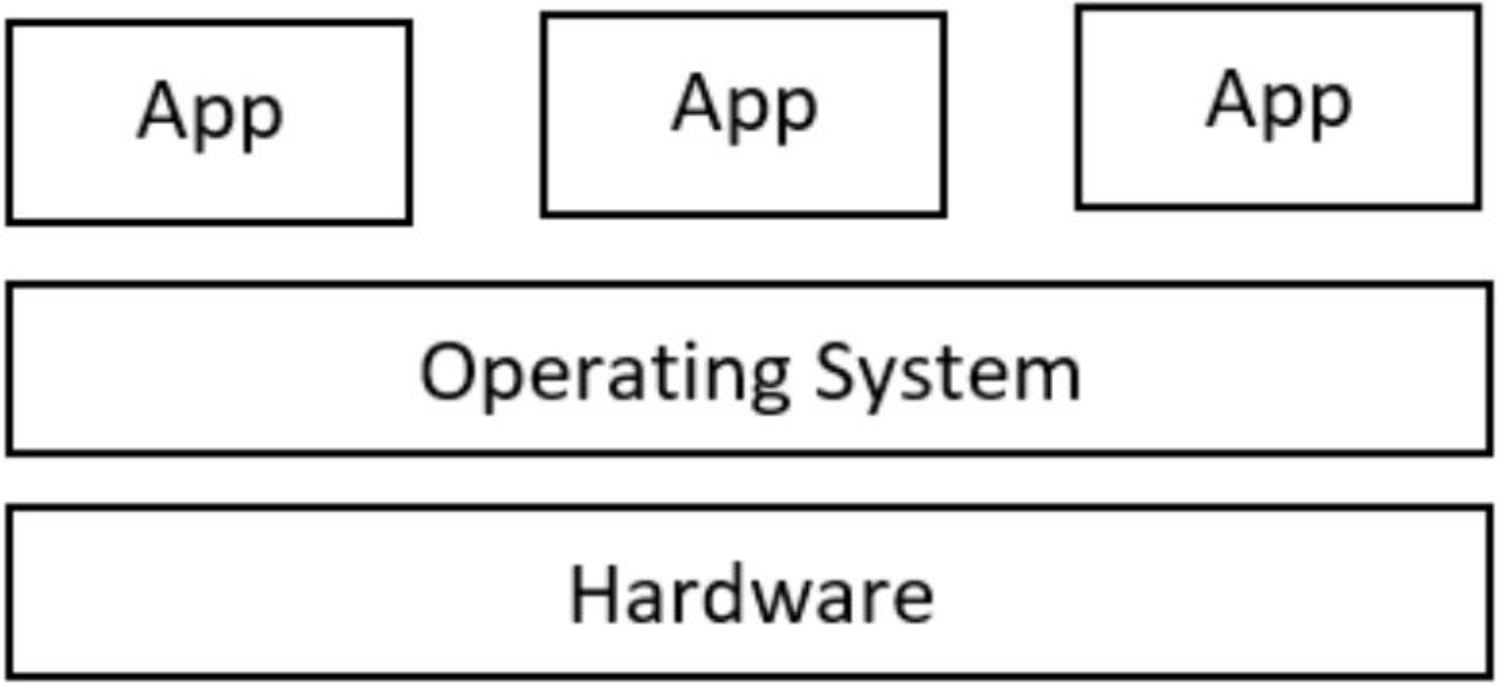
Docker 101 Tutorial

02/19/2023

Agenda

- Virtualization: Virtual Machine and Container
- Docker Architecture and Principles
- Docker Use Cases
- Podman: Daemonless and Rootless Container Engine

