

Containers, Dockers, and Kubernetes



Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@cse.wustl.edu

These slides and audio/video recordings of this class lecture are at:
<http://www.cse.wustl.edu/~jain/cse570-18/>



1. What is a Container and Why?
2. How Docker helps using containers
3. Docker Commands
4. Orchestration: Swarms and Kubernetes
5. Docker Networking and Security

Key Reference: N. Poulton, "Docker Deep Dive," Oct 2017, ISBN: 9781521822807 (Not a Safari Book)

Advantages of Virtualization

- ❑ Minimize hardware costs (CapEx)
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 (Lower OpEx)
Simplified provisioning/administration of hardware and software
- ❑ Scalability and Flexibility: Multiple operating systems



Ref: http://en.wikipedia.org/wiki/Platform_virtualization

Ref: K. Hess, A. Newman, "Practical Virtualization Solutions: Virtualization from the Trenches," Prentice Hall, 2009,

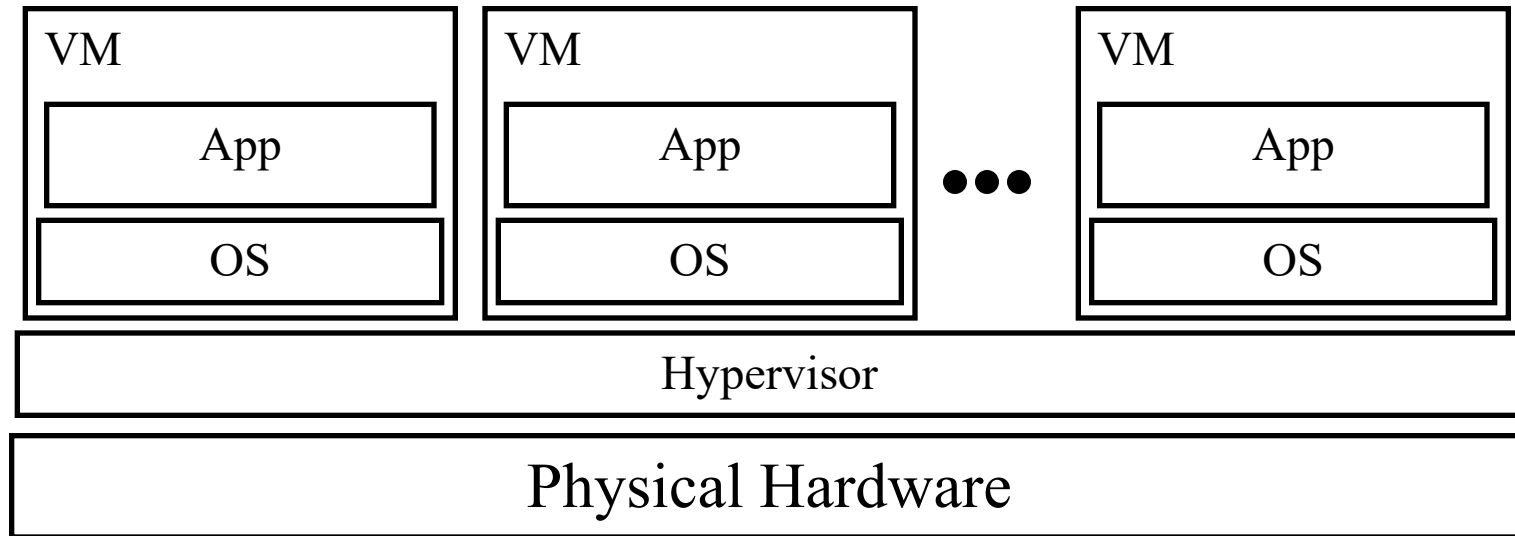
ISBN: 0137142978

Washington University in St. Louis

<http://www.cse.wustl.edu/~jain/cse570-18/>

©2018 Raj Jain

Problems of Virtualization



