Deployment and Environment Isolation with Docker !!

Dec. 10th, 2018

Computer System Lab. (A6)

Oriishi, Kohda, Miyazaki, Nomura, Kawanaka, Sakai, Takashima

AGENDA

1. Virtualization Technology
2. Docker Technology
3. Docker Lifecycle
4. Security Problems of Docker
5. Advanced Technology

1. Virtualization Technology

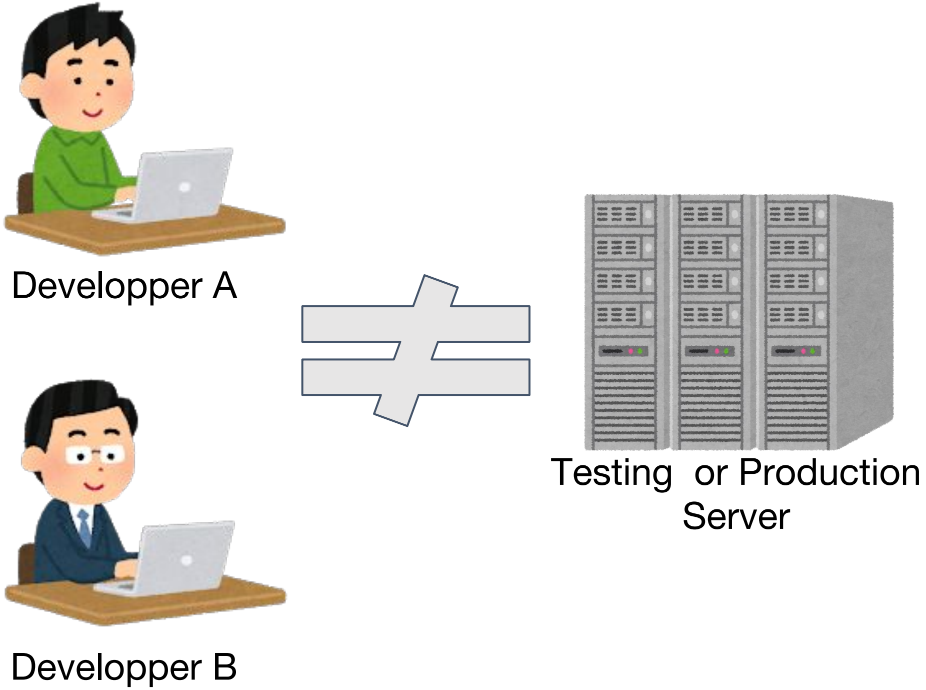
Reason for Virtualization

- ❑ Unify the Environment
- ❑ Reliability, Availability and Serviceability
- ❑ Cost Down
- ❑ Improve computer performance

Unify the Environment

Differences

- ❑ Kernel
- ❑ Runtime
- ❑ MiddleWare



Developper A

≠

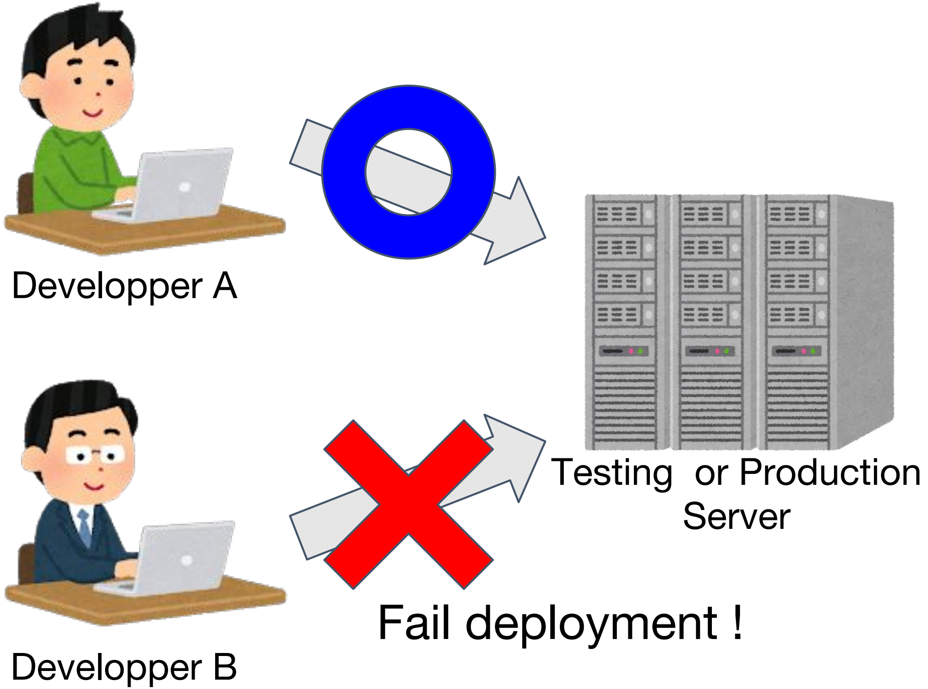
Testing or Production Server

Developper B

Unify the Environment

Differences

- ❑ Kernel
- ❑ Runtime
- ❑ MiddleWare



Developper A

Developper B

Testing or Production Server

Fail deployment !

□ Unify the Environment

Provisioning tools (Infrastructure as Code)

❑ Automate provisioning
(install and setup operations).

❑ Guarantee idempotence.
(Ansible, Chef, Terraform etc.)

Unify the Environment

Virtualization

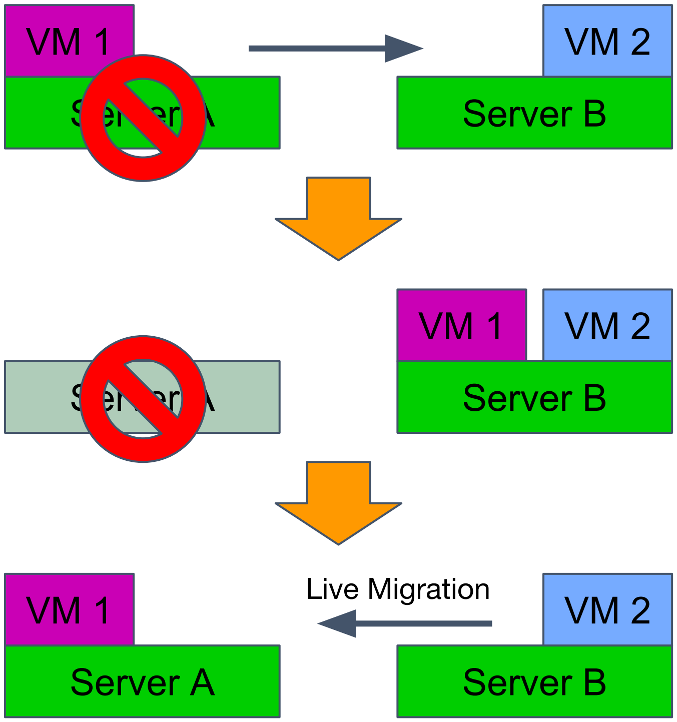
- ❑ Packaging environment.
  (library, runtime, middleware, kernel etc.)
- ❑ Can share same environment with some machines.
  (VirtualBox, Xen, ESXi, KVM, Docker etc.)

Reliability, Availability and Serviceability

Failover

Shorten system failure time

1. If Server-A Failed
2. Restart VM1 on Server-B
3. After Server A Recovery,
   VM 1 Returns to Server A

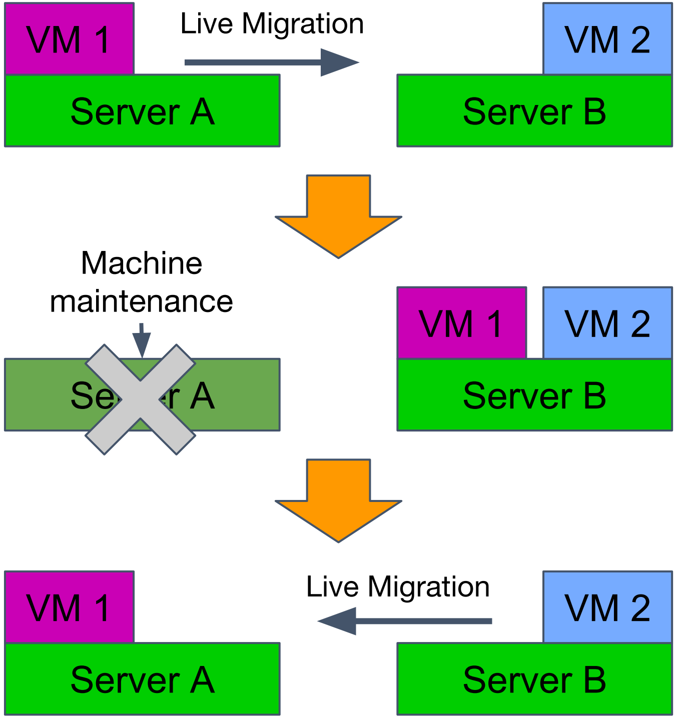


󰀀 Reliability, Availability and Serviceability

Non stop service

Non stop the service by maintenance

1. Move VM1 to Server B
2. Maintenance Server A
3. VM 1 return to Server A

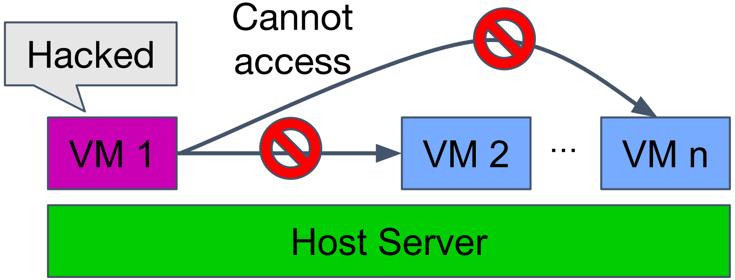


Reliability, Availability and Serviceability

Secure

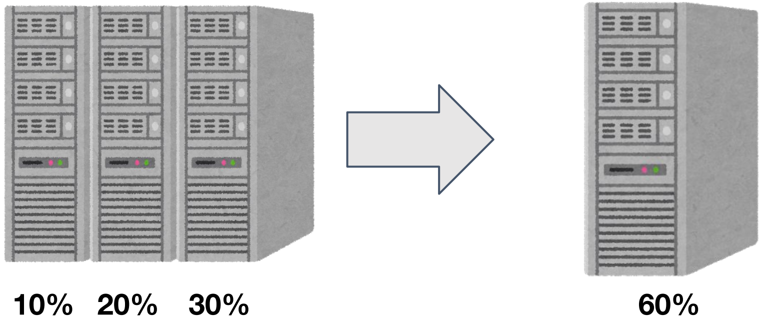
Completely separate from other VMs

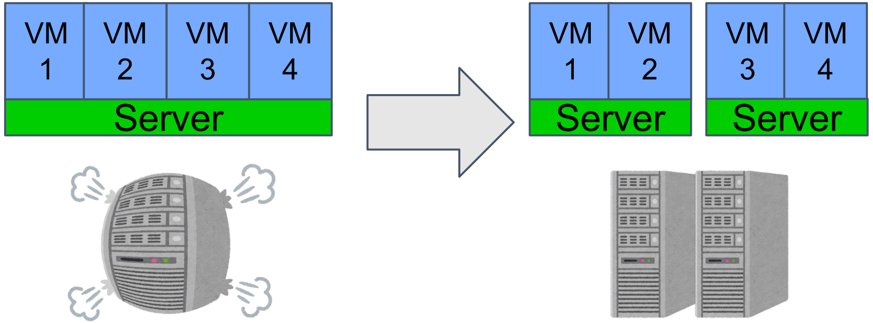
- If VM1 is hacked,
  it is possible to protect other VM's access



Cost Down

- ❑ Reduce required server equipment
- ❑ Usually, horizontal scaling is low-cost than vertical scaling.
  (ref. Moor's law)





□ X86 Virtualization with Hardware support

- ❑ Virtualize on the hardware side
- ❑ Possible to operate without modifying the guest OS
- ❑ Significantly improved performance
- ❑ Hardware-Assisted Virtualization
  - – Intel : Intel VT
  - – AMD : AMD-V

Virtualization Type

❏ Native hypervisors

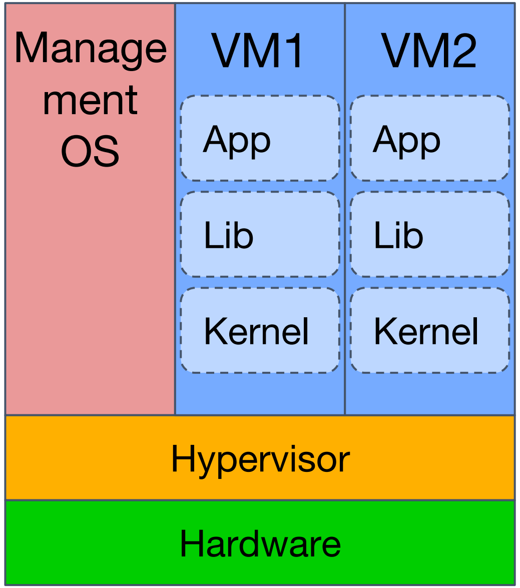
❏ Hosted hypervisors

❏ KVM (Kernel-based Virtual Machine)

❏ LXC (LinuX Container)

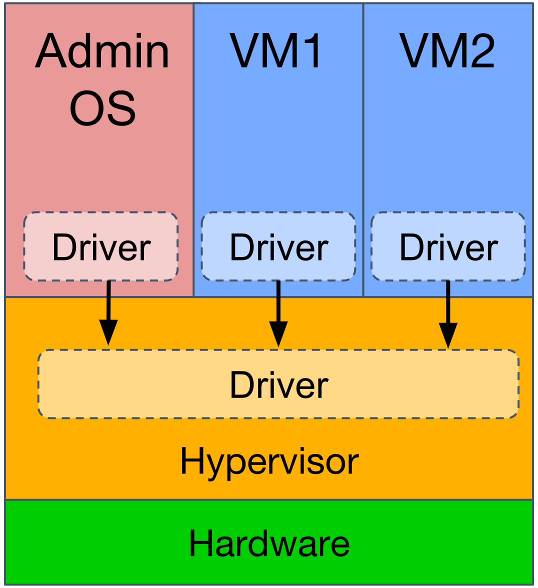
Native Hypervisor

- ❑ A hypervisor provide emulated hardware to VMs.
- ❑ VMs contain kernel, libraries and application binaries.
- ❑ Divided into two subcategories: Monolithic Kernel and MicroKernel
- ❑ Xen, ESXi, Hyper-V etc.

