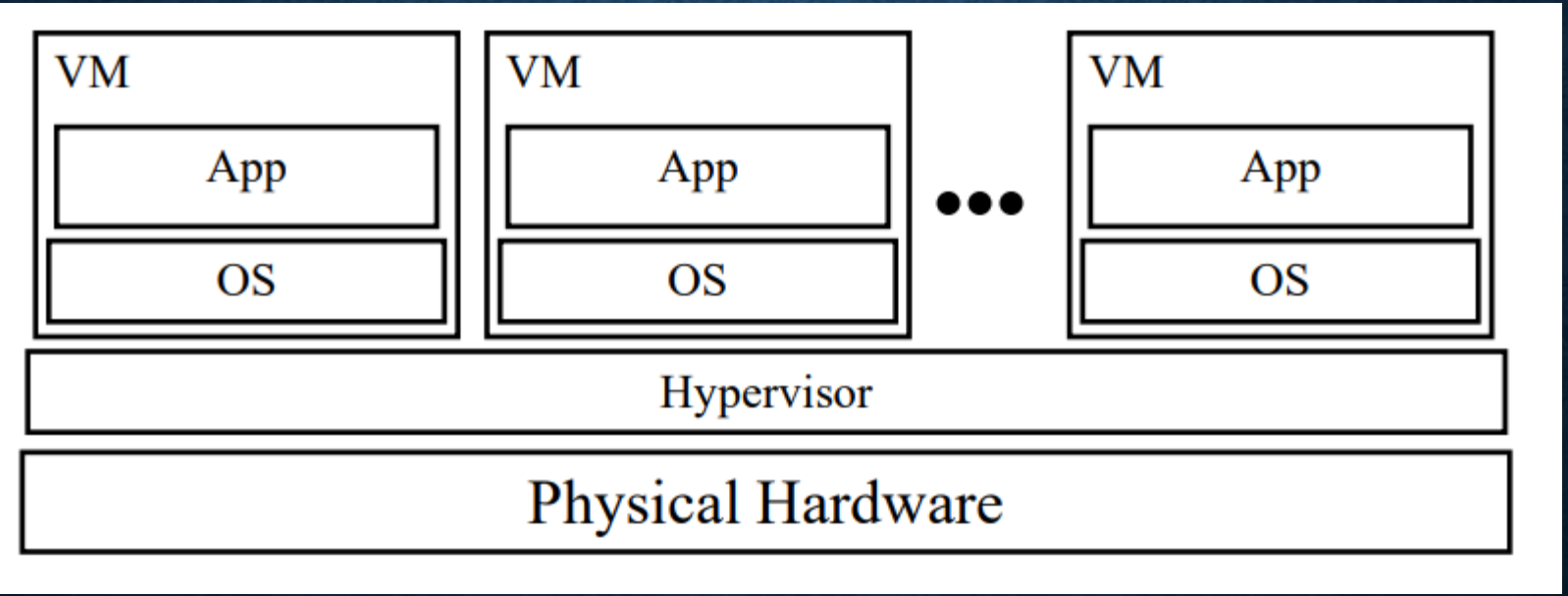
VIRTUALIZATION

VIRTUALIZATION

- Creating a Virtual version of something as opposed to a physical representation. The "something" can be hardware components, platforms, resources (disk, network, etc), and system software.
- Has existed since the first multi-user Operating systems of the 1960's
- Some types:
- Hardware Virtualization
- Machine snapshots
- Migration and Failover Techniques
- Simulation and Emulation
- Desktop Virtualization
- Containerization

REPRESENTATION OF A VIRTUALIZED SYSTEM



ADVANTAGES

- Minimize hardware costs
  - Multiple virtual servers on one physical hardware
- Easily move VMs to other data centers
  - Provide disaster recovery. Hardware maintenance.
  - Follow the sun (active users) or follow the moon (cheap power)
- Consolidate idle workloads. Usage is bursty and asynchronous.
  - Increase device utilization
- Conserve power
  - Free up unused physical resources
- Easier automation
  - Simplified provisioning/administration of hardware and software
- Scalability and Flexibility: Multiple operating systems

