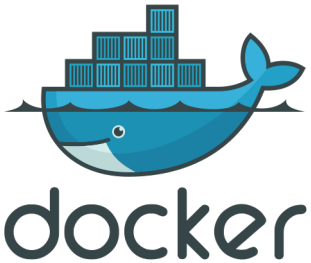
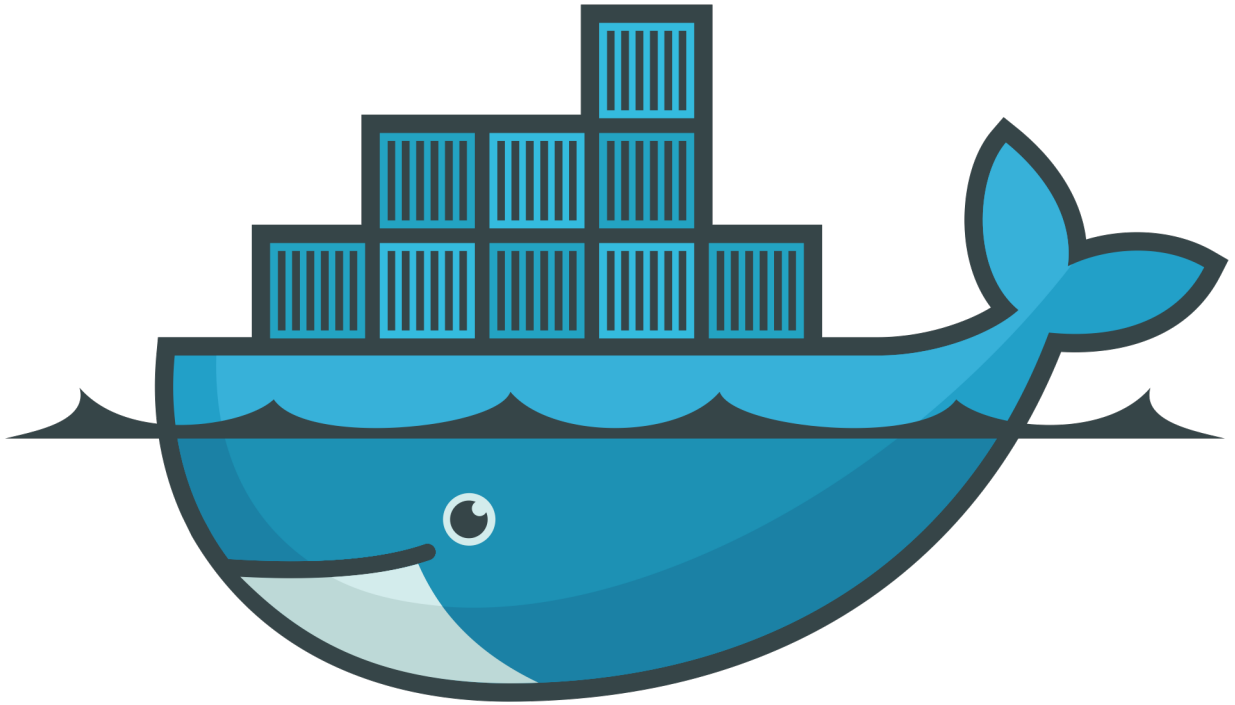
System Administration & Security

COMP 175 | Fall 2021 | University of the Pacific | Jeff Shafer

Docker

(Containerized Applications)





docker

Docker Overview

https://www.youtube.com/watch?v=eGz9DS-aIeY

Containers

Containers

- Operating System Level Virtualization
  - Container holds *executables* + *libraries* + *configuration files*
  - Everything a program or microservice needs to run!
- Modern examples
  - Docker - https://www.docker.com/ (released 2013)
  - Linux LXC - https://linuxcontainers.org/
  - Linux OpenVZ - https://openvz.org/
  - FreeBSD Jails
  - Solaris Containers

