Containerization

CS398 - ACC

Prof. Robert J. Brunner

Ben Congdon
Tyler Kim

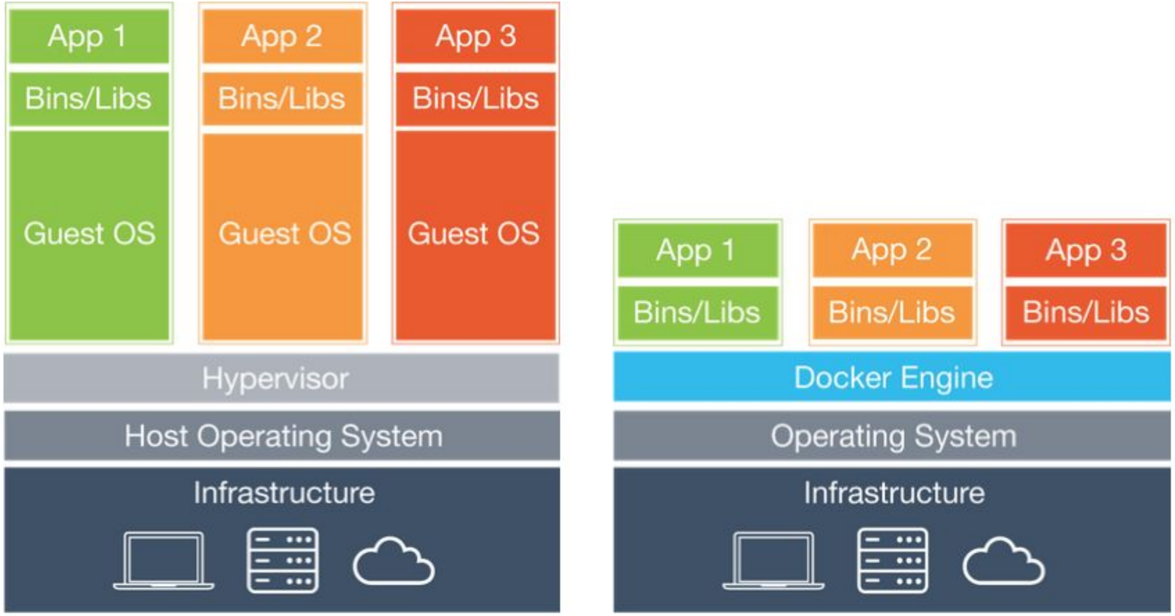
What is Containerization?

- OS-level virtualization.
- Deploy and Run distributed applications without VMs
- Isolated Environment
  - Each container can have different files, environment variables, libraries, and different OS.
  - Multiple isolated application can run on a single host and access the same OS kernel

Motivation for Containerization

- VMs are great, but have high runtime overhead
  - Can we scale faster and more easily?
- What if we could sandbox VMs, but share the OS kernel?
- Enables different software architectures and practices