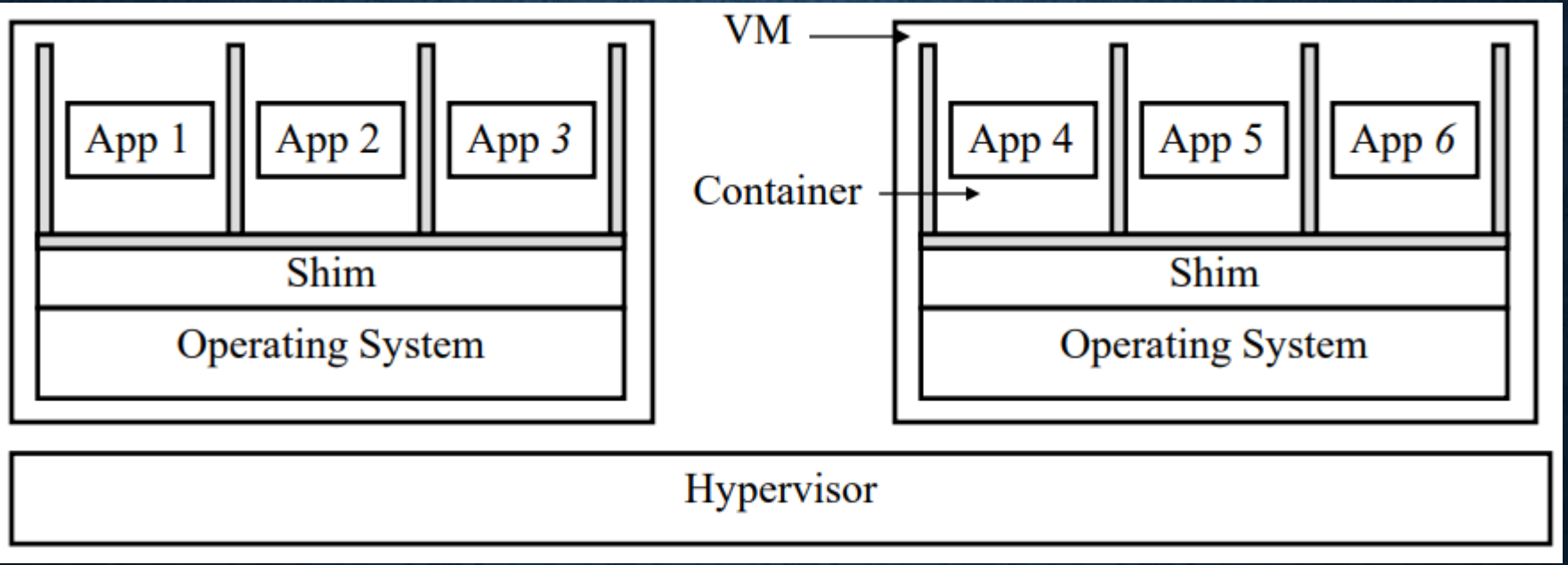
ISSUES

- Each VM requires an operating system (OS)
- Each OS requires a license ⇒ CapEx (Capital Expenditure)
- Each OS has its own compute and storage overhead
- Needs maintenance, updates ⇒ OpEx (Operationa Expenditure)
- VM Tax = added CapEx + OpEx

SOLUTION: LIGHTWEIGHT VIRTUALIZATION (CONTAINERS)

- Run many applications in the same virtual machine
- These applications share the OS and its overhead
- But these applications
  - Can't interfere with each other
  - Can't access each other's resources without explicit permission

REPRESENTATION OF A CONTAINERIZED SYSTEM



CONTAINERS

- Multiple containers run on one operating system on a virtual/physical machine
- All containers share the operating system ⇒ CapEx and OpEx
- Containers are isolated ⇒ cannot interfere with each other
- Own file system/data, own networking ⇒ Portable