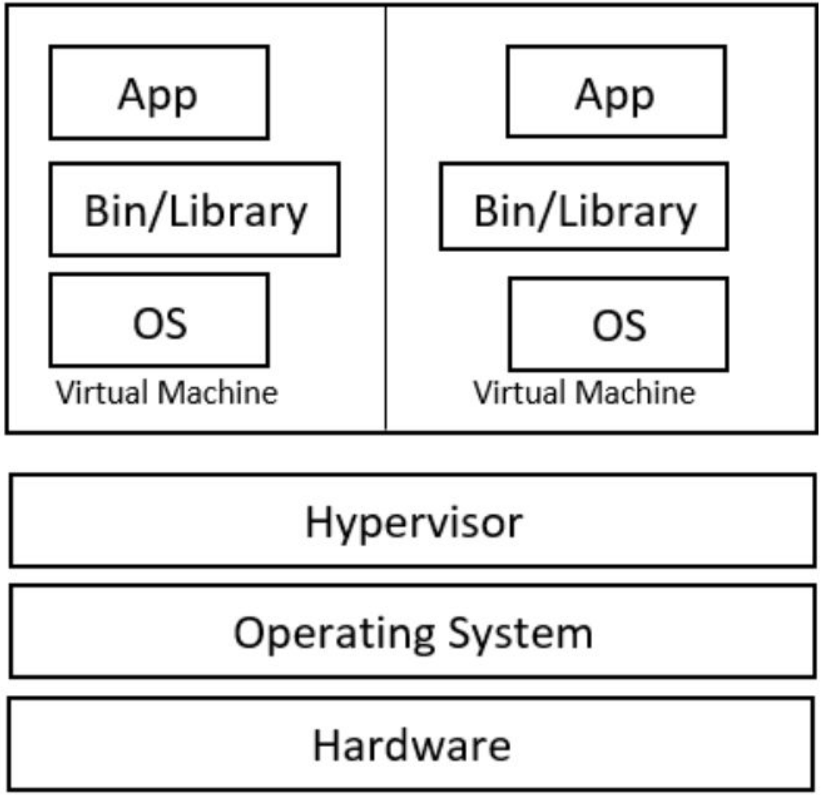
Physical Server Architecture



Physical Server Problems

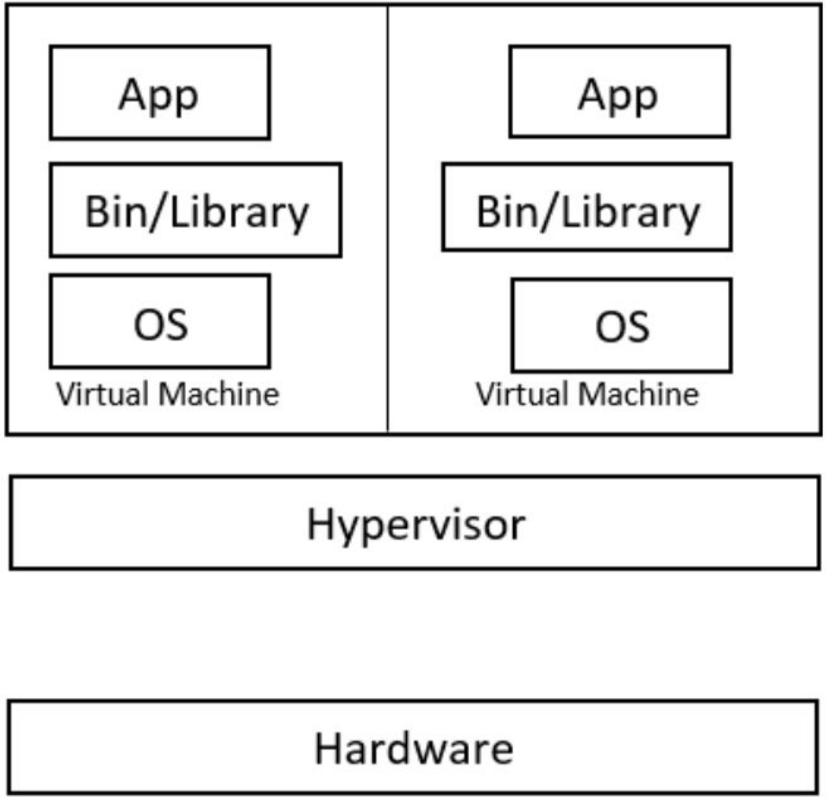
- Limited servers
- Run old and incompatible software
- Develop cross-platform applications
- Try new operating systems
- Sandbox environment, such as virus testing