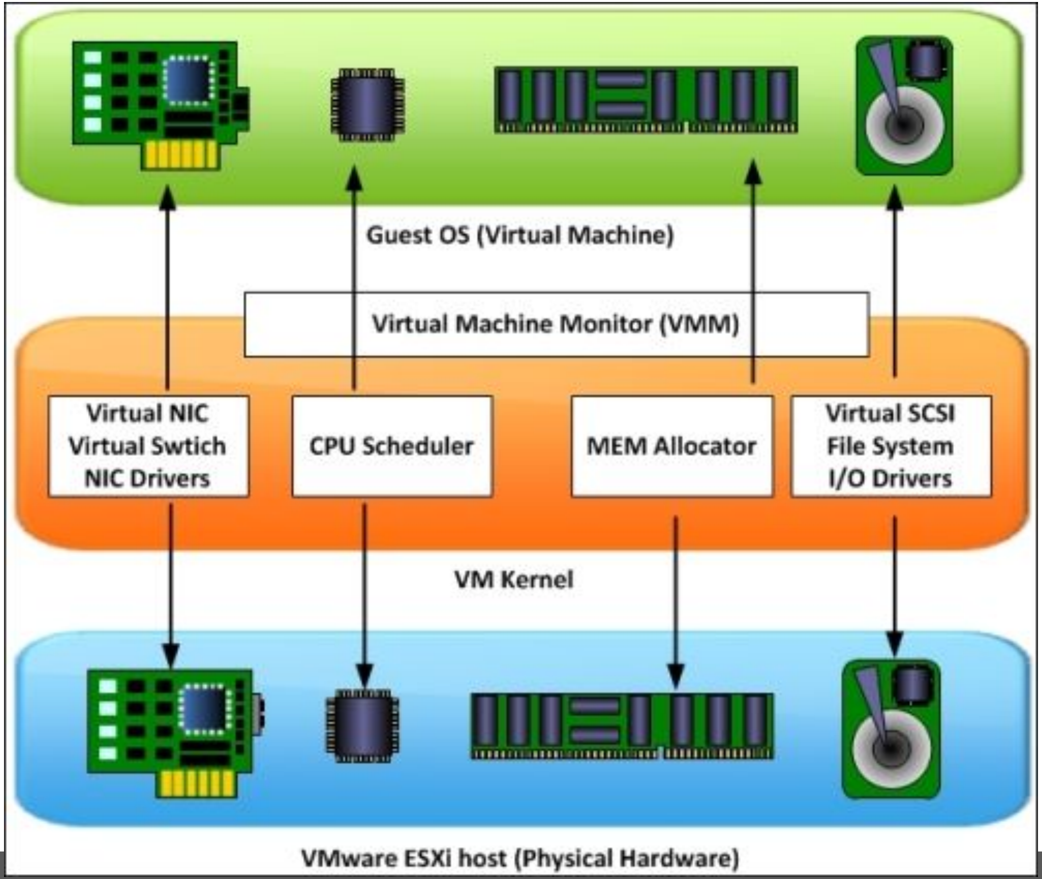


Monolithic Kernel

- ❑ Not separate address space
- ❑ Have to change device driver
- ❑ All processes are process with the hypervisor
- ❑ Non context switch



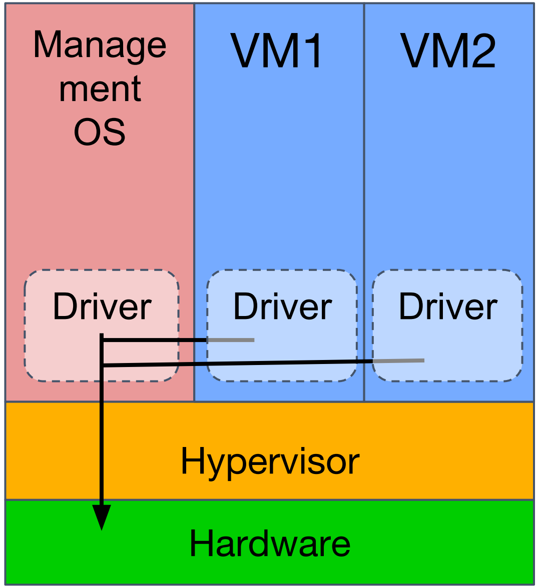
ESXi Architecture



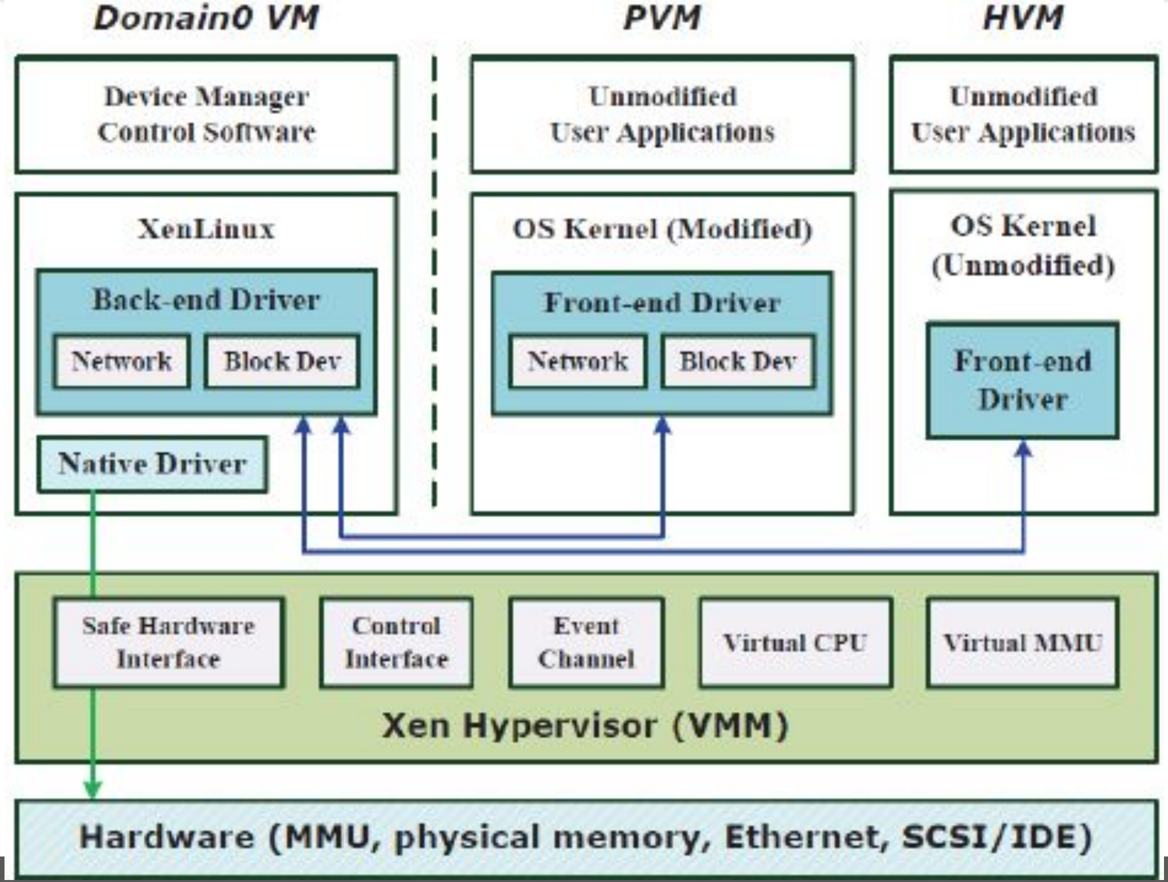
https://www.packtpub.com/mapt/book/virtualization_and_cloud/9781782174851/3/ch03lvl1sec21/the-vmware-vsphere-esxi-architecture

Microkernel

- ❑ Separate address space
- ❑ Can use the default device driver
- ❑ Minimal hypervisor function



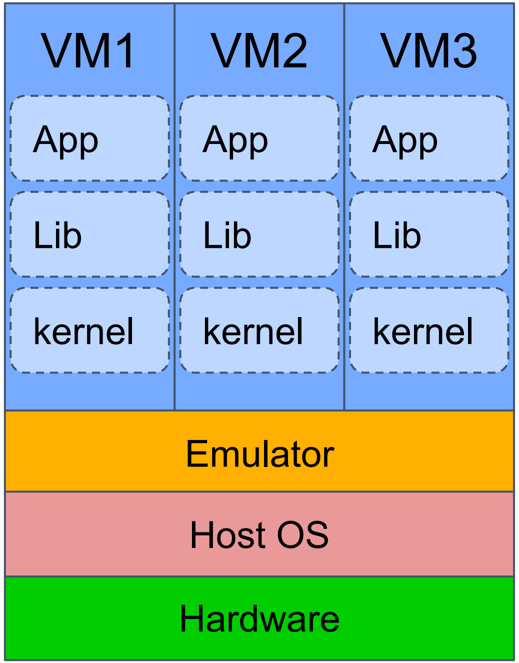
Xen Architecture



Peijie Yuz *et all.*, Real-time Enhancement for Xen hypervisor, IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2010

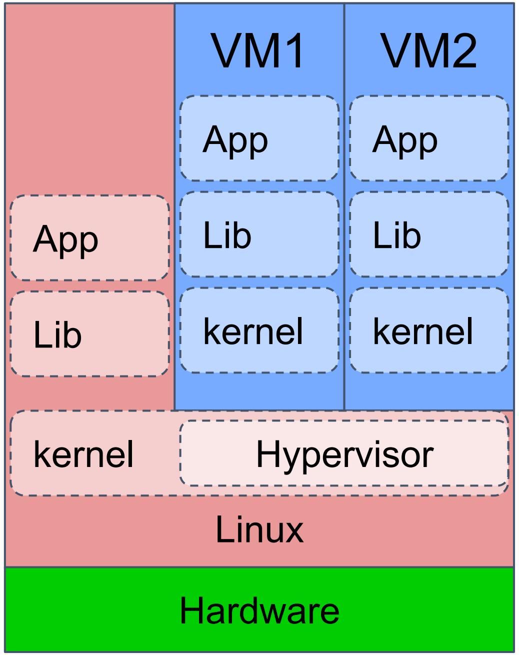
Hosted Hypervisor

- ❑ An emulator program runs on Host OS.
- ❑ The emulator program provides emulated hardware resources.
- ❑ VirtualBox, VMWare, QUEM etc.

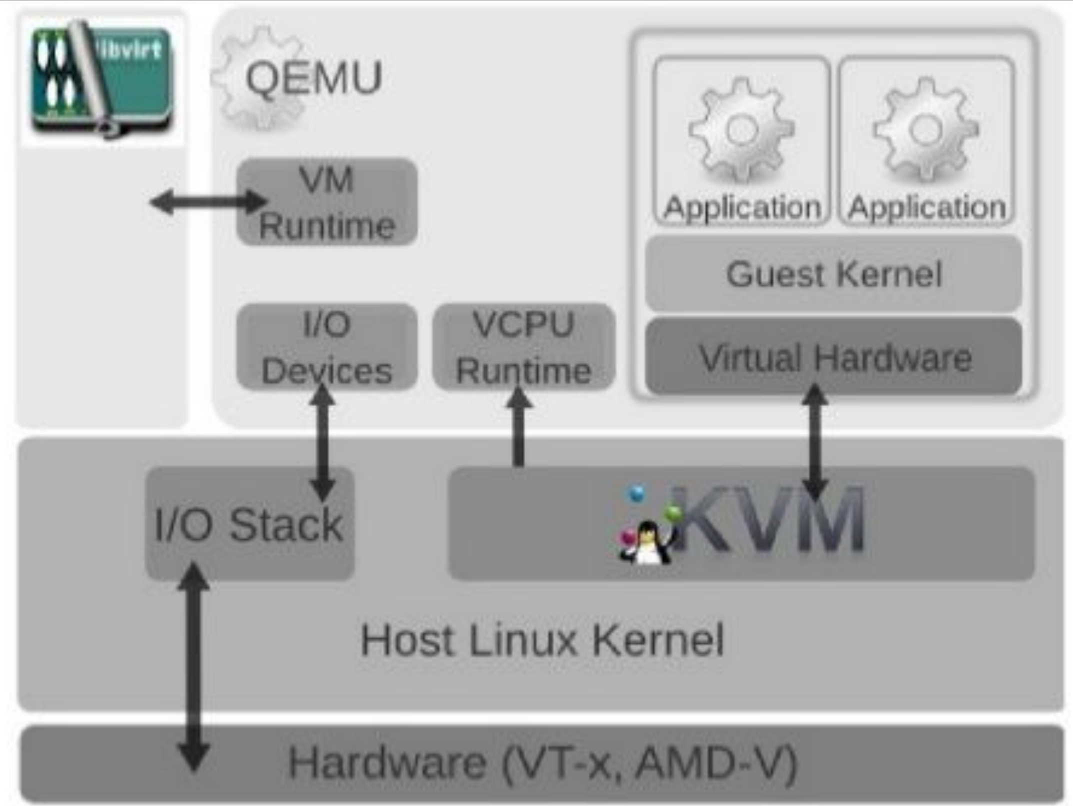


KVM (Kernel-based Virtual Machine)

