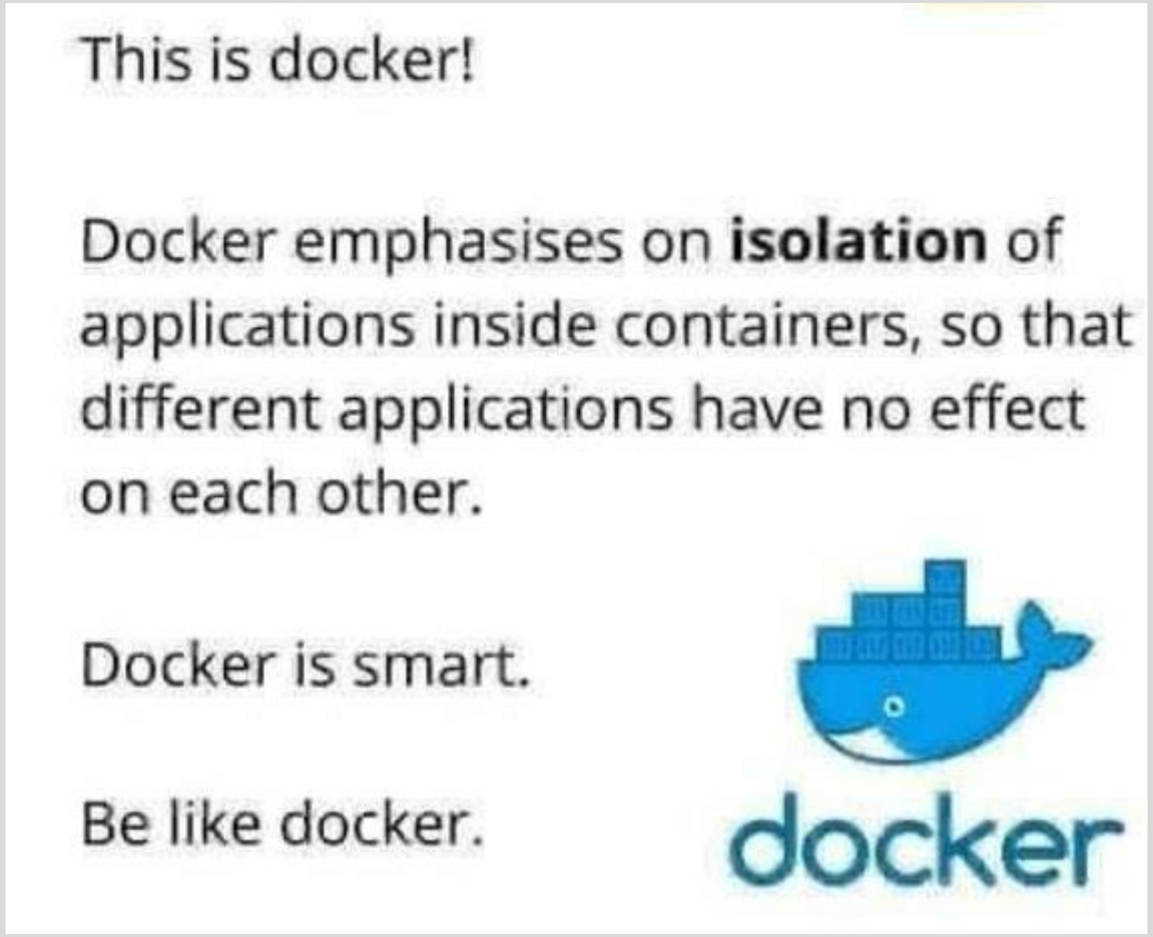
How Docker solves these problems

You can run several Microservices in the same VM by running various Docker containers for each Microservices.



In Dev there can be a software that is upgraded and in Prod the old version of software might be present.