Not a new idea! These existed before virtual machines were the next big thing…

Container Features

- **Q:** What can I customize inside a container?
  - Unique users (including root users!)
  - Unique memory
  - Unique files
  - Unique applications (processes)
  - Unique network configuration
  - Unique system libraries

Container Features

- **Q:** What can I *not* customize in a container?
- **A:** Run a different OS than the host OS
  - If your host is Linux, your containers are Linux
  - If you host is FreeBSD, your containers are FreeBSD
- **A:** Change the kernel
  - It's the **same kernel** underlying all containers and the host OS

Container Implementation

- How does this work? (in Linux)
  - `cgroups` ("control groups") is kernel feature that isolates *hardware resources* used by a collection of processes
  - Resource = CPU, memory, disk I/O, network I/O, …
    - Can set limits on groups
    - Can prioritize groups
    - Can freeze/checkpoint/research groups

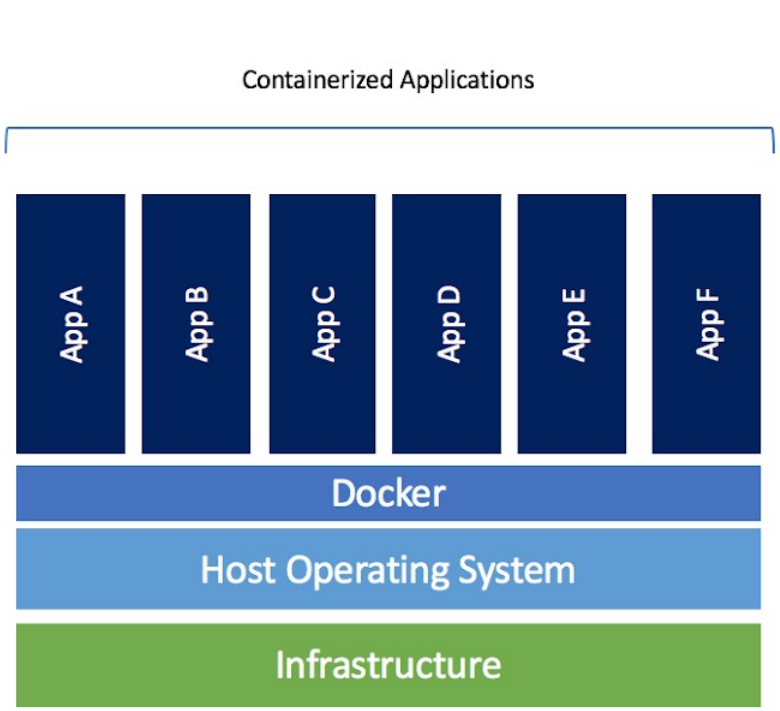
Container Implementation

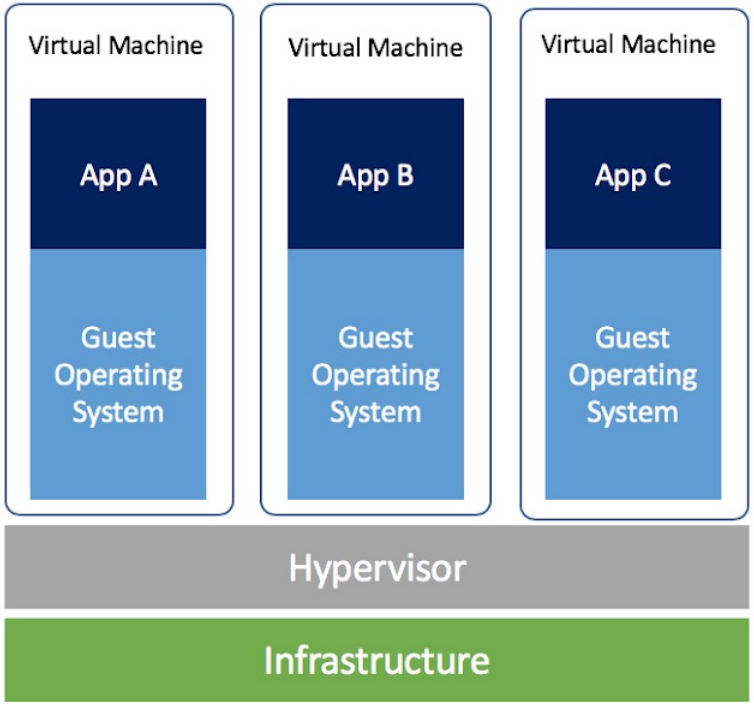
- How does this work? (in Linux)
  - `namespaces` is kernel feature that virtualizes *system resources* used by a collection of processes
  - Resource = Process IDs, hostname, user IDs, interprocess communication, filesystem mount points, network interface names…
- Container systems like LXC, OpenVZ, Docker rely on kernel features like `cgroups` and `namespaces`

