KHOA CÔNG NGHỆ THÔNG TIN-ĐHKHTN
CSC11004 - MẠNG MÁY TÍNH NÂNG CAO

VIRTUALIZATION & CONTAINER

Lê Ngọc Sơn

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

VIRTUALIZATION

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

What is Virtualization ?

vir•tu•al (adj): existing in essence or effect, though not in actual fact

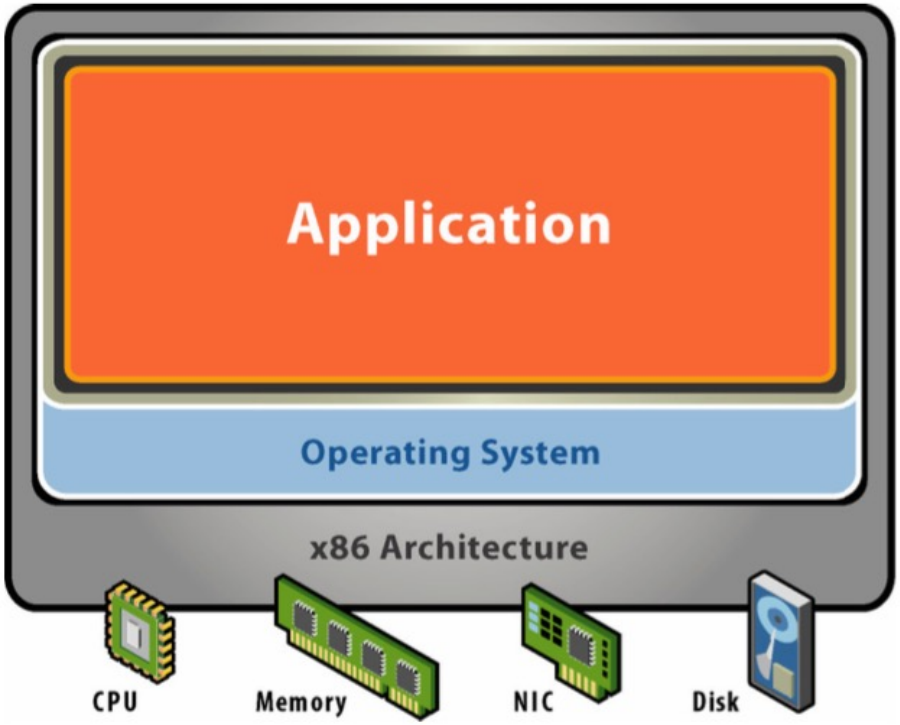
Virtual systems

- Abstract physical components using logical objects
- Dynamically bind logical objects to physical configurations

Examples

- Network – Virtual LAN (VLAN), Virtual Private Network (VPN)
- Storage – Storage Area Network (SAN), LUN
- Computer – Virtual Machine (VM), simulator

Starting Point: A Physical Machine



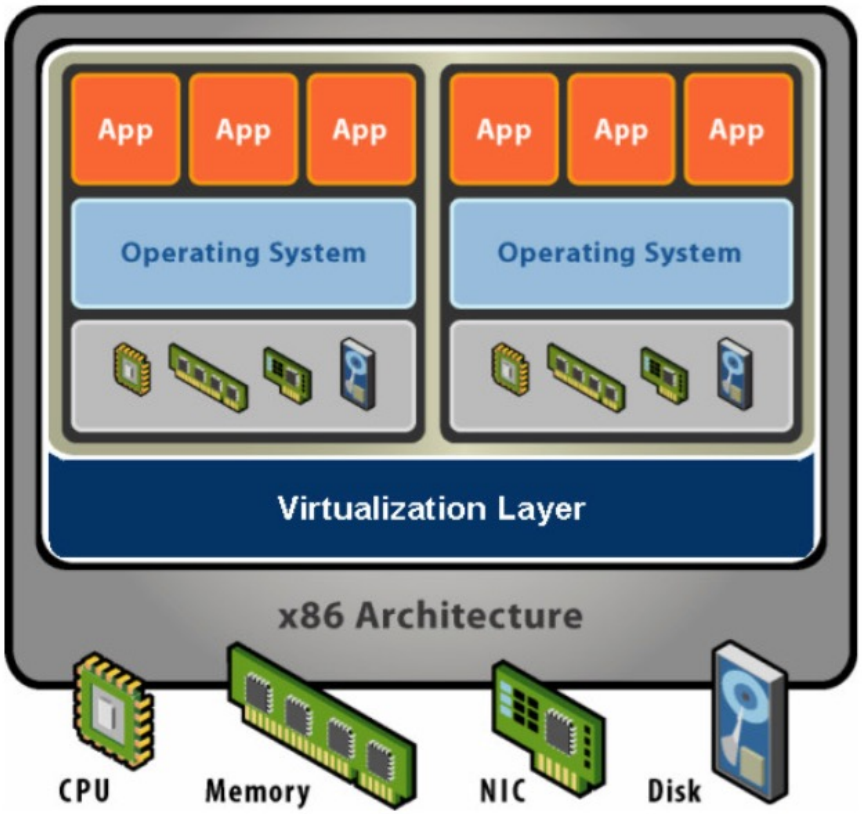
Physical Hardware

- Processors, memory, chipset, I/O bus and devices, etc.
- Physical resources often underutilized

Software

- Tightly coupled to hardware
- Single active OS image
- OS controls hardware

What is a Virtual Machine ?



Hardware-Level Abstraction

- Virtual hardware: processors, memory, chipset, I/O devices, etc.
- Encapsulates all OS and application state

Virtualization Software

- Extra level of indirection decouples hardware and OS
- Multiplexes physical hardware across multiple “guest” VMs
- Strong isolation between VMs
- Manages physical resources, improves utilization

VM Isolation

Secure Multiplexing

- Run multiple VMs on single physical host
- Processor hardware isolates VMs, *e.g.* MMU

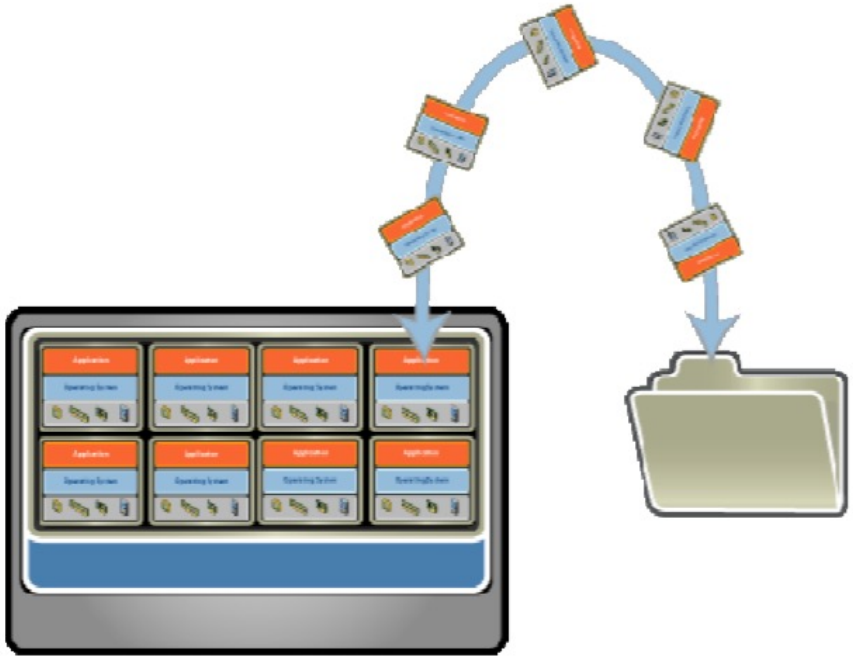
Strong Guarantees

- Software bugs, crashes, viruses within one VM cannot affect other VMs

Performance Isolation

- Partition system resources
- Example: VMware controls for reservation, limit, shares

VM Encapsulation



Entire VM is a File

- OS, applications, data
- Memory and device state

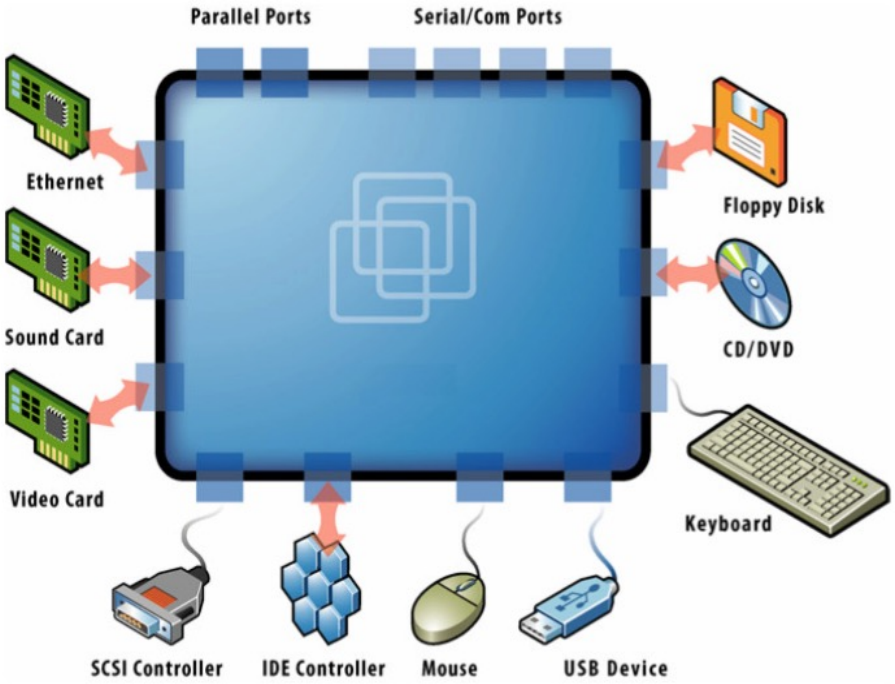
Snapshots and Clones

- Capture VM state on the fly and restore to point-in-time
- Rapid system provisioning, backup, remote mirroring

Easy Content Distribution

- Pre-configured apps, demos
- Virtual appliances

VM Compatibility



Hardware-Independent

- Physical hardware hidden by virtualization layer
- Standard virtual hardware exposed to VM

Create Once, Run Anywhere

- No configuration issues
- Migrate VMs between hosts

Legacy VMs

- Run ancient OS on new platform
- *E.g.* DOS VM drives virtual IDE and vLance devices, mapped to modern SAN and GigE hardware

Virtualization Comes in many forms

Virtual Memory Each application sees its own logical **memory,** independent of physical memory

Virtual Networks Each application sees its own logical **network,** independent of physical network

