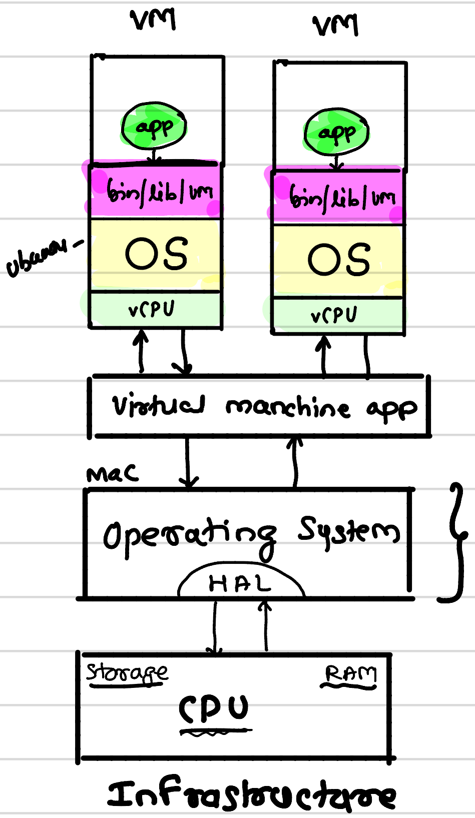
Docker

Containerization

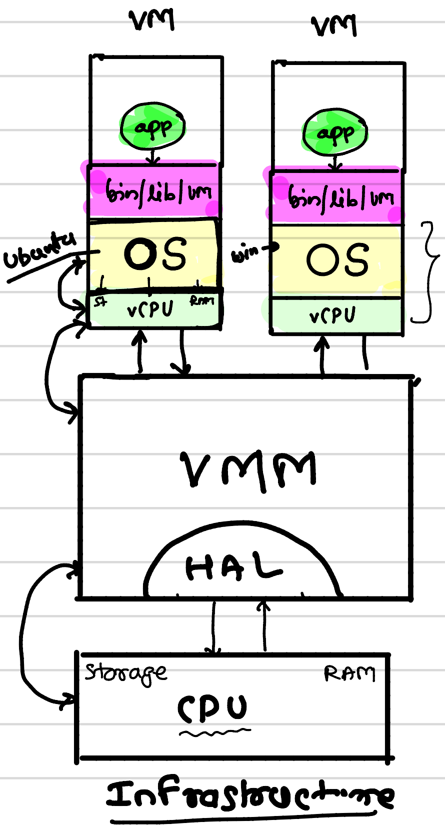
- lightweight alternative to a virtual machine
- involves encapsulating an application in a container with its own operating system
- foundation of Containerization lies in the Linux Container (LXC) Format
- containers only work with Linux based machines and can only run Linux applications

Virtualization vs Containerization

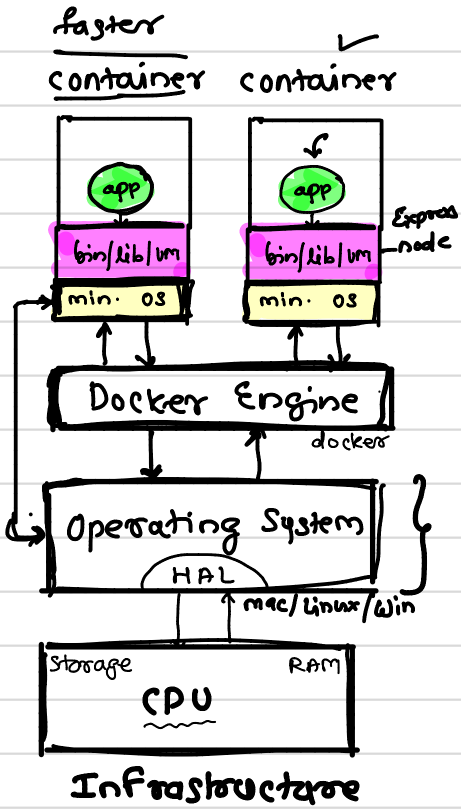
Type II



Type I



Containerization



angular → ng

java – JDK

Containers vs Virtual machines

Virtual Machine	Container
Hardware level virtualization	OS virtualization
Heavyweight (bigger in size)	Lightweight (smaller in size)
Slow provisioning	Real-time and fast provisioning
Limited Performance	Native performance (faster)
Fully isolated	Process-level isolation
More secure	Less secure
Each VM has separate OS	Each container can share OS resources
Boots in minutes	Boots in seconds
Pre-configured VMs are difficult to find and manage	Pre-built containers are readily available
Can be easily moved to new OS	Containers are destroyed and recreated
Creating VM takes longer time	Containers can be created in seconds