Virtual Machine: Type-2 Hypervisor



- Hypervisor: Virtual Machine Manager/Monitor(VMM)
- Hosted Hypervisor
- Commercial products in 2000s
- Typical Type-2 Hypervisor Products
  - VMWare Workstation Player/Pro
  - Oracle VM VirtualBox
  - QEMU

Virtual Machine: Type-1 Hypervisor



- ➢ Bare metal hypervisor

- ➢ Typical Type-1 Hypervisor Products
  - VMWare ESX/ESXi(vSphere)
  - Microsoft Hyper-V
  - Oracle VM Server
  - Citrix Hypervisor(XenServer)
  - KVM

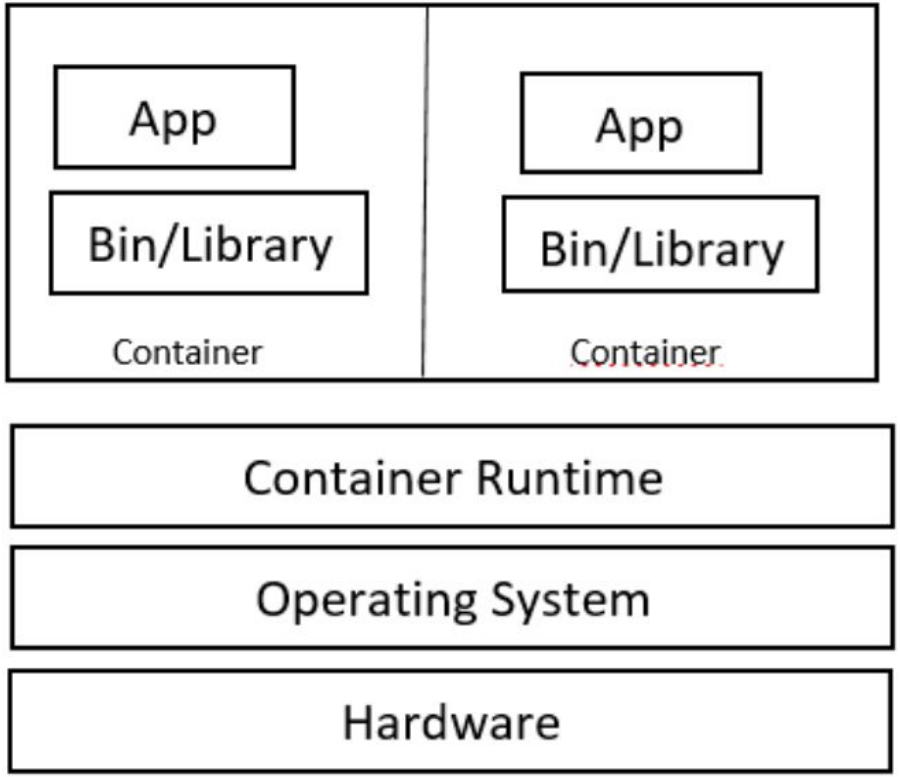
Type-1 Virtual Machine Real World Examples

- AWS Elastic Compute Cloud(EC2)
- Microsoft Azure Virtual Machine(Azure)
- Google Cloud Platform(GCP)
- Alibaba Cloud Elastic Compute Service(ECS)
- Tencent Cloud Virtual Machine(CVM)
- SmartX(native hypervisor ELF, KVM based)