Container Implementation - Filesystem

- **Layers**
- Original filesystem is marked *read-only*, and each container layers a new filesystem on top storing only modifications
  - Space efficient!
  - Easy to distribute! (no huge VM image)
- **Union mount** – combine multiple directories into single directory with unified view

Containers -vs- Virtual Machines





https://www.docker.com/blog/containers-replacing-virtual-machines/

Virtualization

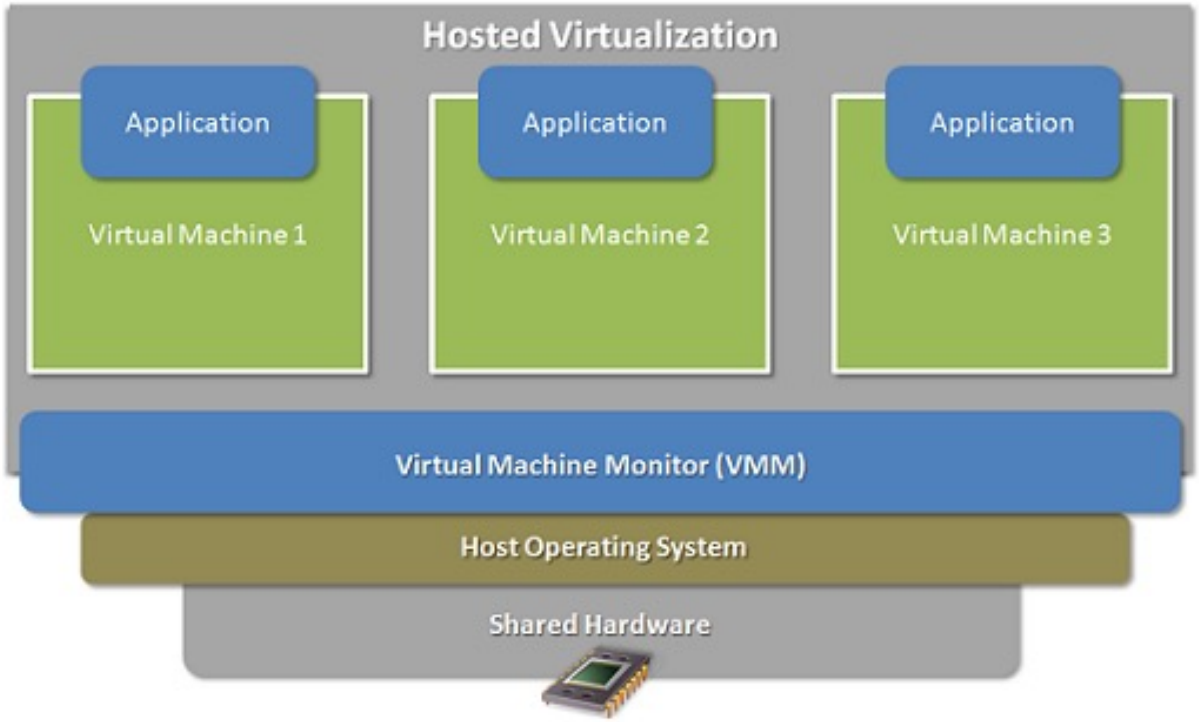
Terminology

- Hypervisor / Virtual Machine Monitor
  - Hardware + software that creates and runs virtual machines
  - Think “supervisor”, but where “hyper” is a stronger type of supervisor