How Docker solves these problems



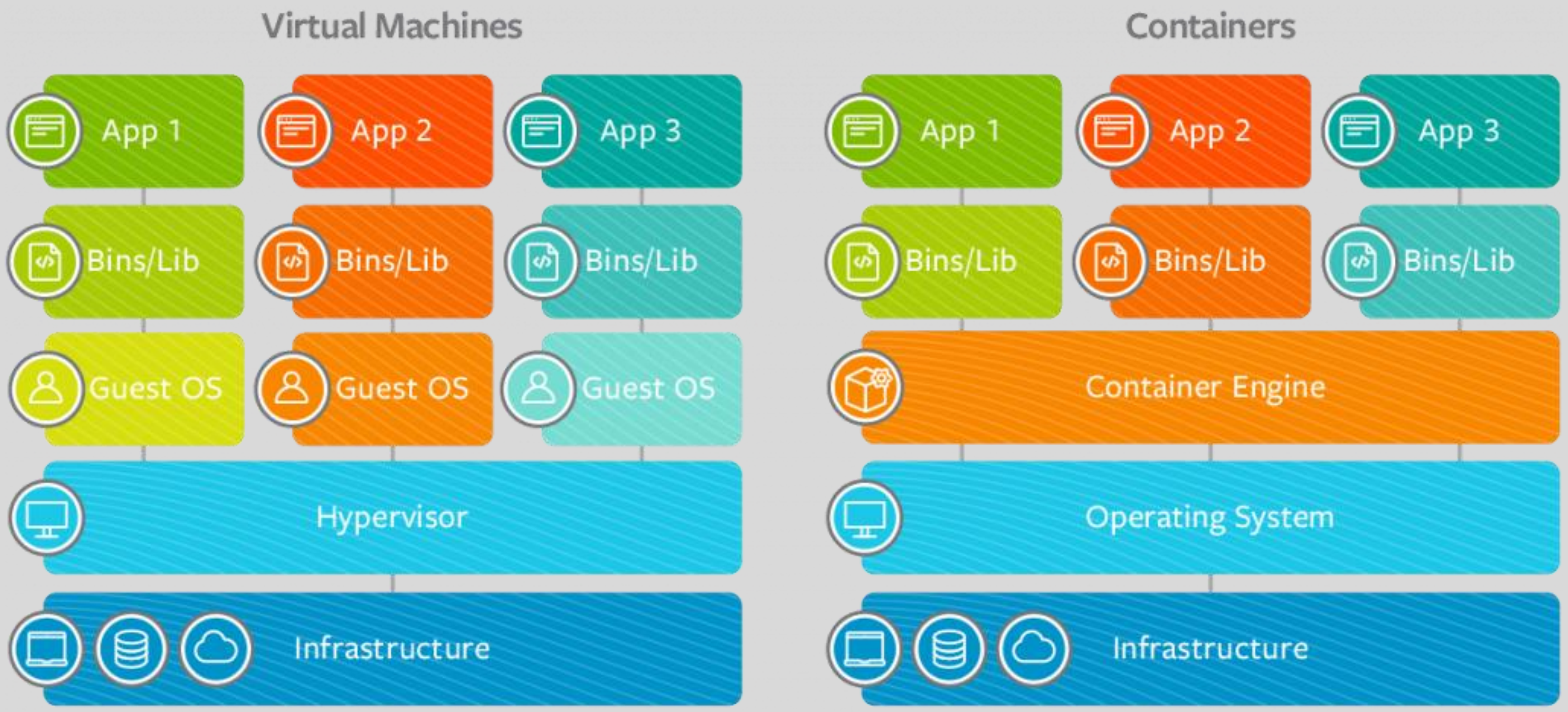
This is docker!

Docker emphasises on **isolation** of applications inside containers, so that different applications have no effect on each other.

Docker is smart.

Be like docker.