Virtual Machine Problems

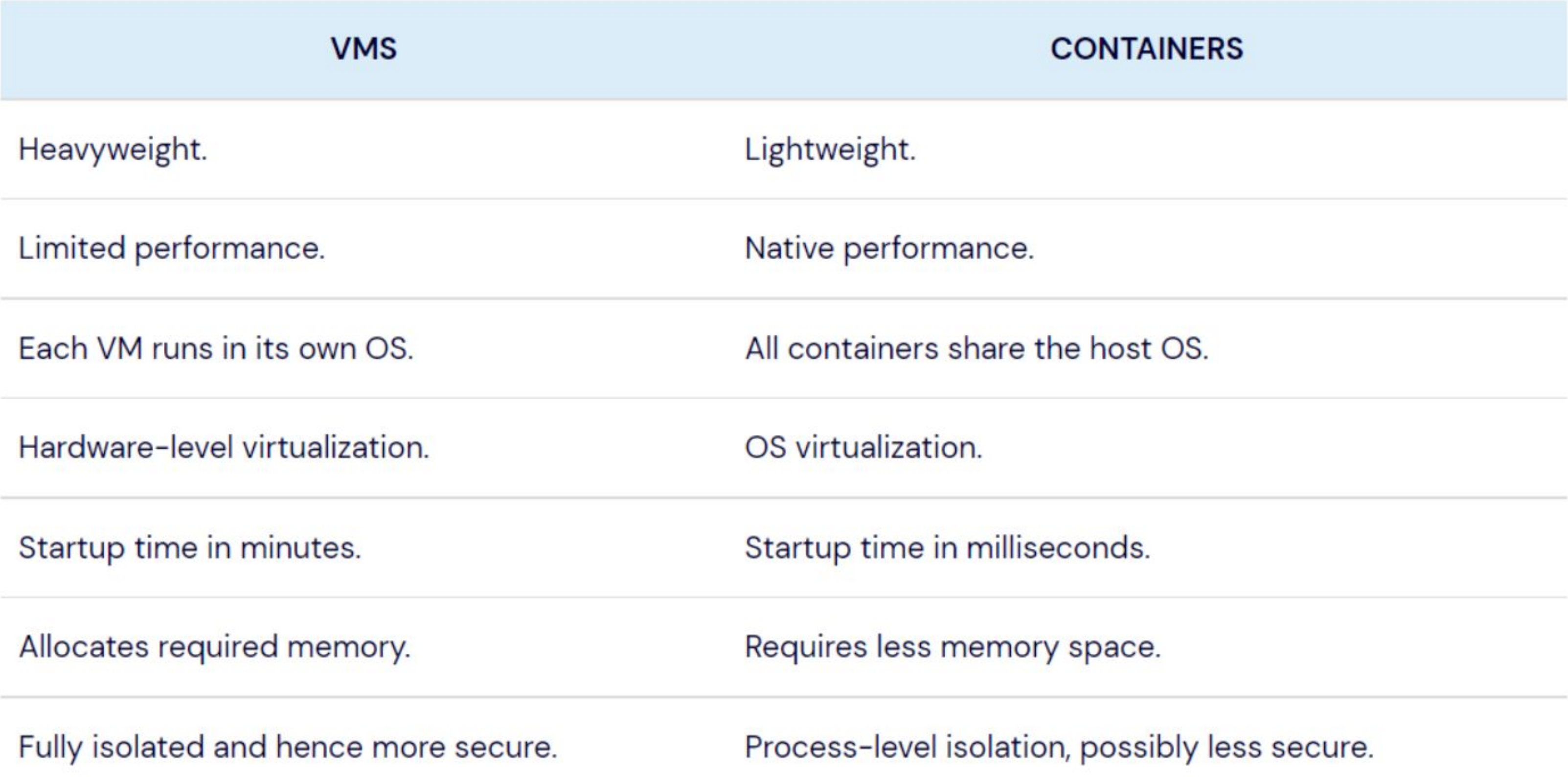
- Resource Intensive
- Slow startup and shutdown
- Deployment complexity
- Scalability