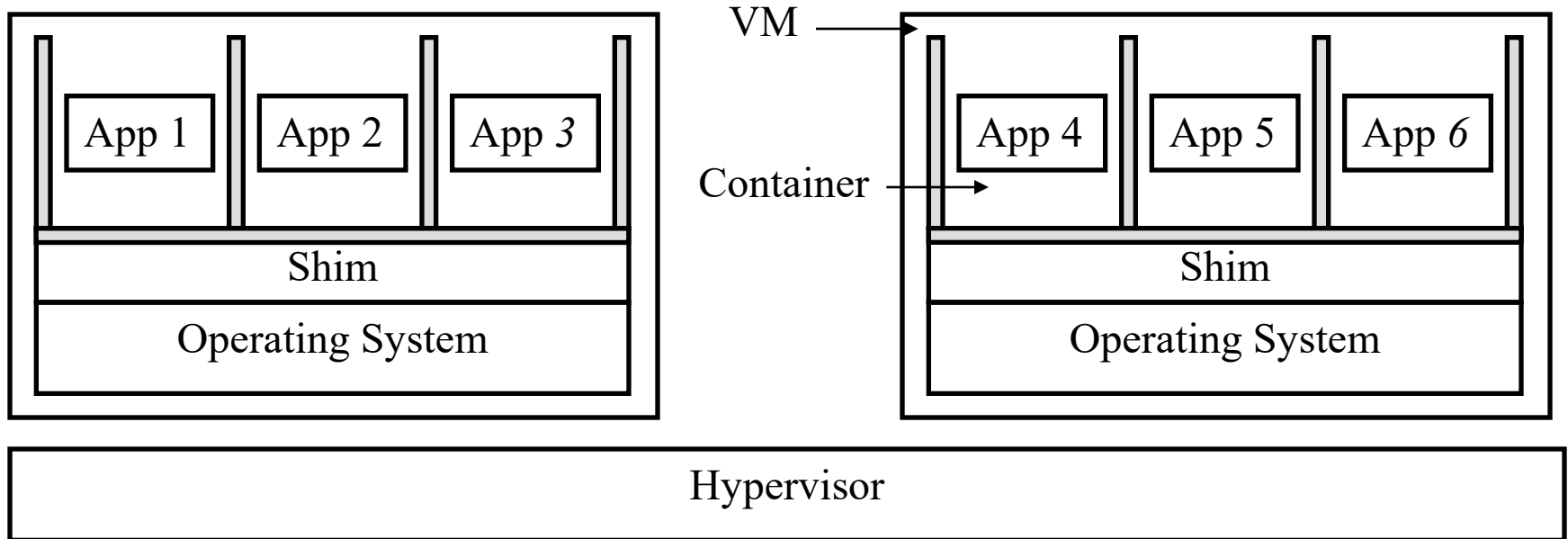
- ❑ Each VM requires an operating system (OS)
 - Each OS requires a license \Rightarrow CapEx
 - Each OS has its own compute and storage overhead
 - Needs maintenance, updates \Rightarrow OpEx
 - VM Tax = added CapEx + OpEx

Solution: Containers

- ❑ Run many apps in the same virtual machine
 - These apps share the OS and its overhead
 - But these apps can't interfere with each other
 - Can't access each other's resources without explicit permission
 - Like apartments in a complex
- ⇒ Containers



Containers



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

Containers (Cont)

- ❑ 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 \Rightarrow 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, “Show Process” (ps on Linux) 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

VM 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-Medium	Medium-High
OS Flexibility	Excellent	Poor
Management	Excellent	Evolving
Impact on Legacy application	Low-Medium	High

Ref: M. K. Weldon "The Future X Network: A Bell Labs Perspective," CRC Press, 2016, 476 pp., ISBN:9781498779142