PROPERTIES OF CONTAINERS

Containers have all the good properties of VMs

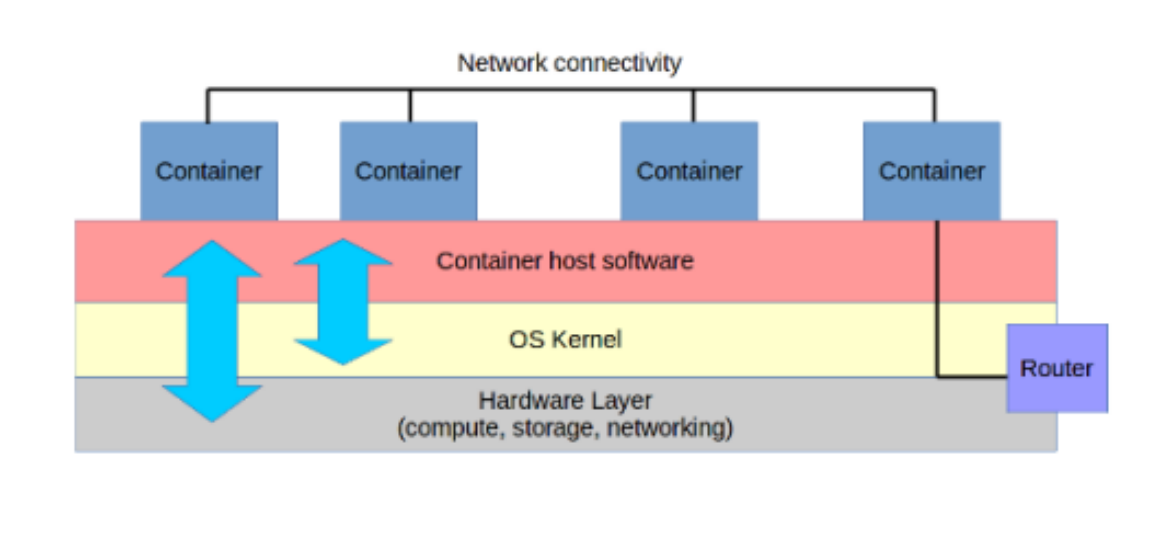
- Come complete with all files and data that you need to run
- Multiple copies can be run on the same machine or different machine ⇒ Scalable
- Same image can run on a personal machine, in a data center or in a cloud
- Operating system resources can be restricted or unrestricted as designed at container build time
- Isolation: For example, ps command in a container will show only the processes in the container
- Can be stopped. Saved and moved to another machine or for later run

VIRTUAL MACHINES VS CONTAINERS

Criteria	VM	Containers
Image Size	3x	x
Boot Time	>10s	~1s
Computer Overhead	>10%	<5%
Disk I/O Overhead	>50%	Negligible
Isolation	Good	Fair
Security	Low/Med-low	Medium-High
Flexibility	Excellent	Poor
Management	Excellent	Still Evolving
Legacy Application Impact	Low-Medium	High

DOCKER

- Provides the isolation among containers
- Helps them share the OS
- Docker = Dock worker ⇒ Manage containers
- Developed initially by docker.com
- Downloadable for Linux, Windows, and Mac from docker.com
- Customizable with replacement modules from others



DOCKER ENGINE

- Docker Engine: Runtime
- Two Editions:
  - Community Edition (CE): Free for experimentation
  - Enterprise Edition (EE): For deployment with paid support

Written in Go.

DOCKER ENGINE COMPONENTS

- daemon: API and other features
- containderd: Execution logic. Responsible for container lifecycle. Start, stop, pause, unpause, delete containers.
- runc: A lightweight runtime CLI
- shim: runc exists after creating the container. shim keeps the container running. Keep stdin/stdout open.