Container



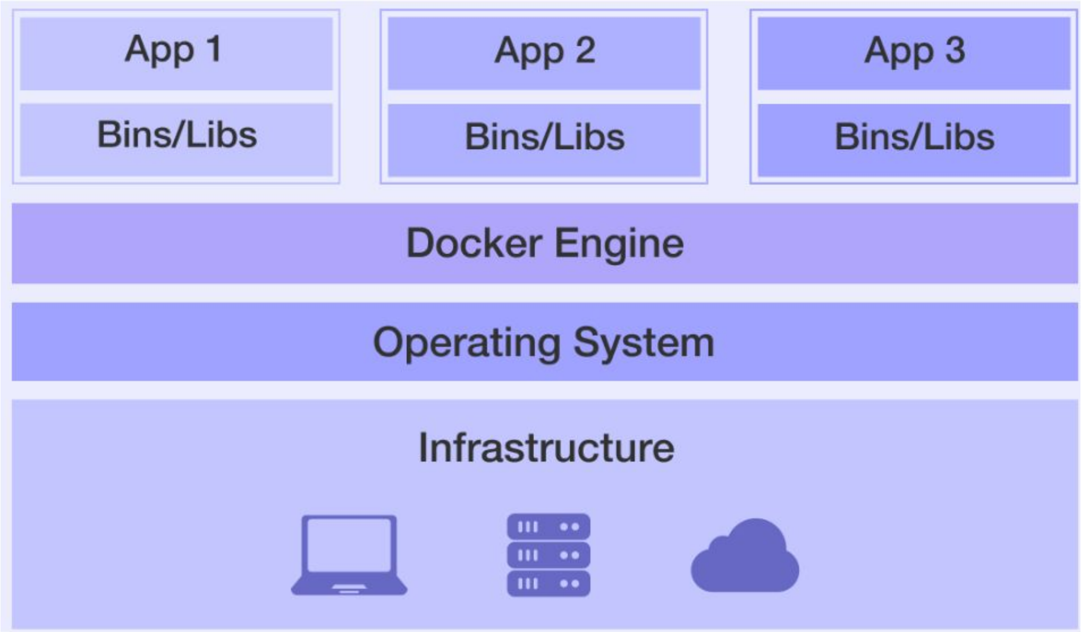
- OS-level virtualization, share OS kernel
- Isolated process
- LXC: Linux Container(2008)
- Namespaces and CGroups

Virtual Machine vs Container



VMS	CONTAINERS
Heavyweight.	Lightweight.
Limited performance.	Native performance.
Each VM runs in its own OS.	All containers share the host OS.
Hardware-level virtualization.	OS virtualization.
Startup time in minutes.	Startup time in milliseconds.
Allocates required memory.	Requires less memory space.
Fully isolated and hence more secure.	Process-level isolation, possibly less secure.