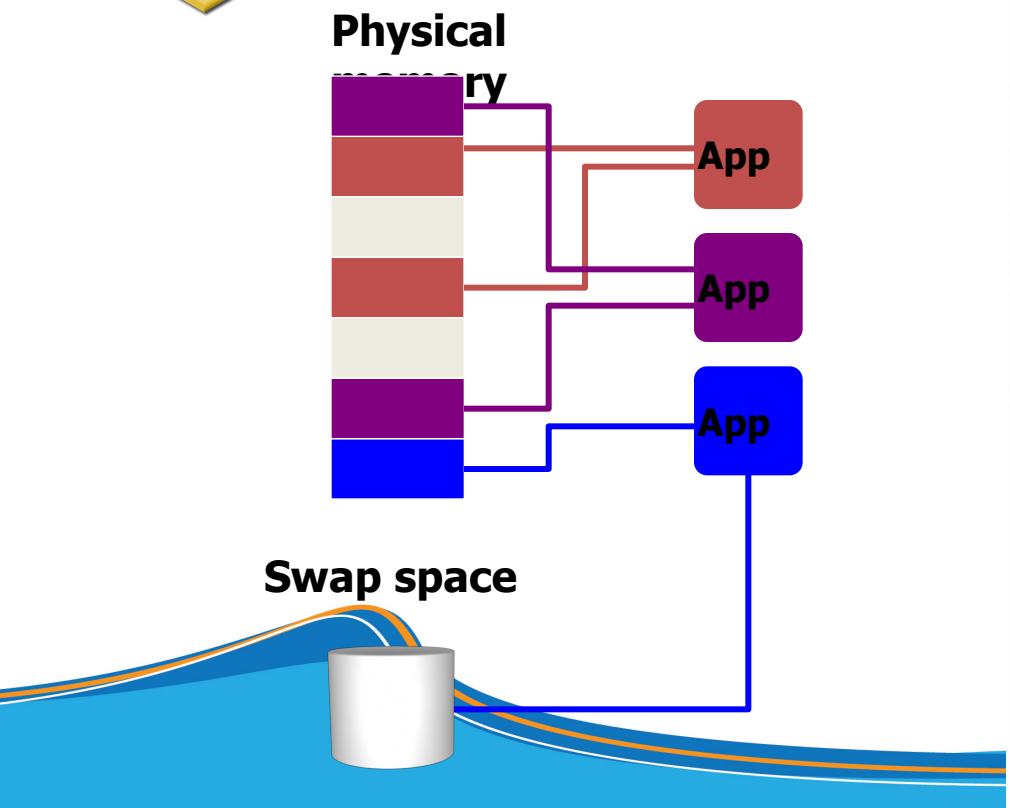
Virtual Servers Each application sees its own logical **server,** independent of physical servers

Virtual Storage Each application sees its own logical **storage,** independent of physical storage

Memory Virtualization

Virtual Memory

Each application sees its own logical **memory,** independent of physical memory



Benefits of Virtual Memory

- Remove physical-memory limits
- Run multiple applications at once

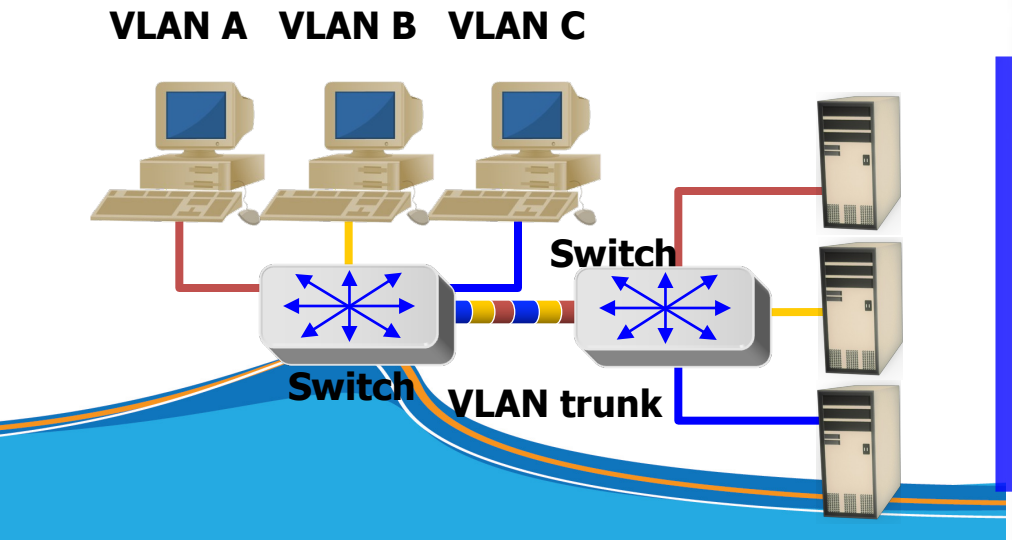
cdio 4.0

fit@hcmus

Network Virtualization

Virtual Networks

Each application sees its own logical **network,** independent of physical network

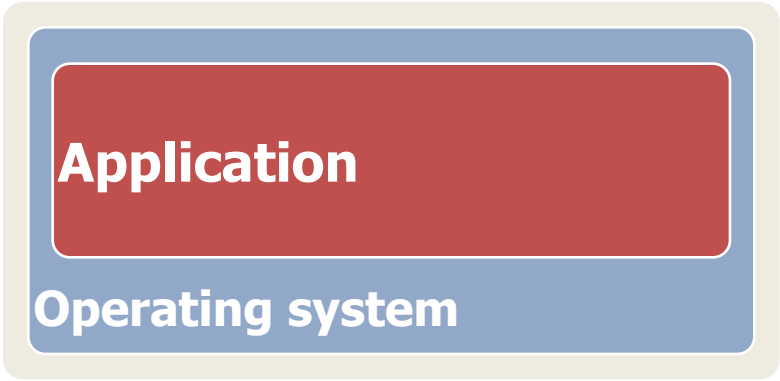


Benefits of Virtual Networks

- Common network links with access-control properties of separate links
- Manage logical networks instead of physical networks
- **Virtual SANs** provide similar benefits for storage-area networks

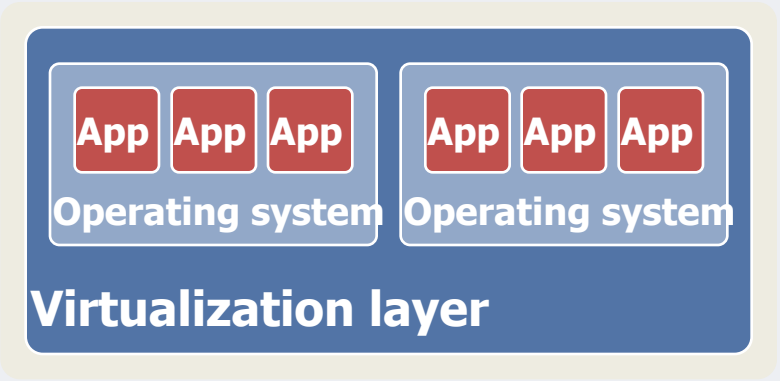
Server Virtualization

Before Server Virtualization:



- Single operating system image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources

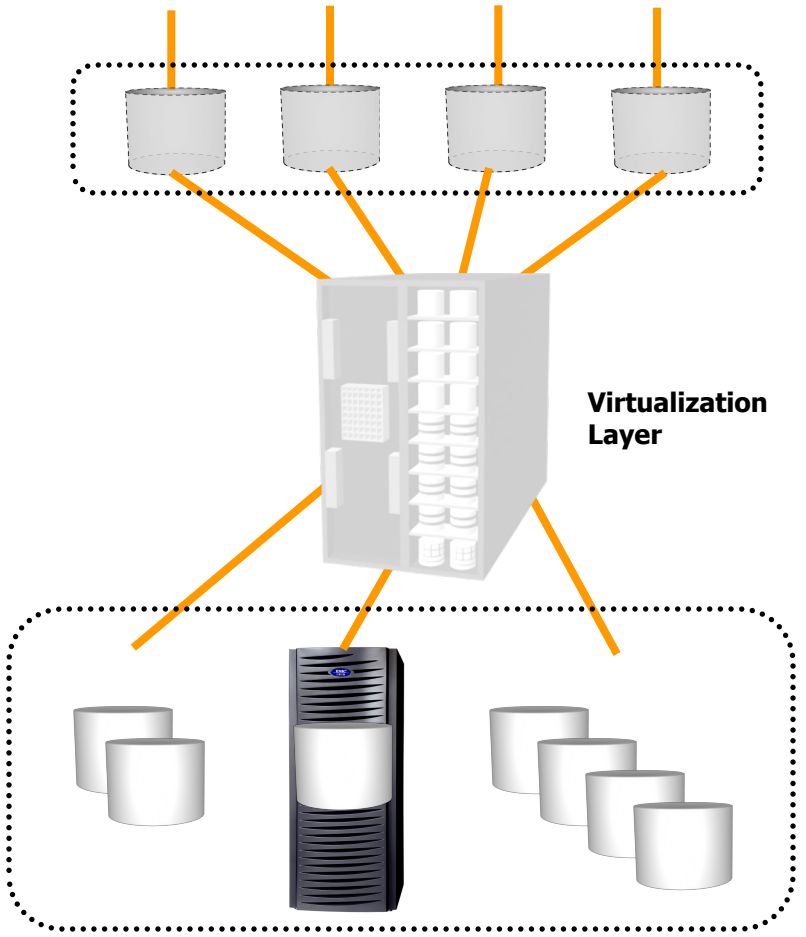
After Server Virtualization:



- Virtual Machines (VMs) break dependencies between operating system and hardware
- Manage operating system and application as single unit by encapsulating them into VMs
- Strong fault and security isolation
- Hardware-independent

Storage Virtualization

- Process of presenting a logical view of physical storage resources to hosts
- Logical storage appears and behaves as physical storage directly connected to host
- Examples of storage virtualization are:
  - Host-based volume management
  - LUN creation
  - Tape virtualization
- Benefits of storage virtualization:
  - Increased storage utilization
  - Adding or deleting storage without affecting application's availability
  - Non-disruptive data migration



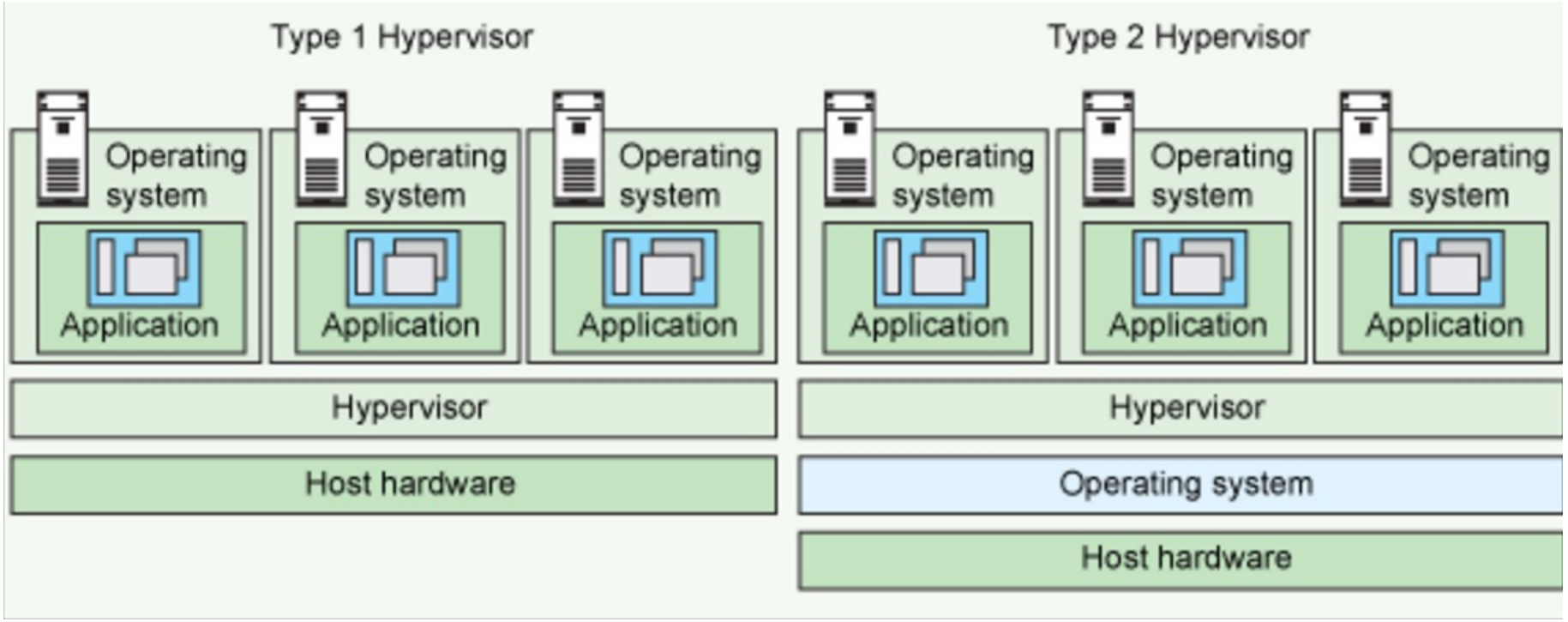
Desktop Virtualization

- Virtual Desktop Infrastructure (VDI) is a desktop delivery model which allows client desktop workloads (operating system, application, user data) to be hosted and executed on servers in the data center
- Users can communicate with their virtual desktops through a client device that supports remote desktop protocols such as RDP
- This allows you to virtualize Windows desktops in the datacenter and deliver them on demand to any user — anywhere