REPRESENTATION OF A DOCKER ENGINE

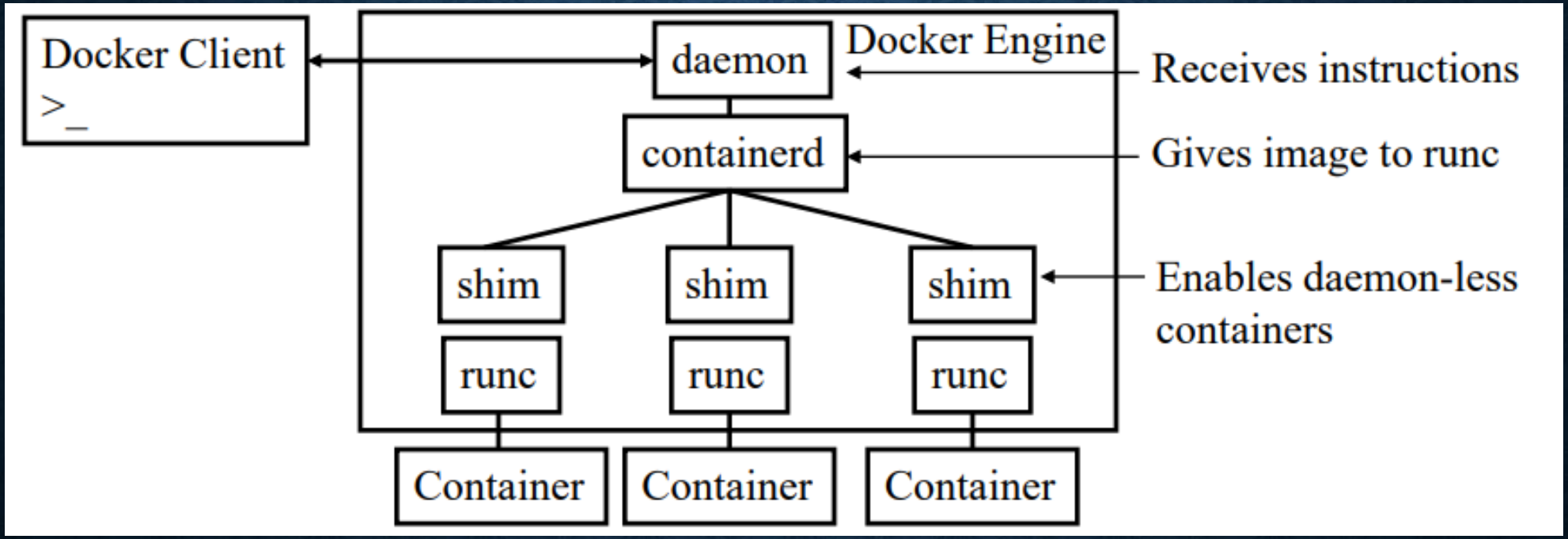
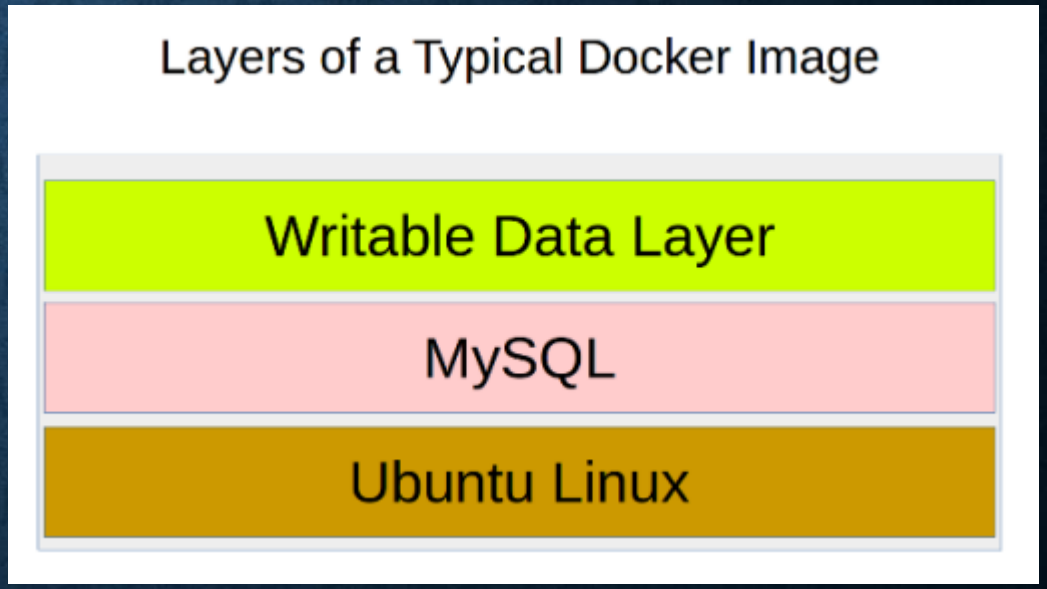


