VIRTUALIZATION

INTERNET ENGINEERING

Fall 2022

@1995parham

- IaaS, PaaS, SaaS
- Virtual Machine
- Containers

- IaaS, PaaS, SaaS
- Virtual Machine
- Containers

IAAS

Infrastructure as a service (IaaS) refers to online services that *abstract* the user from the details of **infrastructure** like physical computing resources, location, data partitioning, scaling, security, backup etc.

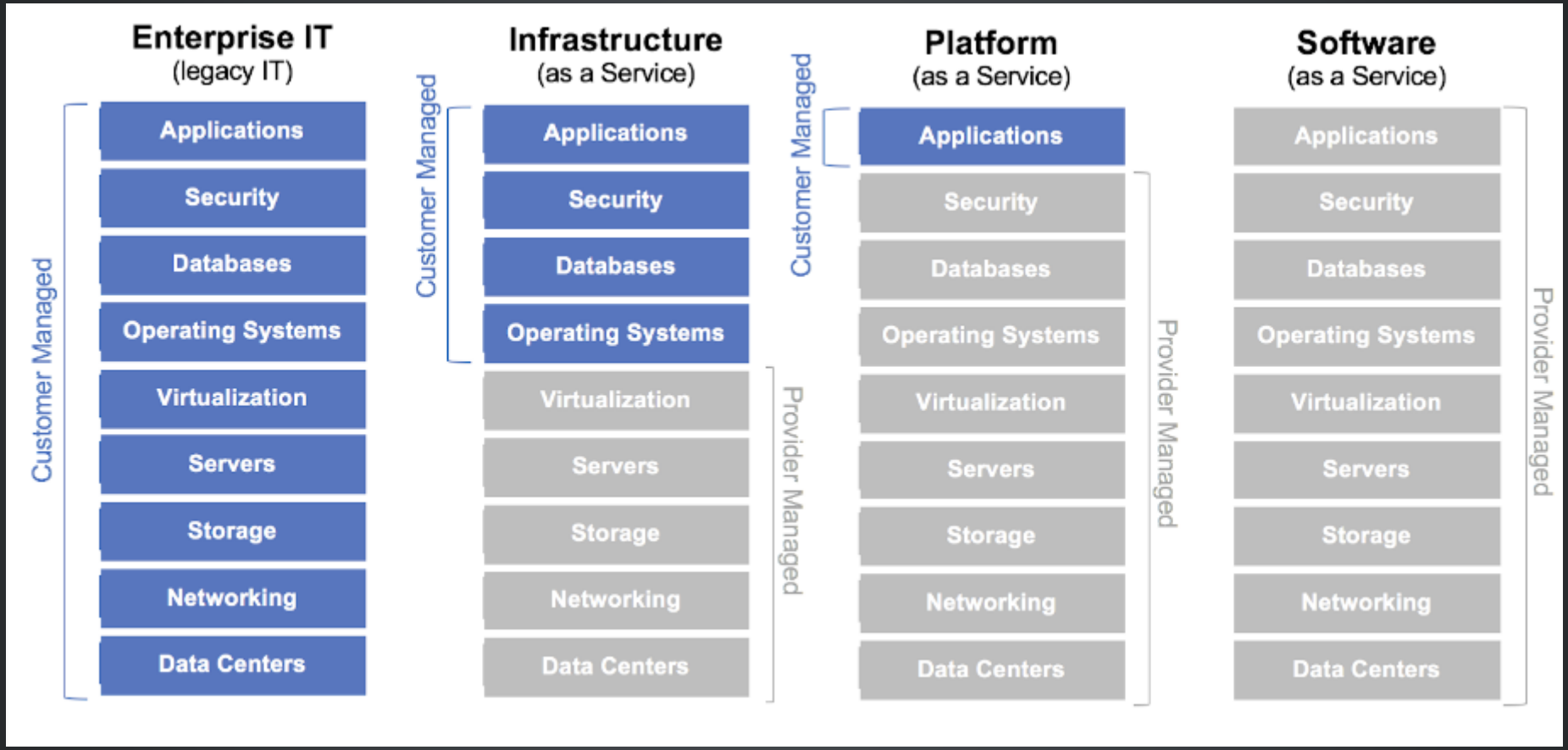
- Example providers:
  - AWS Amazon
  - Virtual Box
  - VMware

PAAS

Platform as a service (PaaS) or application platform as a service (aPaaS) is a category of cloud computing services that *provides a platform* allowing customers to **develop**, **run**, and **manage** applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app.