- ❑ Linux kernel turn into a hypervisor
- ❑ Merge to Linux kernel in version 2.6.20
- ❑ Virtualize CPU and available in /dev/kvm
- ❑ Used by some cloud vendors (GCP, AWS)



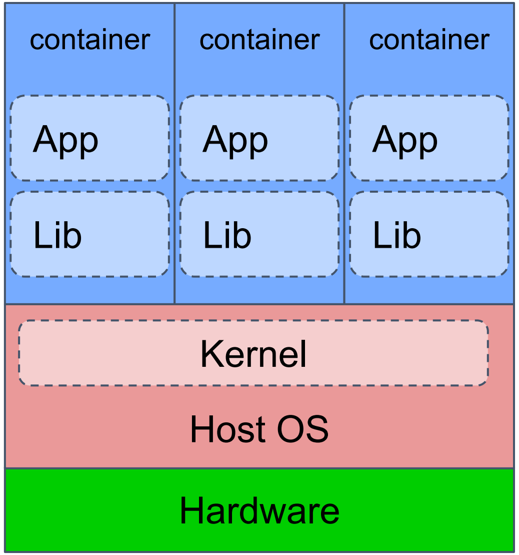
KVM Architecture



https://www.slideshare.net/pradeepkumarsuvce/virtualization-architecture-kvm

LXC (LinuX Container)

❏ Containers are processes which isolated by
- chroot (file system)
- cgroup (process resource)
- namespace (system resource)
- capabilities (privilege)



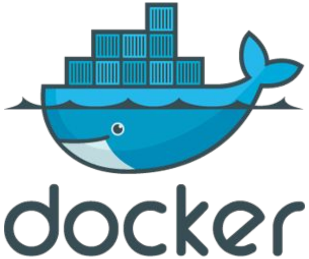
Compare LXC with VM

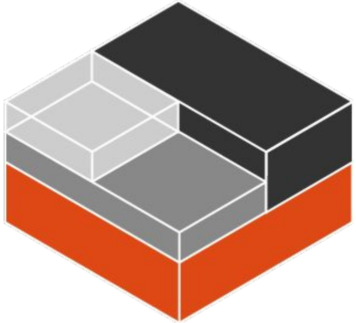
❑ LXC isolate processes on Host OS.
(Make environments for execute applications)

❑ All containers share kernel and its parameters.
(Cannot optimize kernel parameters
of each container)

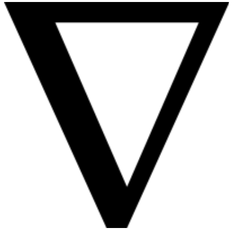
❑ LXC runs fast than VM because
LXC doesn’t emulate hardware resources.

Container Runtime





LXC

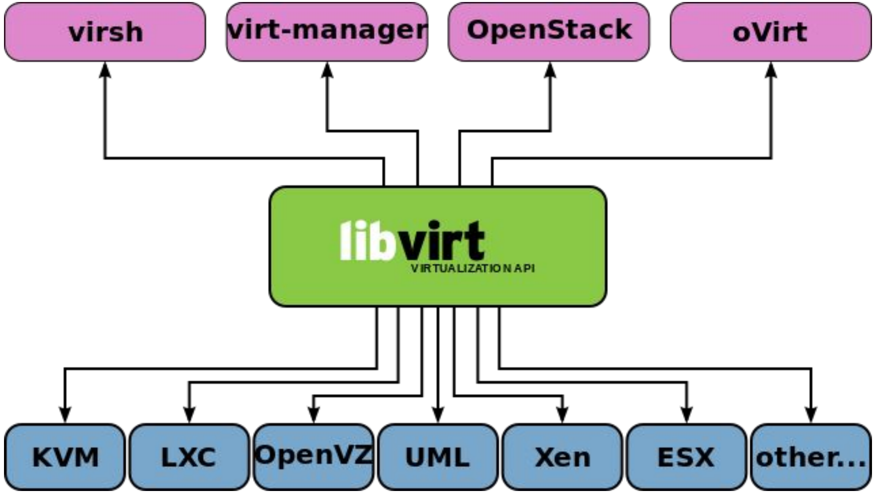


Nabla

Libvirt

- ❑ Control virtual machines library
- ❑ Many hypervisors support

    KVM, QEMU, LXC,
    Xen, ESXi and other



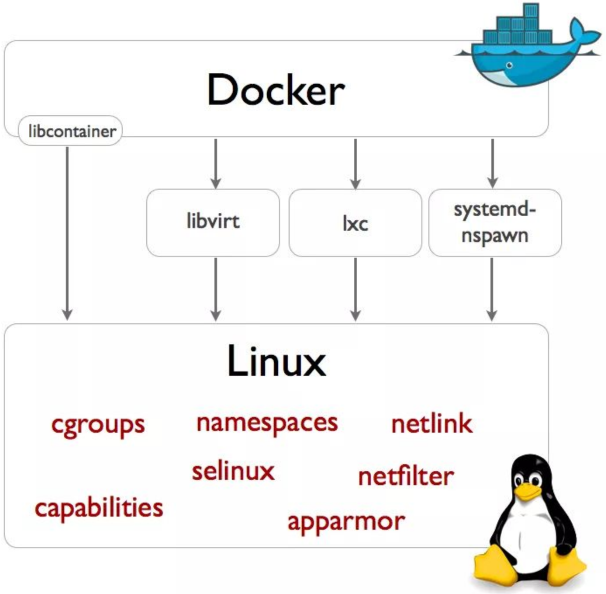
https://en.wikipedia.org/wiki/Libvirt

2. Docker Technology

Docker

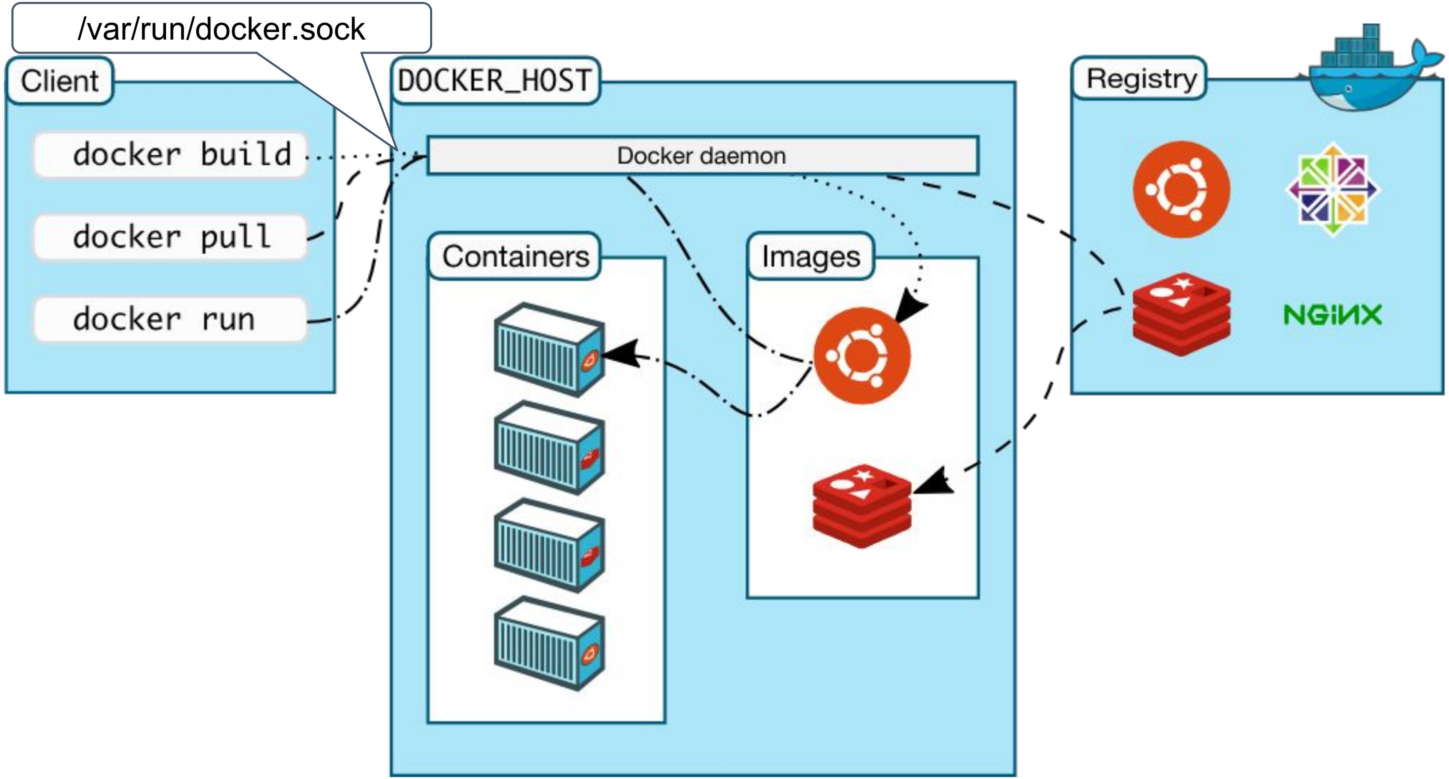
- Docker accesses kernel’s container API through a driver. (default driver is libcontainer)
- Docker provides an ecosystem
  - container portability
  - version management
  - easy building

  to users.



https://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer

Components of Docker



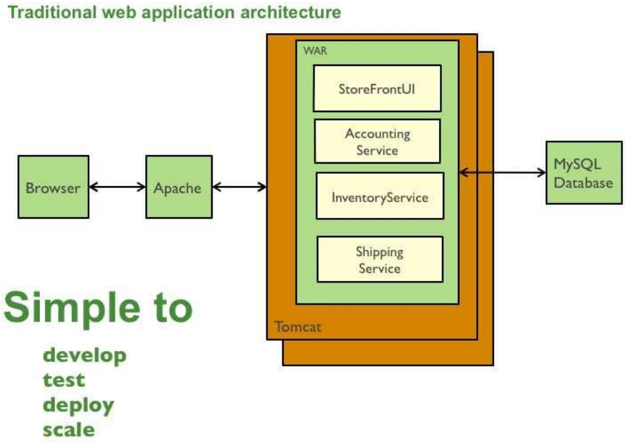
https://docs.docker.com/engine/docker-overview/#docker-architecture

Main Applications of Docker

- ❏ Microservices architecture
- ❏ Cloud platforms
- ❏ CI/DI tools
- ❏ Distribute environments with Docker registry

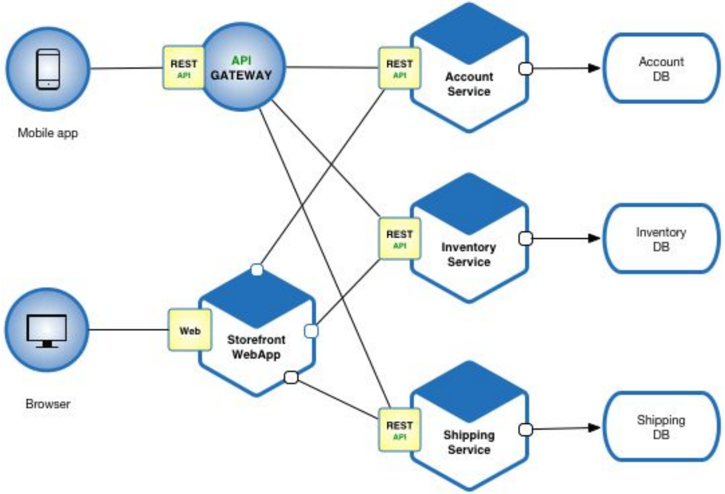
Architectures of WEB APPs

Monolithic architecture



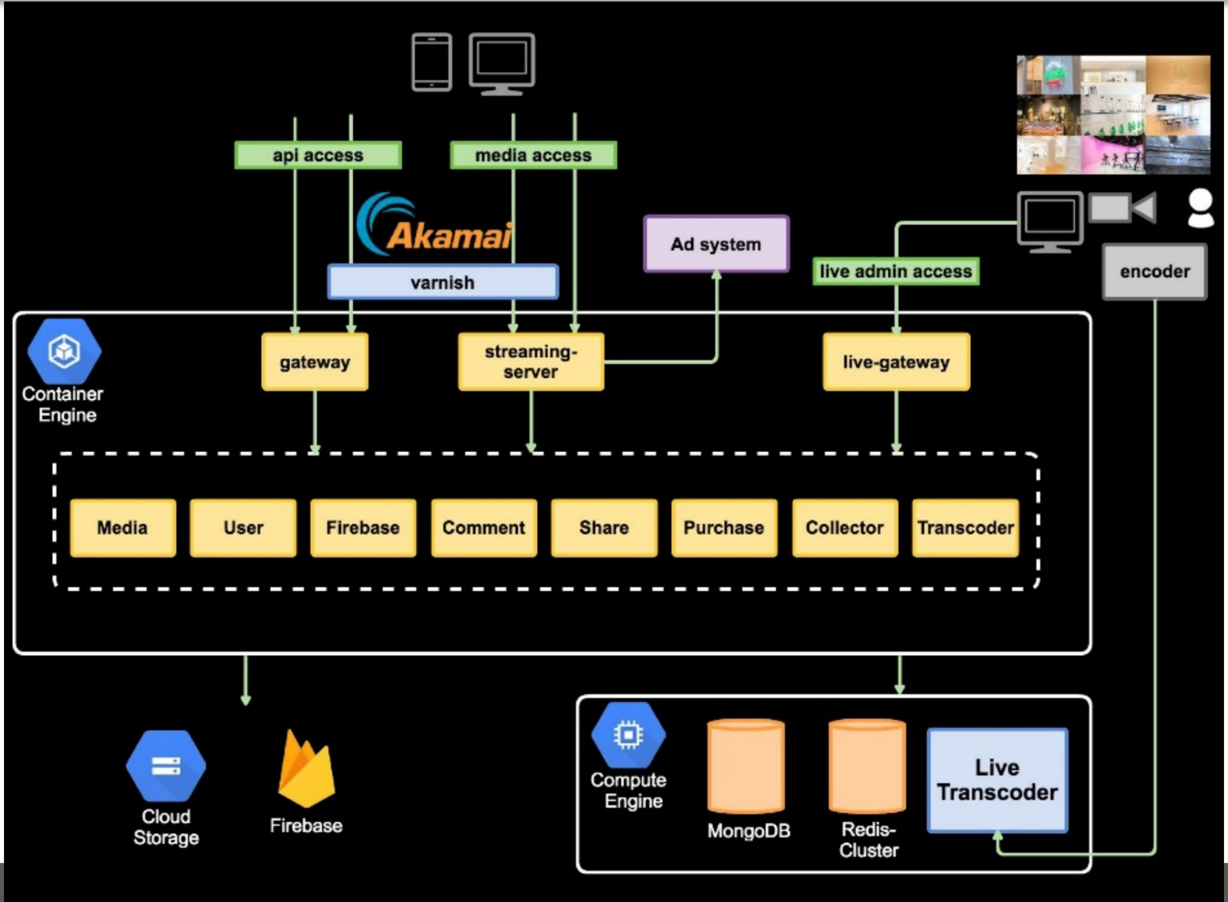
https://microservices.io/patterns/monolithic.html