Docker vs Virtual Machines

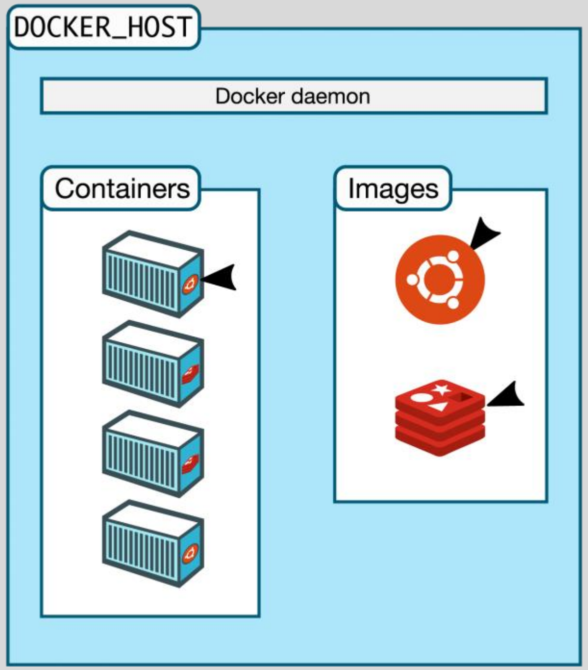


Why Docker?

- Isolation
- Lightweight
- Simplicity
- Workflow

What is Docker?

What is Docker?



- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.
- Docker containers are lightweight alternatives to Virtual Machines and it uses the host OS.
- You don’t have to pre-allocate any RAMS in containers.

Docker in a Nutshell

- In simplest terms, Docker containers consist of applications and all their dependencies.
- They share the kernel and system resources with other containers and run as isolated systems in the host operating system.
- The main aim of Docker containers is to get rid of the infrastructure dependency while deploying and running applications.
- Technically, they are just the runtime instances of Docker images.

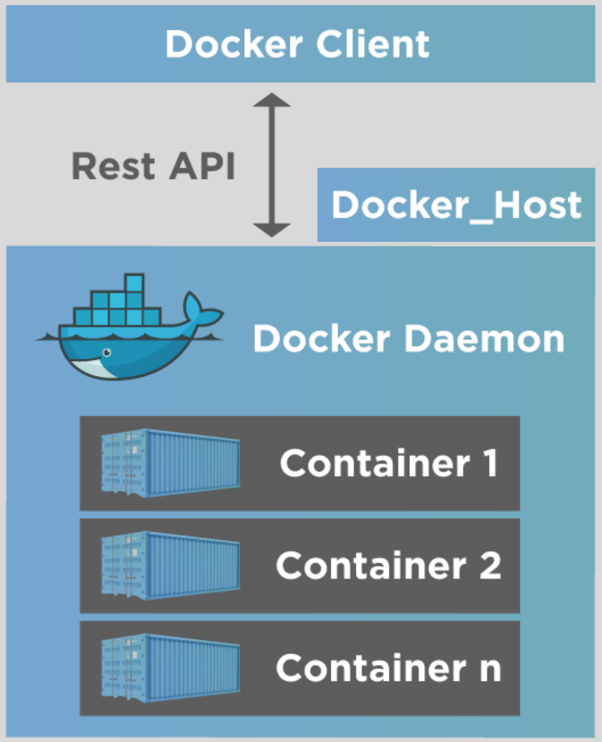
Docker Platform

Docker Engine

+ Docker Hub

= Docker Platform

Docker Engine



Docker Engine

- The software that hosts the containers is named **Docker Engine**.
- Docker Engine is a Client – Server based application.
- A Client or a Command Line Interface (CLI).
- Docker Engine has a daemon process that acts as a server. A Docker daemon is a background process. The daemon process is responsible for managing images, containers, storage volumes, and networks.