Hosted Virtualization

Type 2 Hypervisor

- Run on top of commodity OS
- Examples
  - VMWare Workstation, Player, Fusion
  - Virtualbox
  - Parallels Desktop

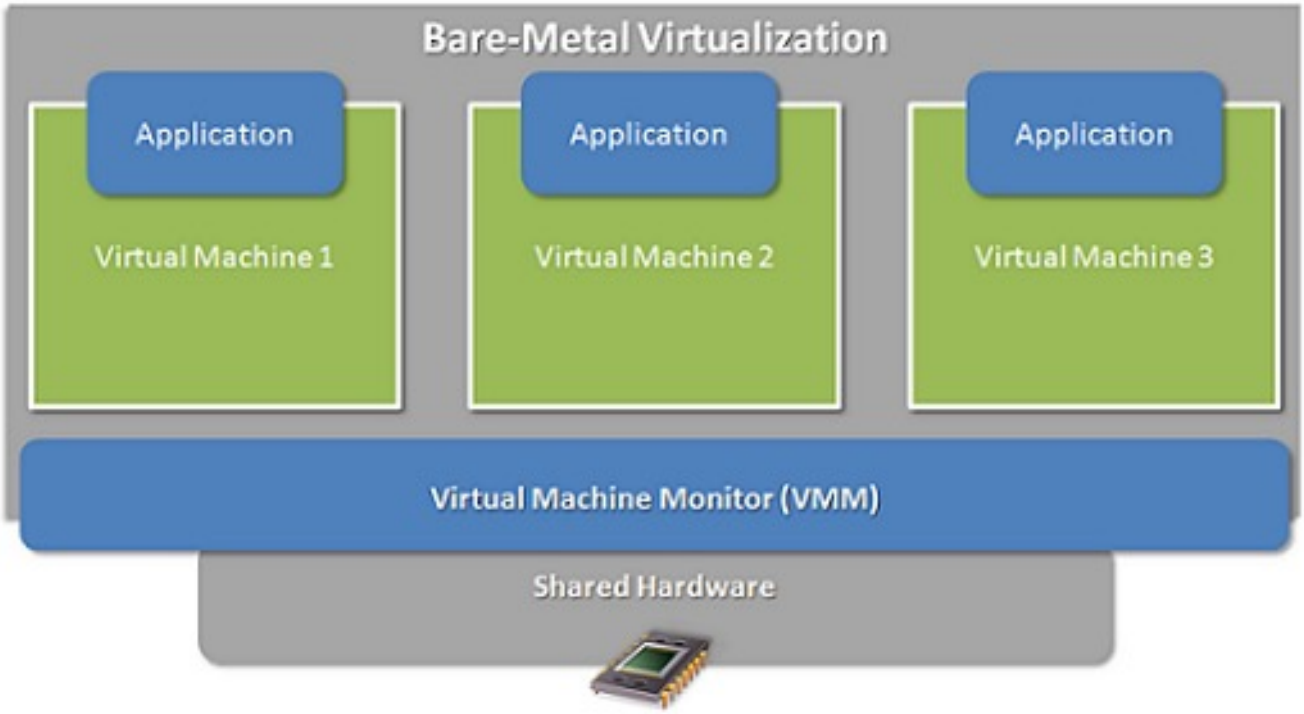


Native / Bare Metal Virtualization

***Type 1* Hypervisor**

More efficient, but not as easy to install.

