9. Docker - Containers

9.1 VM vs Containers

VM

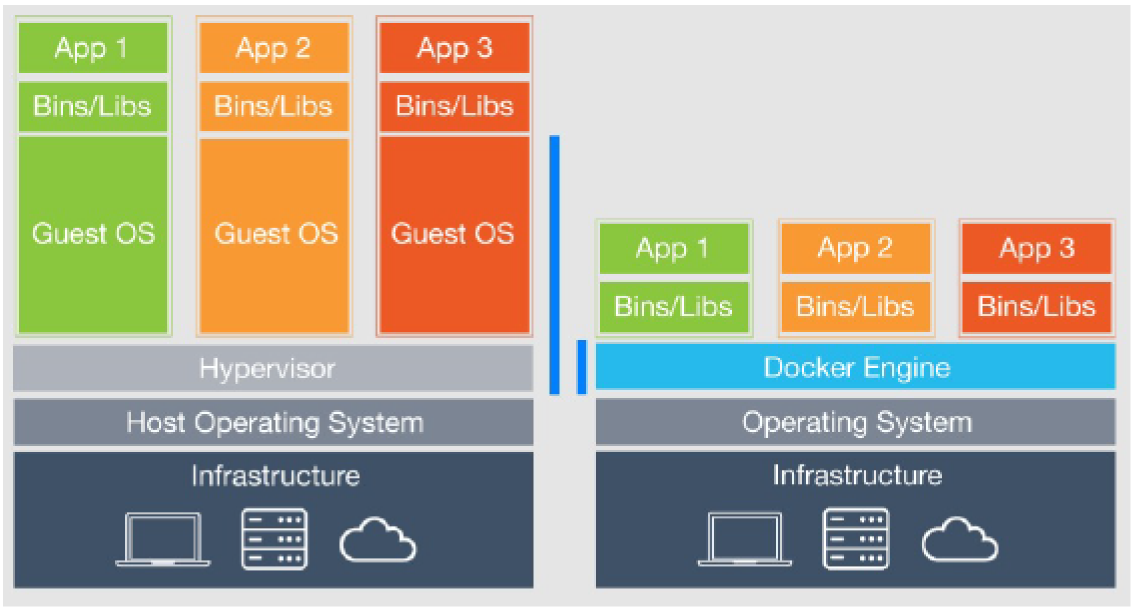
Includes the application, the necessary binaries and libraries and an entire guest operating system.

Container

They have similar resource isolation and allocation benefit as VM but they share the kernel with other containers, so they run as an isolated process in userspace on the host operating system.

Much more portable and efficient.

Images are constructed from layered.



Container (lightweight process virtualization) technology is not new, mainstream support in the vanilla kernel however is.

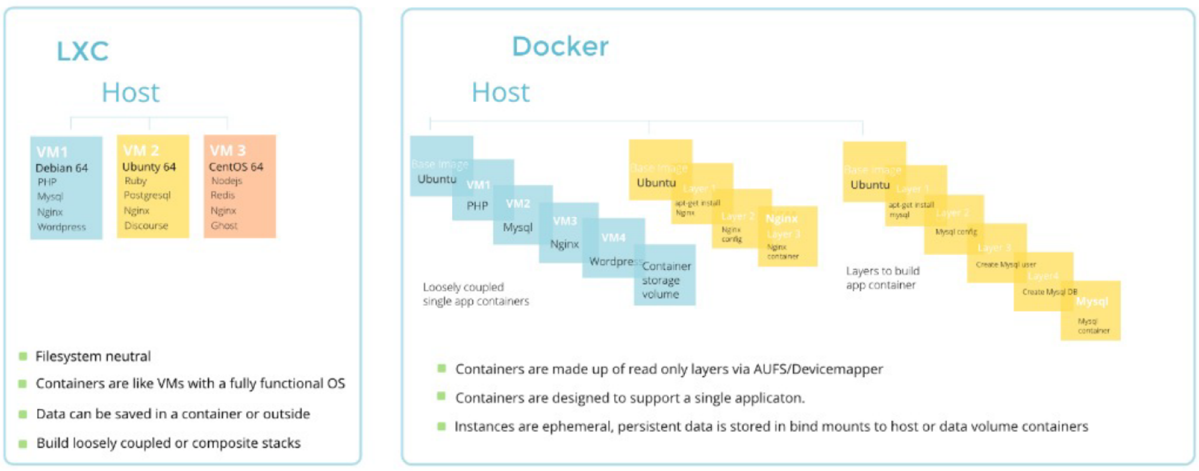
FreeBSD has Jails, Solaris has Zones etc.