Docker Engine

- Docker Daemon
- Docker CLI

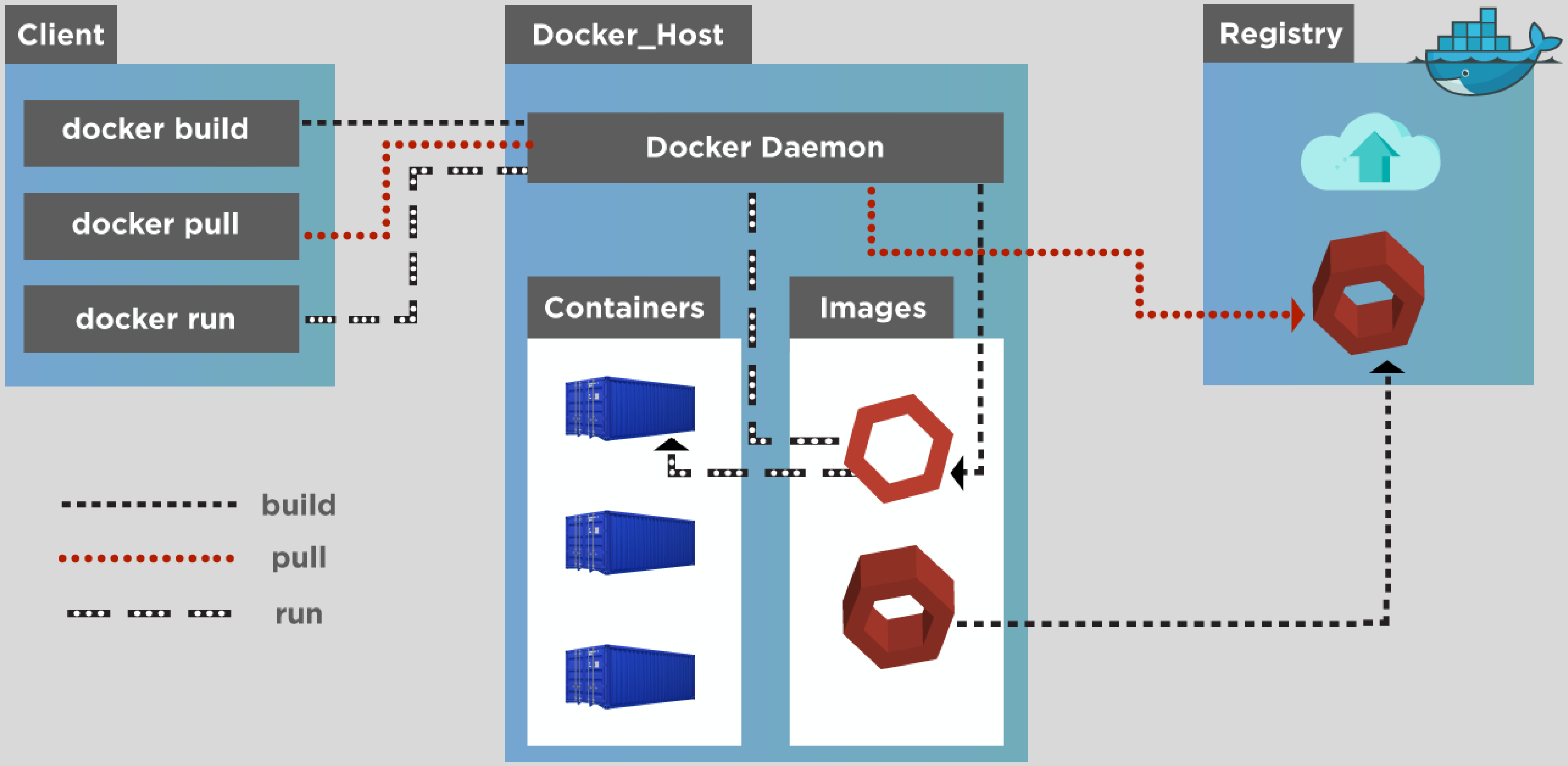
Docker Daemon

- Builds Images
- Runs and Manages Containers
- RESTful API

Docker CLI

• docker build	#Build an image from a Docker file
• docker images	#List all images on a Docker host
• docker run	#Run an image
• docker ps	#List all running containers
• docker stop	#Stop a running Docker Instances
• docker rm	#Remove an instance
• docker rmi	#Remove an image