IMAGE REGISTRIES

- Containers are built from images and can be saves as images
- Images are stored in registries
  - Local registry on the same host
  - Docker Hub Registry: Globally shared
  - Private registry on Docker.com
- Any component not found in the local registry is downloaded from specified location
- Official Docker Registry: Images vetted by Docker
- Unofficial Registry: Images not vetted (Use with care)
- Each image has several tags, e.g., v2, latest, …
- Each image is identified by its 256-bit hash

IMAGE LAYERS

- Each image has many layers
- An Image is built layer by layer
- Layers in an image can be inspected by Docker commands
- Each layer has its own 256-bit hash
- For example:
  - Ubuntu OS is installed, then
  - Python package is installed, then
  - a security patch to the Python is installed
- Layers can be shared among many containers



BUILDING CONTAINER IMAGES

- Create a Dockerfile that describes the application, its dependencies, and how to run it

```
FROM Alpine                                  ← Start with Alpine Linux
LABEL maintainer="xx@gmail.com"              ← Who wrote this container
RUN apk add –update nodejs nodejs –npm       ← Use apk package to install nodejs
COPY . /src                                  ← Copy the app files from build context
WORKDIR /src                                 ← Set working directory
RUN nmp install                              ← Install application dependencies
EXPOSE 8080                                  ← Open TCP Port 8080
ENTRYPOINT ["node", "./app.js"]              ← Main application to run
```

DOCKER COMMANDS

- docker container run: Run the specified image
- docker container ls: list running containers
- docker container exec: run a new process inside a container
- docker container stop: Stop a container
- docker container start: Start a stopped container
- docker container rm: Delete a container
- docker container inspect: Show information about a container

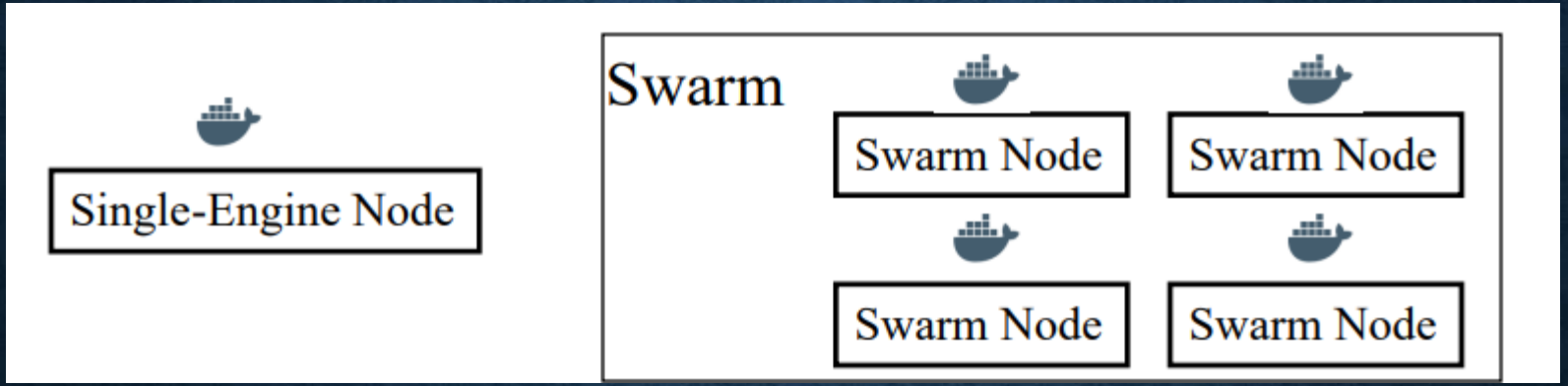
THE OPEN CONTAINER INITIATIVE (OCI)