Advantages

- ✓ Multi-Cloud platform
  - AWS
  - GC
  - Azure
- ✓ Shares same OS
- ✓ Reduced size
- ✓ Testing and CI-CD
- ✓ Portability
  - Win
  - Linux
  - macOS
- ✓ Version Control — Dockefile
- ✓ Cost efficient
- ✓ Faster than VM

v1 → x

v2 → y

Disadvantages

- ✓ Security concern
- ✓ Monitoring → Docker swarm

  Kubernetes

Popular container providers

- ✓ Linux Containers (LXC)
- ✓ Docker
- ✓ Windows Server

Overview

- 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.
- It is a tool that allows developers, sys-admins etc. to easily deploy their applications in a sandbox (called *containers*) to run on the host operating system i.e. Linux
- Docker is a container management service
- It allows users to **package an application with all of its dependencies into a standardized unit** for software development

Dockerfile

Images

- In Docker, everything is based on Images
- An image is a combination of a file system and parameters

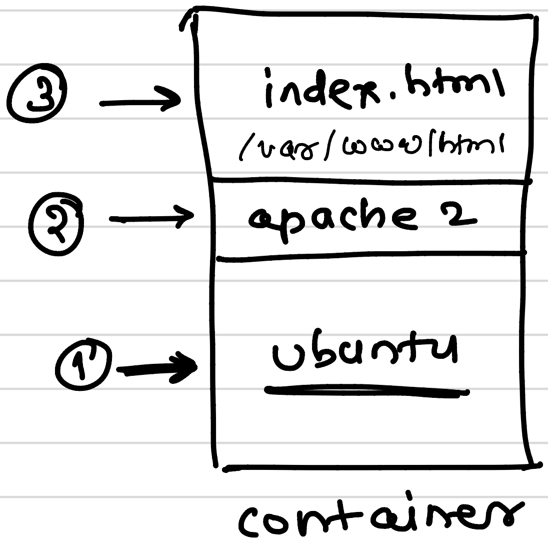
* instructions to create a container

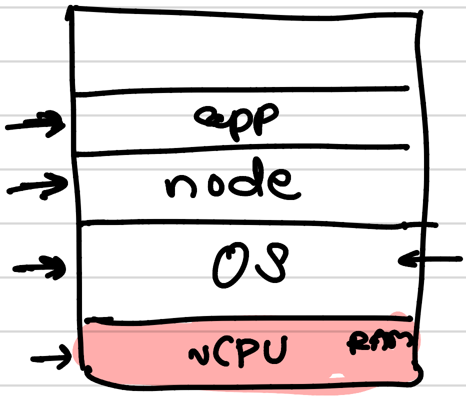
* template to create a container

Containers

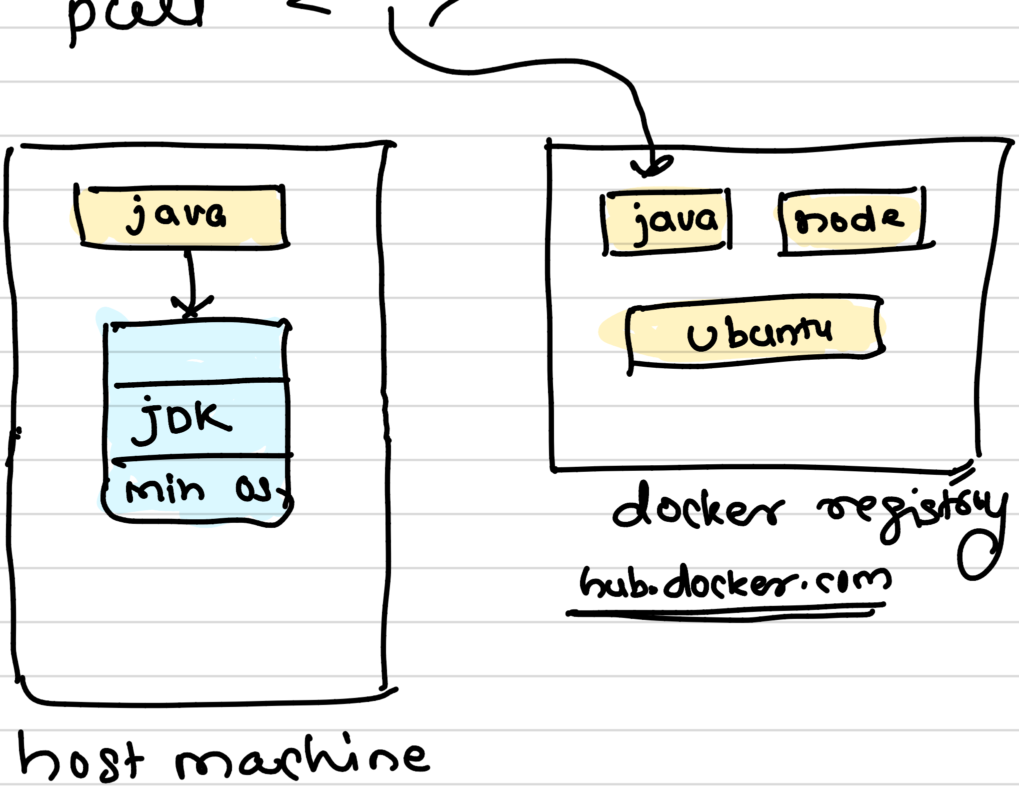
- Containers are instances of Docker images that can be run using the Docker run command
- The basic purpose of Docker is to run containers
- Create a custom container use Dockerfile