Microservices architecture



https://microservices.io/patterns/microservices.html

Microservices Architecture -AbemaTV-



https://www.slideshare.net/RyotaNishio/abematvmicroservices-architecture

Cloud Platforms

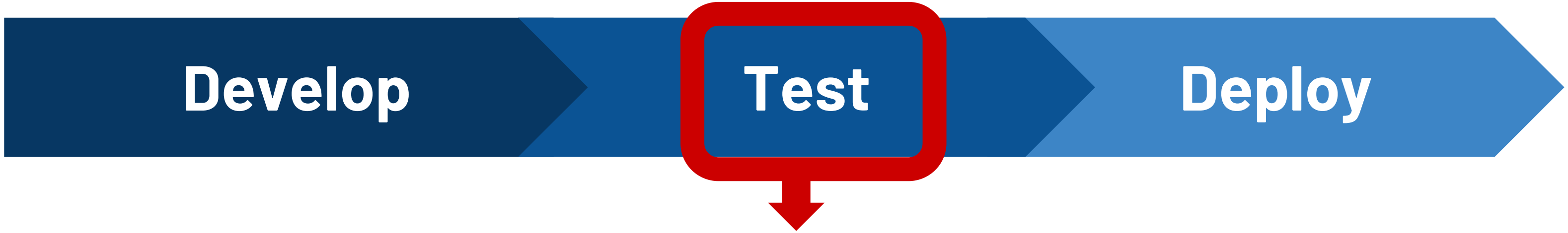
CaaS (Container as a Service)

CaaS provides environments for run containers.

Developers upload containers to CaaS and run them.

□ CI/CD (Continuous Integration/Delivery)

Automatically execution by CI/CD tools.
(Jenkins, CircleCI, Screwdriver ...etc.)



CI tools use Docker containers
with testing environments.

How docker work on Windows

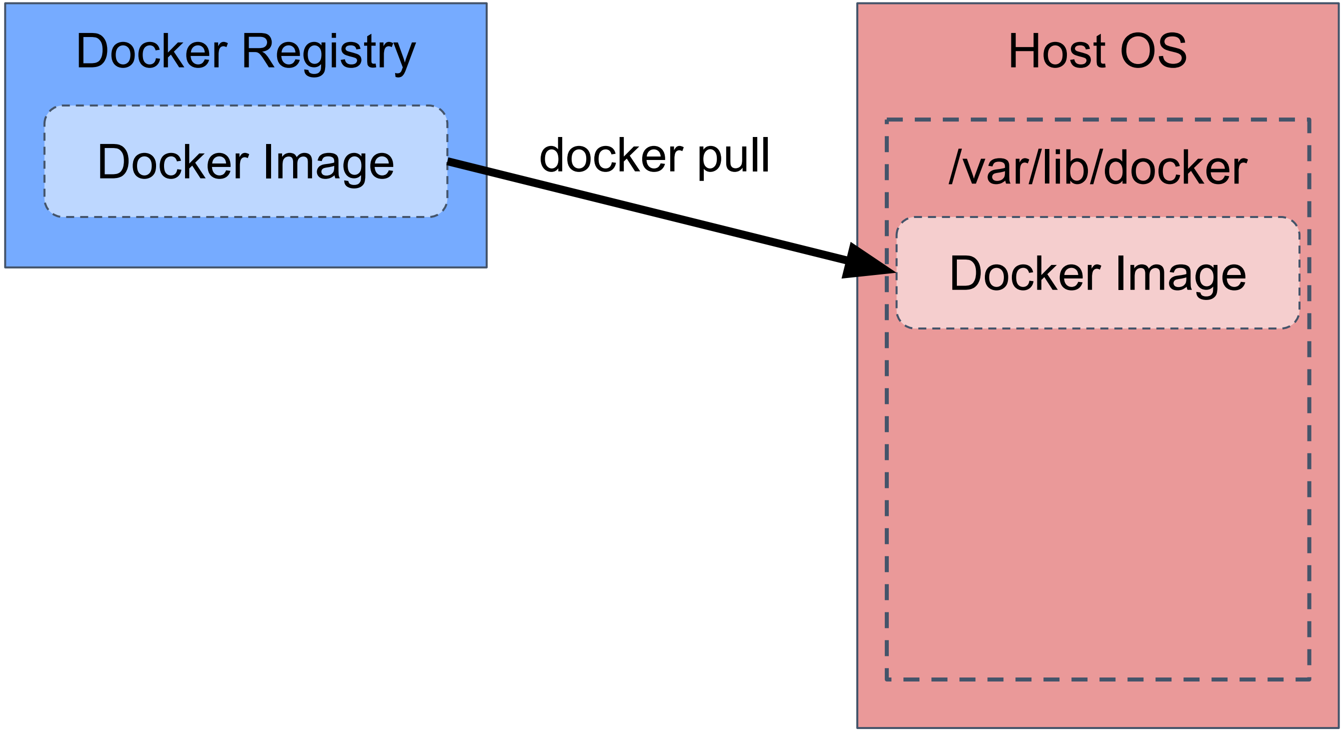
❑ Run Docker on linux VM

❑ Docker client operate Docker engine on linux VM

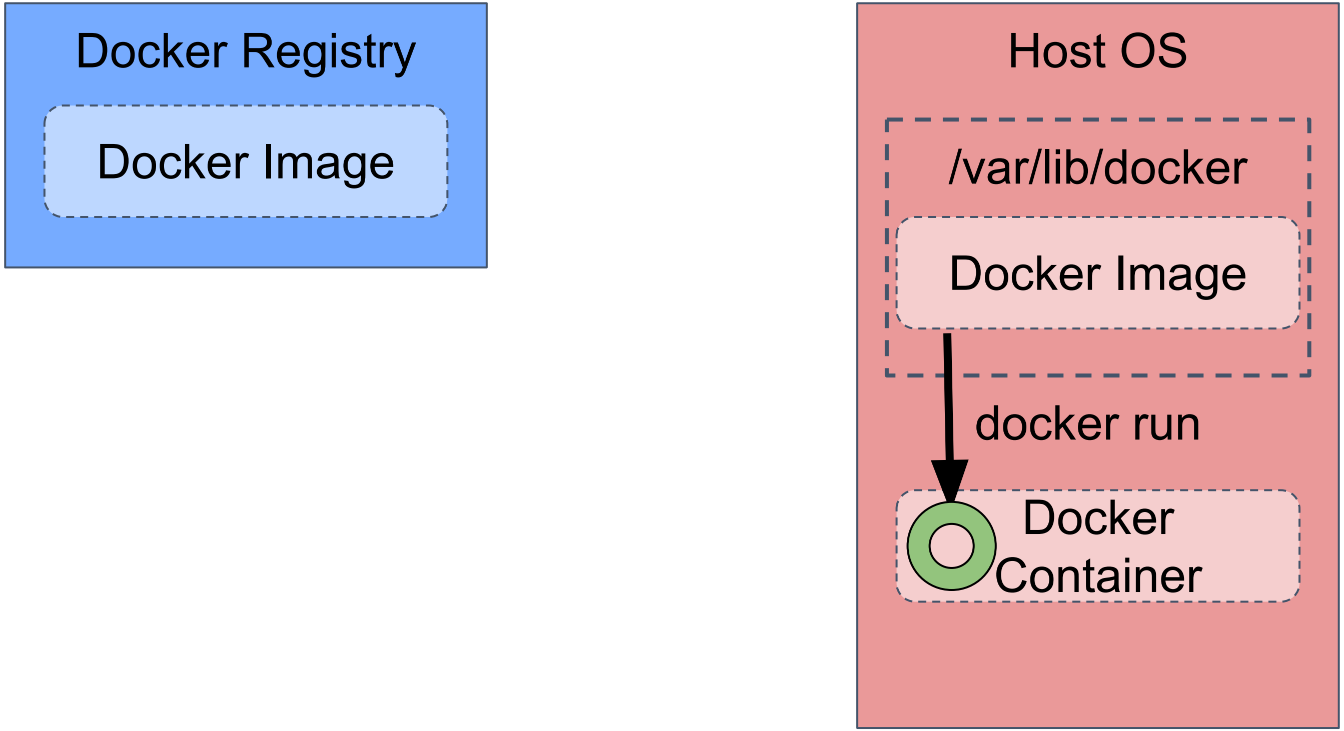
- ❖ Make a linux VM and run on it
- ❖ Docker Toolbox : Use VirtualBox
- ❖ Docker for Windows : Use Hyper-V