- CoreOS defined an alternative image format and container runtime API's
- Led to the formation of OCI under the Linux Foundation to govern container standards
- OCI Image specifications
- OCI Runtime specifications
- Everyone including Docker is now moving to OCI

CONTAINER SWARMS

- Orchestrating thousands of containers
- Swarm: A group of nodes collaborating over a network
- Two modes for Docker hosts:
- Single Engine Mode: Not participating in a swarm
- Swarm Mode: Participating in a Swarm
- A service may run on a swarm
- Each swarm has a few managers that dispatch tasks to workers. Managers are also workers (i.e., execute tasks)

CONTAINER SWARMS

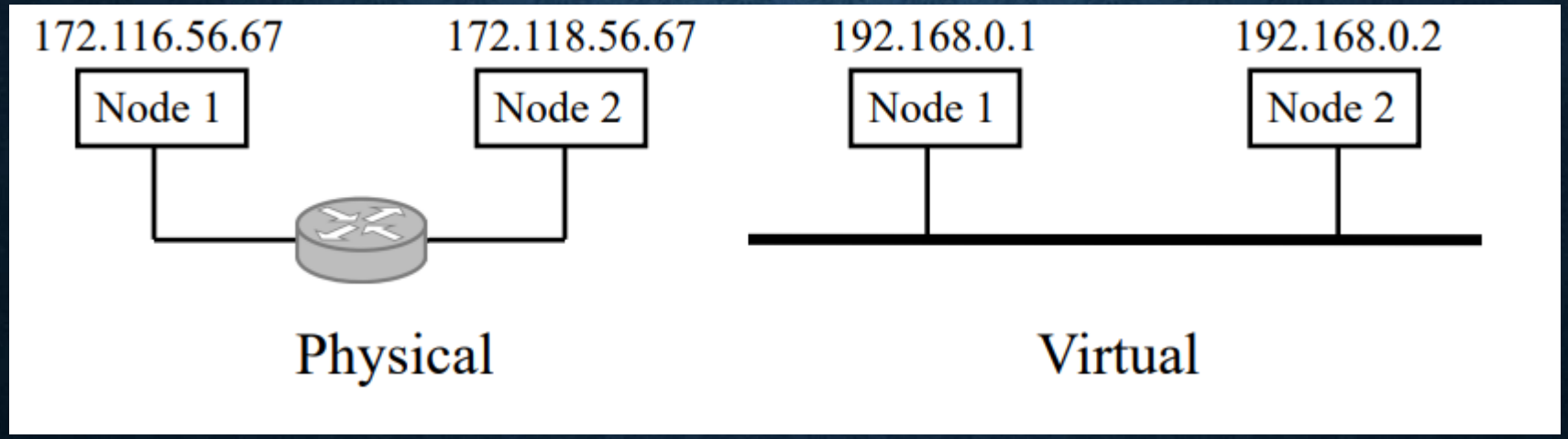


CONTAINER SWARMS

- The managers select a leader, who really keeps track of the swarm
- Assigns tasks, re-assigns failed worker's tasks, …
- Other mangers just monitor passively and re-elect a leader if leader fails
- Services can be scaled up or down as needed
- Several Docker commands:
  - docker service : Manage services
  - docker swarm: Manage swarms
  - docker node: Manage nodes