Docker



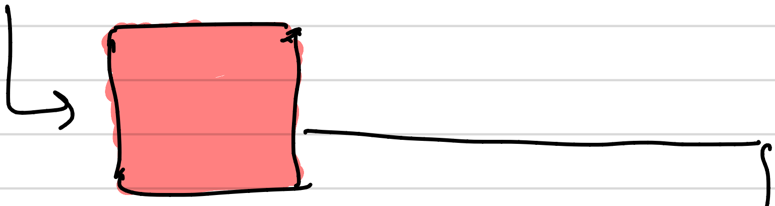


→ docker pull < >

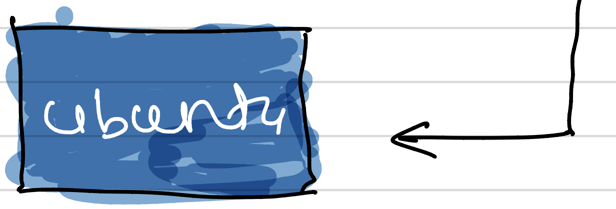


① custom image

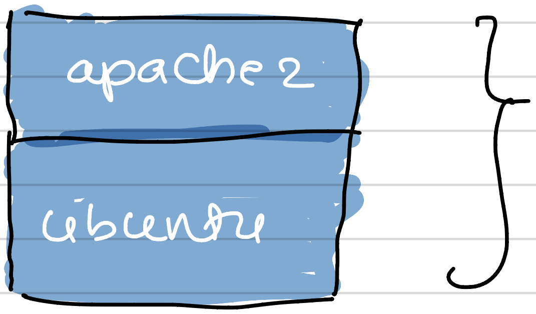
① docker pull ubuntu



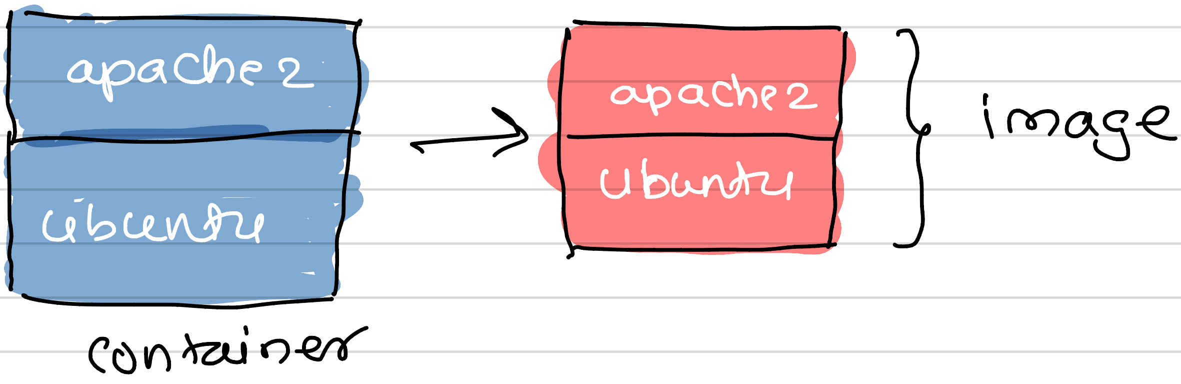
② docker run ubuntu

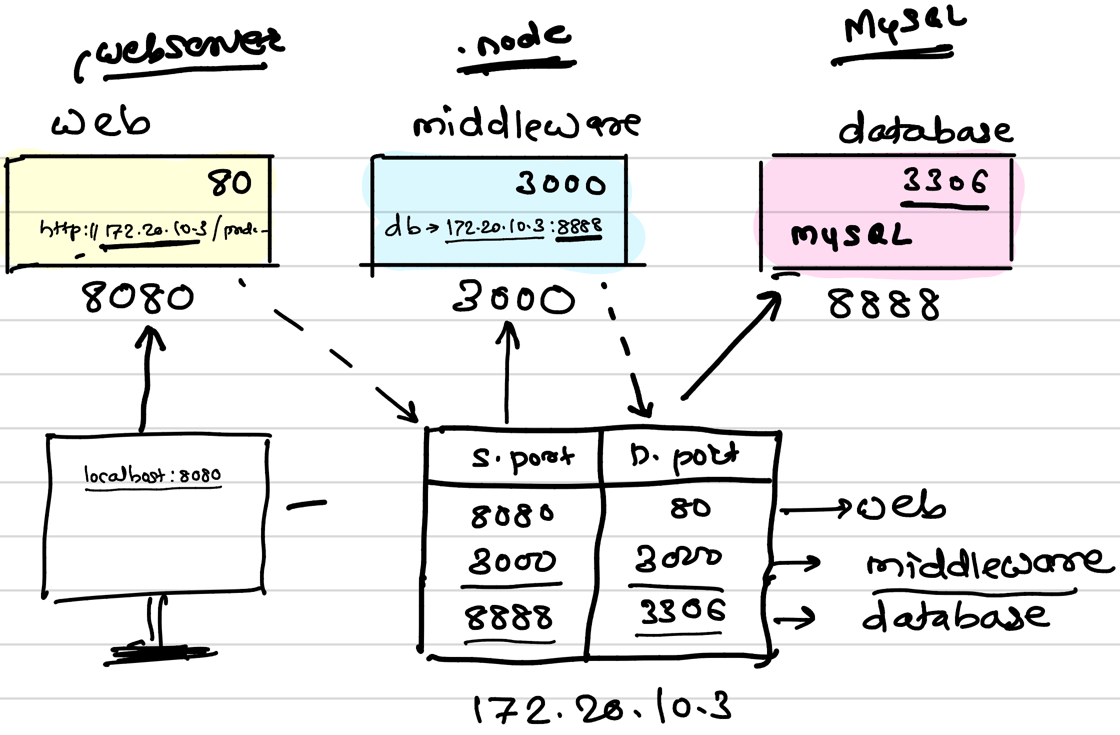


③ apt-get update ; apt-get install apache2



④ docker commit <container id> <image name>





S. port	D. port
8080	80
3000	3000
8888	3306

① localhost : 8080 →

↓

localhost : 3000 →

↓

localhost : 8888 →

Dockerfile

- Dockerfile defines what goes on in the environment inside your container
- Access to resources like networking interfaces and disk drives is virtualized inside this environment, which is isolated from the rest of your system
- Steps
  - Create Dockerfile with the configuration
  - Build the image

Dockerfile commands

- FROM
- ENV
- RUN
- CMD
- EXPOSE
- WORKDIR
- ADD
- COPY
- LABEL
- MAINTAINER
- ENTRYPOINT

Microservices

Overview

- Microservice architecture, or simply microservices, is a distinctive method of developing software systems that tries to focus on building single-function modules with well-defined interfaces and operations
- Is an architectural style that structures an application as a collection of services that are
  - Highly maintainable and testable
  - Loosely coupled
  - Independently deployable
  - Organized around business capabilities

Microservices

- Decoupling
- Componentization
- Continuous Delivery
- Responsibility
- Decentralized Governance
- Agility

Advantages

- Independent development
- Independent deployment
- Fault Isolation
- Mixed technology stack
- Granular Scaling