3. Docker Lifecycle

Containers Life Cycle of Docker



Containers Life Cycle of Docker



Docker Registry

Docker Image

Host OS

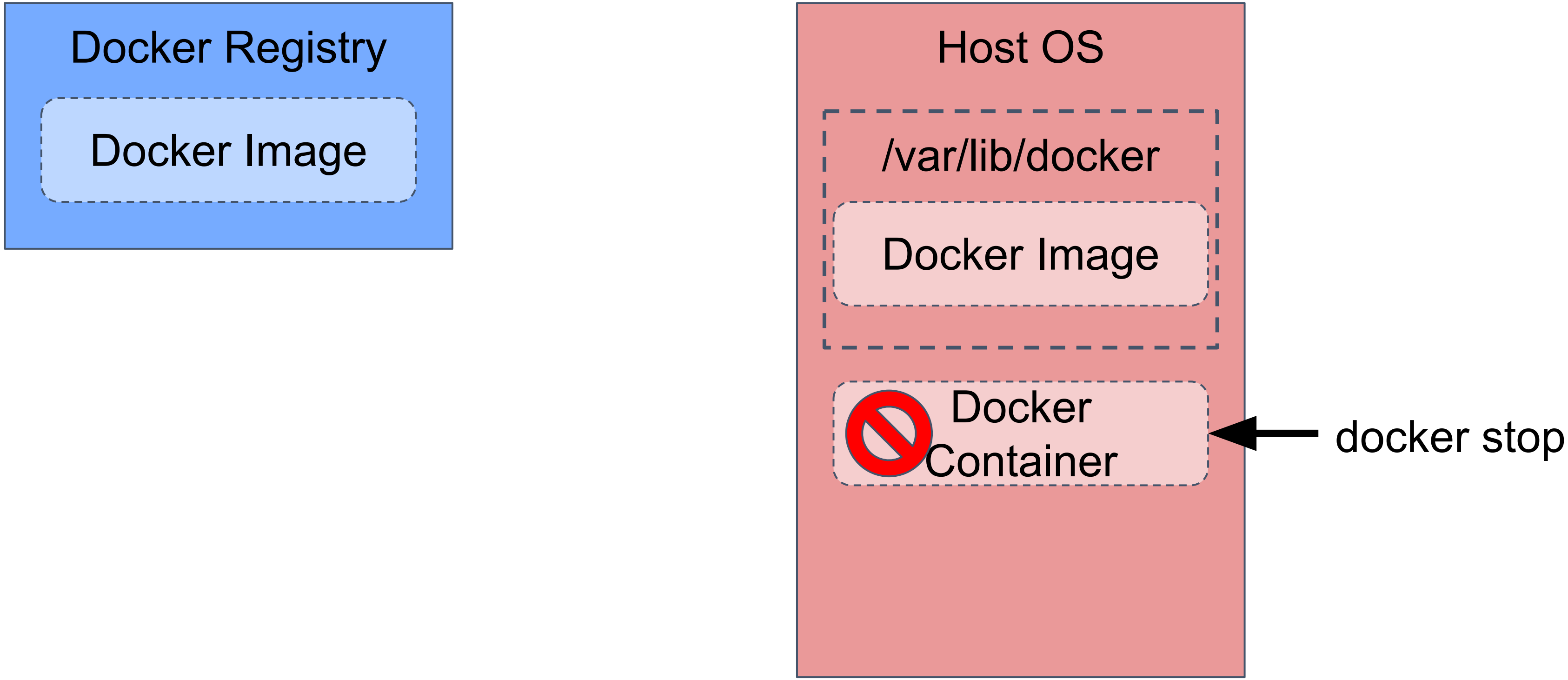
/var/lib/docker

Docker Image

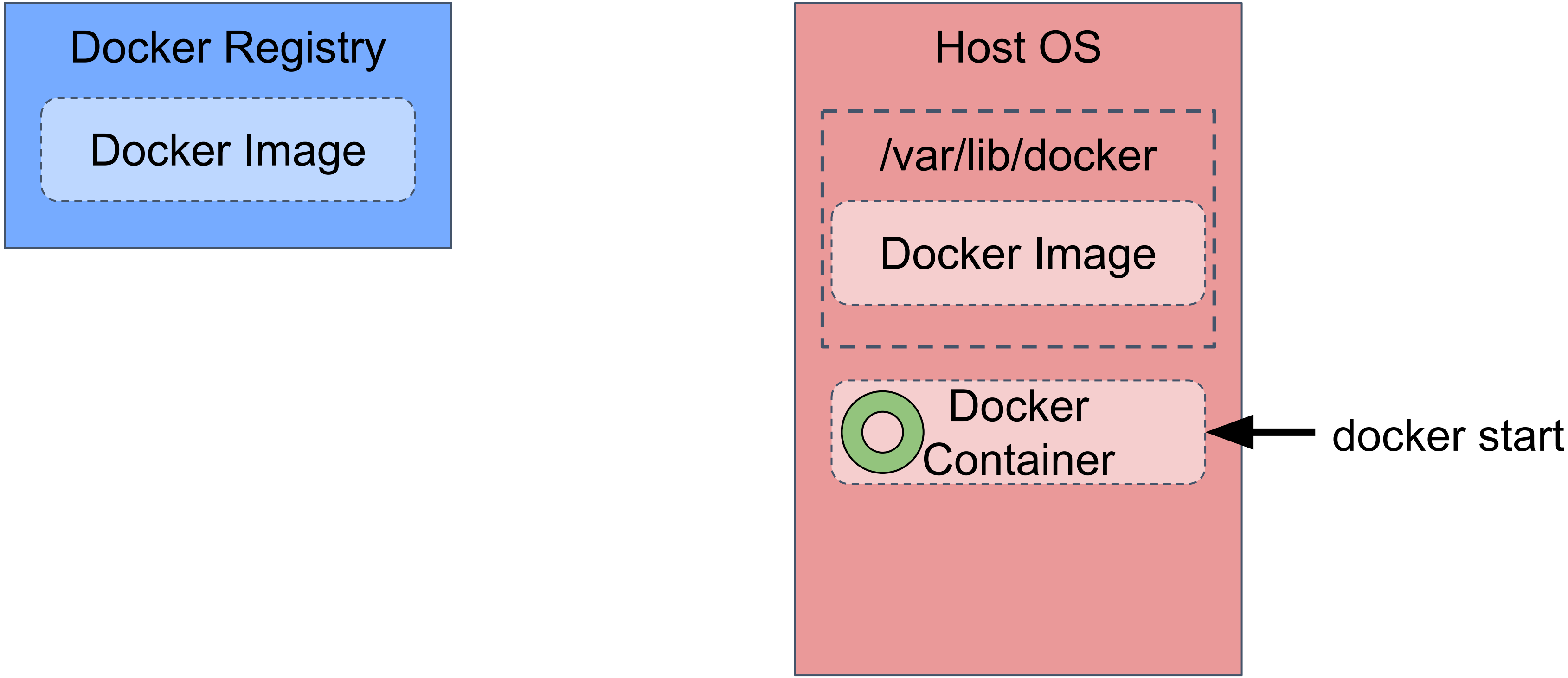
docker run

Docker Container

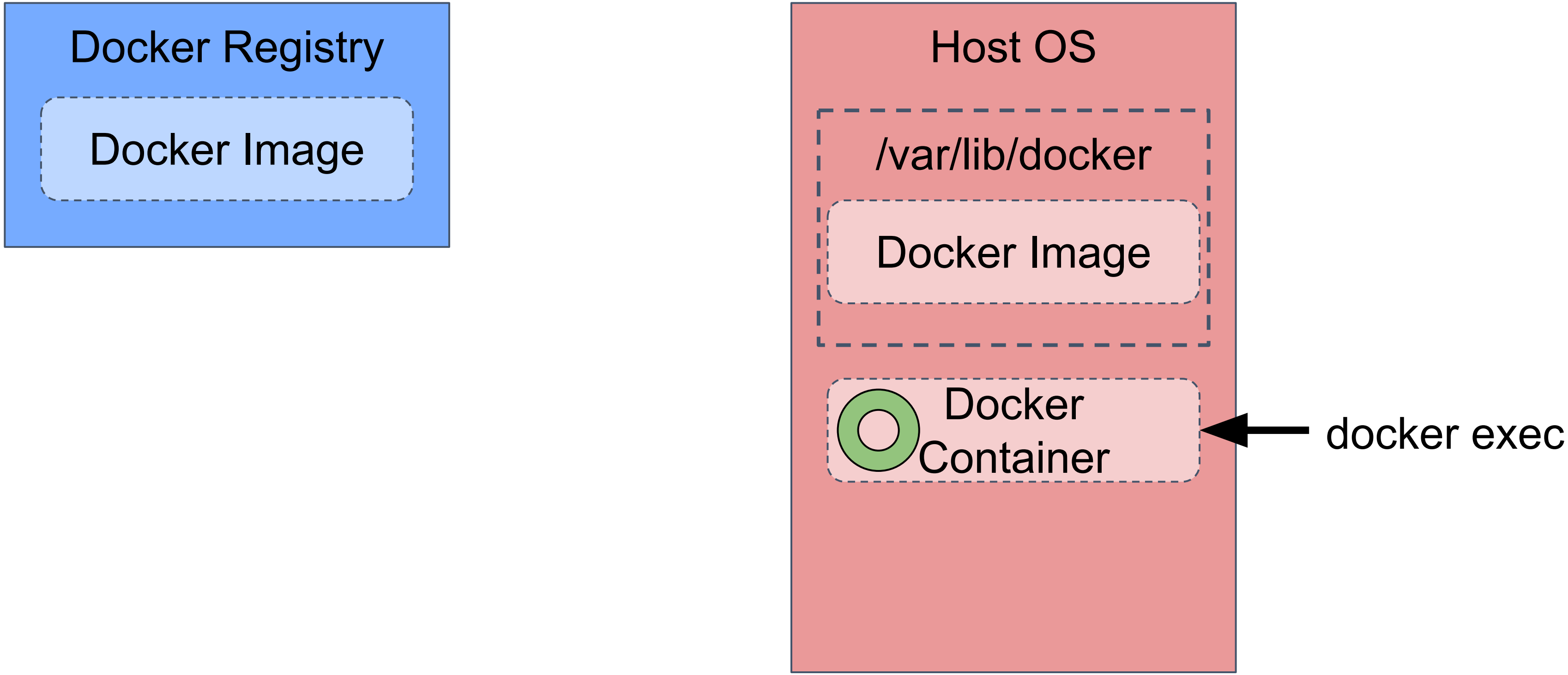
Containers Life Cycle of Docker



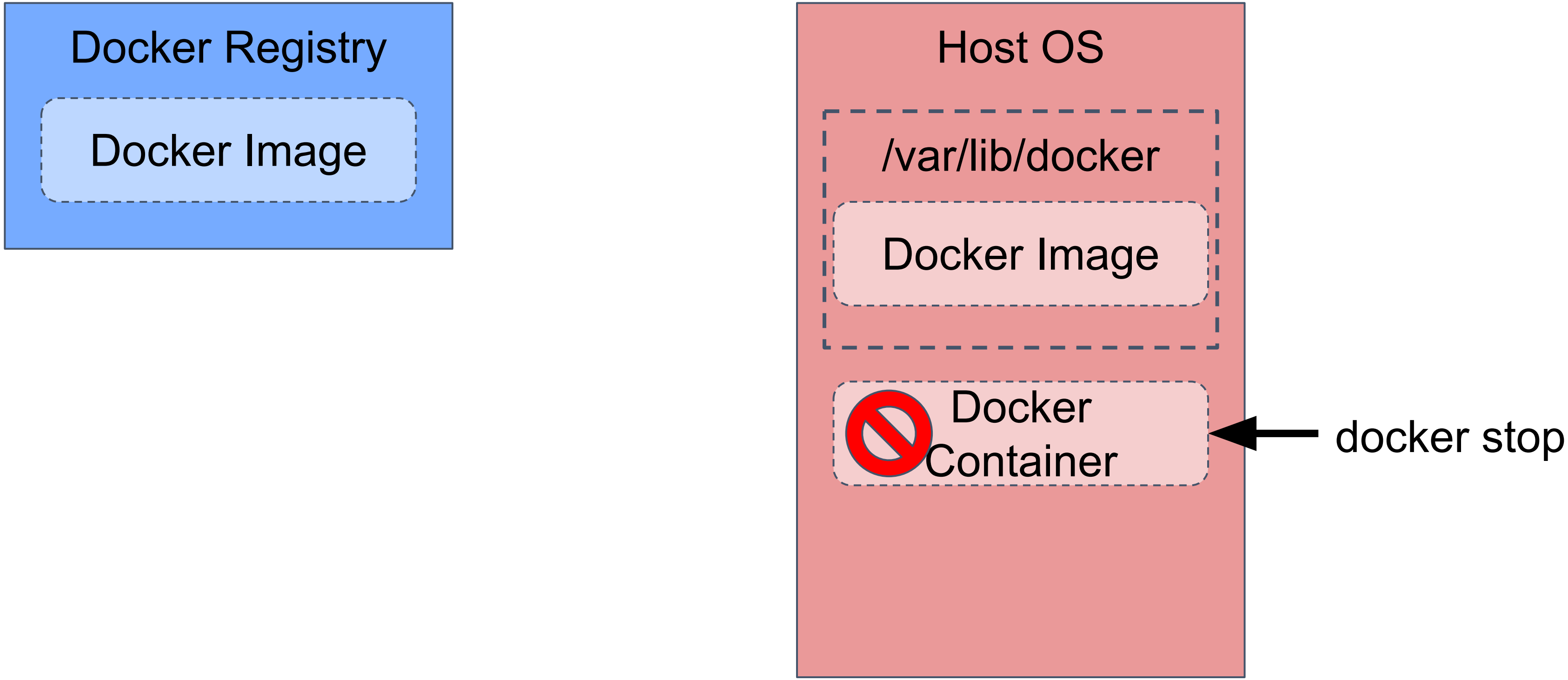
Containers Life Cycle of Docker



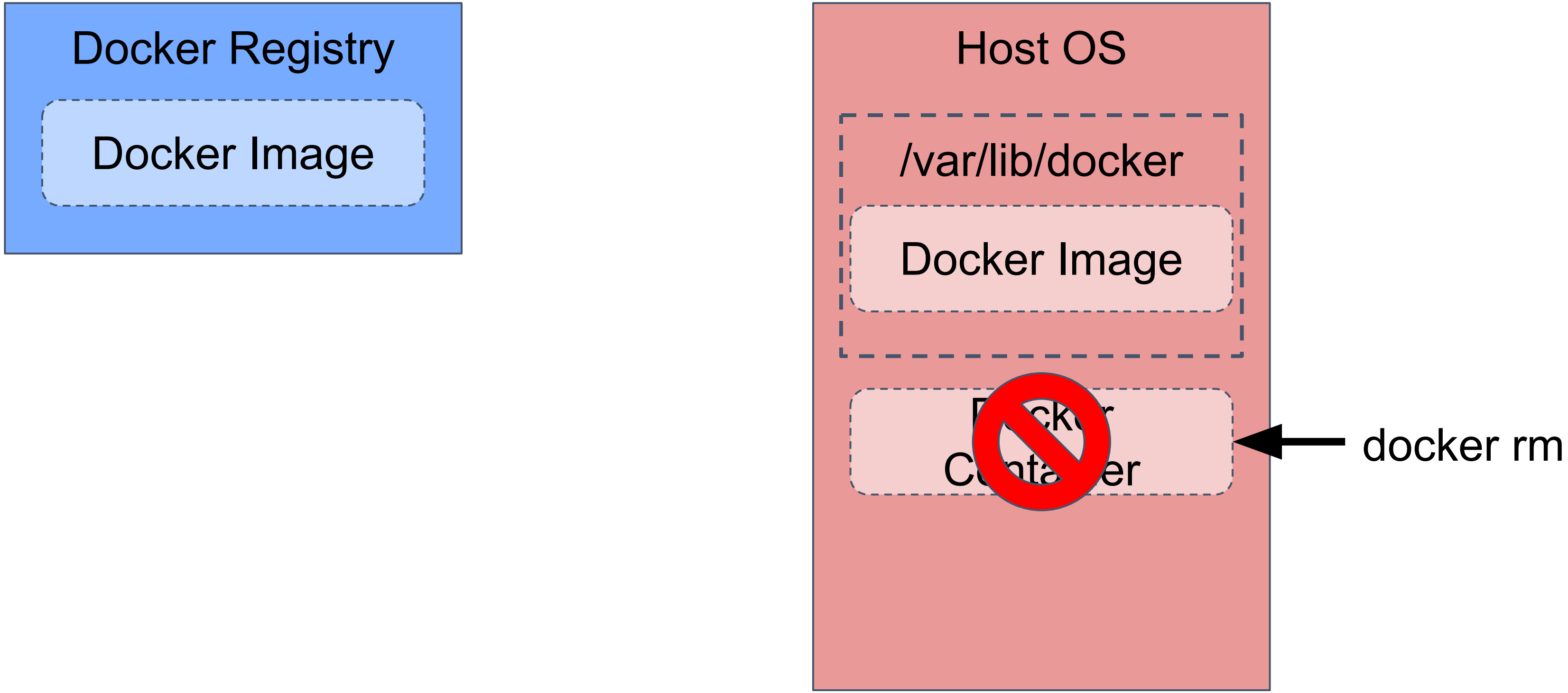
Containers Life Cycle of Docker



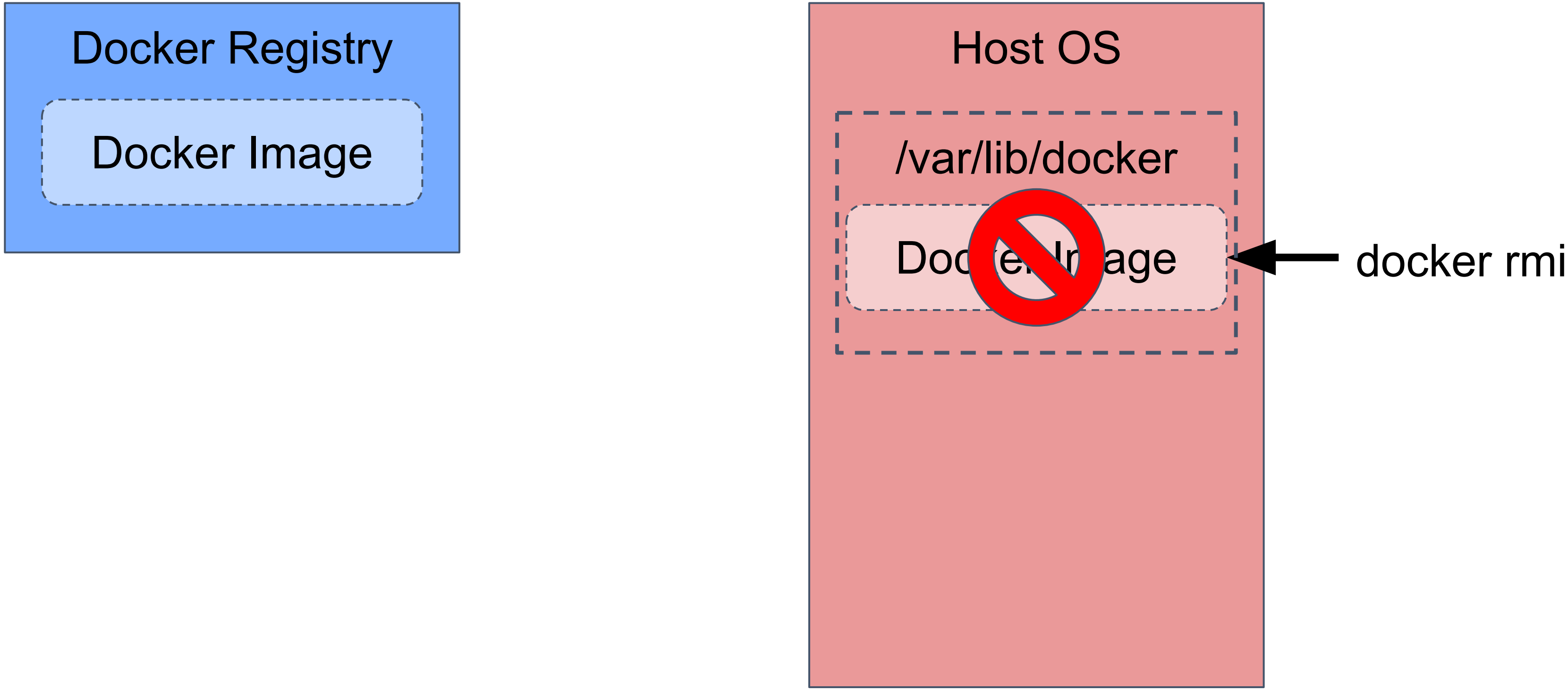
Containers Life Cycle of Docker



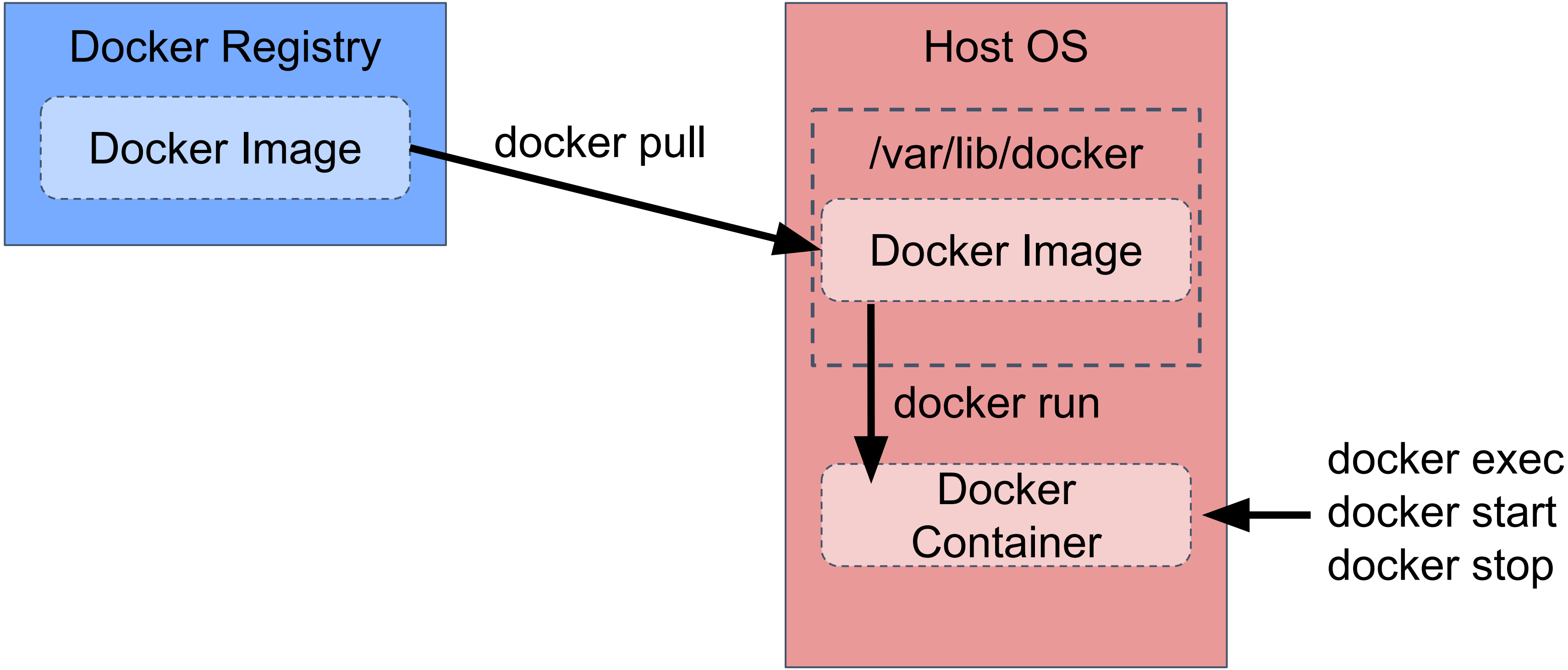
Containers Life Cycle of Docker



Containers Life Cycle of Docker

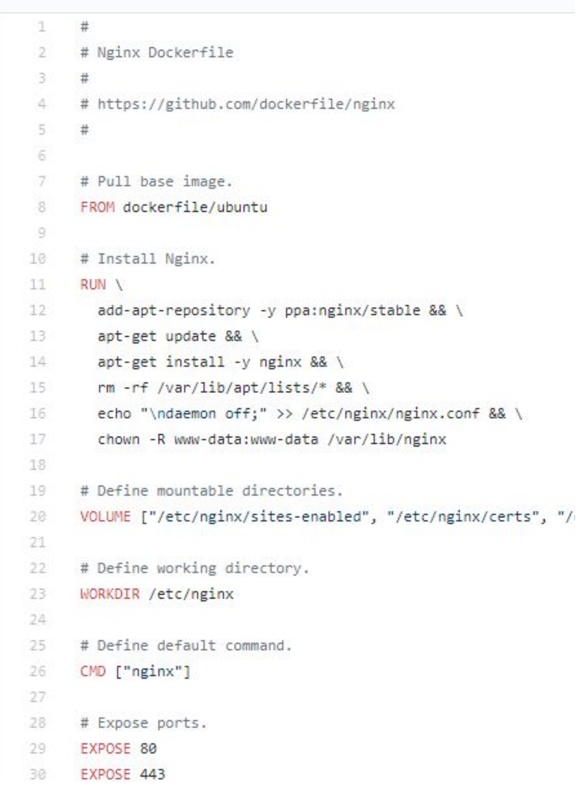


Containers Life Cycle of Docker



Dockerfile

- ❑ Dockerfile is written a procedure of container construction.
- ❑ Dockerfile allows developers to distribute arbitrary images easily.
- ❑ “docker build” builds a image which based on Dockerfile.



```