Docker is a project (2013) based on LXC project to build single app containers. Docker has developed its own implementation.

9.2 LXC vs. Docker

LXC and Docker are userland container managers that use kernel namespaces to provide containers.

Difference: LXC has an init and can run multiple processes (Docker single).



- Filesystem neutral
- Containers are like VMs with a fully functional OS
- Data can be saved in a container or outside
- Build loosely coupled or composite stacks

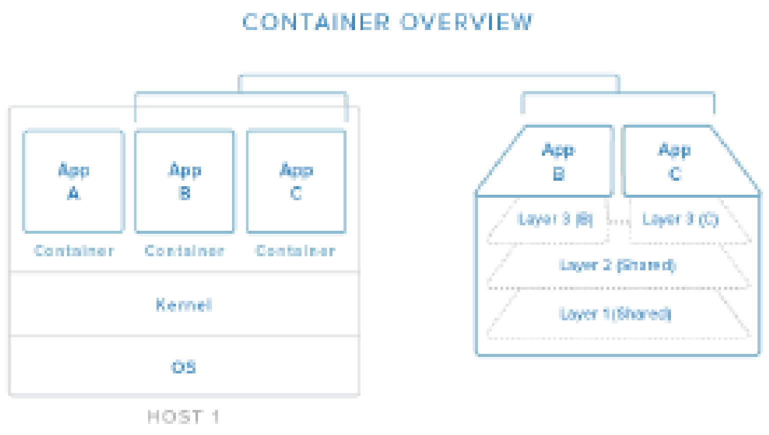
- Containers are made up of read only layers via AUFS/Devicemapper
- Containers are designed to support a single applicaton.
- Instances are ephemeral, persistent data is stored in bind mounts to host or data volume containers

9.3 Docker

Allows to package application with al dependencies into a standardized unit for sw development.

Docker containers wrap a piece of sw in a complete filesystem that contains everything need for running;

Is platform-independent



Containers isolate individual applications and use operating system resources that have been abstracted by Docker.

It works best to break out functionality in individual containers (micro-service architecture).

Easy scale and updatable

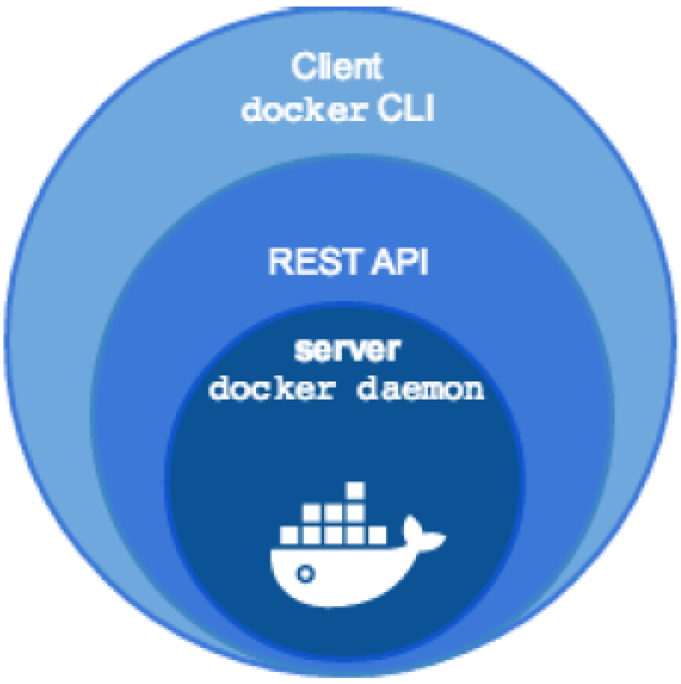
Advantages

Lightweight resource utilization (not use virtualization of OS)