Docker Infrastructure

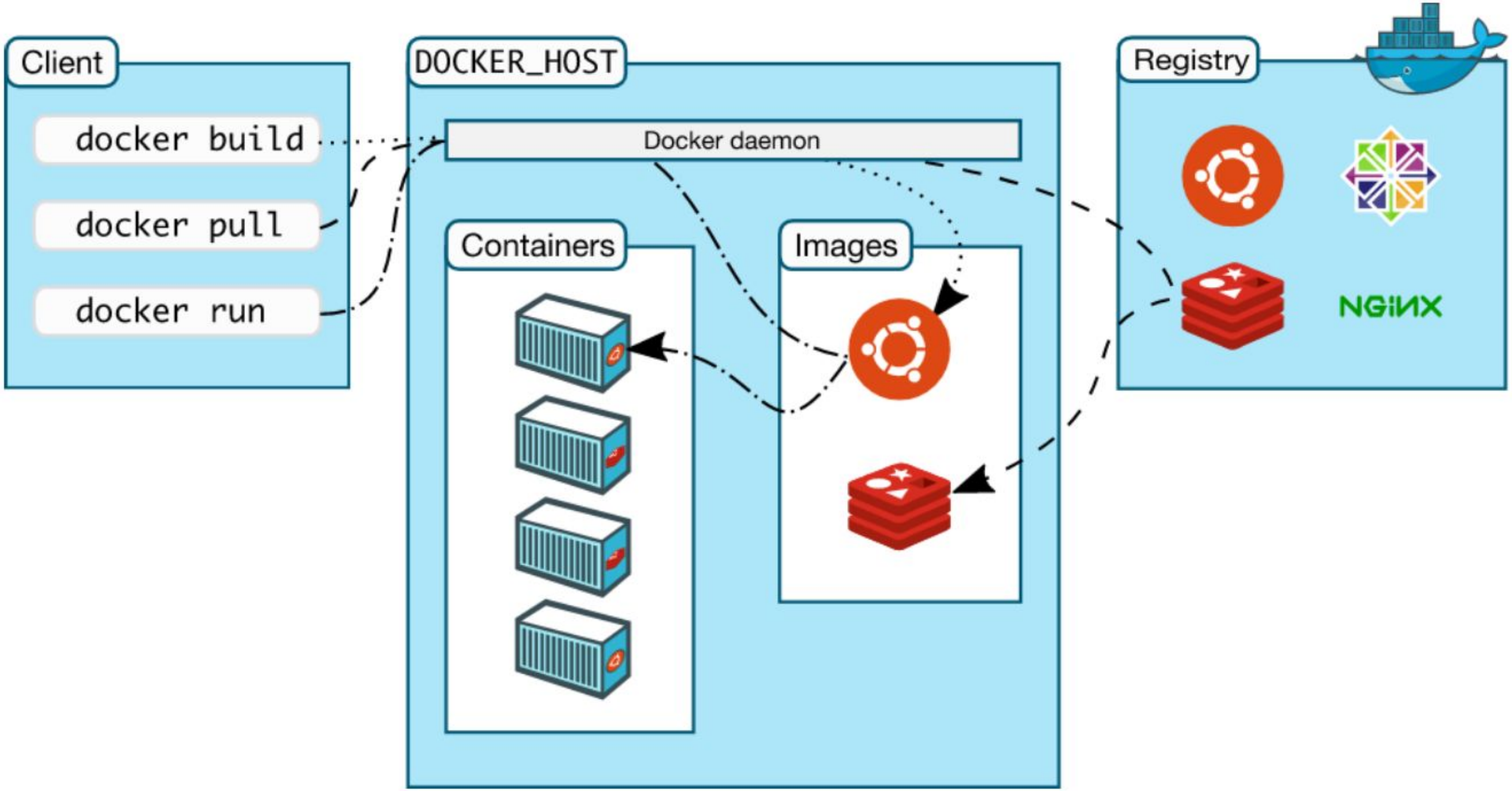


- ➢ Container technology based
- ➢ OS-level virtualization
- ➢ Innovation on LXC(2013)
- ➢ Docker Engine = dockerd + containerd + runC

Docker Added to LXC

- Portable deployment across machines
- Application-centric
- Automatic build: Dockerfile
- Versioning
- Component reuse
- Sharing
- Tool ecosystem

Docker Architecture: client/server



Docker Engine

- Server: It is the docker daemon called dockerd. It can create and manage docker images, containers, networks, etc.
- Rest API: It is used to instruct docker daemon what to do.
- Command Line Interface (CLI): It is a client which is used to enter docker commands.

Docker Registry

- Similar to git repository
- Store docker images
- Public registry: Docker Hub
- Private registry