#
# Nginx Dockerfile
#
# https://github.com/dockerfile/nginx
#

# Pull base image.
FROM dockerfile/ubuntu

# Install Nginx.
RUN \
  add-apt-repository -y ppa:nginx/stable && \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx

# Define mountable directories.
VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/

# Define working directory.
WORKDIR /etc/nginx

# Define default command.
CMD ["nginx"]

# Expose ports.
EXPOSE 80
EXPOSE 443
```

https://github.com/dockerfile/nginx/blob/master/Dockerfile

4. Security Problems of Docker

Malicious Docker Images

- ❏ Should not use docker images,
  if it is not exposed Dockerfile.
- ❏ Some docker images are malicious.

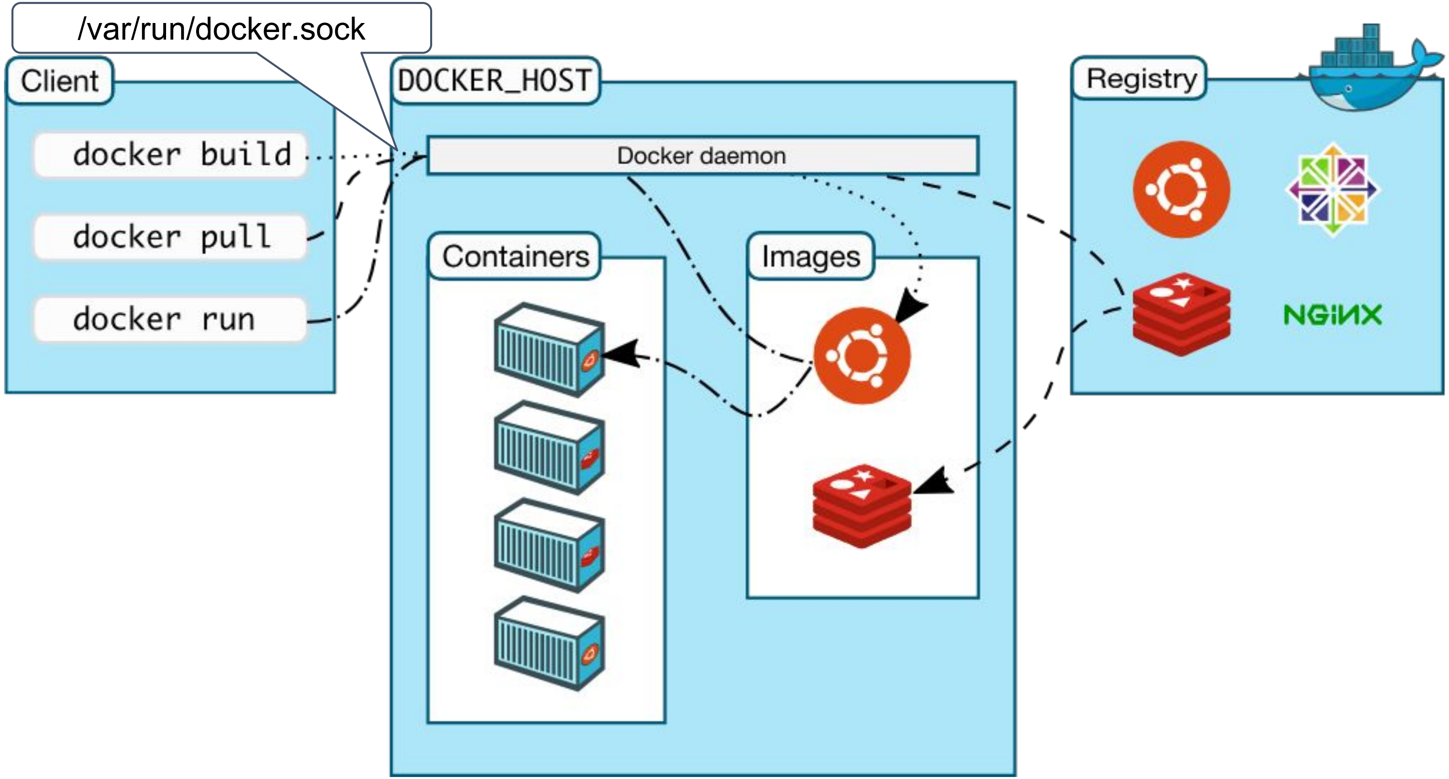
Exploit Arbitrary Code on Host OS

❑ Do not expose /var/run/docker.sock to a container !
Because container can get root privilege
on Host OS.

❑ This attack is allowed by volume function
(Mount directories of Host OS to a container)
and default user of containers is “root”.