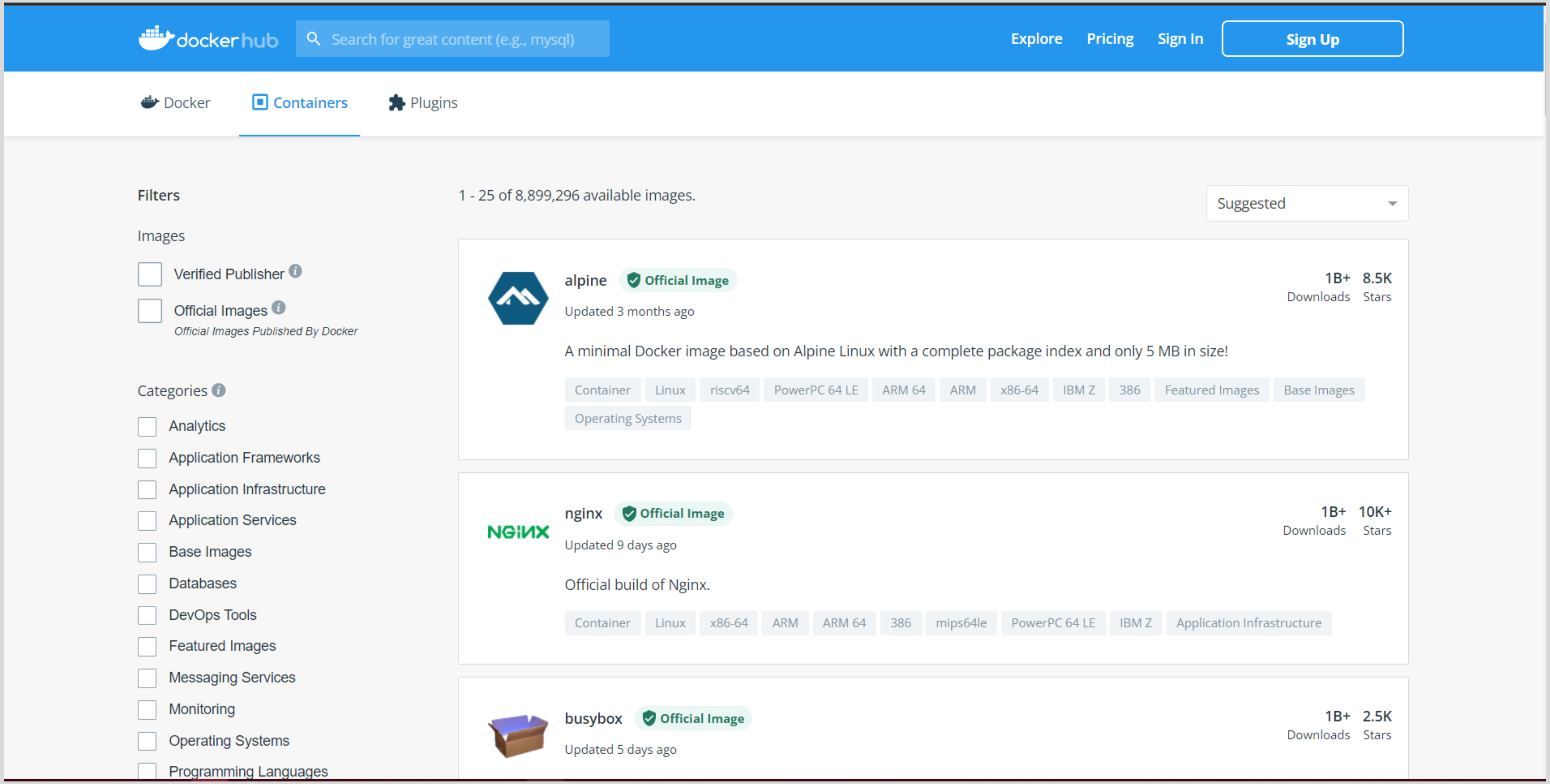
Docker Architecture



Docker Architecture

- **Docker Client :** client triggers interactions with the Docker Host
- **Docker Host** : Docker host runs a Docker daemon and a docker registry
- **The Docker Daemon :** is a component that runs inside the Docker Host and handles the images and containers.
- **Docker registry :** The docker registry stores the docker images. Docker hub is one such registry, but you can have your own too.

Docker Hub



Filters

Images

- Verified Publisher
- Official Images
  Official Images Published By Docker

Categories

- Analytics
- Application Frameworks
- Application Infrastructure
- Application Services
- Base Images
- Databases
- DevOps Tools
- Featured Images
- Messaging Services
- Monitoring
- Operating Systems
- Programming Languages

1 - 25 of 8,899,296 available images.