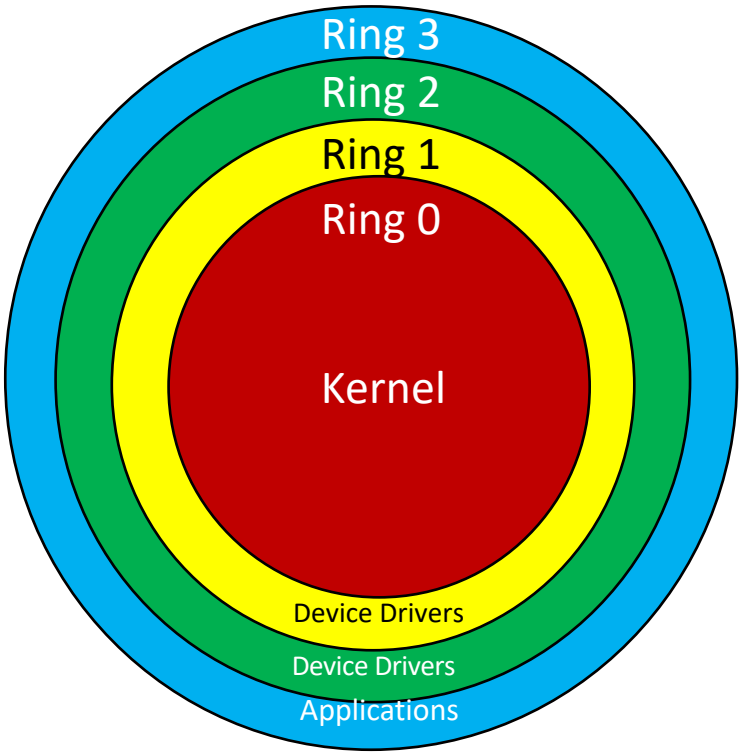
The virtual machine monitor acts like an operating system itself!

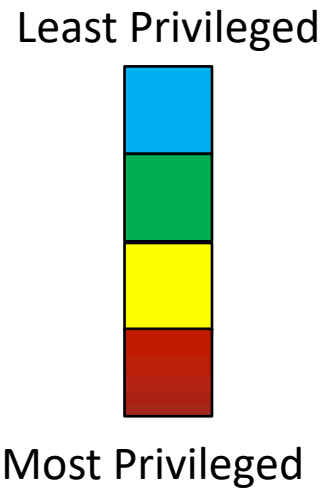


Examples: Xen, VMWare ESX/ESXi

Protection Rings

- Concept: Domains with varying privilege levels
  - Privilege = ability to do actions





Protection Rings

- Hardware protection mechanism
  - Ring 0 – most privileged
  - Ring *n* - least privileged
- Goals
  - Protect system integrity
  - Protect OS kernel from device drivers / services
  - Protect device drivers / services from applications
  - Etc…
- **Program can not call code of higher privilege directly**

x86 Virtualization

- **Q: How do we support Virtual Machines?**
  - Multiple operating systems running on single computer
  - Each OS kernel thinks they "own" the hardware – how do we share?

x86 Virtualization

- **A1: Software Emulation**
  - **Dynamically recompile guest OS** and emulate hardware instructions in software
  - Very complex / high overhead
  - Examples
    - QEMU
      - https://www.qemu.org/
    - PearPC (PowerPC emulator on x86)
      - http://pearpc.sourceforge.net/
    - Unicorn (ARM, 68K, MIPS, SPARC)
      - http://www.unicorn-engine.org/

x86 Virtualization

- **A2: Paravirtualization**
  - **Modify guest OS code**
    - Find all code that requires ring 0 permission
    - Emulate that code in hypervisor
    - Replace OS code with call to hypervisor
  - Pros: High performance
  - Cons:
    - Limited OS support
    - Must keep up with OS kernel development
  - Example
    - Xen VMM paravirtualization

x86 Virtualization

- **A3: Crazy software tricks**
  - Move the guest OS into ring 1
  - Set “traps” on all instructions that cannot run at ring 1 or require adjustments to share resources with other guests
    - Trap into virtual machine manager and emulate in software
  - Pros: Works with any OS
  - Cons:
    - Slow due to emulation of some instructions (better than full emulation)
    - Very complicated - http://www.virtualbox.org/manual/ch10.html#idp13729504
  - Example
    - VirtualBox (x86-only in software virtualization mode)