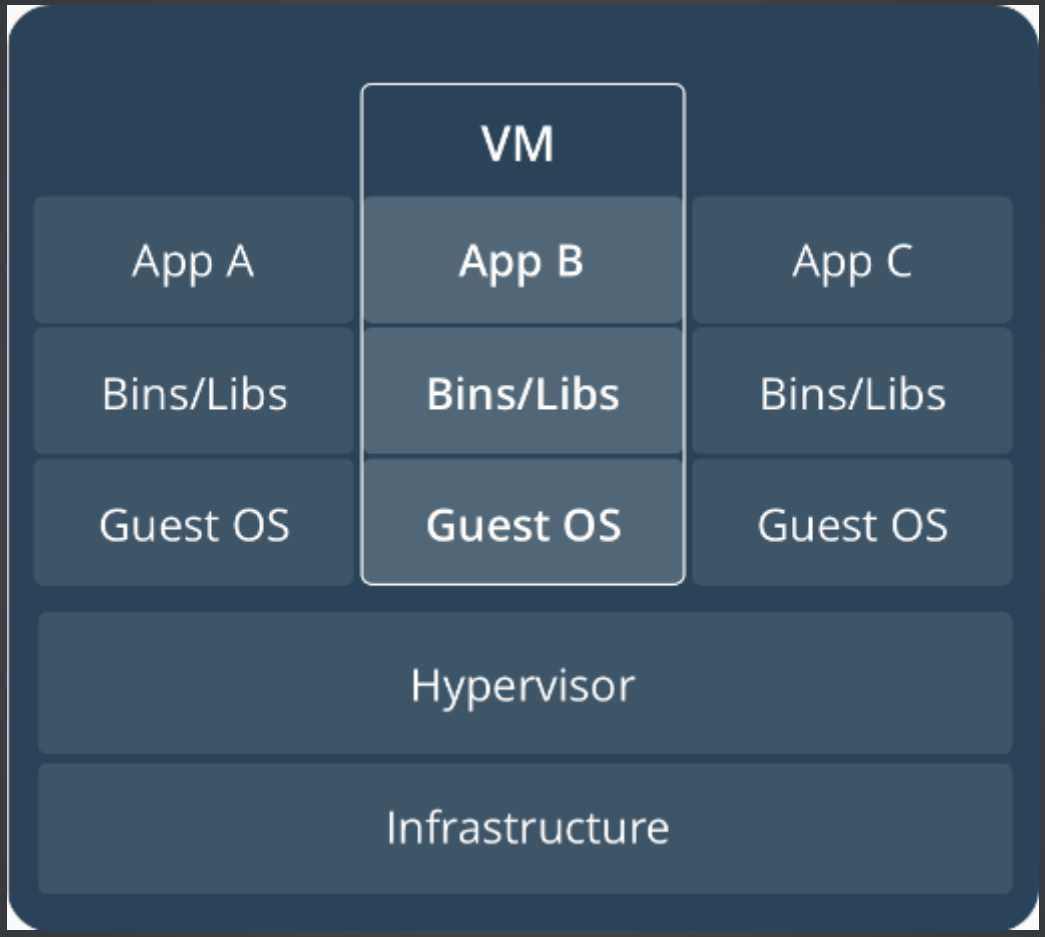
SAAS

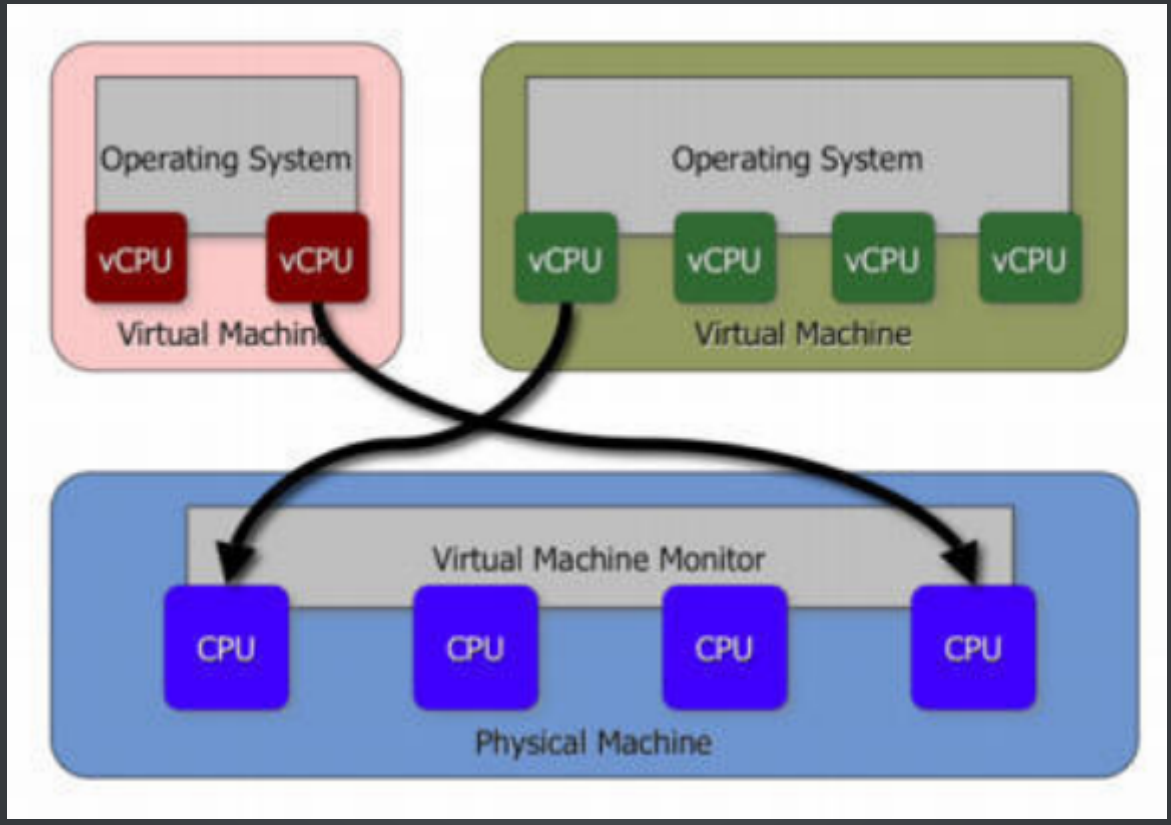
SaaS applications are sometimes called *Web-based software*, on-demand software, or hosted software. Whatever the name, SaaS applications **run on a SaaS provider's servers**. The provider manages access to the application, including security, availability, and performance.


Enterprise IT
(legacy IT)
Applications
Security
Databases
Operating Systems
Virtualization
Servers
Storage
Networking
Data Centers
Customer Managed
Infrastructure
(as a Service)
Applications
Security
Databases
Operating Systems
Virtualization
Servers
Storage
Networking
Data Centers
Customer Managed
Provider Managed
Platform
(as a Service)
Applications
Security
Databases
Operating Systems
Virtualization
Servers
Storage
Networking
Data Centers
Customer Managed
Provider Managed
Software
(as a Service)
Applications
Security
Databases
Operating Systems
Virtualization
Servers
Storage
Networking
Data Centers
Provider Managed

- IaaS, PaaS, SaaS
- Virtual Machine
- Containers

In computing, a virtual machine (VM) is an emulation of a computer system.


VM
App A
App B
App C
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure


Operating System
vCPU
vCPU
Virtual Machine
Operating System
vCPU
vCPU
vCPU
vCPU
Virtual Machine
Virtual Machine Monitor
CPU
CPU
CPU
CPU
Physical Machine

VIRTUALIZATION TECHNIQUES

- The **VM-based** approach virtualizes the complete OS. The abstraction it presents to the VM are virtual devices like virtual disk, virtual CPUs, and virtual NICs. With virtual machines, multiple OSes can share the same hardware resources, with virtualized representations of each of the resources available to the VM.
- **Container-based** form of virtualization doesn't abstract the hardware but uses techniques within the Linux kernel to isolate access paths for different resources. It carves out a logical boundary within the same operating system.

HYPERVISORS

A special piece of software is used to virtualize the OS.

CPU VIRTUALIZATION

- **Binary Translation in the Case of Full Virtualization:** In this case, the guest OS is used without any changes. The instructions are trapped and emulated for the target environment. This cause a lot of performance overhead, as lots of instructions have to be trapped into the host/hypervisor and emulated.
- To avoid the performance problems related to binary translation when using full virtulization, we use **paravirtualization** wherein the guest knows that it is running in a virtualized environment and its interaction with the host is optimized to avoid excessive trapping.