Suggested

alpine Official Image
Updated 3 months ago
1B+ Downloads 8.5K Stars
A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!
Container Linux riscv64 PowerPC 64 LE ARM 64 ARM x86-64 IBM Z 386 Featured Images Base Images Operating Systems

nginx Official Image
Updated 9 days ago
1B+ Downloads 10K+ Stars
Official build of Nginx.
Container Linux x86-64 ARM ARM 64 386 mips64le PowerPC 64 LE IBM Z Application Infrastructure

busybox Official Image
Updated 5 days ago
1B+ Downloads 2.5K Stars

Docker Hub

- Provides Docker Services
- Library of public images
- Storage for your images
- Automated builds (link github/bitbucket repo; trigger build on commit)

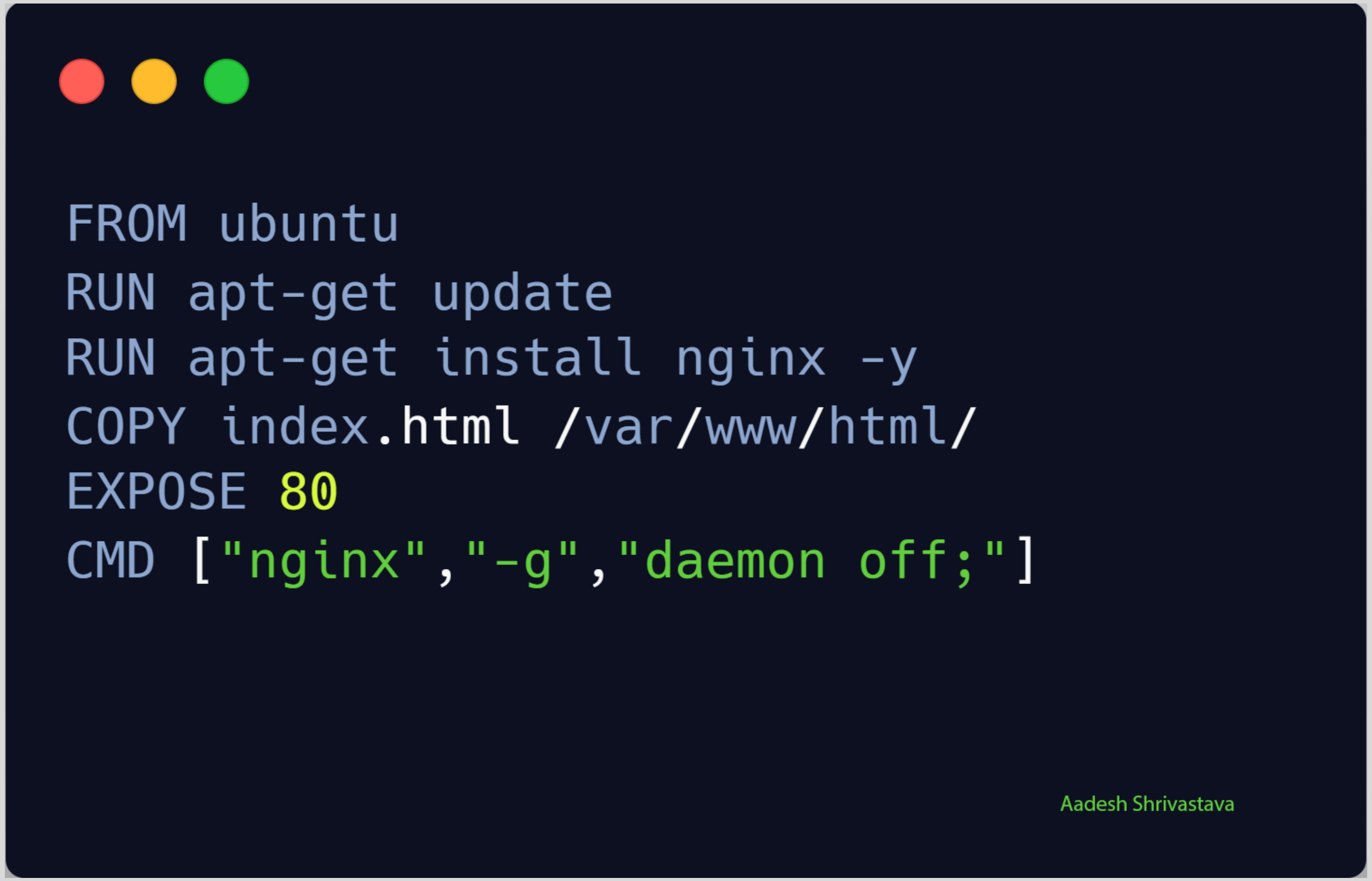
Basically, It's a public cloud-based registry provided by Docker for storing public images of the containers along with the provision of finding and sharing them.