Hypervisor

- A **hypervisor** or **virtual machine monitor** (**VMM**) is a piece of computer software, firmware or hardware that creates and runs virtual machines.
- Two major types:
  - Type-I
  - Type-II

Hypervisor

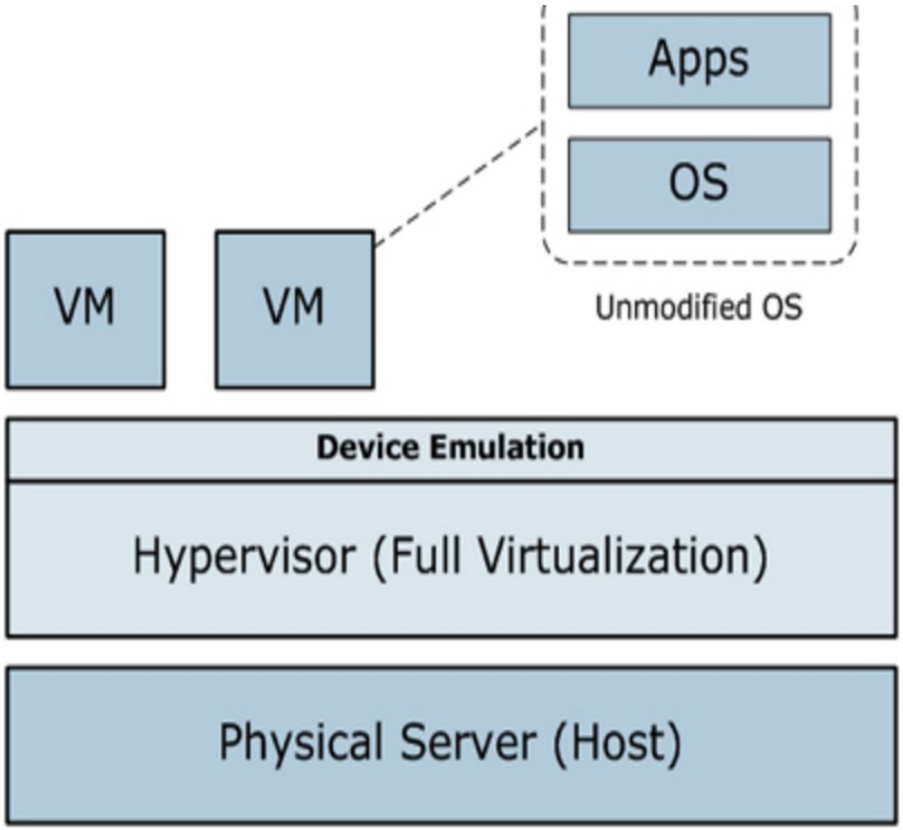

Type 1 Hypervisor
Operating system
Application
Operating system
Application
Operating system
Application
Hypervisor
Host hardware
Type 2 Hypervisor
Operating system
Application
Operating system
Application
Operating system
Application
Hypervisor
Operating system
Host hardware

Hardware Virtualization Techniques

- CPU installed on the host is only one set, but each VM that runs on the host requires their own CPU
- It means CPU needs to virtualized, done by hypervisor

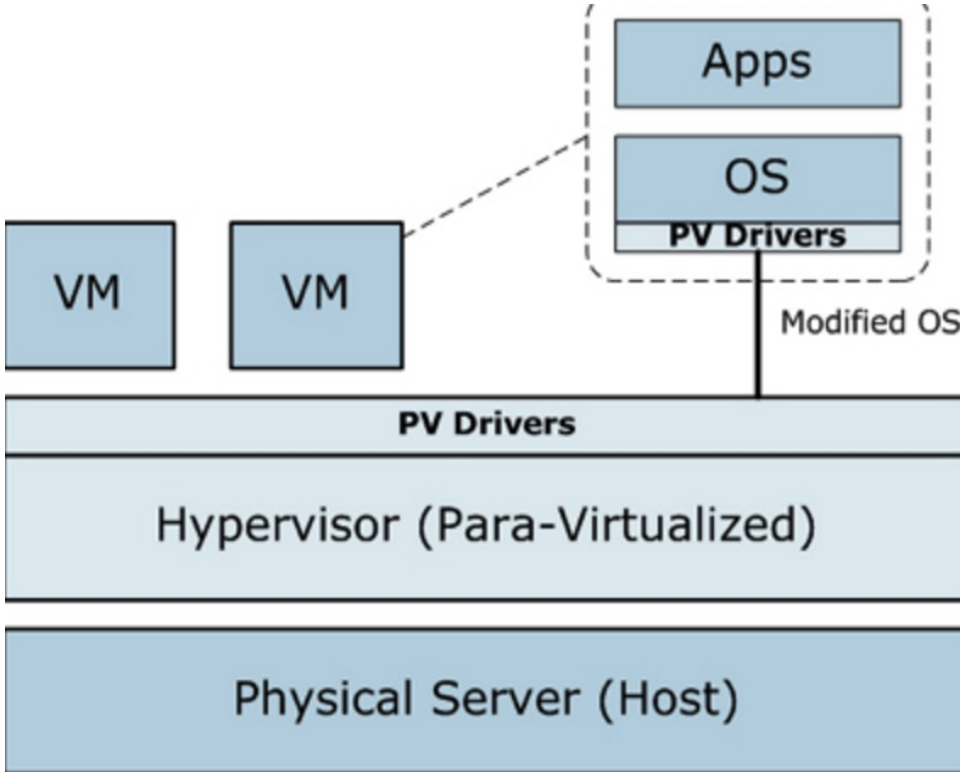
Full virtualization

- Ability to run program (OS) directly on top of a VM and without any modification
- Advantages:
  - Complete isolation
  - Enhanced security
  - Easy of emulation of different architectures and



Paravirtualization

- Not-transparent virtualization
- Guest OS need to be modified
- Simply transfer the execution of instructions which were hard to virtualized, directly to the host.



Common Virtualization Uses

Test and Development – Rapidly provision test and development servers; store libraries of pre-configured test machines

Server Consolidation and Containment – Eliminate server sprawl by deploying systems into virtual machines that can run safely and move transparently across shared hardware

Business Continuity – Reduce cost and complexity by encapsulating entire systems into single files that can be replicated and restored onto any target server

Enterprise Desktop – Secure unmanaged PCs without compromising end-user autonomy by layering a security policy in software around desktop virtual machines

INTRODUCTION TO CONTAINERS

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

Content

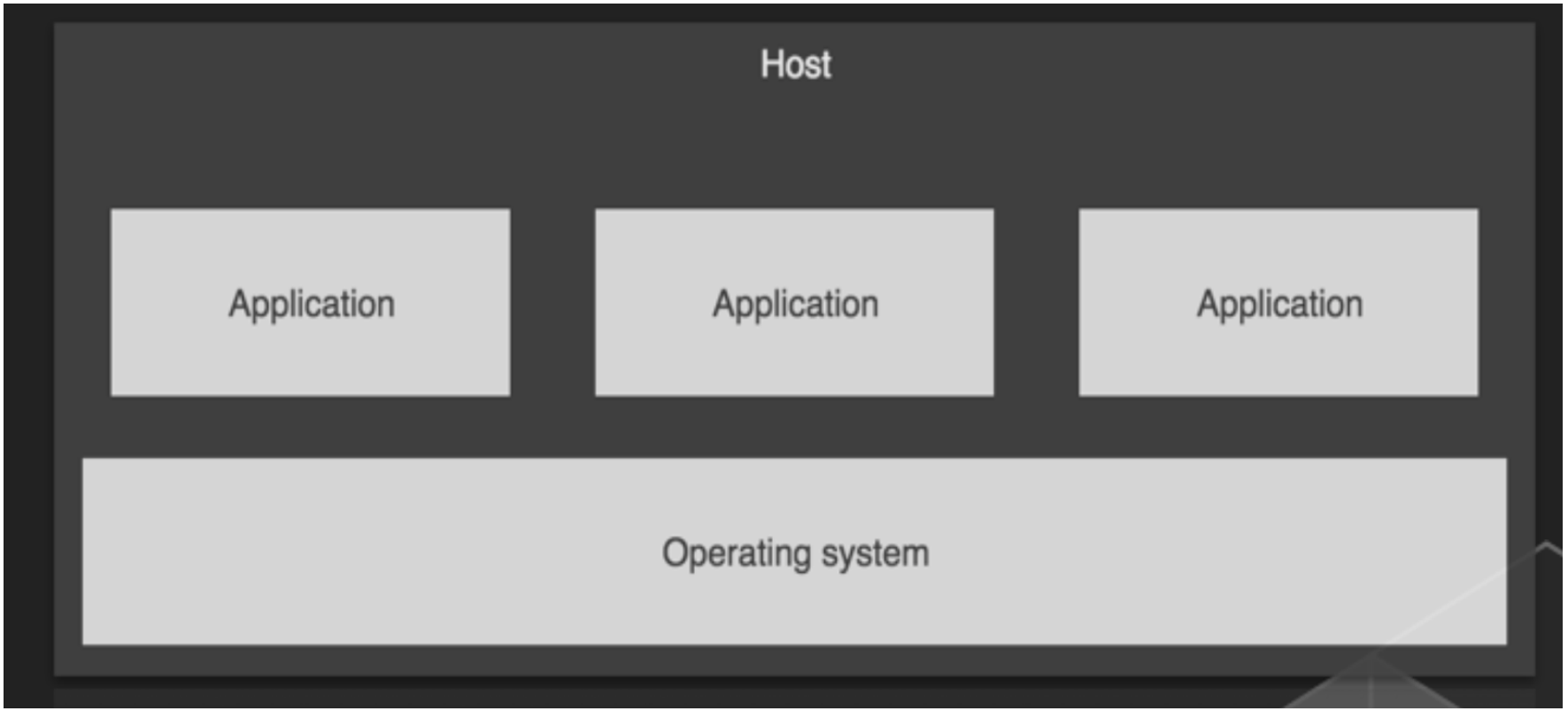
- History of Containers
- Containers vs Virtual Machine
- Docker platform Overview and Terminology
- Introduction to Images
- Getting Started with Containers

