Docker 101 session 1/3

Luis Alberto Carvajal Martínez

luis.carvajal@definityfirst.com luisalbcarvajal@gmail.com

Course Overview

- 3 sessions.
- no prior docker knowledge required.
- images, containers, swarms, services, secrets, stacks, logging. All in one course...
- some other tooling besides docker.
- theory, simply explained.
- mostly (99.9%) windows oriented.
- hands on exercises.

Introduction

What is Docker?

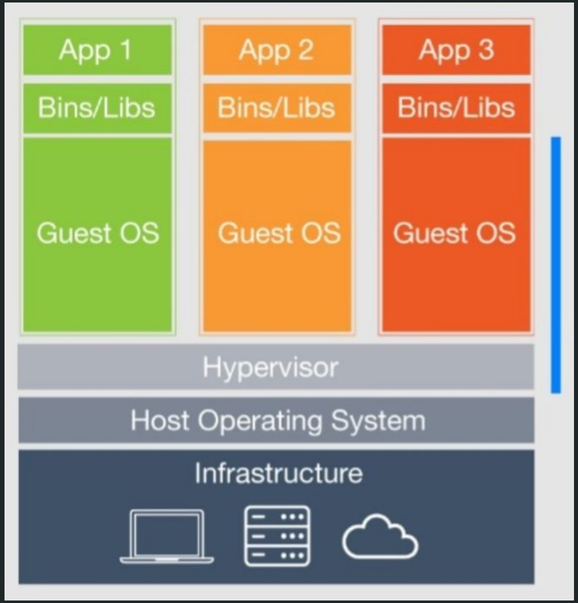
an open-source project that automates the deployment of software applications inside containers by providing an additional layer of abstraction and automation of OS-level virtualization ~~on Linux~~.

What is a Container?

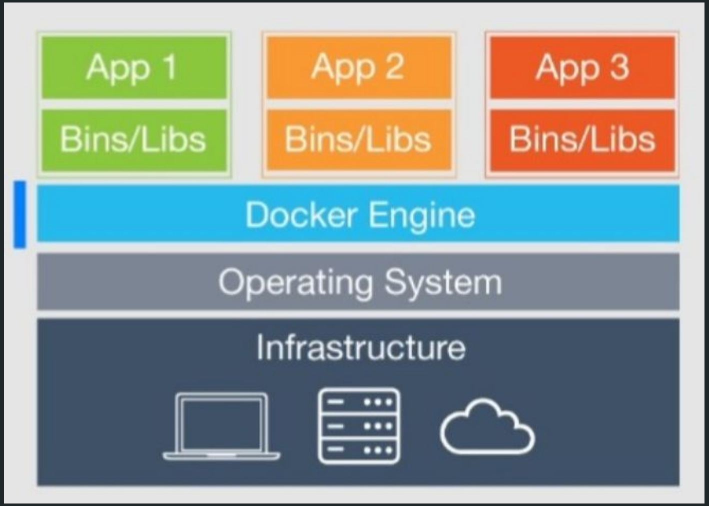
kind of like a VM … but not.

resource isolation and allocation. Different architectural approach, portable and efficient.

Old school virtualization



Docker Containers



Docker Daemon

The background service running on the host that manages building, running and distributing Docker containers. The daemon is the process that runs in the operating system to which clients talk to.

Docker Client

The command line tool that allows the user to interact with the daemon. More generally, there can be other forms of clients too - such as Kitematic which provide a GUI to the users.

Docker Hub

A registry of Docker images. You can think of the registry as a directory of all available Docker images. If required, one can host their own Docker registries and can use them for pulling images.

Dockerfile

defines what goes on in the environment inside your container.

Installing Community Edition

- **Open** https://store.docker.com/editions/community/docker-ce-desktop-windows
- **Run** Docker for Windows Installer to install
- **Open** a command-line terminal like PowerShell.
- **Run** `docker version` to check the version.
- **Run** `docker info`
- **Run** `docker run hello-world`.