Docker File

Docker File

- It is a text document that contains necessary commands which on execution helps assemble a Docker Image.
- Docker image is created using a Docker file.

How to write a Docker File



```
FROM ubuntu
RUN apt-get update
RUN apt-get install nginx -y
COPY index.html /var/www/html/
EXPOSE 80
CMD ["nginx","-g","daemon off;"]
```

How to write Docker File Contd…

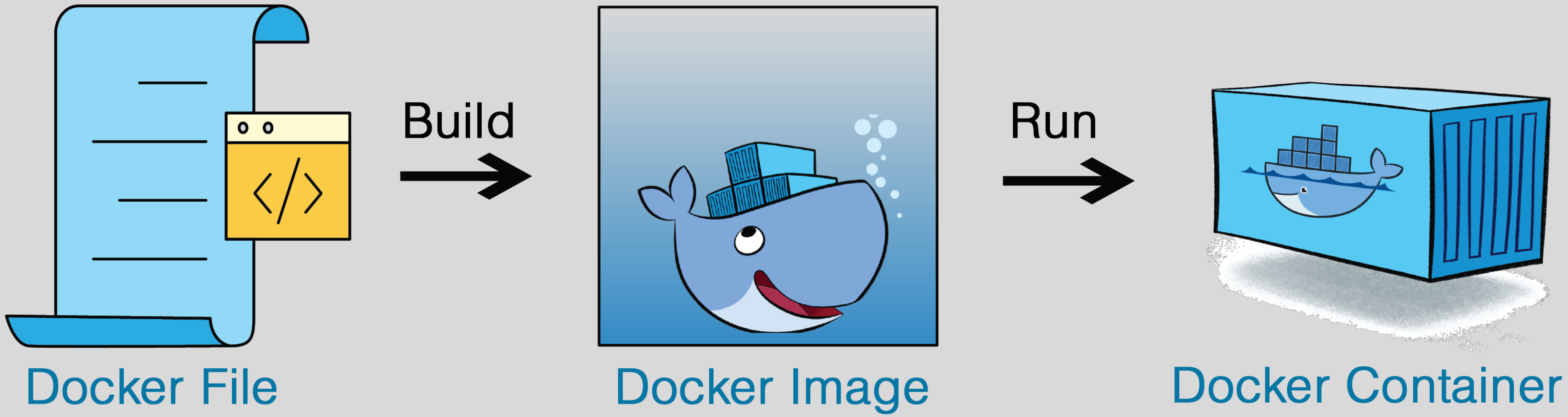
- FROM — is where we are pulling our official image from ubuntu is an official image provided by Docker
- WORKDIR — you guessed it…it is setting the current location (directory) of where are files are located(not used here but an option for your future use)
- RUN — provides an instruction to download and acquire all packages updates from the internet
- RUN — the next run instruction install nginx web server on our container

How to write Docker File Contd…

- COPY — adds files from your Docker client’s current directory
- EXPOSE — is a documentation instruction letting the container know we are exposing the standard port 80 (TCP)
- CMD — provide defaults for an executing container