Introduction to Containers Technology

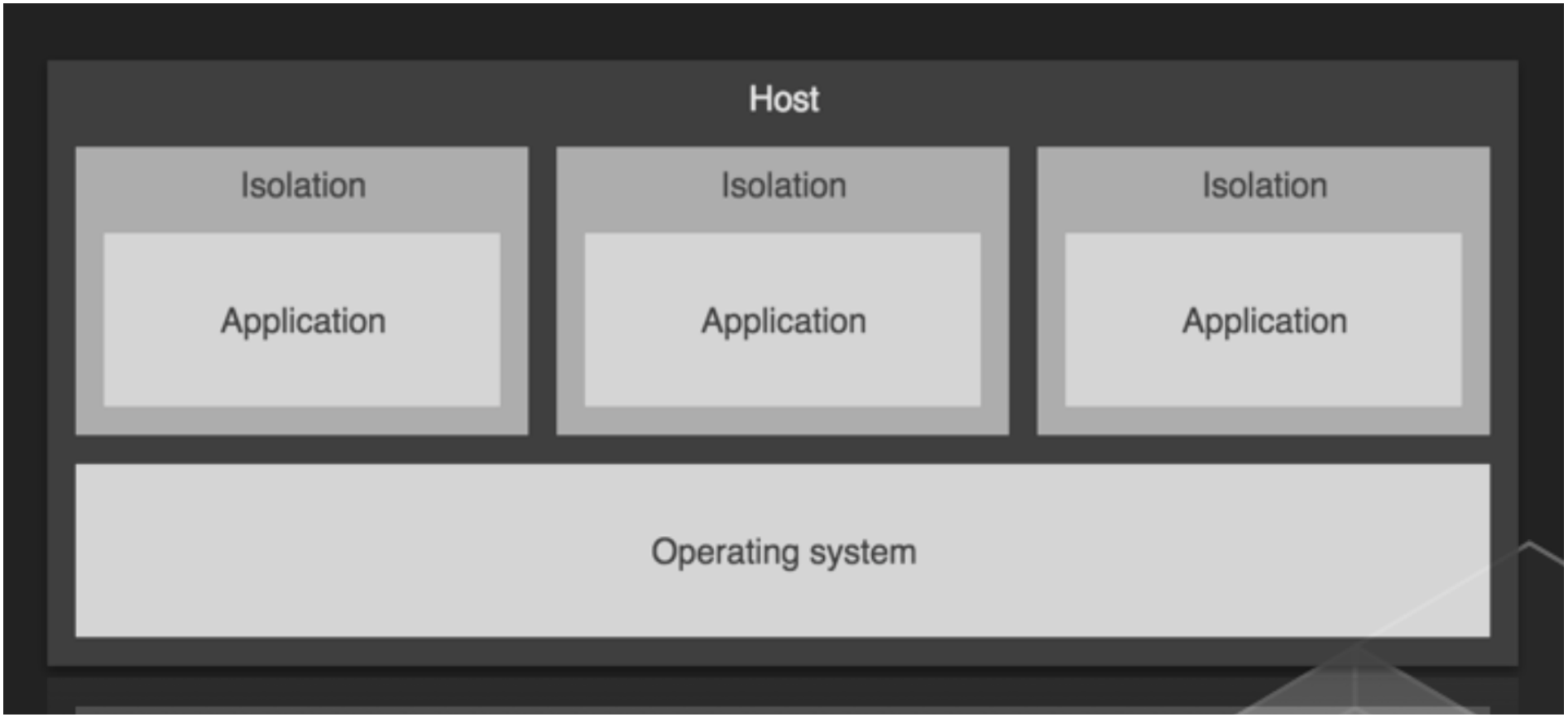
HOW TO DEPLOY AND ISOLATE AN APPLICATION ANYWHERE WITHOUT TAKING CARE ABOUT THE ENVIRONMENT?