Big Idea: Containers have less OS overhead

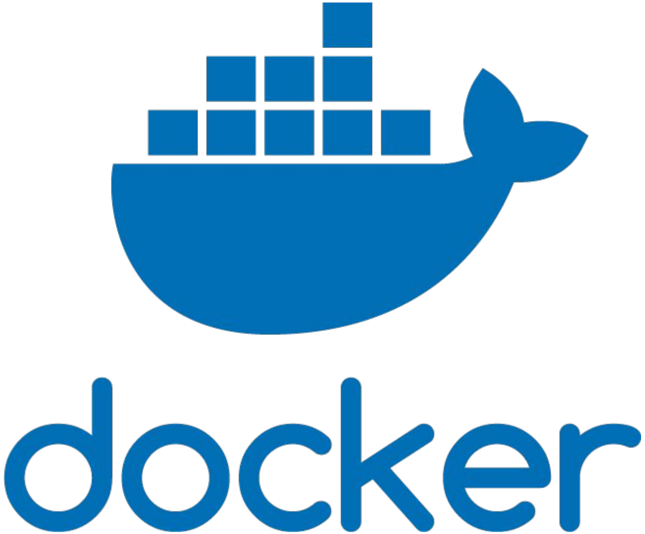


Traditional VM-based Infrastructure

Container Infrastructure

https://newsroom.netapp.com/blogs/containers-vs-vms/

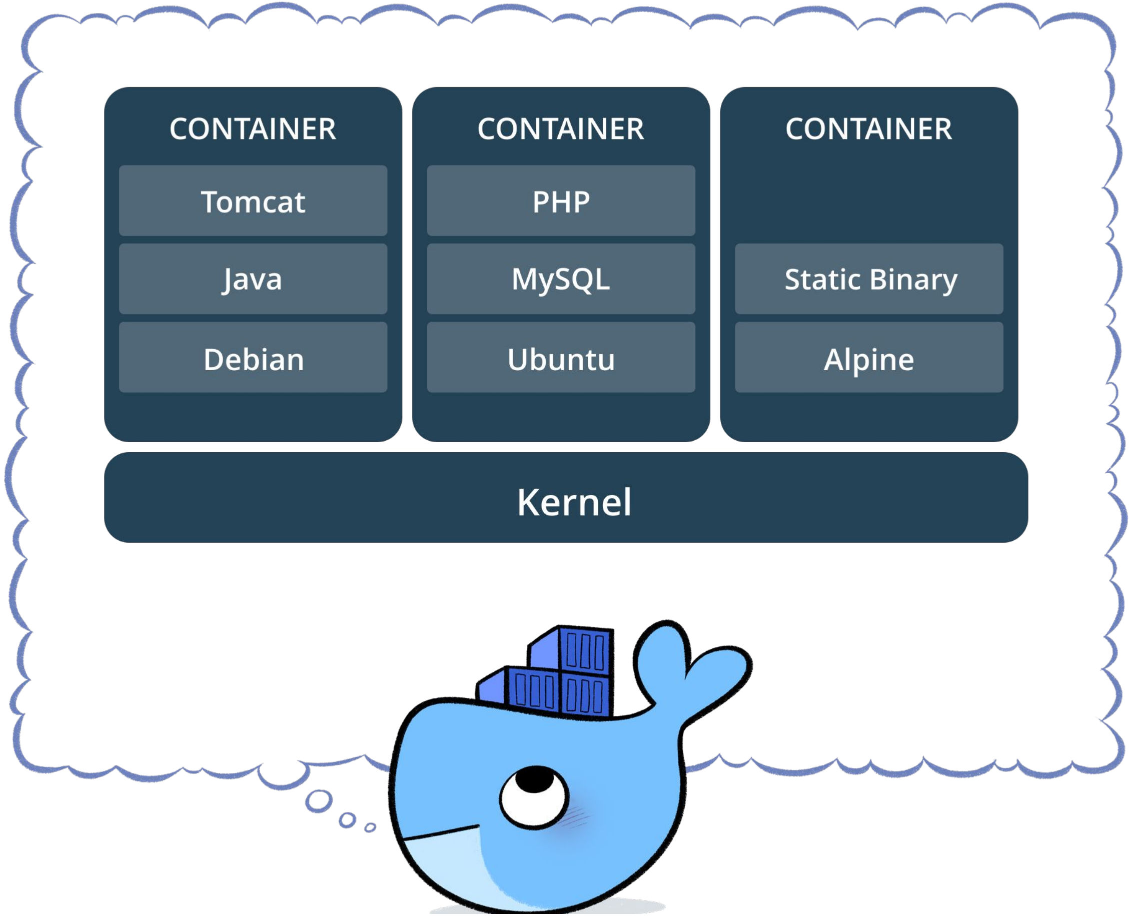
Popular Containerization Platforms



kubernetes

What is Docker?

- Open platform to **develop**, **deploy**, and **run** applications with containers.
- Abstracts hardware virtualization for containers.
- Client-Server Architecture



CONTAINER

Tomcat

Java

Debian

CONTAINER

PHP

MySQL

Ubuntu

CONTAINER

Static Binary

Alpine

Kernel

Docker Concepts

- **Image**
  - Frozen description of an environment
- **Container**
  - Running instantiation of an image
- **Volume**
  - Persistent data storage

Docker Concepts

- Dockerfile
  - Describes everything your container needs:
    - Dependencies
    - Source Code / Binaries

The Dockerfile

Overview Guide to Running Code in Docker:

1. Inherit from a parent OS/platform container
2. Install any packages / libraries you need
3. Add any source code you need
4. Attach any volumes you need for data persistence
5. Set a command to be run at startup

The Dockerfile

Essential Commands:

- **FROM** - Inherit from a parent container
  - i.e. “FROM ubuntu”
- **RUN** - Runs a command *during* the build process
  - i.e. “RUN apt-get install python3”
- **ADD** - Copies files from the build directory into the image
  - i.e. “ADD hello_world.py /usr”
- **EXPOSE** - Register a port that the image will listen on
  - i.e. “EXPOSE 80”
- **CMD** - Set the default command to be executed on startup
  - i.e. “CMD python /usr/hello_world.py”

The Dockerfile

- Each command in a Dockerfile creates an intermediate image
  - Useful for caching!
- Structure your Dockerfiles to take advantage of caching
  - Install packages first, then add source code
  - Within reason, “Funnel down” from most general to most specific

Where to go from here

- **Docker Swarm**
  - Pools multiple Docker engines into a combined virtual host
  - Allows multiple VMs to collaborate to host clustered Docker containers
- **Docker Compose**
  - Orchestrate multiple-container applications
  - Declarative format for configuring volumes, container networking, and scaling

MP7 - Docker

- Released Tonight.
- Due next Tuesday at 11:59pm (as normal)

Wednesday:

- Docker Demo
- Docker MP Office Hours.