CPU VIRTUALIZATION (CONTD)

- In 2005, x86 finally become virtualizable. Advantages of **hardware-assisted virtulization** are two fold:
  - No binary translation
  - No OS modification

IO VIRTUALIZATION

- With **full virtualization**, the guest does not know it's running on a hypervisor and the guest OS doesn't need any changes to run on a hypervisor.
- In **paravirtualization** case, the guest OS is made aware that it's running in a vritualized environment and special drivers are loaded into the guest to take care of the I/O.

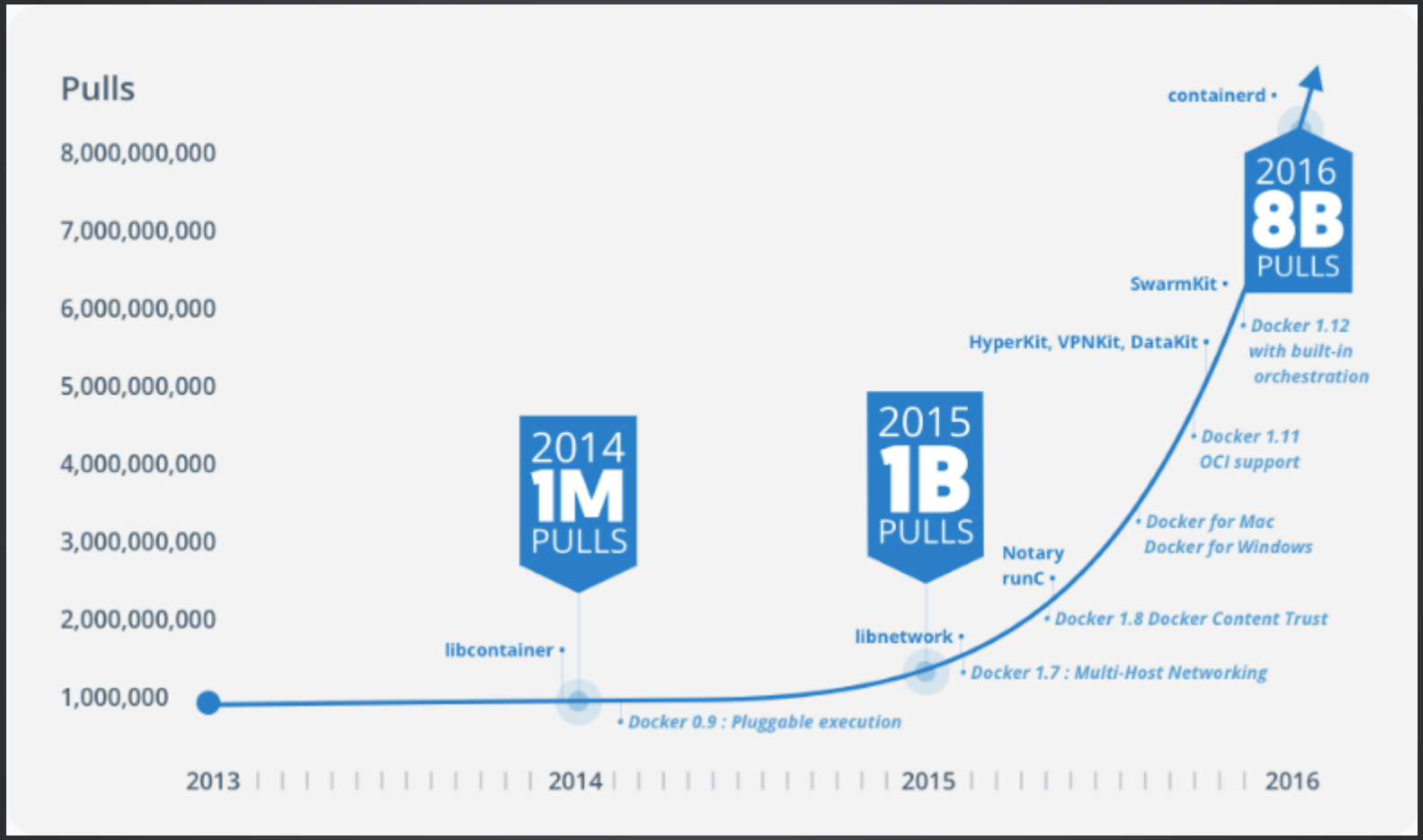
THE QUICK EMULATOR (QEMU)