Container-based Virtualization

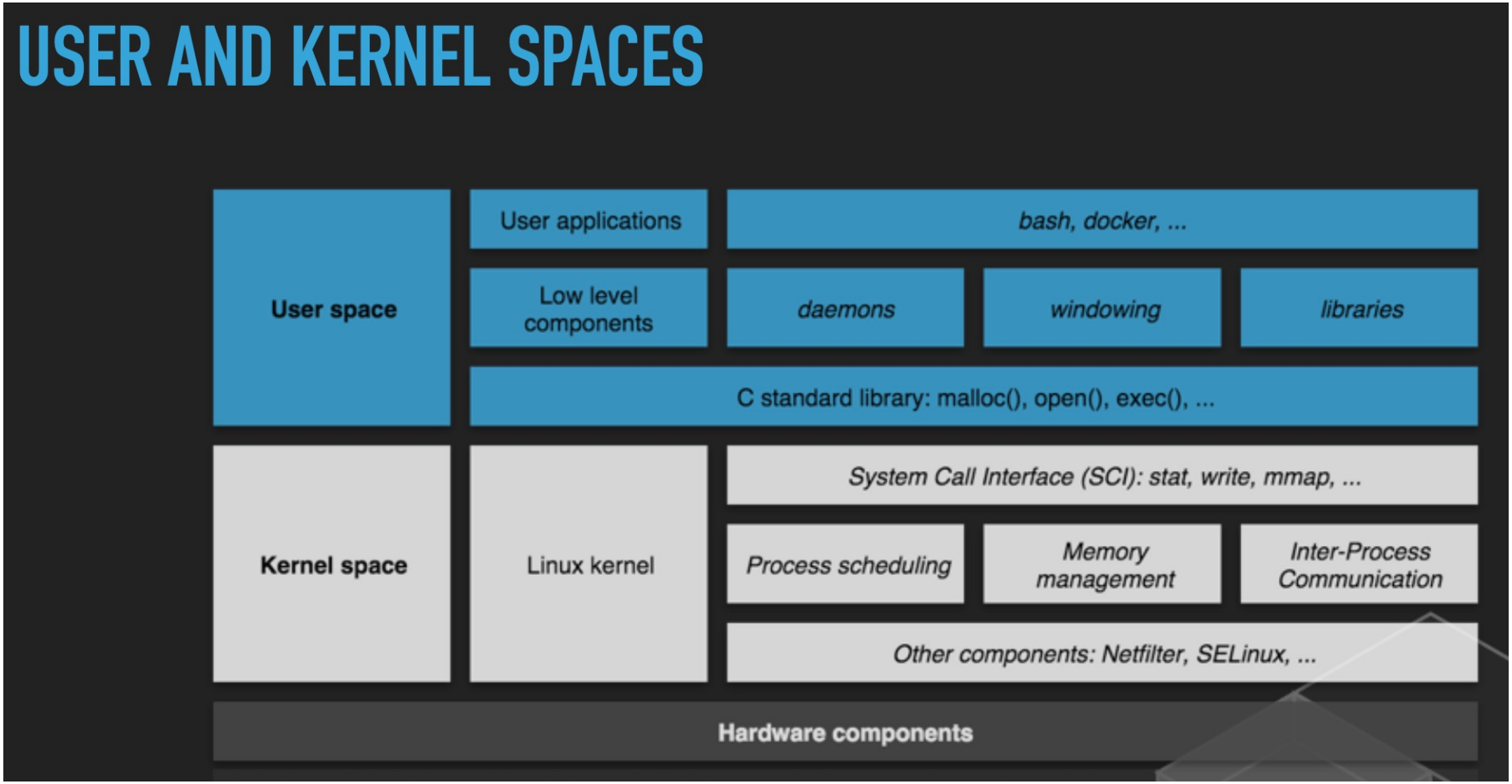
Container-based virtualization



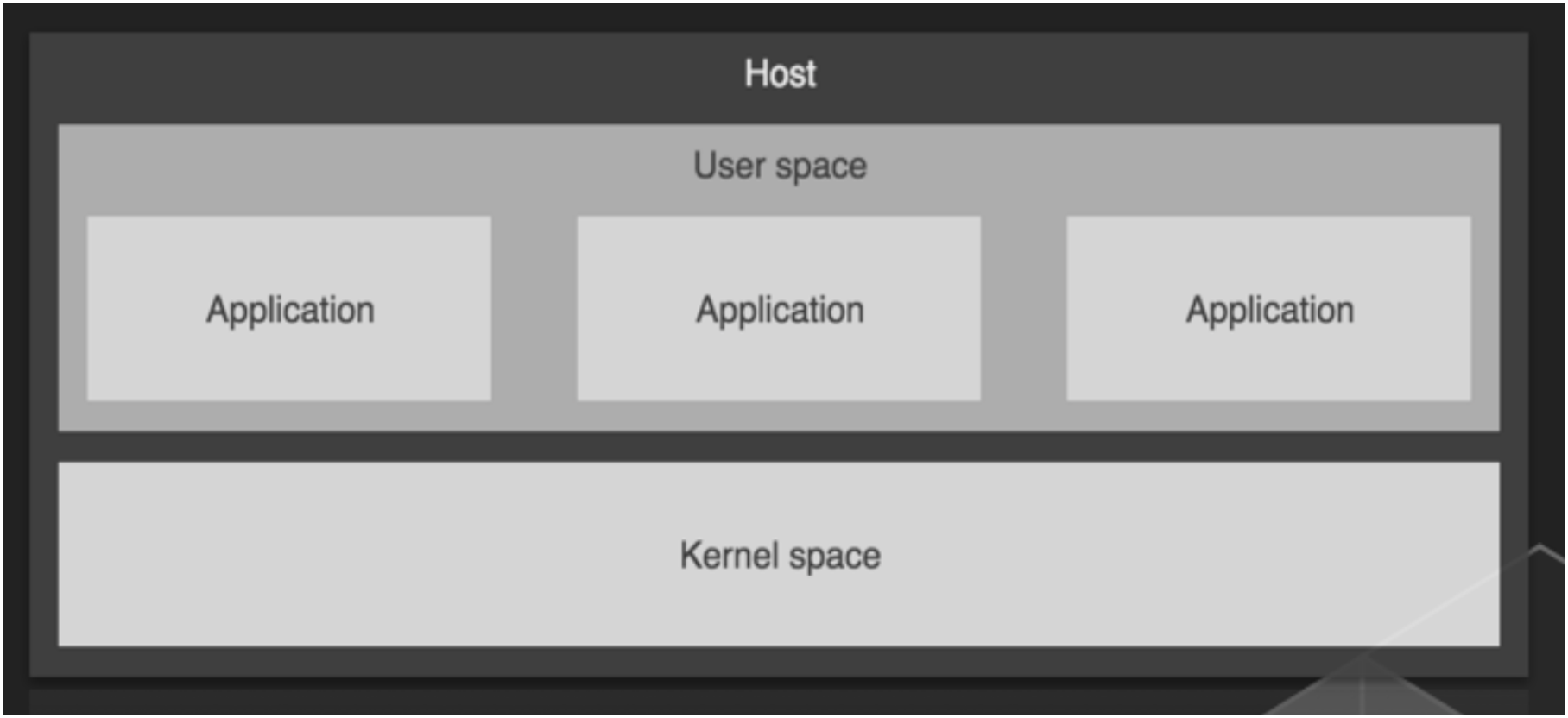
Container-based virtualization



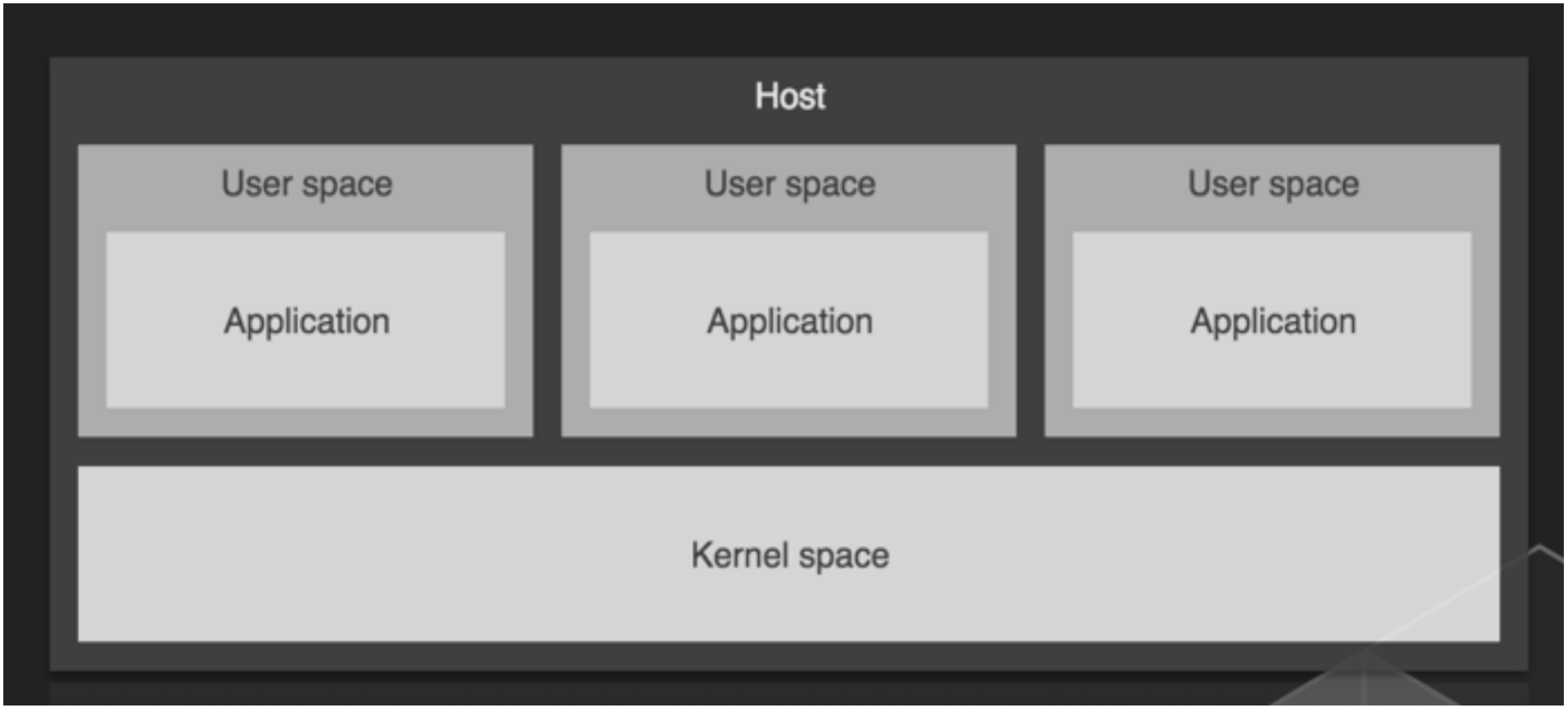
Container – Architecture



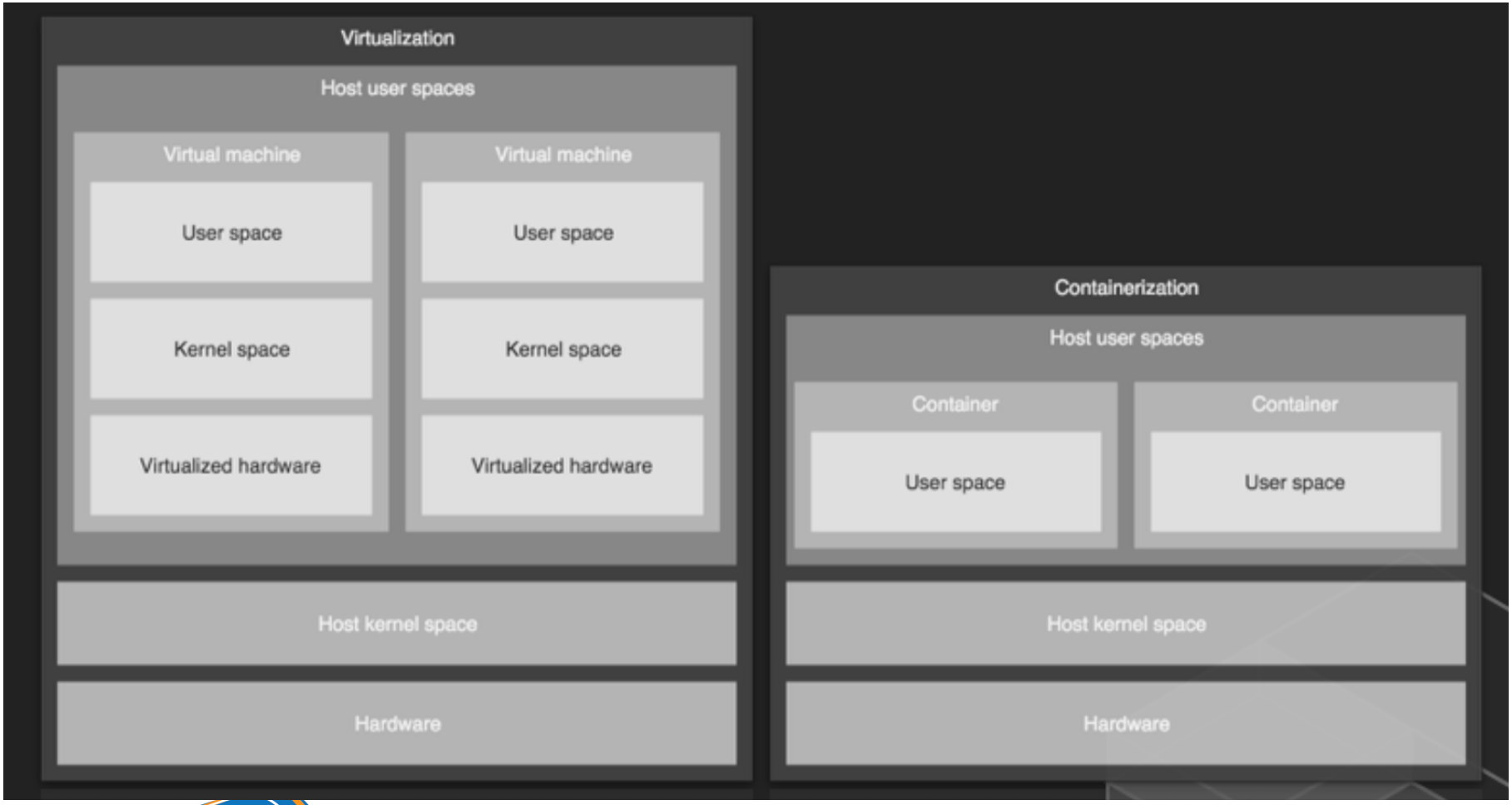
Non-Isolated Applications



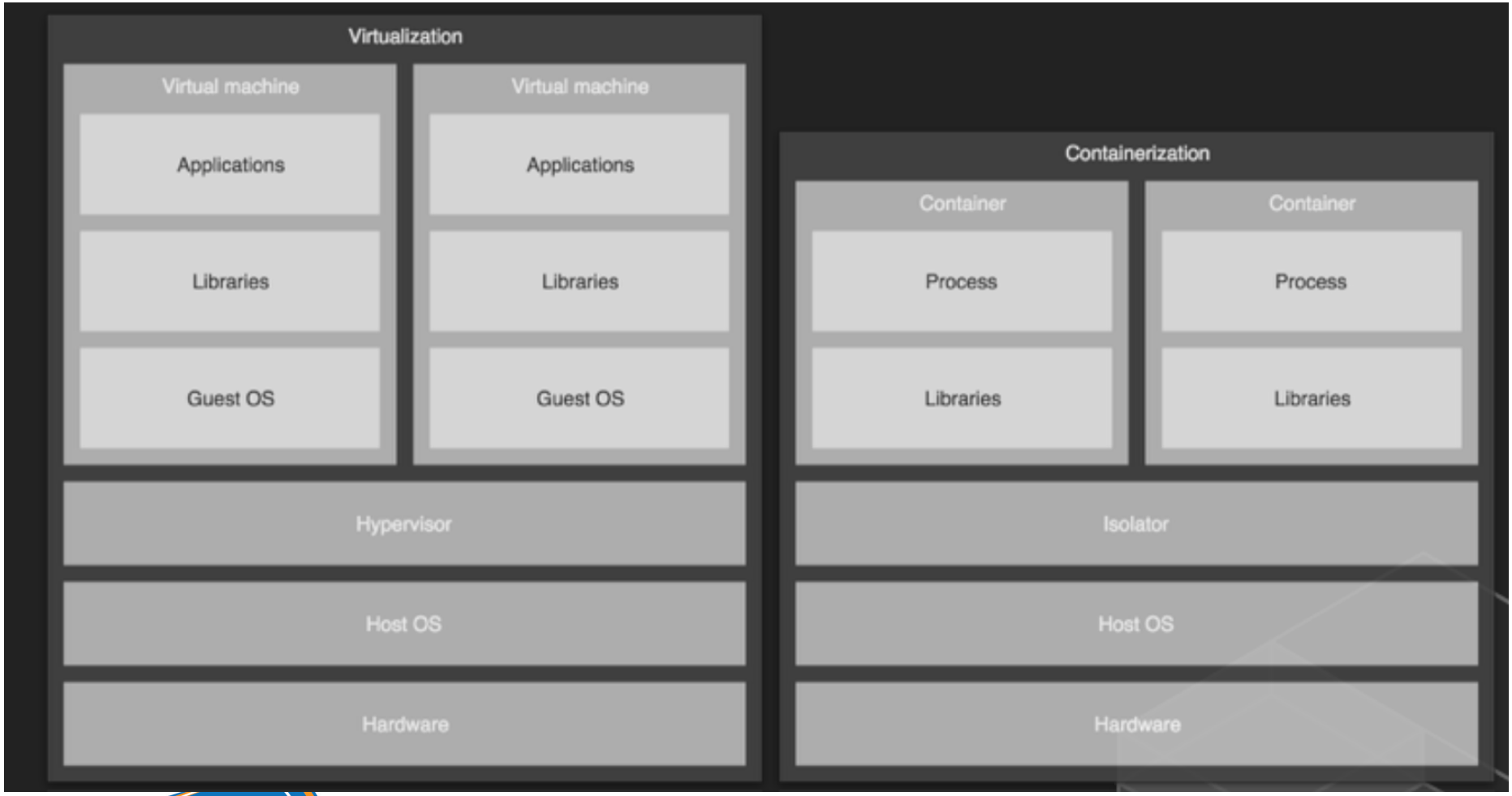
Isolated Applications



Containers vs Virtual Machines



Container vs Virtual Machines



Benefits of VM

- Better resource pooling
  - One physical machine divided into multiple virtual machines
- Easier to scale
- VM's in the cloud
  - Rapid elasticity
  - Pay as you go model

Limitations of VMs

- Each VM stills requires
  - CPU allocation
  - Storage
  - RAM
  - An entire guest operating system
- The more VM's you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed

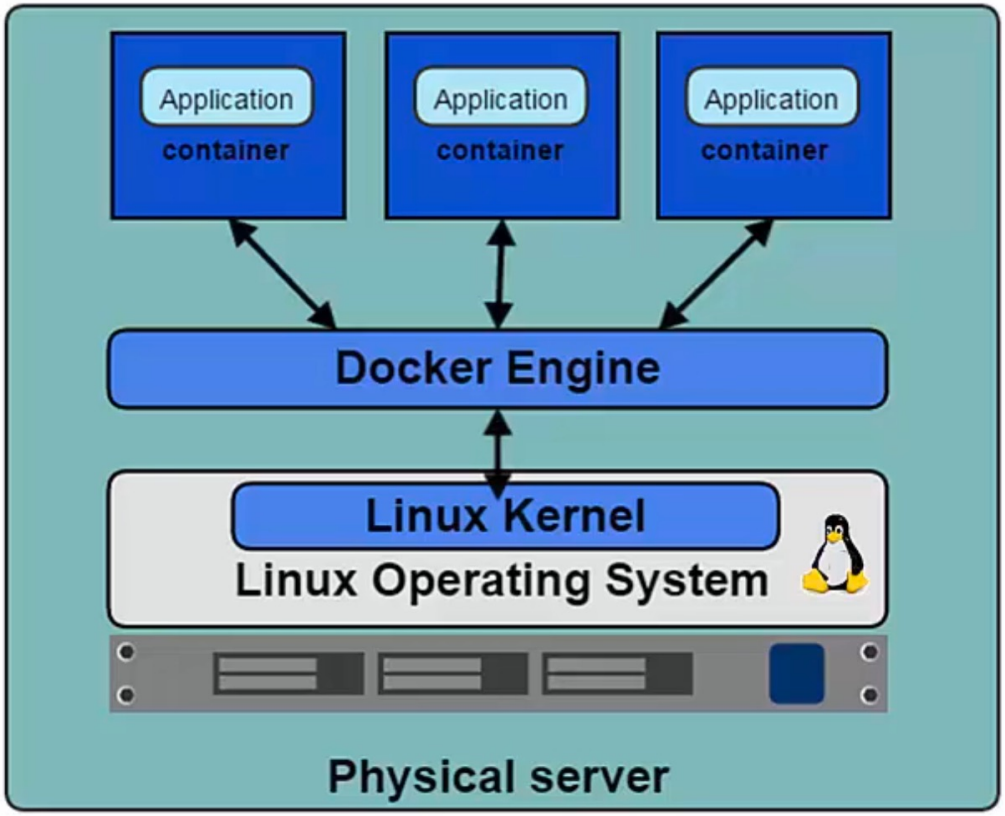
What is Docker ?