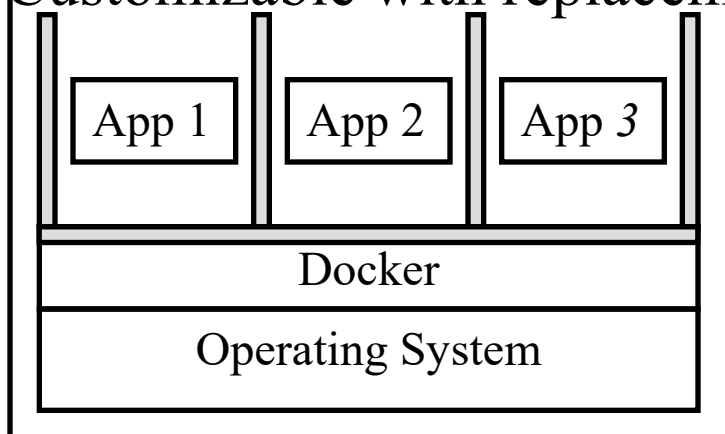
Washington University in St. Louis

<http://www.cse.wustl.edu/~jain/cse570-18/>

©2018 Raj Jain

Docker

- ❑ Provides the isolation among containers
- ❑ Helps them share the OS
- ❑ Docker = Dock worker \Rightarrow Manage containers
- ❑ Developed initially by Docker.com
- ❑ Downloadable for Linux, Windows, and Mac from [Docker.com](https://www.docker.com)
- ❑ Customizable with replacement modules from others



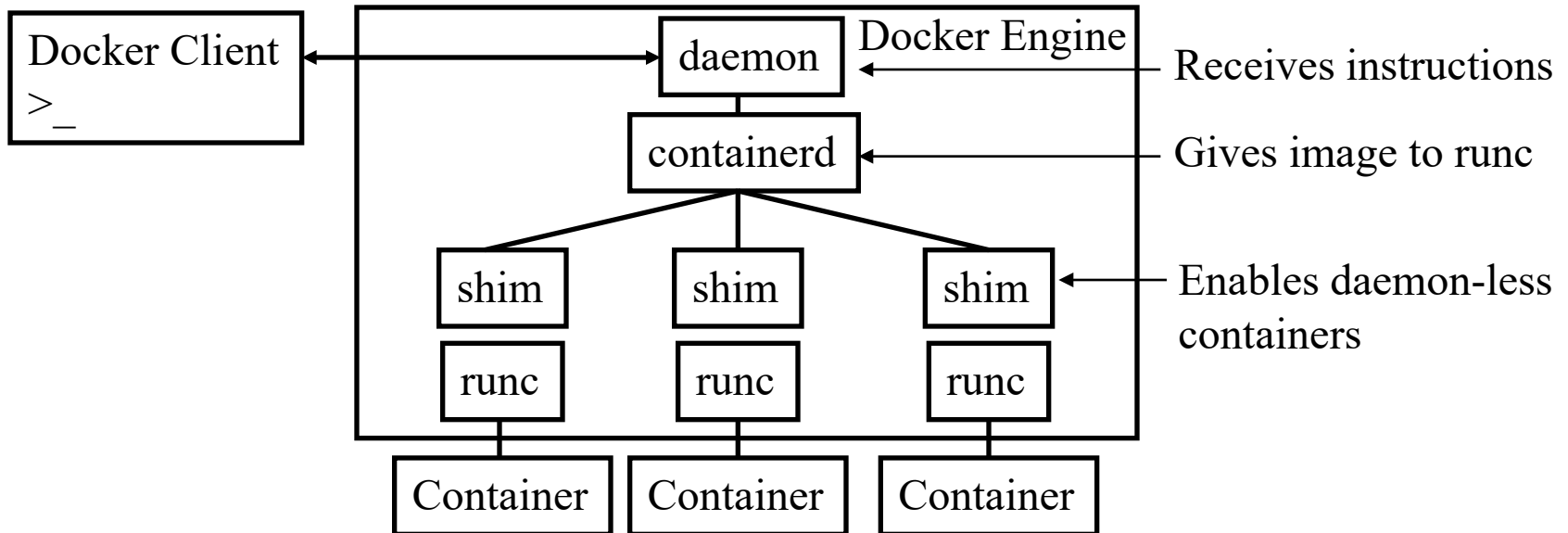
Docker

- ❑ Docker Engine: Runtime
- ❑ Two Editions:
 - Community Edition (CE): Free for experimentation
 - Enterprise Edition (EE): For deployment with paid support
- ❑ Written in “Go” programming language from Google
- ❑ Now open source project under mobyproject.org
<https://github.com/moby/moby>
- ❑ Download the community edition and explore

Ref: <https://golang.org/>

Docker Engine Components

- ❑ daemon: API and other features
- ❑ containerd: 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.