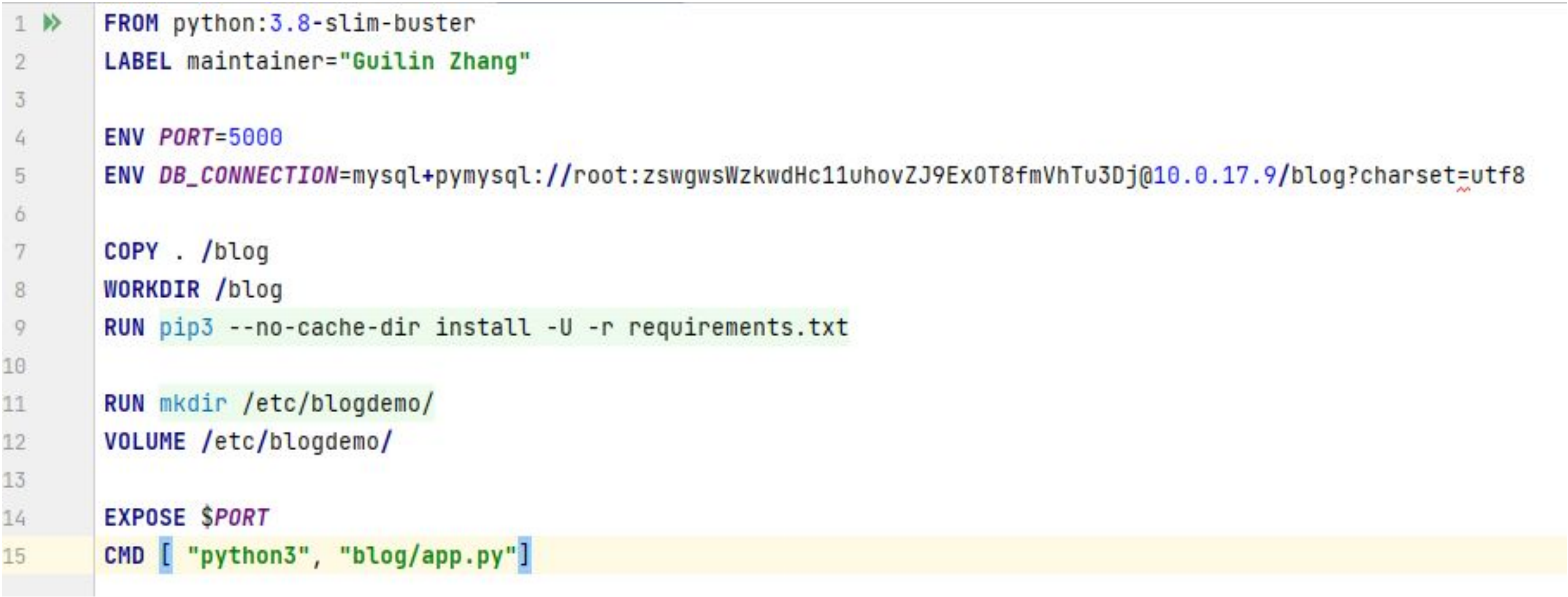
Docker Objects

- images
- containers
- volumes
- networks

Docker Images

- read-only template with instructions for creating a Docker container
- image = base image + customization
- package: code+configuration+dependencies
- Dockerfile -> docker image

Dockerfile

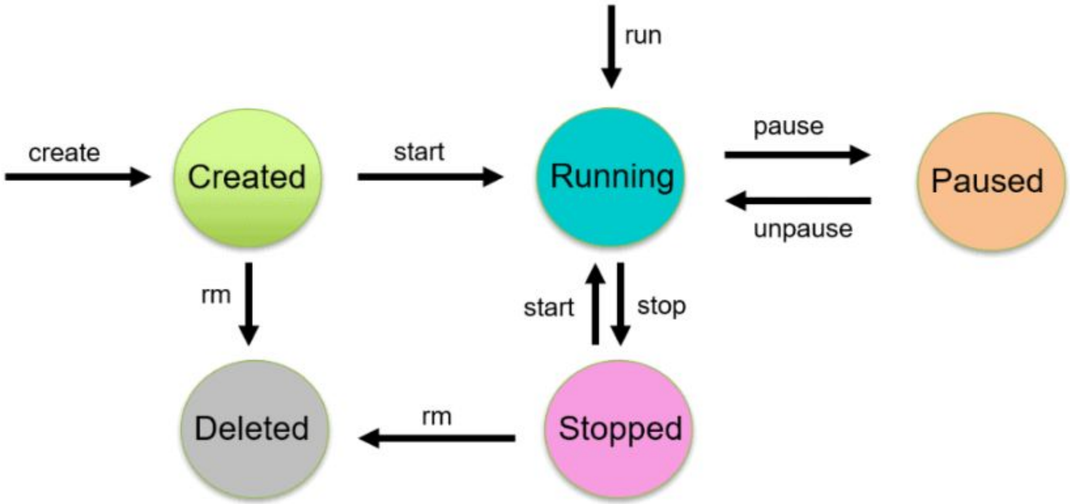


```dockerfile
FROM python:3.8-slim-buster
LABEL maintainer="Guilin Zhang"

ENV PORT=5000
ENV DB_CONNECTION=mysql+pymysql://root:zswgwsWzkwdHc11uhovZJ9ExOT8fmVhTu3Dj@10.0.17.9/blog?charset=utf8

COPY . /blog
WORKDIR /blog
RUN pip3 --no-cache-dir install -U -r requirements.txt

RUN mkdir /etc/blogdemo/
VOLUME /etc/blogdemo/

EXPOSE $PORT
CMD [ "python3", "blog/app.py"]
```