Docker *is a platform for developing, shipping and running applications using container virtualization technology*

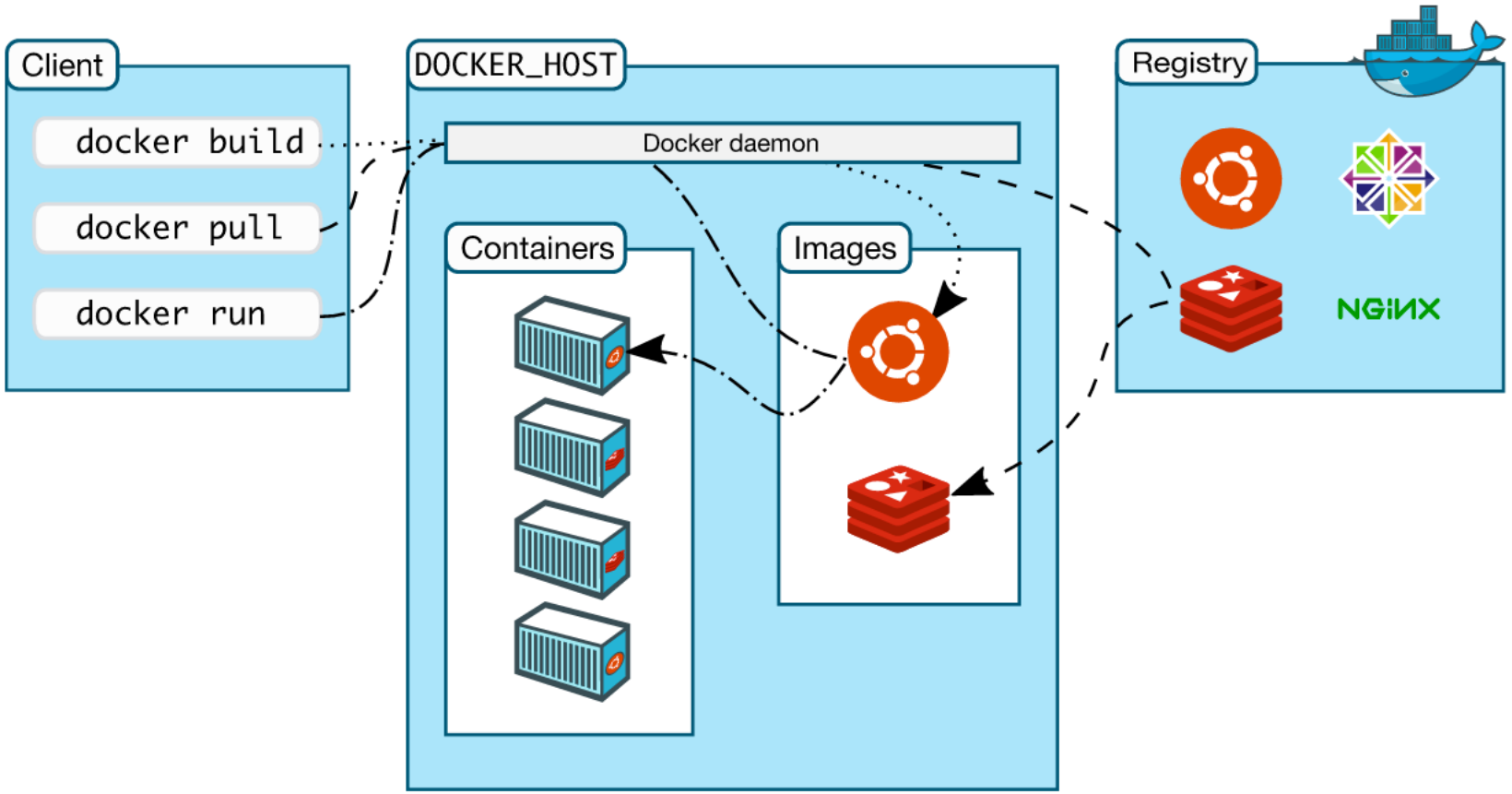
- The Docker Platform consists of multiple products/tools
  - Docker Engine
  - Docker Hub
  - Docker Machine
  - Docker Swarm
  - Docker Compose
  - Kitematic

Docker and the Linux kernel

- **Docker Engine** is the program that enables containers to be built, shipped and run.
- Docker Engine uses Linux Kernel namespaces and control groups
- Namespaces give us the isolated workspace

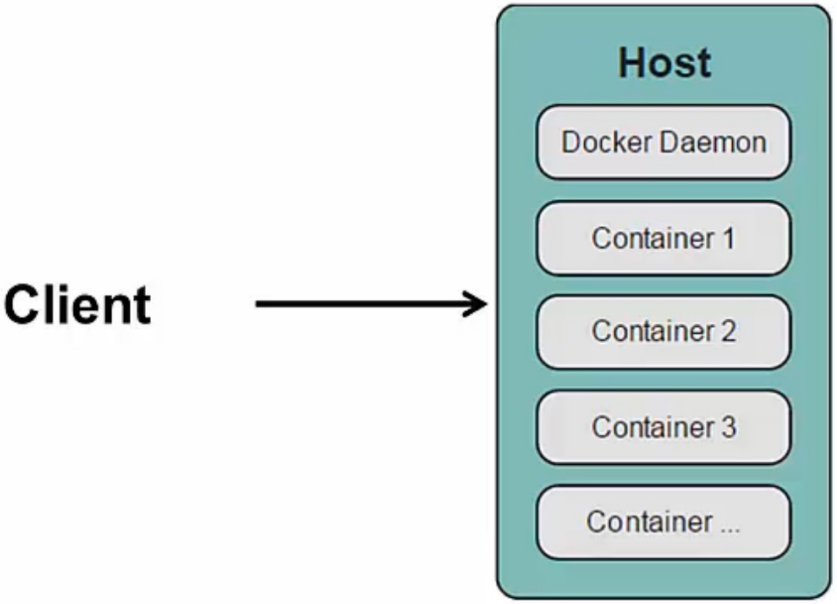


Docker Architecture



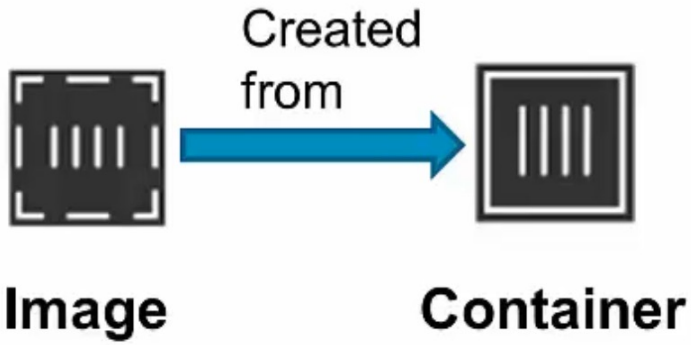
Docker Client and Daemon

- Client / Server architechture
- Client takes user inputs and send them to the daemon
- Daemon builds, runs and distributes containers
- Client and daemon can run on the same host or on different hosts
- CLI client and GUI (Kitematic)

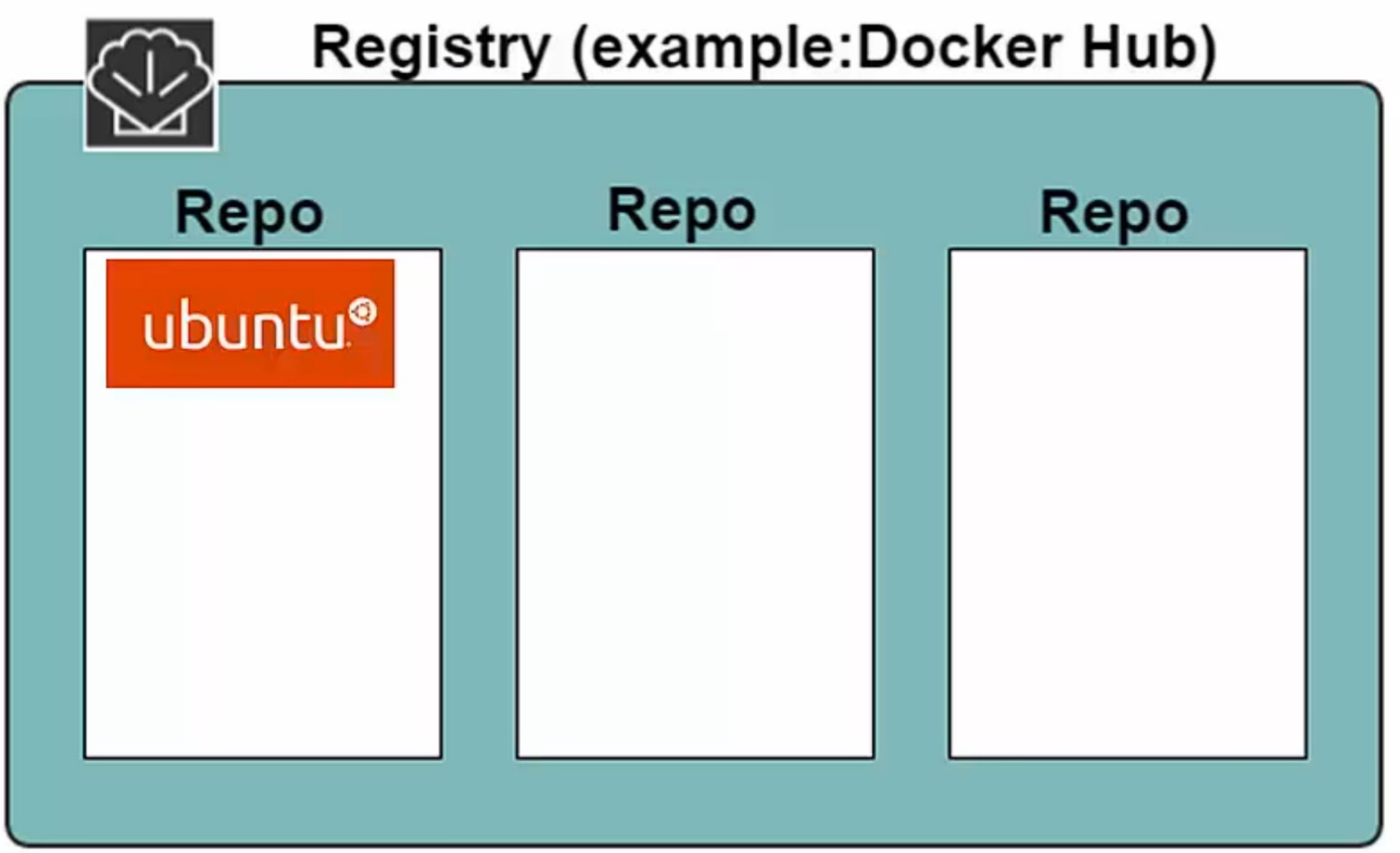


Docker Containers & Images

- Images
  - Read only template used to create containers
  - Built by you or other Docker users
  - Stored in the Docker Hub or your local Registry
- Containers
  - Isolated application platform
  - Contains everything needed to run your application
  - Based on one or more images