Ref: N. Poulton, "Docker Deep Dive," Oct 2017, ISBN: 9781521822807 (Not a Safari Book)

Washington University in St. Louis

<http://www.cse.wustl.edu/~jain/cse570-18/>

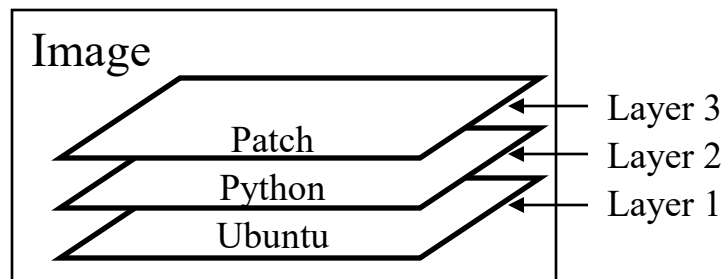
©2018 Raj Jain

Image Registries

- ❑ Containers are built from images and can be saved 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

Layers

- ❑ Each image has many layers
- ❑ 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 npm install	← Install application dependencies
EXPOSE 8080	← Open TCP Port 8080
ENTRYPOINT ["node", "./app.js"]	← Main application to run

RUN npm install	← Layer 4
Copy . /src	← Layer 3
RUN apk add ...	← Layer 2
FROM Alpine	← Layer 1

Note: WORKDIR, EXPOSE, ENTRYPOINT result in tags. Others in Layers.

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

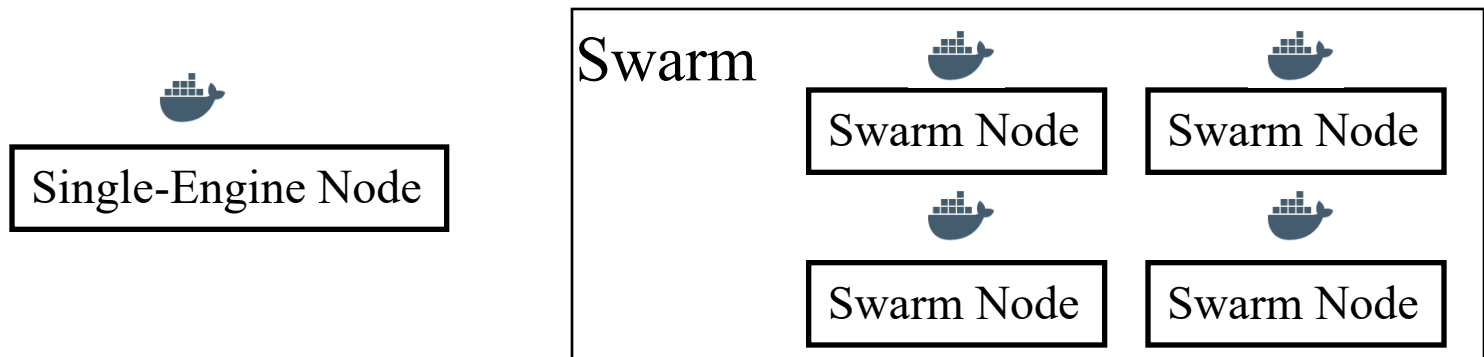
Open Container Initiative (OCI)

- ❑ A company called CoreOS defined alternative image format and container runtime API's
- ❑ Led to formation of OCI under Linux Foundation to govern container standards
 - OCI Image spec
 - OCI Runtime spec
- ❑ Everyone including Docker is now moving to OCI



Swarm

- ❑ 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)



