Docker Commands

Module Overview

* What are Containers ?
* What is Docker ?
* Install Docker in AWS EC2
* First Docker Command

Containers

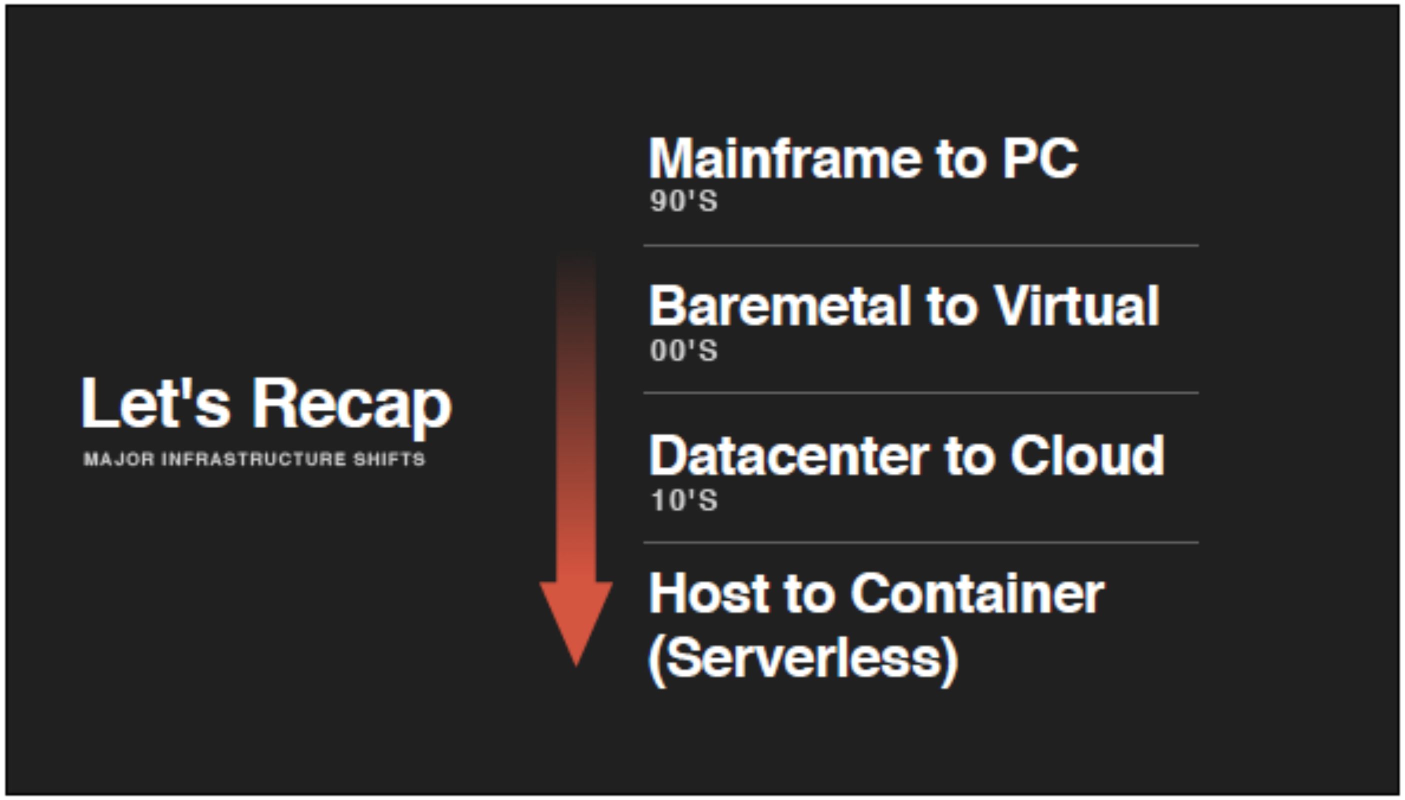
- What are containers ?
  - A metallic box with standard dimensions
  - Means to package applications & their dependencies in a standardized way
  - Build, ship, & run anywhere
- Why are containers important ?
  - Applications are more secure
  - Simulate production like environment
  - Operators can concentrate on provisioning infrastructure, running and monitoring applications
  - Applications are like black boxes to operators
- Efficiency
  - Security, automation and standardization
  - Speeds up development and reduces maintenance cost

Containerization & Virtualization

- Containers are an abstraction at the app layer that packages code and dependencies together.
- Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space
- Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

- Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers
- The hypervisor allows multiple VMs to run on a single machine
- Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

Infrastructure Shifts



Let's Recap

MAJOR INFRASTRUCTURE SHIFTS

Mainframe to PC
90'S

Baremetal to Virtual
00'S

Datacenter to Cloud
10'S

Host to Container (Serverless)

What is Docker ?