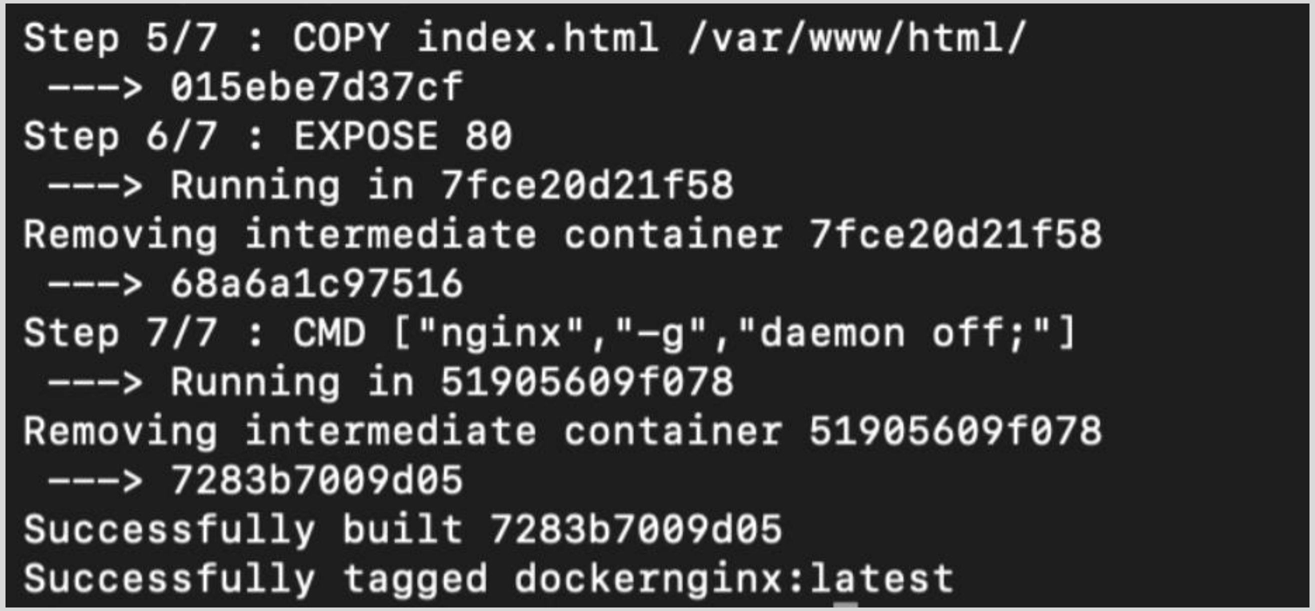
How containers are created?

How containers are created?



How containers are created?

Docker Build



```
Step 5/7 : COPY index.html /var/www/html/
 ---> 015ebe7d37cf
Step 6/7 : EXPOSE 80
 ---> Running in 7fce20d21f58
Removing intermediate container 7fce20d21f58
 ---> 68a6a1c97516
Step 7/7 : CMD ["nginx","-g","daemon off;"]
 ---> Running in 51905609f078
Removing intermediate container 51905609f078
 ---> 7283b7009d05
Successfully built 7283b7009d05
Successfully tagged dockernginx:latest
```

docker build -t <name of tag of container> . ← . is to symbolize current directory.

Docker Commands Cheat Sheet

Docker Cheat Sheet

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
```

Docker Management

All commands below are called as options to the base `docker` command. Run `docker <command> --help` for more information on a particular command.

`app*`	***Docker Application***
`assemble*`	***Framework-aware builds (Docker Enterprise)***
`builder`	***Manage builds***
`cluster`	***Manage Docker clusters (Docker Enterprise)***
`config`	***Manage Docker configs***
`context`	***Manage contexts***
`engine`	***Manage the docker Engine***
`image`	***Manage images***
`network`	***Manage networks***
`node`	***Manage Swarm nodes***
`plugin`	***Manage plugins***
`registry*`	***Manage Docker registries***
`secret`	***Manage Docker secrets***
`service`	***Manage services***
`stack`	***Manage Docker stacks***
`swarm`	***Manage swarm***
`system`	***Manage Docker***
`template*`	***Quickly scaffold services (Docker Enterprise)***
`trust`	***Manage trust on Docker images***
`volume`	***Manage volumes***

*Experimental in Docker Enterprise 3.0.

The End