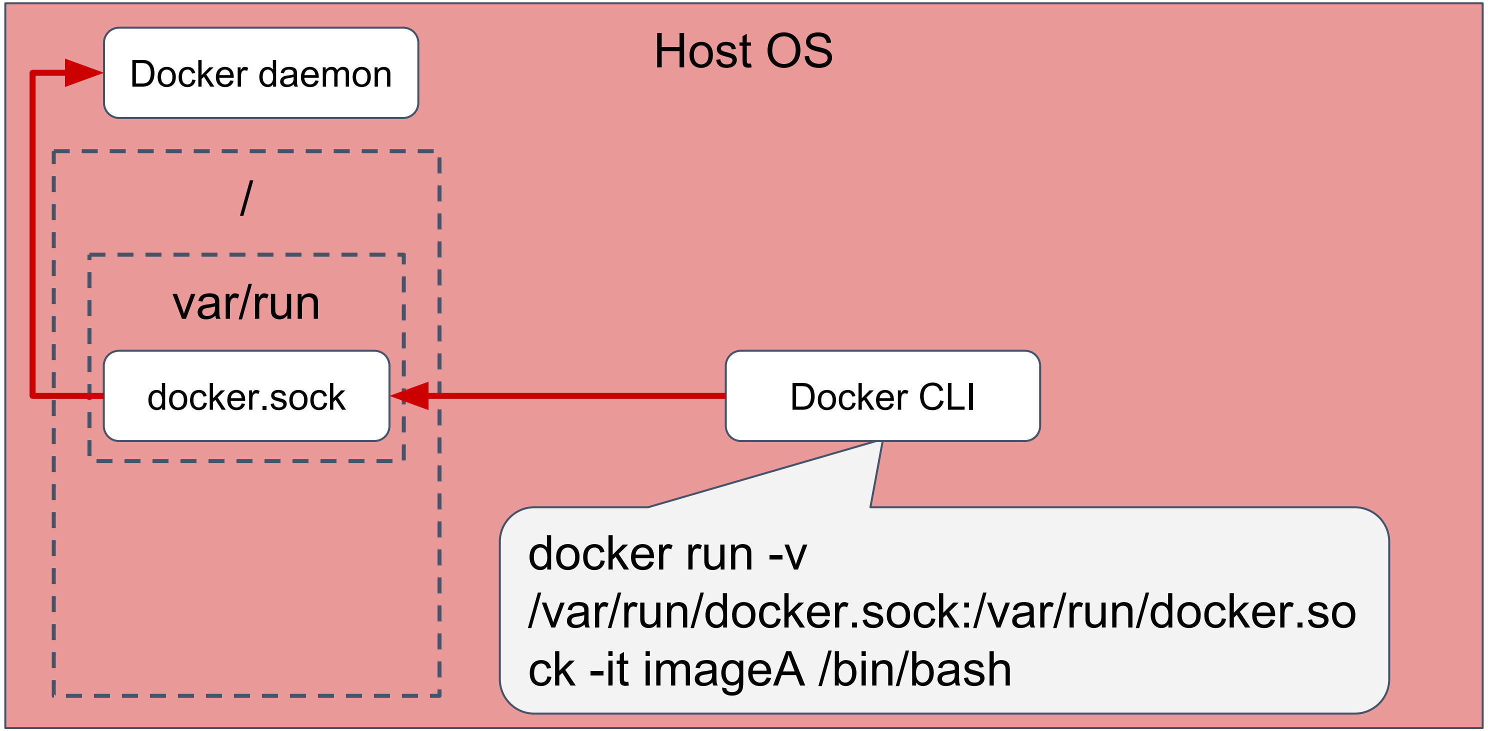
❑ Should change execution user.

Components of Docker



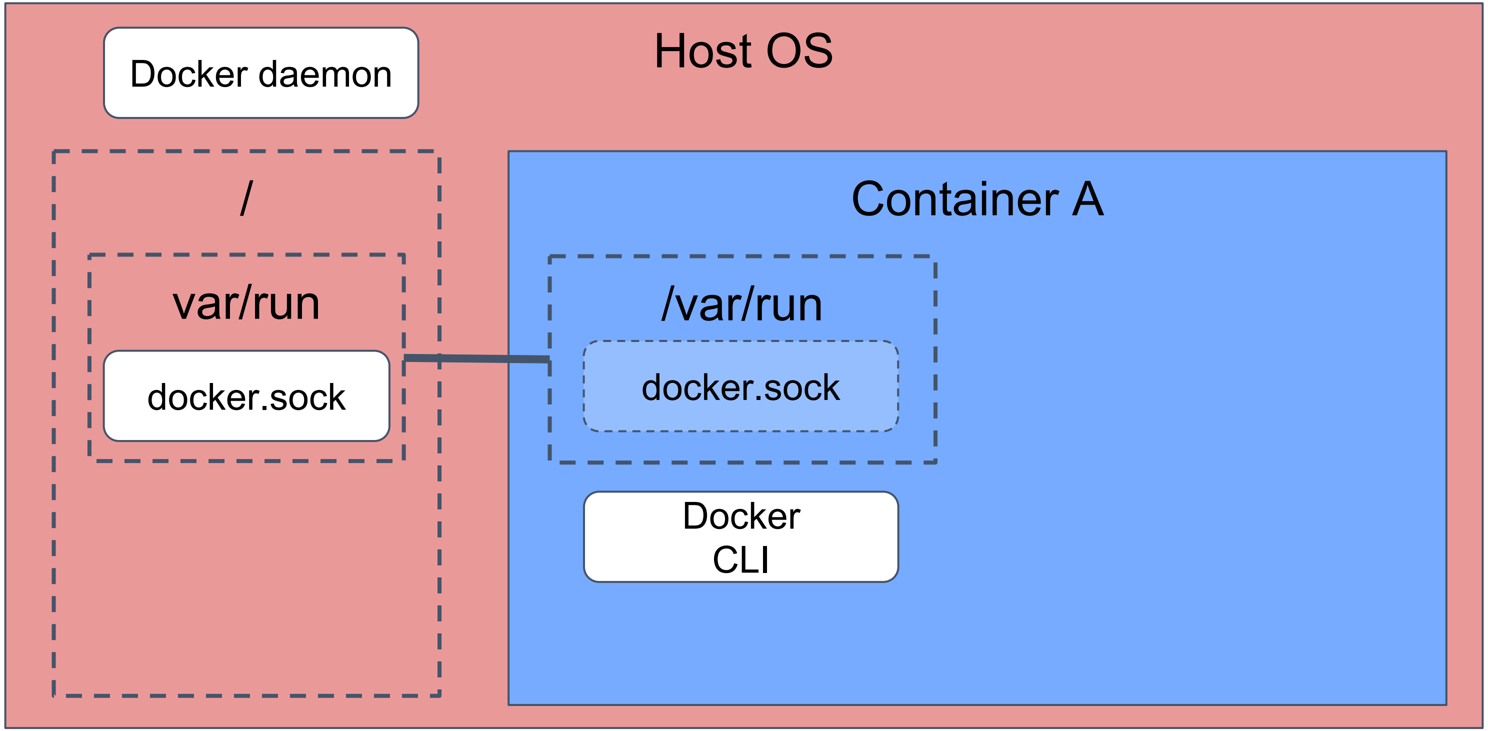
https://docs.docker.com/engine/docker-overview/#docker-architecture

Exploit Arbitrary Code on Host OS

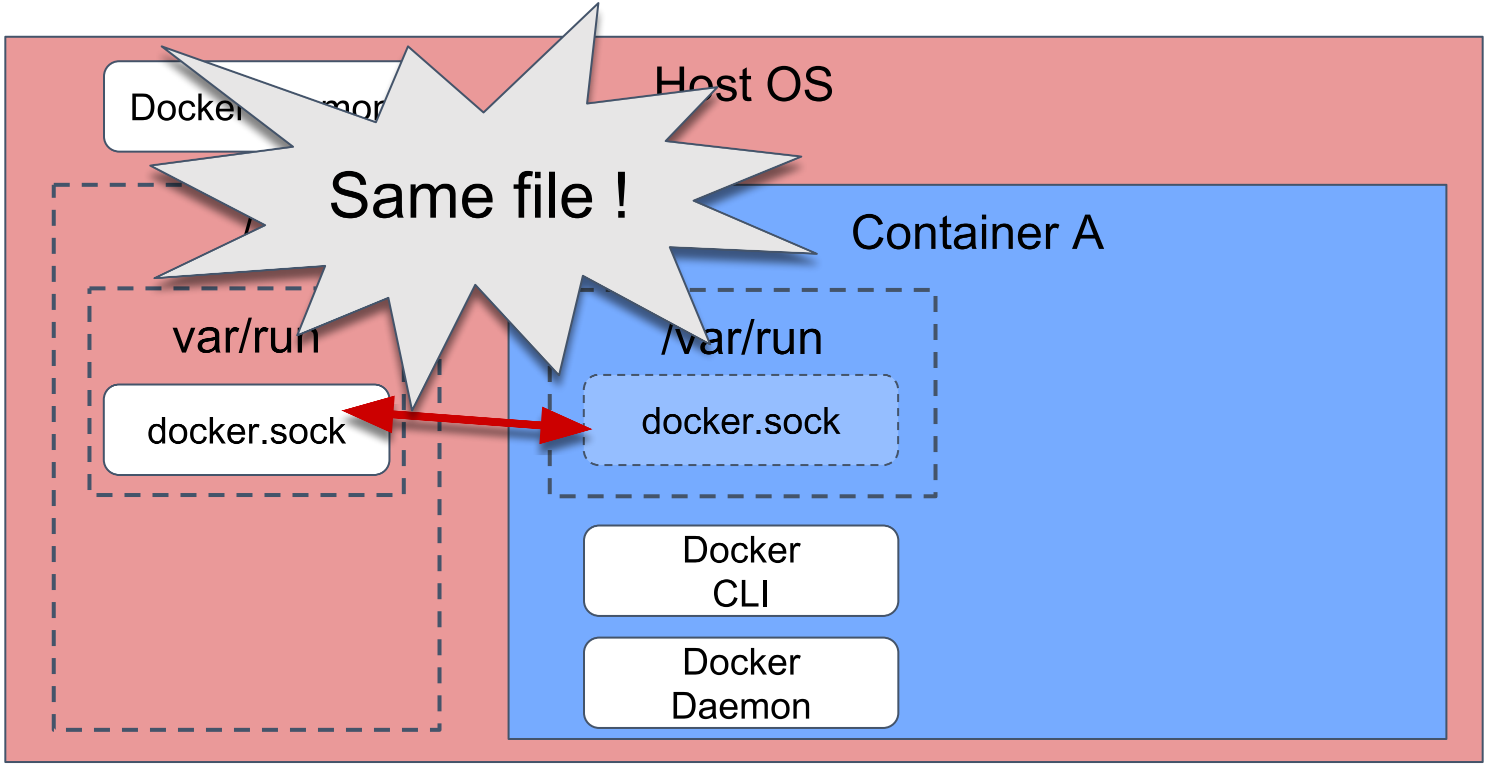


```
docker run -v
/var/run/docker.sock:/var/run/docker.so
ck -it imageA /bin/bash
```