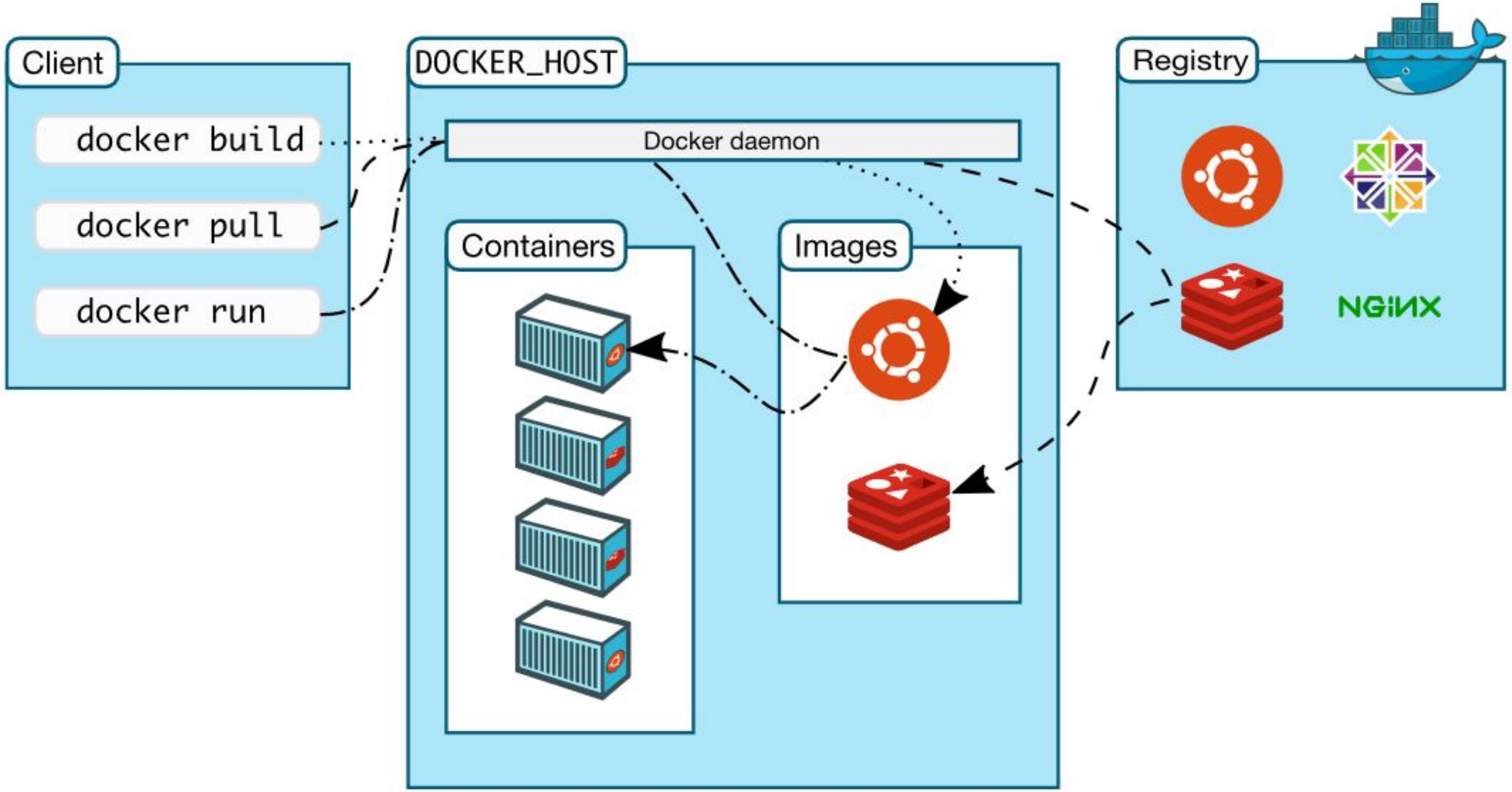
Tell me more!

Mis descubrimientos

¿Qué has aprendido de las pruebas?

1. Escribe aquí tu texto Escribe aquí tu texto
2. Escribe aquí tu texto
3. Escribe aquí tu texto Escribe aquí tu texto

Docker architecture


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

- The Docker daemon
- The Docker client
- Docker registries
- Docker objects
  - Images
  - Containers
  - **Services**

Docker Services

Services allow you to scale containers across multiple Docker daemons, which all work together as a swarm with multiple managers and workers. A service allows you to define the desired state, such as the number of replicas of the service that must be available at any given time. To the consumer, the Docker service appears to be a single application.

The underlying technology

- Namespaces
- Control groups
- Union file systems
- Container format

Namespaces

Provides the isolated workspace called the container.

Control groups

Allows Docker Engine to share available hardware resources to containers and optionally enforce limits and constraints. For example, you can limit the memory available to a specific container.

UnionFS

Are file systems that operate by creating layers, making them very lightweight and fast. Docker Engine uses UnionFS to provide the building blocks for containers.

Working with images

1. Build your own image
2. Run locally
3. Push image to a registry
4. Deploy to the cloud

Build your own image

1. Download https://github.com/carvajalluis/angular-architecture-patterns
2. Create Dockerfile
3. Create even better Dockerfile (Layers and layers)

2. Create DockerFile, then...

- when you finished:
  - `docker build -t app-serve .` to create the image
  - `docker run -p 1235:4200 -d app-serve` to start the container
- build a multistage image:
  - `docker build -f n-stg.Dockerfile  -t app-nginx .`
  - `docker run --name 2stage-demo --rm -p 80:80 -d app-nginx`

3. Push image to a registry

- `docker tag app-nginx luiscarvajal/101:nginx`
- `docker push luiscarvajal/101:nginx`
- `docker run -p 8080:80 -d luiscarvajal/101:nginx`