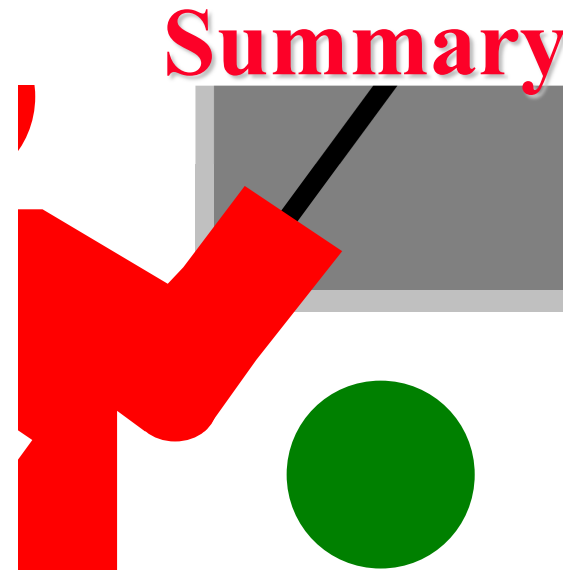
Swarms (Cont)

- ❑ The managers select a leader, who really keeps track of the swarm
- ❑ Assigns tasks, re-assigns failed worker's tasks, ...
- ❑ Other managers 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

Kubernetes

- ❑ Open Source Container Orchestration alternative
- ❑ Original source released by Google
- ❑ Cloud Native Computing Foundation (CNCF) project in Linux Foundation
- ❑ Pre-cursor 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”





- ❑ Virtual Machines provide scalability, mobility, and cost reduction but need OS which increase resource requirements
- ❑ Containers provide isolation on a single OS and are lightweight
- ❑ Docker allows managing containers
- ❑ Docker Swarm and Kubernetes allow orchestrating a large number of containers
- ❑ Docker provides overlay networking and security

Acronyms

❑ API	Application Programming Interface
❑ CapEx	Capital Expenditure
❑ CE	Community Edition
❑ CLI	Command Line Interface
❑ CNCF	Native Computing Foundation
❑ DCT	Docker Content Trust
❑ EE	Enterprise Edition
❑ ID	Identifier
❑ ISBN	International Standard Book Number
❑ LAN	Local Area Network
❑ OpEx	Operational Expenses
❑ OS	Operating System
❑ TCP	Transmission Control Protocol
❑ VM	Virtual Machine

References

- ❑ N. Poulton, "Docker Deep Dive," Oct 2017, ISBN: 9781521822807 (Not a Safari Book) **Highly Recommended.**
- ❑ Parminder Singh Kocher, "Microservices and Containers, First edition," Addison-Wesley Professional, April 2018, 304 pp., ISBN:978-0-13-459838-3 (Safari Book).
- ❑ Russ McKendrick; Pethuru Raj; Jeeva S. Chelladurai; Vinod Singh, "Docker Bootcamp," Packt Publishing, April 2017, 196 pp., ISBN:978-1-78728-698-6 (Safari Book).
- ❑ Russ McKendrick; Scott Gallagher, "Mastering Docker - Second Edition," Packt Publishing, July 2017, 392 pp., ISBN:978-1-78728-024-3 (Safari Book).
- ❑ Jeeva S. Chelladurai; Vinod Singh; Pethuru Raj, "Learning Docker - Second Edition," Packt Publishing, May 2017, 300 pp., ISBN:978-1-78646-292-3 (Safari Book).

Wikipedia Links

- ❑ [https://en.wikipedia.org/wiki/Docker_\(software\)](https://en.wikipedia.org/wiki/Docker_(software))
- ❑ https://en.wikipedia.org/wiki/Operating-system-level_virtualization
- ❑ <https://en.wikipedia.org/wiki/Kubernetes>
- ❑ <https://en.wikipedia.org/wiki/Microservices>
- ❑ <https://en.wikipedia.org/wiki/DevOps>
- ❑ <https://en.wikipedia.org/wiki/OpenShift>
- ❑ <https://en.wikipedia.org/wiki/LXC>