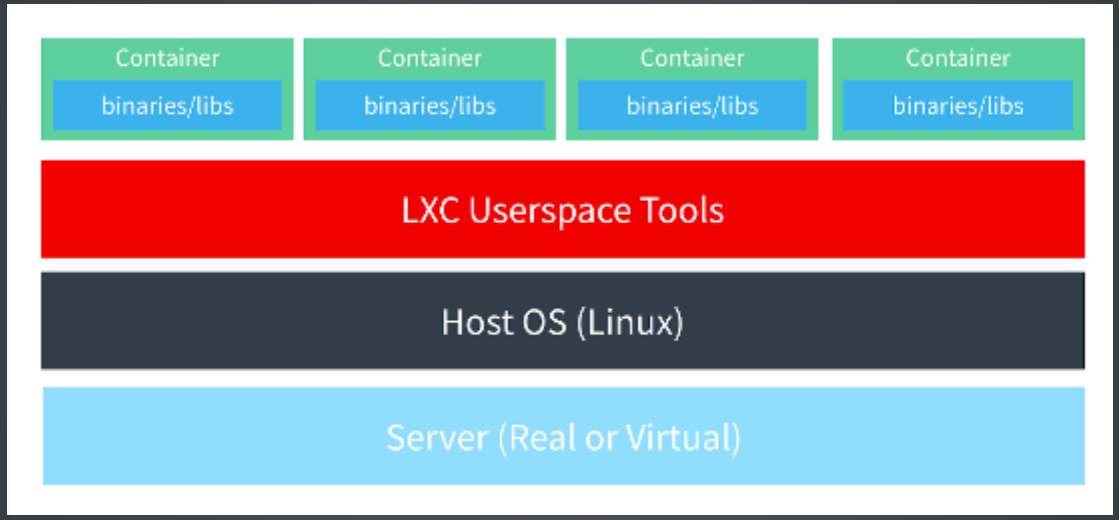
- The QEMU runs as a user process and handles the KVM kernel module.
- It uses the `vt-x` extensions to provide the guest with an isolated environment from a memory and cpu perspective.
- The QEMU also dedicates a separate thread for I/O. This thread runs an event loop and is based on the non-blocking mechanism.
- The QEMU can use paravirtualized drivers.

- IaaS, PaaS, SaaS
- Virtual Machine
- Containers

CONTAINERS

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.


Pulls
8,000,000,000
7,000,000,000
6,000,000,000
5,000,000,000
4,000,000,000
3,000,000,000
2,000,000,000
1,000,000
2013
2014
2015
2016
2014 1M PULLS
2015 1B PULLS
2016 8B PULLS
libcontainer •
• Docker 0.9 : Pluggable execution
libnetwork •
• Docker 1.7 : Multi-Host Networking
Notary
runC •
• Docker 1.8 Docker Content Trust
• Docker for Mac
Docker for Windows
• Docker 1.11
OCI support
HyperKit, VPNKit, DataKit •
SwarmKit •
• Docker 1.12
with built-in
orchestration
containerd •


Container
binaries/libs
Container
binaries/libs
Container
binaries/libs
Container
binaries/libs
LXC Userspace Tools
Host OS (Linux)
Server (Real or Virtual)

CONTAINERS VS VMS (PROS)

- Lightweight (MBs vs GBs)
- Easier Deployment
- Easier Portability

CONTAINERS VS VMS (CONS)

- Better control in virtual machines
- Have been tested over the years

Linux containers are made of three Linux kernel primitives:

- Linux namespaces
- cgroups
- Layered file systems

NAMESPACE

- A *namespace* is a logical isolation within the Linux kernel.
- A namespace controls visibility within the kernel.
- All controls are defined at the process level.

CONTAINER ENGINES

- podman
- rtk> 
- docker (moby)

403 FORBIDDEN

You cannot access docker from Iran

SO WHAT?!

- HTTP Proxy
- Shecan
- etc.

HELLO WORLD :)

```
docker run hello-world
```

DAYS BEFORE DOCKER

- Installing dependencies of your app on host machine
- Creates conflict with previous installation
- Separate Environment - Separate Runtime
- Code that runs on one system and doesn’t on the other

DAYS AFTER DOCKER

- Dependencies and runtime environment are all in the same place
- No need to install any dependency 💃
- Ensuring your app and your dependencies travel together

DOCKER IMAGE

An image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files.