docker build -t python-docker-image:tag path/to/Dockerfile

https://docs.docker.com/engine/reference/builder/

Docker Containers



- container: runnable instance of image
- All the applications and their environment run inside the container
- Docker API or CLI to operate container

Volumes

- Store the persisting data generated by docker and used by Docker containers.
- Volume's content exists outside the lifecycle of a container.

Networks

- **Bridge**: It is the default network driver for a container.
- **Host**: no network isolation between host and container.
- **Overlay**: This network enables swarm services to communicate with each other.
- **None**: disable all the networking.
- **macvlan**: Assigns mac address to containers to make them look like physical devices.

Docker Scenarios

- Application isolation
- Build portable environment
- Microservices
- CI/CD

Docker Problems

- Root privilege: daemon binds unix socket
- Docker user group
- Security problems
- Rootless mode: Docker Engine v20.10, limitations
- https://docs.docker.com/engine/release-notes/
- https://docs.docker.com/engine/security/rootless/

Podman: Pod Manager

- Daemonless and rootless container engine
- Docker command compatible
- alias docker=podman
- https://podman.io/

Key takeaways

- Virtualization Techniques: VM and Container
- VM: type-1 hypervisor vs. type-2 hypervisor
- Docker: A Container Engine
- Docker can't replace VM
- Package up application and all its dependencies
- Podman: rootless container engine

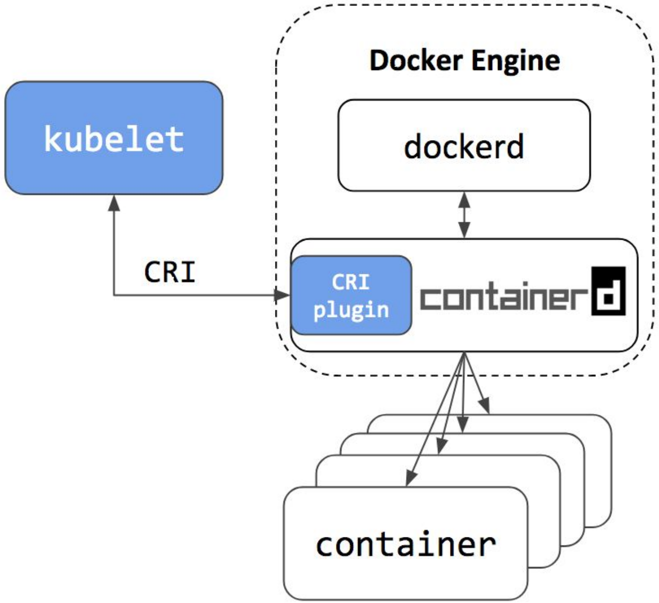
Thoughts

- Multiple containers on a single host machine?
    - Docker Compose

- Multiple containers across multiple host machines?
    - Docker Swarm
    - Kubernetes(K8s)

References

- 官网: https://www.docker.com/
- podman: https://podman.io/
- K8S宣布“弃用”Docker，意味着什么
  - https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/
  - https://acloudguru.com/blog/engineering/kubernetes-is-deprecating-docker-what-you-need-to-know
  - https://mdnice.com/writing/9337a24f4ceb47c4b7517d567e2fa5e7


Docker Engine
kubelet
dockerd
CRI
CRI plugin
containerd
container