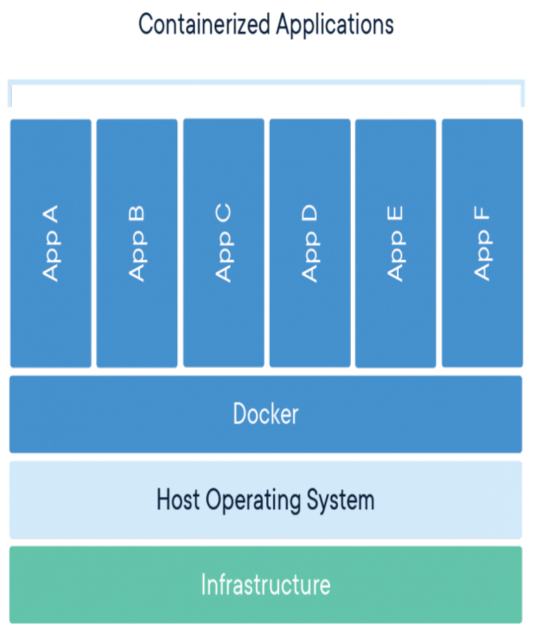
- A container is a standardized unit of software.
- A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Docker containers that run on Docker Engine:

Standard: Docker created the industry standard for containers, so they could be portable anywhere

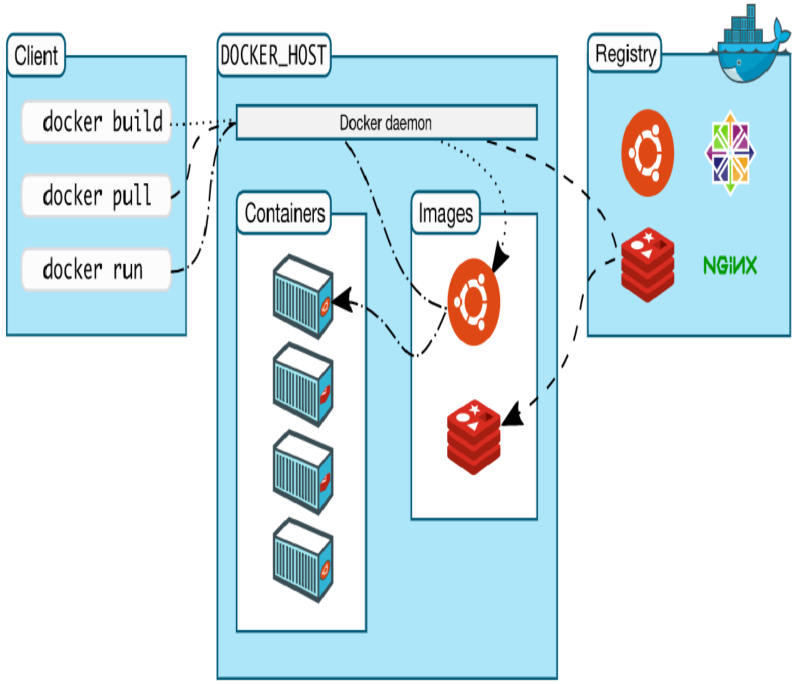
Lightweight: Containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs

Secure: Applications are safer in containers and Docker provides the strongest default isolation capabilities in the industry



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 communicate using a REST API, over UNIX sockets or a network interface.
- The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon



Docker Images & Containers

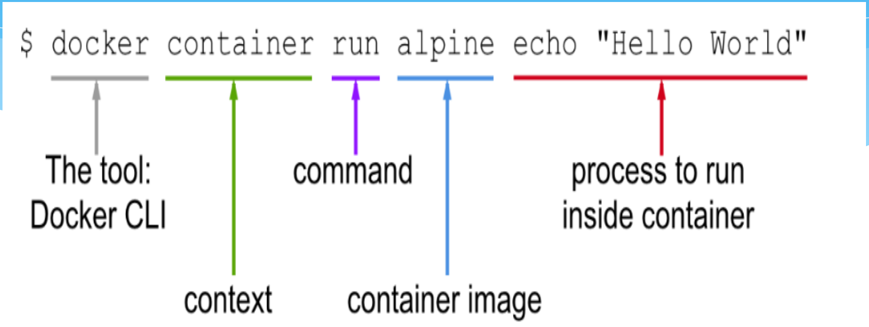
* Images
  * An *image* is a read-only template with instructions for creating a Docker container
  * To build your own image, you create a *Dockerfile* with a simple syntax for defining the steps needed to create the image and run it.
  * Each instruction in a Dockerfile creates a layer in the image
* Containers
  * A container is a runnable instance of an image
  * You can create, start, stop, move, or delete a container
  * A container is defined by its image as well as any configuration options you provide to it when you create or start it

Docker Installation

- This is Lab.
- Create an account on AWS to launch EC2 instance.
- Install Docker on EC2 instance
- Please refer to course documentation

First Docker Command

* Docker looks for the alpine image locally in image cache, doesn’t find anything
* Then look in remote image repository(defaults to docker hub)
* Downloads the latest version
* Creates a new container based on that image
* Starts the container
* Runs the echo command
* Terminates the container



```
$ docker container run alpine echo "Hello World"
```

Lab

* Create account on AWS
* Launch an instance
* Install Docker on EC2
* Running The First Container!