DOCKER CONTAINER

A container is a runtime instance of an image – what the image becomes in memory when actually executed. It runs completely isolated from the host environment by default, only accessing host files and ports if configured to do so.

DOCKERFILE

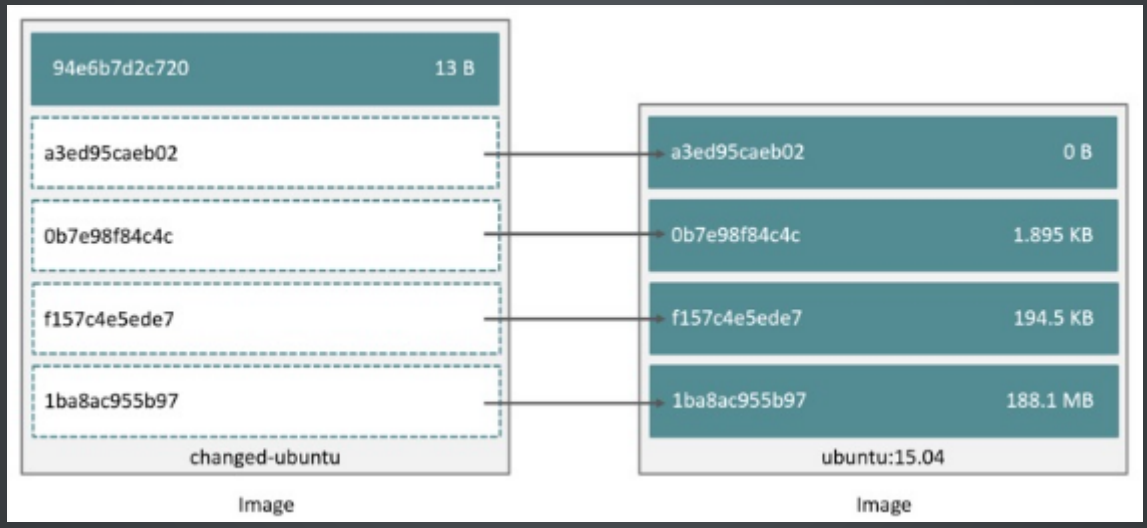
A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

```
# Format: FROM    repository[:version]
FROM       ubuntu:latest
# Installation:
# Import MongoDB public GPG key AND create a MongoDB list file
RUN apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
RUN apt-get install -y --no-install-recommends software-properties-common
RUN echo "deb http://repo.mongodb.org/apt/ubuntu $(cat /etc/lsb-release | grep DI

# Update apt-get sources AND install MongoDB
RUN apt-get update && apt-get install -y mongodb-org
# Create the MongoDB data directory
RUN mkdir -p /data/db
# Expose port 27017 from the container to the host
EXPOSE 27017

# Set usr/bin/mongod as the dockerized entry-point application
ENTRYPOINT ["/usr/bin/mongod"]
```

IMAGE LAYERS



DOCKER CLI (CHEATSHEET)

```
docker build -t friendlyname .          # Create image using this directory's Docke
docker run -p 4000:80 friendlyname      # Run "friendlyname" mapping port 4000 to 8
docker run -d -p 4000:80 friendlyname # Same thing, but in detached mode
docker ps                               # See a list of all running containers
docker stop hash                        # Gracefully stop the specified container
docker ps -a                            # See a list of all containers, even the on
docker kill hash                        # Force shutdown of the specified container
docker rm hash                          # Remove the specified container from this m
docker rm $(docker ps -a -q)            # Remove all containers from this machine
docker images -a                        # Show all images on this machine
docker rmi imagename                    # Remove the specified image from this mach
docker rmi $(docker images -q)          # Remove all images from this machine
```

DOCKER VOLUMES

- Data volumes can be shared and reused among containers.
- Changes to a data volume are made directly.
- Changes to a data volume will not be included when you update an image.
- Data volumes persist even if the container itself is deleted.

DOCKER COMPOSE

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application's services.

DOCKER COMPOSE

```
version: '3'

services:
   db:
     image: mysql:latest
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress
  wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       "8000:80"
```

REFERENCES 

- Iman Tabrizian's Virtualization Workshop, 9th Linux Festival