Portability (all of the dependencies are bundled inside the Container)

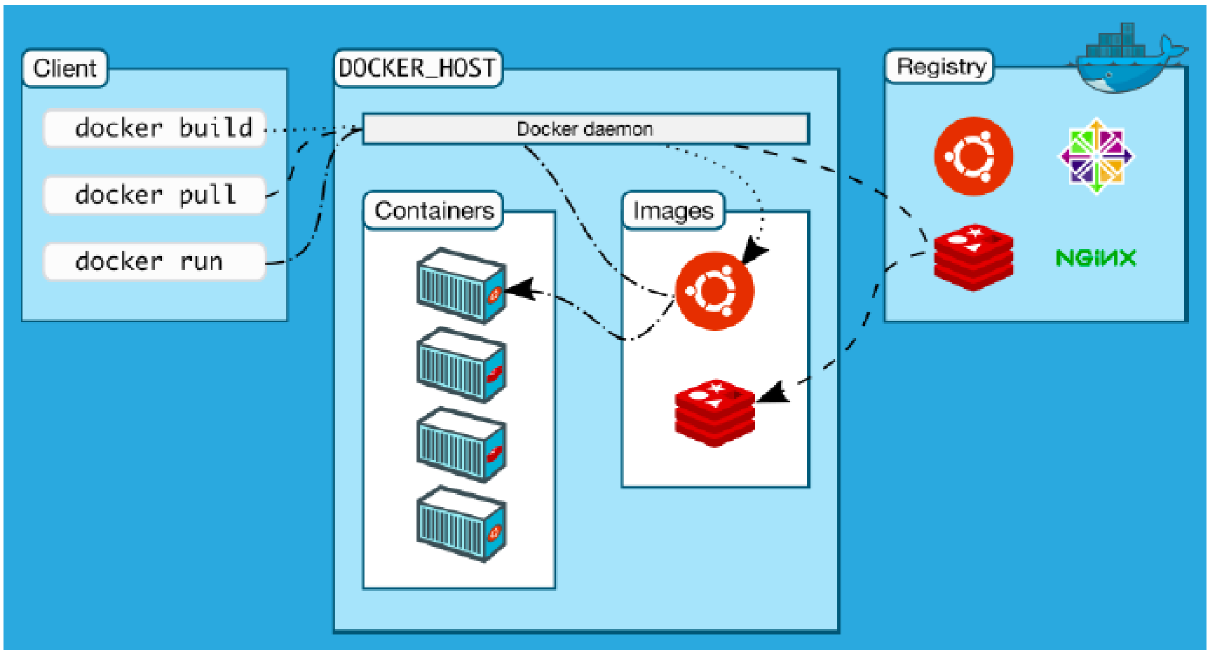
Predictability (The host and the container don't care about each other)

9.4 Docker Engine



Client/Server app made up of the Docker daemon, a REST API that specifies how to interact with the Daemon, and a CLI that talks with the daemon.