DOCKER OVERLAY NETWORKING

- Nodes in a swarm may not be in the same LAN
- VXLAN is used to provide virtual overlay networking



DOCKER SECURITY

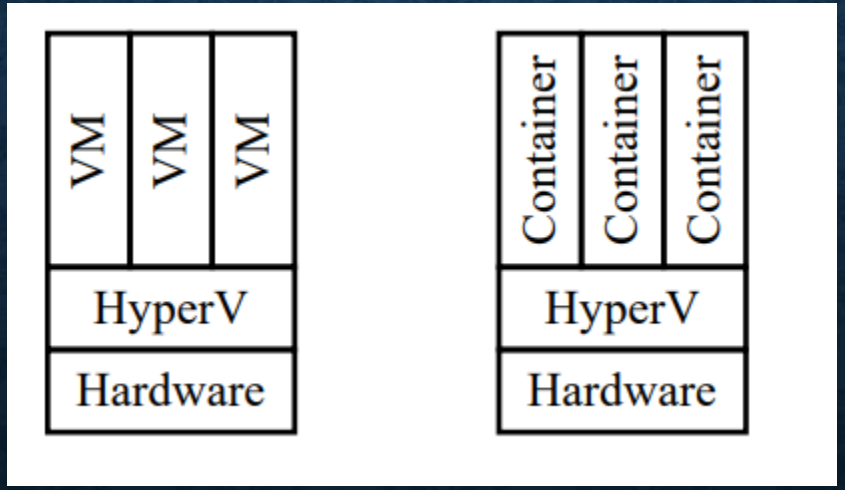
- All built-in security mechanisms in Linux are used and more
- Cryptographic node IDs
- Mutual Authentication
- Automatic Certificate Authority configuration
- Automatic Certificate Renewal on expiration
- Encrypted Cluster Store
- Encrypted Network traffic
- Signed images in Docker Content Trust (DCT)
- Docker Security Scanning detects vulnerabilities
- Docker secrets are stored in encrypted cluster store, encrypted transmission over network, and stored in in-memory file system when in use

KUBERNETES

- Open Source Container Orchestration alternative
- Original source released by Google
- Cloud Native Computing Foundation (CNCF) project in Linux Foundation
- Precursor to Swarms
- Facilities similar to Swarms
- A set of related containers is called a “Pod” A Pod runs on a single host.
- Swarm is called a “Cluster”

HYPER-V CONTAINERS

- Microsoft allows two kinds of containers:
- Windows Server Containers: Multiple containers on a single VM (like Docker containers)
- Hyper-V containers: Each container runs on its own VM ⇒ No need for a Linux



INTEL CLEAR CONTAINERS

- Started 2015 to address security concerns (Dirty COW) in containers
- Idea: Allow lightweight VMs using Intel Virtualization Technology
- Own lightweight OS and a dedicated kernel ⇒ Isolation of network, memory, and I/O
- Help by hardware enforced isolation
- No need for full VMs for containers
- Merged with HyperV to form Kata containers on Dec 5, 2017

KATA CONTAINERS

- Lightweight virtual machines
- Dedicated VMs to run one and only one container
- Combines "Intel Clear Containers" and "HyperV runV"
- Open source project under OpenStack Foundation
- Compatible with the OCI specs for Docker containers
- Compatible with CRI for Kubernetes
- Performance like containers, isolation and security like VMs
- Six Components: Agent, Runtime, Proxy, Shim, Kernel and QEMU 2.9
- Kubernetes will be extended to provision VMs (Kata Containers)
- OpenStack's VM orchestration engine (Nova) will be extended to handle containers
- Package once and run anywhere
  - VMware, Google, and Amazon are all moving towards this approach