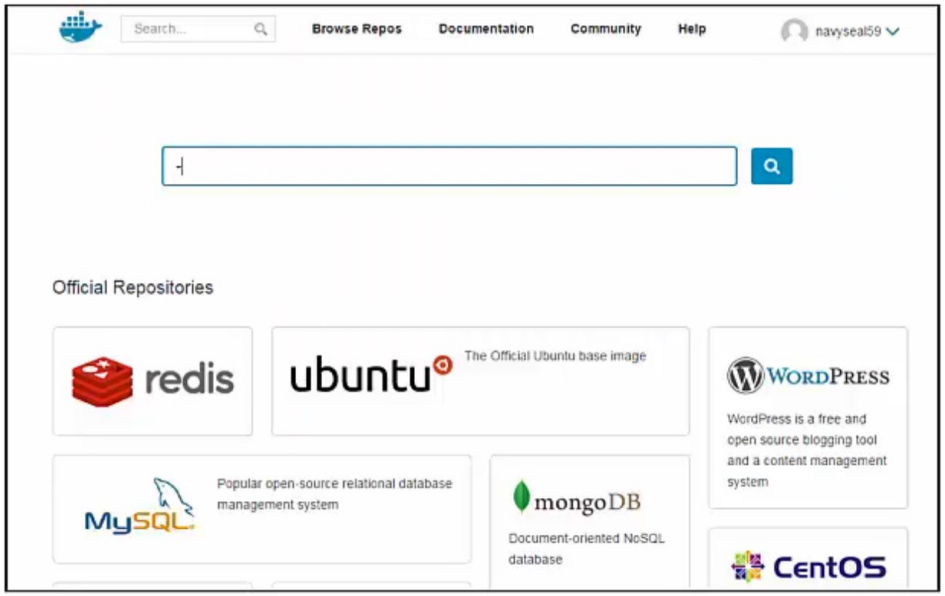


Registry and Repository



Docker Hub

Docker Hub *is the public registry that contains a large number of images available for your use*



Docker Orchestration

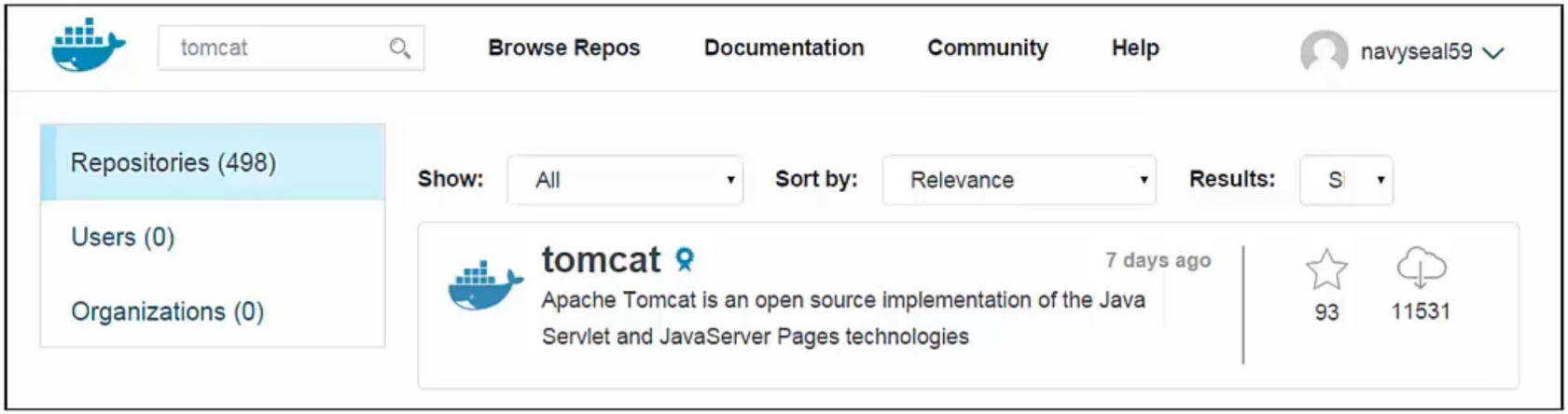
- Three tools for orchestrating distributed applications with Docker
- Docker Machine
  - Tool that provisions Docker hosts and installs the Docker Engine on them
- Docker Swarm
  - Tool that clusters many Engines and schedules containers
- Docker Compose
  - Tool to create and manage multi-container applications
- Covered in Docker Operations course

Benefits of Docker

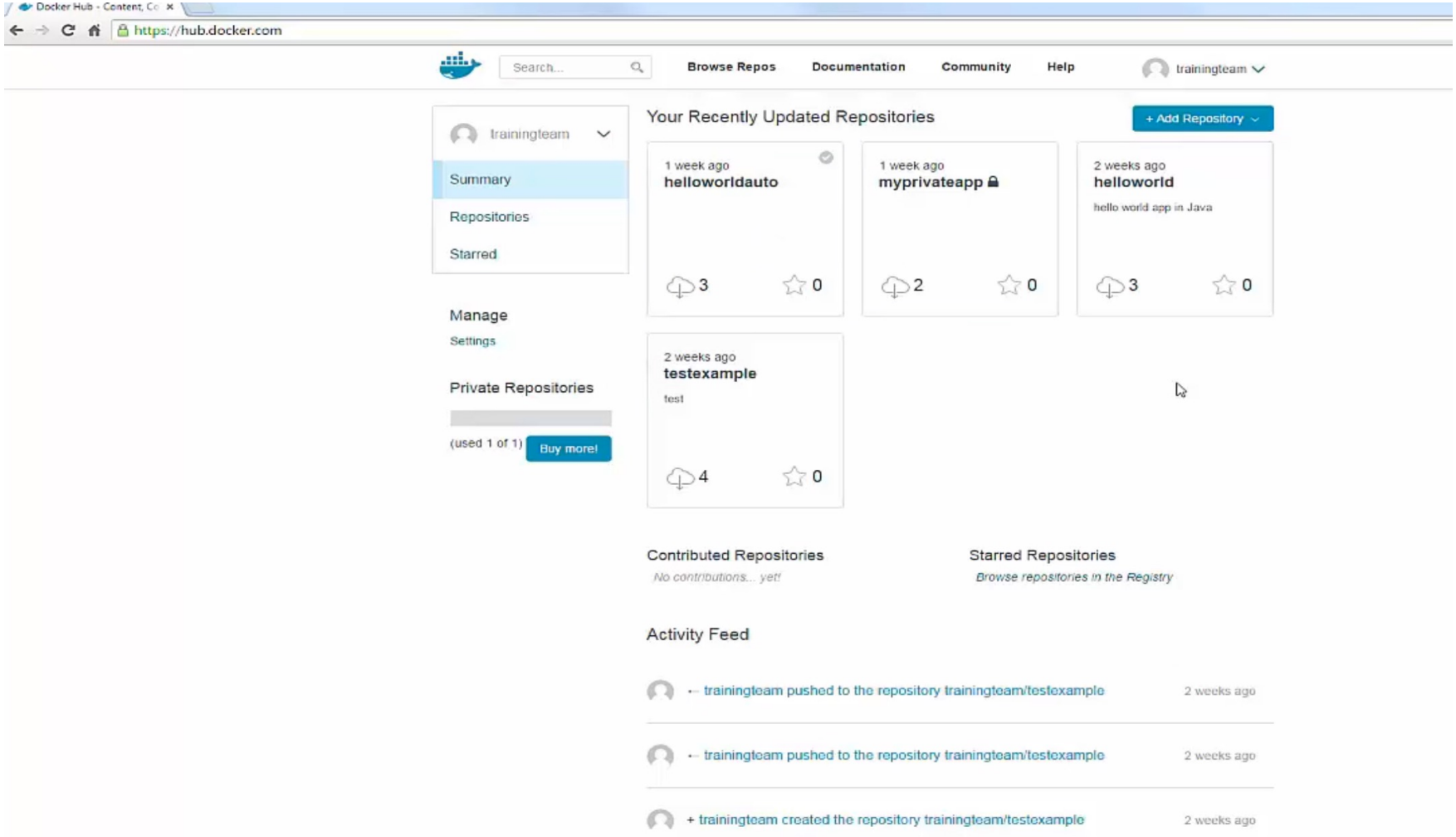
- Separation of concerns
  - Developers focus on building their apps
  - System admins focus on deployment
- Fast development cycle
- Application portability
  - Build in one environment, ship to another
- Scalability
  - Easily spin up new containers if needed
- Run more apps on one host machine

Intro to Images

- Lots of Images available for use
- Images reside in various Repositories



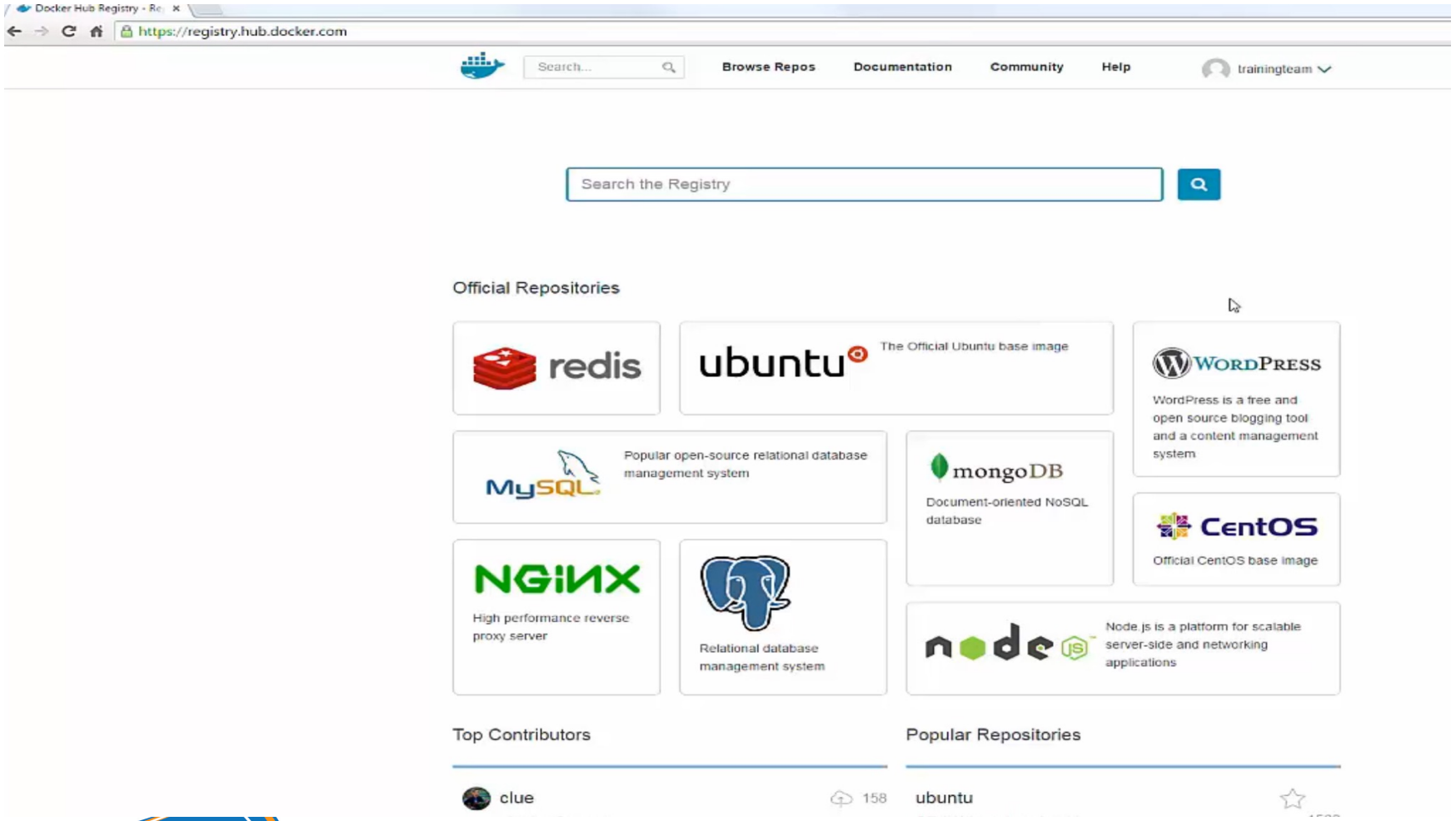
Docker Hub


Docker Hub - Content, Co
https://hub.docker.com
Search...
Browse Repos
Documentation
Community
Help
trainingteam
trainingteam
Summary
Repositories
Starred
Manage
Settings
Private Repositories
(used 1 of 1)
Buy more!
Your Recently Updated Repositories
+ Add Repository
1 week ago
helloworldauto
3
0
1 week ago
myprivateapp
2
0
2 weeks ago
helloworld
hello world app in Java
3
0
2 weeks ago
testexample
test
4
0
Contributed Repositories
No contributions... yet!
Starred Repositories
Browse repositories in the Registry
Activity Feed
trainingteam pushed to the repository trainingteam/testexample
2 weeks ago
trainingteam pushed to the repository trainingteam/testexample
2 weeks ago
trainingteam created the repository trainingteam/testexample
2 weeks ago