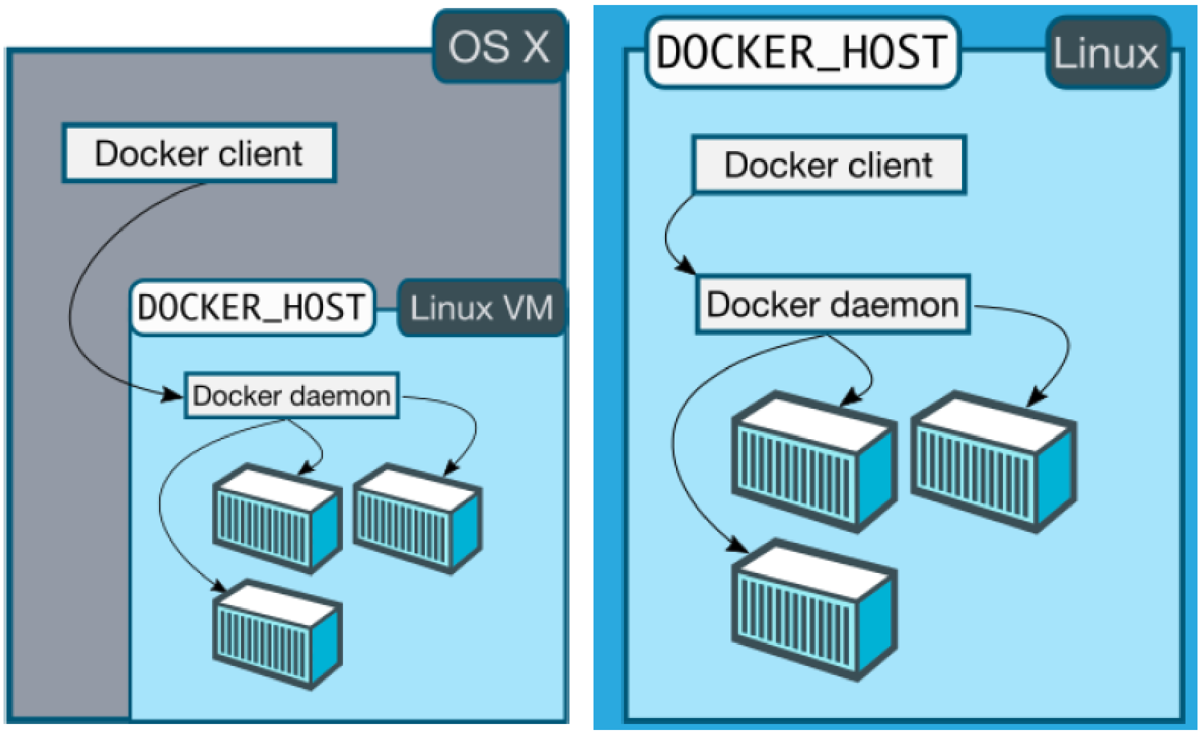
9.5 Docker architecture



The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.

Both the Docker client and the daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.

The Docker client and daemon communicate via sockets or through a RESTful API.



SEE Example p. 18 - 26

9.6 Docker Compose

It's a tool for defining and running multi-container Docker apps. You use a Compose file to configure services.

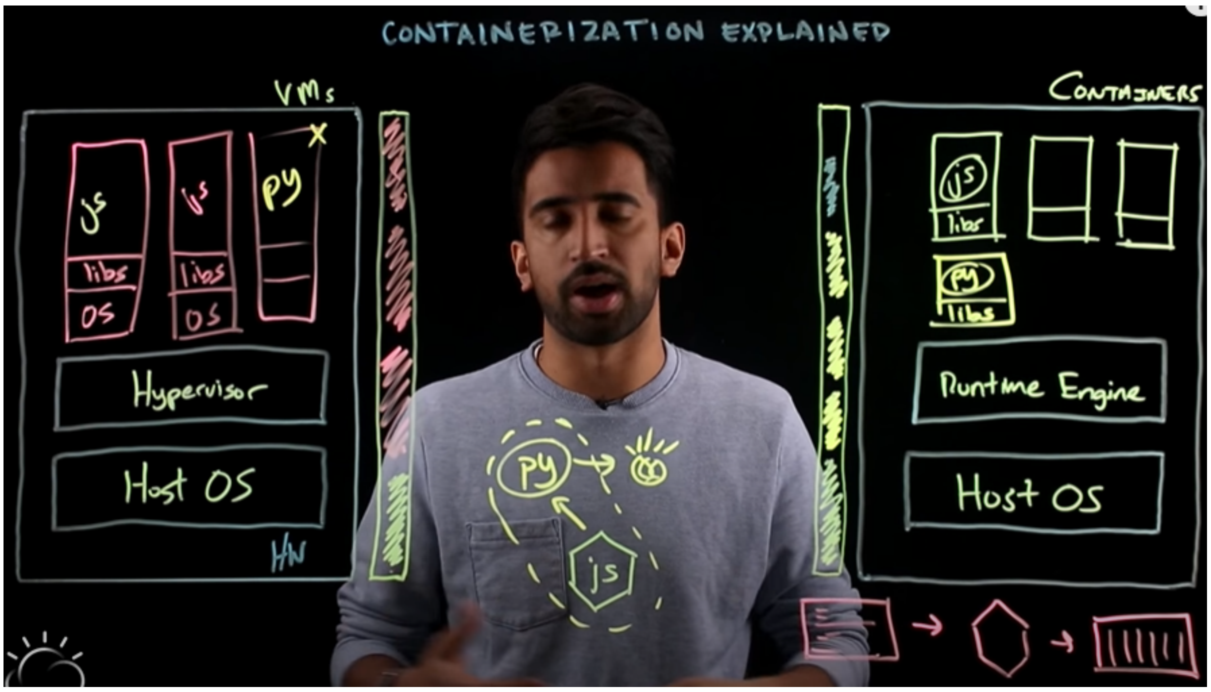
3 steps:

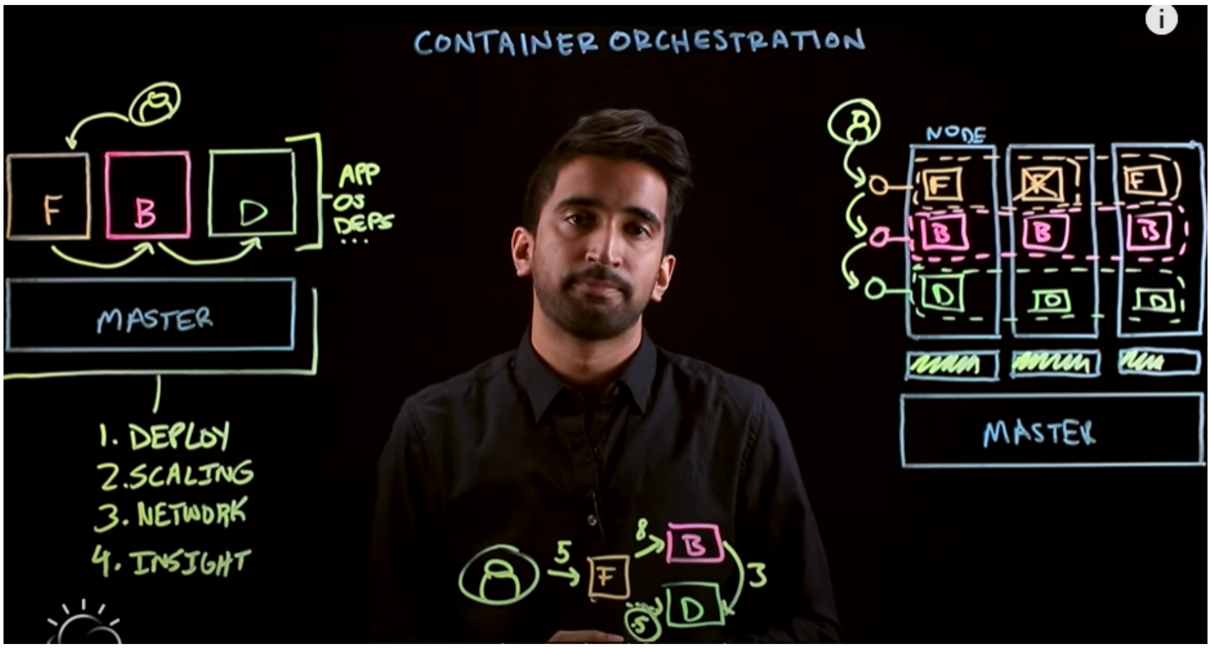
1) Define app's env (**Dockerfile**)

2) Define services (**compose.yml**)

3) Run **docker-compose up** to start

SEE Tutorial from p. 30




CONTAINER ORCHESTRATION
F
B
D
APP
OS
DEPS
...
MASTER
1. DEPLOY
2. SCALING
3. NETWORK
4. INSIGHT
NODE
MASTER
5
8
3