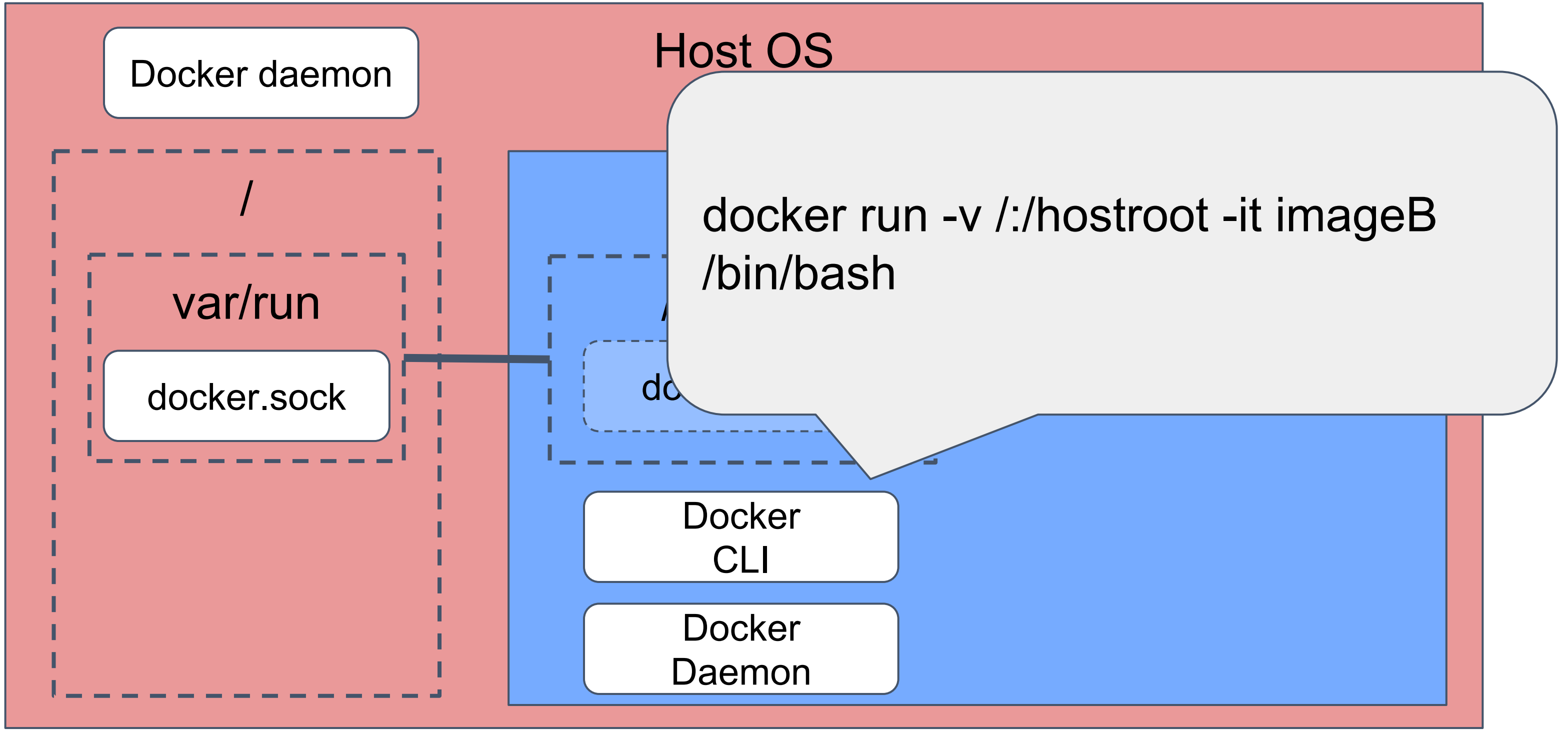
Exploit Arbitrary Code on Host OS



Exploit Arbitrary Code on Host OS

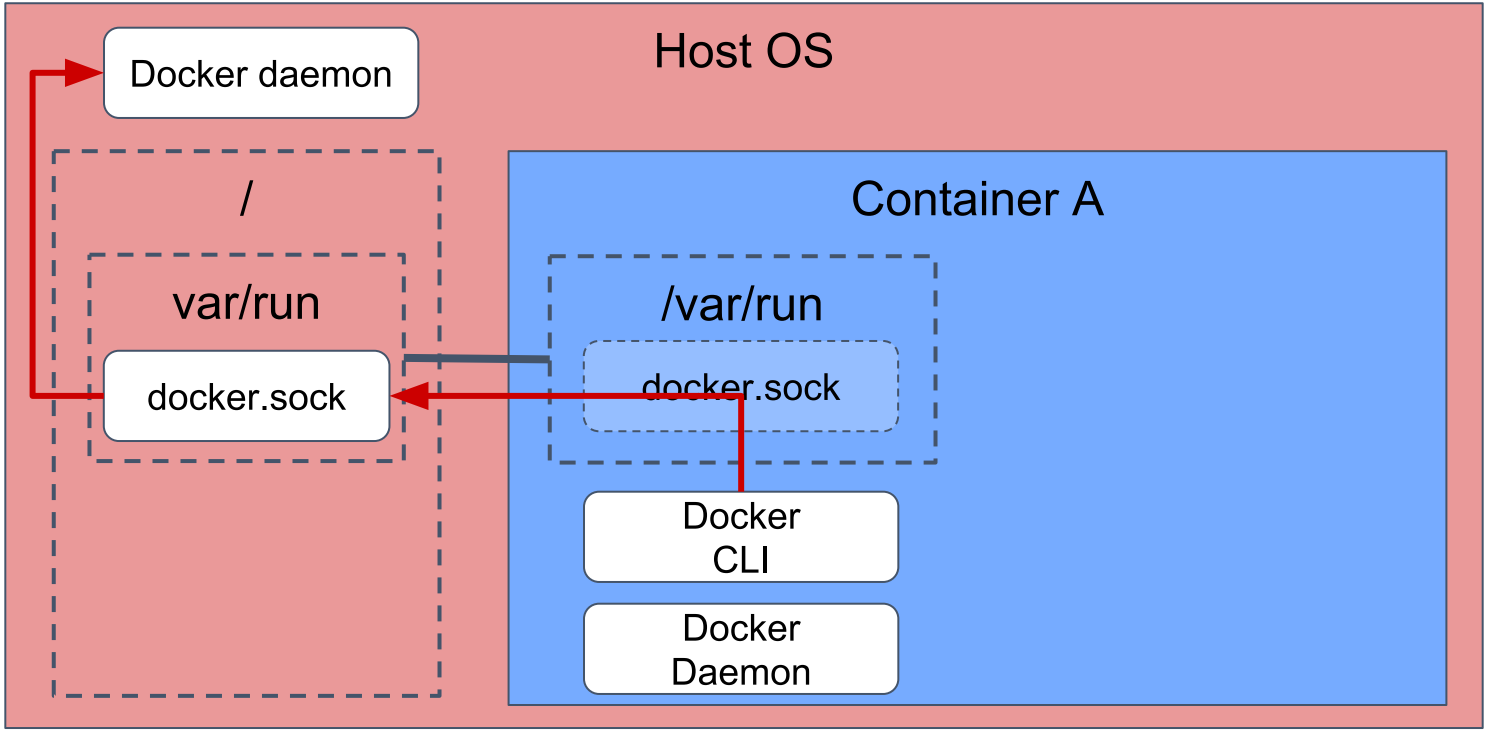


Exploit Arbitrary Code on Host OS

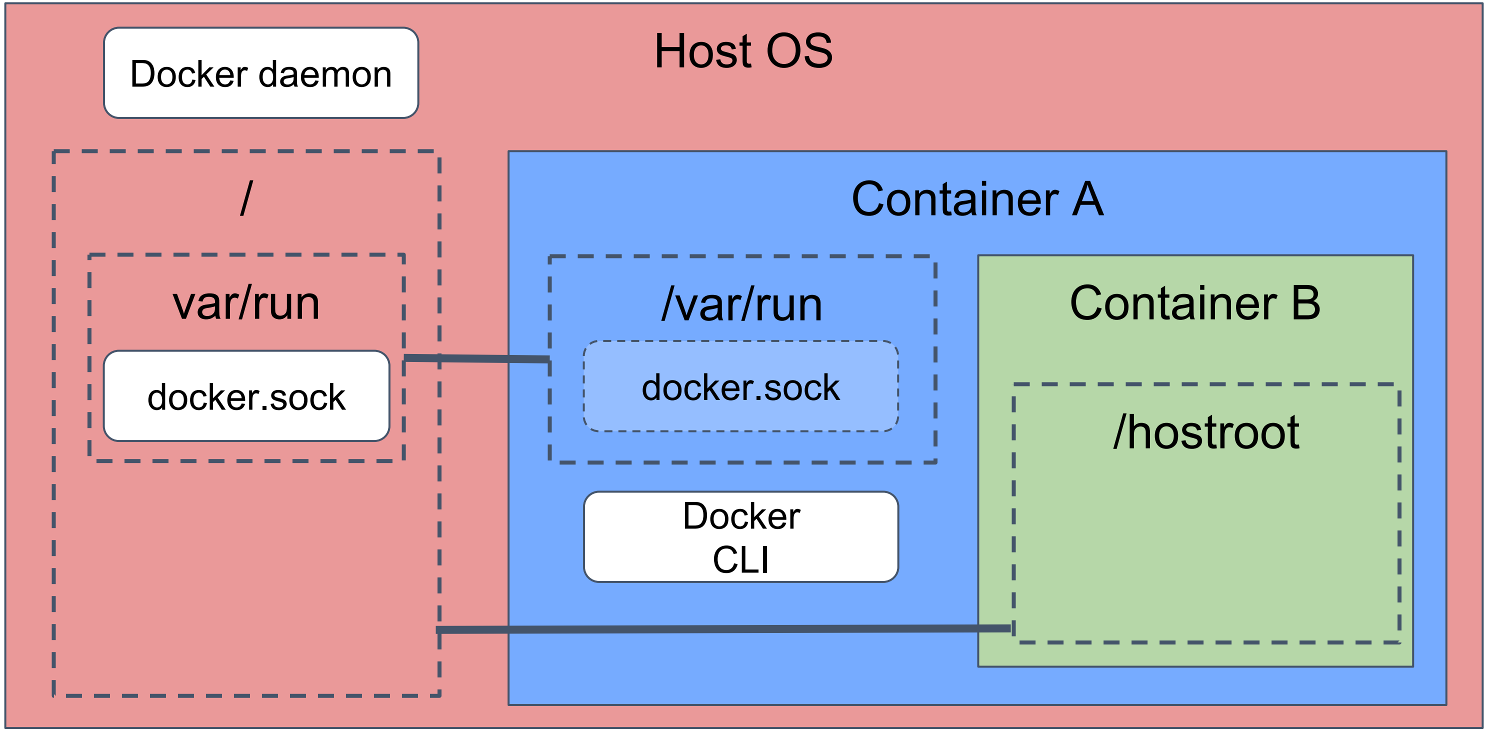


```
docker run -v /:/hostroot -it imageB
/bin/bash
```

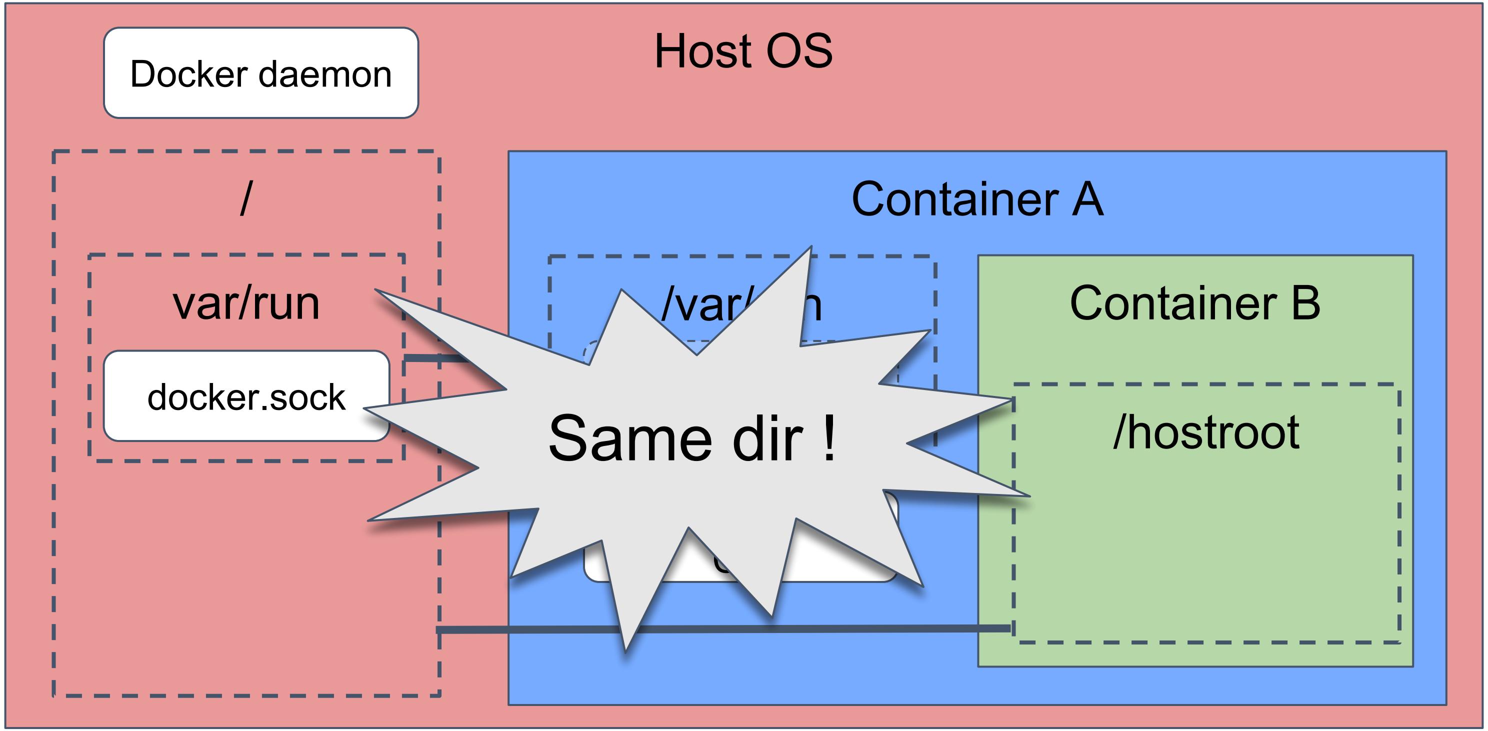
Exploit Arbitrary Code on Host OS



Exploit Arbitrary Code on Host OS



Exploit Arbitrary Code on Host OS



Exploit Arbitrary Code on Host OS

- Root directory of Host OS is mounted to Container B’s /hostroot.

- Container B uses “root” user.

5. Advanced Technology

Container-based Orchestration

❑ Kubernetes

- Provision components of a service.
- Auto scheduling.
- Auto recovery.

❑ Istio

IaC (Infrastructure as Code)

❑ Ansible

❑ Packer

❑ Terrafrom

❑ Docker

❑ Kubernetes

❑ Istio

QUIZ

Quiz

Q1. Docker can change kernel parameters.

True / False

Q2. Docker only provides container management function.

True / False

Q3. Choose all OS which can run Docker directly.

1. Arch Linux, 2. Debian, 3. openSUSE, 4. RHEL
5. Windows 10, 6. Window Server 2018,

Quiz

Q4. Which driver does Docker usually use. (Docker >= 0.9)

1. Libcontainer, 2. Libvirt, 3. LXC

Q5. Docker CLI can directly operate Docker container.

True / False

Q6. Usually cost of vertical scaling is cheaper than horizontal scaling.

True / False

Quiz

Q7. All Docker images on Docker registry are safety.

True / False

Q8. Which file or directory should not mount to Docker container for security reason.

1. /dev/null, 2. /admin, 3. /var/run/docker.sock

Quiz

Q9. Which API is used for isolate filesystem from another containers?

1. cgroup, 2. chroot, 3. cowsay, 4. docker, 5. Lsmod

Q10. Choose all which can operate with libvirt.

1. ESXi, 2. KVM, 3. LXC, 4. Xen

Any question?