cdio 4.0

fit@hcmus

Docker Hub



Create a Docker Hub account

1. Go to https://hub.docker.com/account/signup/ and signup for an account if you do not already have one.
   No credit card details are needed
2. Find your confirmation email and activate your account
3. Browse some of the repositories
4. Search for some images of your favourite dev tools, languages, servers etc…
   a) (examples: Java, Perl, Maven, Tomcat, NGINX, Apache)

Display local images

- Run **`docker images`**
- When creating a container NGINX will attempt to use a local image first
- If no local image is found, the Docker daemon will look in Docker Hub unless another registry is specified

```
johnnytu@dockertraining:~$ sudo docker images
[sudo] password for johnnytu:
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
ubuntu              14.04               2103b00b3fdf        8 days ago          188.3 MB
ubuntu              14.04.2             2103b00b3fdf        8 days ago          188.3 MB
ubuntu              latest              2103b00b3fdf        8 days ago          188.3 MB
ubuntu              trusty              2103b00b3fdf        8 days ago          188.3 MB
ubuntu              trusty-20150228.11  2103b00b3fdf        8 days ago          188.3 MB
java                openjdk-7-jdk       fa5c5774f090        10 days ago         584.9 MB
nginx               latest              b17a02c942e1        10 days ago         93.43 MB
centos              7                   88f9454e60dd        2 weeks ago         210 MB
```

Images Tag

- Images are specified by **repository:tag**
- The same image may have multiple tags
- The default tag is `latest`
- Look up the repository on Docker Hub to see what tags are available

Create a Container

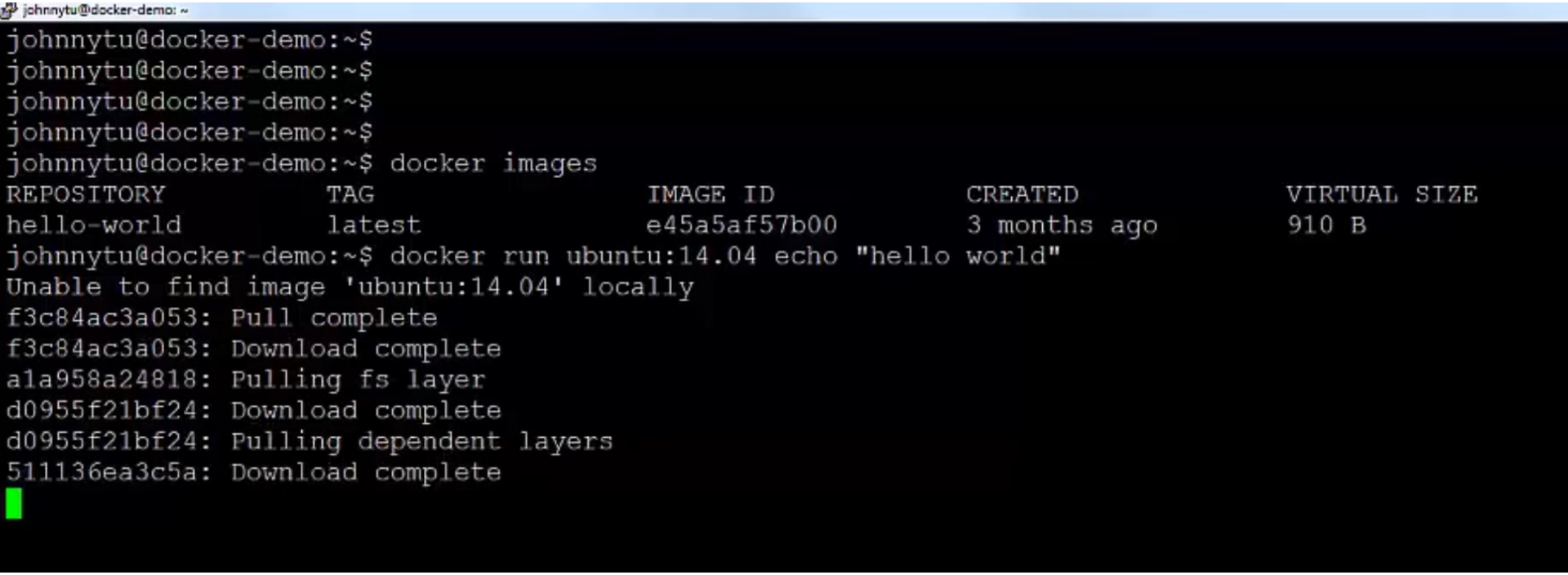
- Use **`docker run`** command
- Syntax
  `sudo docker run [options] [image] [command] [args]`
- Image is specified with `repository:tag`

Examples

```
docker run ubuntu:14.04 echo "Hello World"
docker run ubuntu ps ax
```

Run a simple container

1. On your terminal type
   `docker run ubuntu:14.04 echo "hello world"`
2. Observe the output
3. Then type
   `docker run ubuntu:14.04 ps ax`
4. Observe the output
5. Notice the much faster execution time compared to the first container that was run. This is due to the fact that Docker now has the Ubuntu 14.04 image locally and thus does not need to download the image


johnnytu@docker-demo: ~
johnnytu@docker-demo:~$
johnnytu@docker-demo:~$
johnnytu@docker-demo:~$
johnnytu@docker-demo:~$
johnnytu@docker-demo:~$ docker images
REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE
hello-world latest e45a5af57b00 3 months ago 910 B
johnnytu@docker-demo:~$ docker run ubuntu:14.04 echo "hello world"
Unable to find image 'ubuntu:14.04' locally
f3c84ac3a053: Pull complete
f3c84ac3a053: Download complete
a1a958a24818: Pulling fs layer
d0955f21bf24: Download complete
d0955f21bf24: Pulling dependent layers
511136ea3c5a: Download complete

Container with Terminal

- Use `-i` and `-t` flags with docker run
- The `-i` flag tells docker to connect to STDIN on the container
- The `-t` flag specifies to get a pseudo-terminal
- **Note:** You need to run a terminal process as your command (e.g. `/bin/bash`)

Example

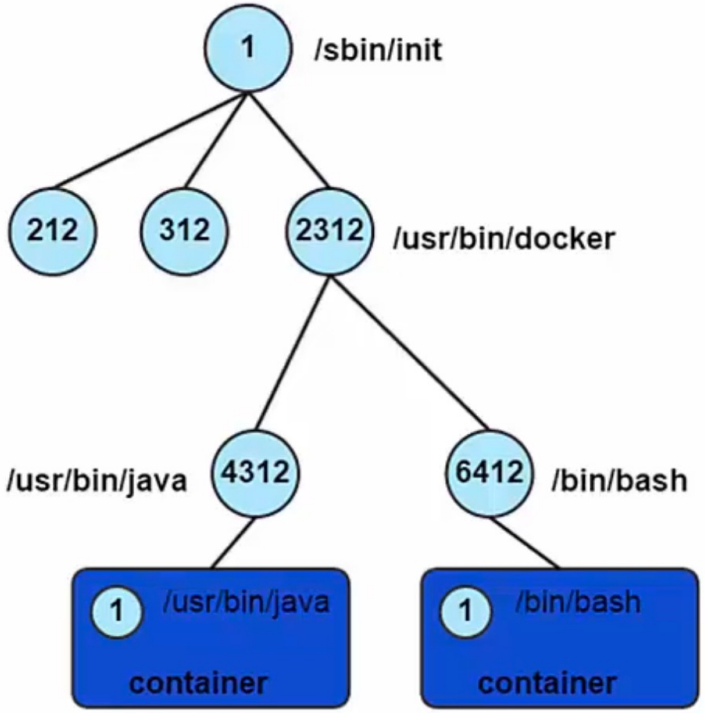
```
docker run -i -t ubuntu:latest /bin/bash
```

Run a Container and get Terminal access

1. Create a container using the ubuntu 14.04 image and connect to STDIN and a terminal
   ```
   sudo docker run -i -t ubuntu:14.04 /bin/bash
   ```
2. In your container, create a new user using your first and last name as the username
   ```
   adduser username
   ```
3. Add the user to the sudo group
   ```
   adduser username sudo
   ```
4. Exit the container
   ```
   exit
   ```
5. Notice how the container shut down
6. Once again run:
   ```
   sudo docker run -i -t ubuntu:14.04 /bin/bash
   ```
7. Try and find your user
8. Notice that it does not exist

Container process

- A container only runs as long as the process from your specified `docker run` command is running
- Your command's process is always PID 1 inside the container



Container ID

- Containers can be specified using their ID or name
- Long ID and short ID
- Short ID and name can be obtained using `docker ps` command to list containers
- Long ID obtained by inspecting a container

cdio 4.0

fit@hcmus

Container ID

```
johnnytu@docker-demo:~$
johnnytu@docker-demo:~$
johnnytu@docker-demo:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
22e50a079686        ubuntu:14.04        "bash"              30 minutes ago      Up 30 minutes                           loving_f

johnnytu@docker-demo:~$ docker ps -a
CONTAINER ID        IMAGE                 COMMAND                  CREATED             STATUS                      PORTS
NAMES
22e50a079686        ubuntu:14.04          "bash"                   30 minutes ago      Up 30 minutes
loving_feynman
b2cee57aaf92        ubuntu:14.04          "echo 'hello world'"     39 minutes ago      Exited (0) 39 minutes ago
compassionate_thompson
901138dca328        ubuntu:14.04          "/bin/bash"              About an hour ago   Exited (0) About an hour ago
jolly_mcclintock
7a0de6083648        ubuntu:14.04          "/bin/bash"              About an hour ago   Exited (0) About an hour ago
ecstatic_thompson
84efc508ab90        ubuntu:14.04          "ps ax"                  About an hour ago   Exited (0) About an hour ago
gloomy_pasteur
731fce6fa185        ubuntu:14.04          "echo 'hello world'"     About an hour ago   Exited (0) About an hour ago
adoring_archimedes
fabf3617079f        hello-world:latest    "/hello"                 24 hours ago        Exited (0) 24 hours ago
jolly_hoover
dcc3d3c46d2a        hello-world:latest    "/hello"                 24 hours ago        Exited (0) 24 hours ago
lonely_davinci
johnnytu@docker-demo:~$
```

Running in Detached Mode

- Also known as running in the background or as a daemon
- Use `-d` flag
- To observe output use **`docker logs [container id]`**

Create a centos container and run the ping command to ping the container itself 50 times

```
docker run -d centos:7 ping 127.0.0.1 -c 50
```

Q&A