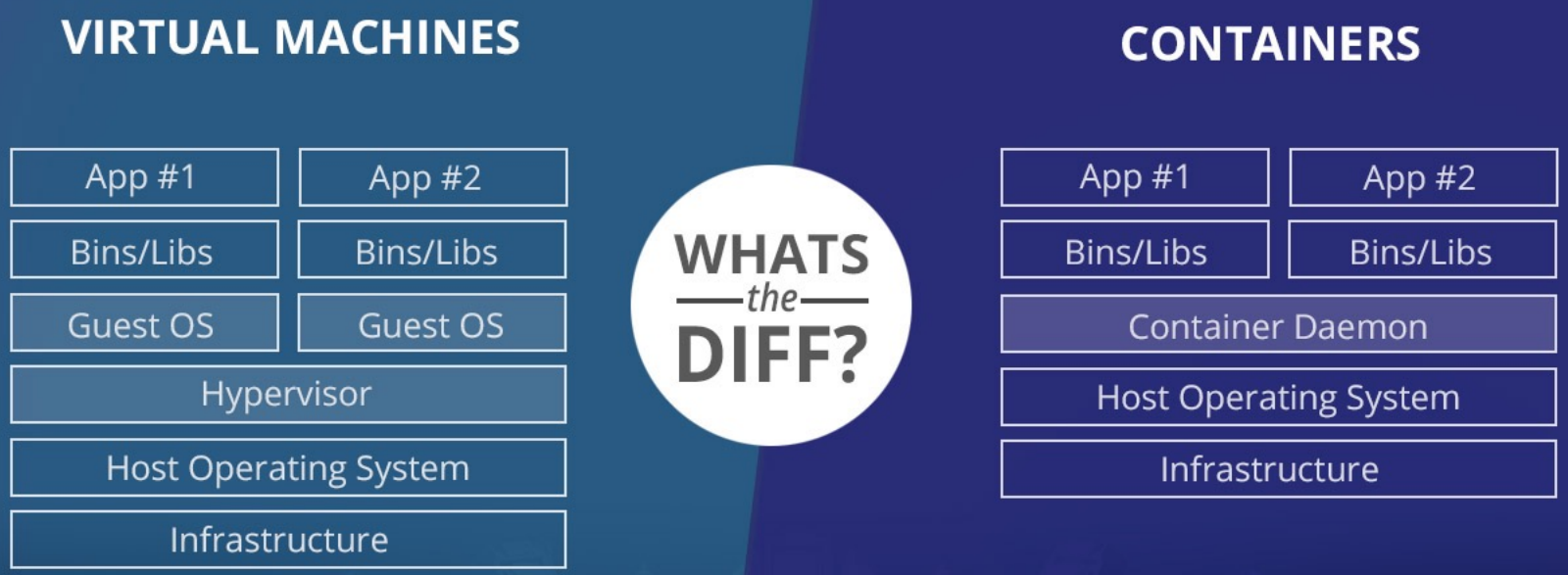
x86 Virtualization

- **A4: Hardware Virtualization**
  - Modify CPU to provide native virtualization support for un-modified operating systems
  - Ring -1 : Hypervisor Mode
  - Examples
    - Intel VT-x – Virtualization Technology for x86
    - AMD-V
  - New machine instructions that only work at ring -1

Hardware-Assisted Virtualization

- Guest OS has address space not shared with hypervisor ☺
- Guest OS kernel runs at ring 0 in non-root mode ☺
- Minimal software emulation ☺
- No need to re-write paravirtualization code to keep up with changes in guest OS kernel ☺

Virtual Machines -vs- Containers



https://www.backblaze.com/blog/vm-vs-containers/

Virtual Machines -vs- Containers

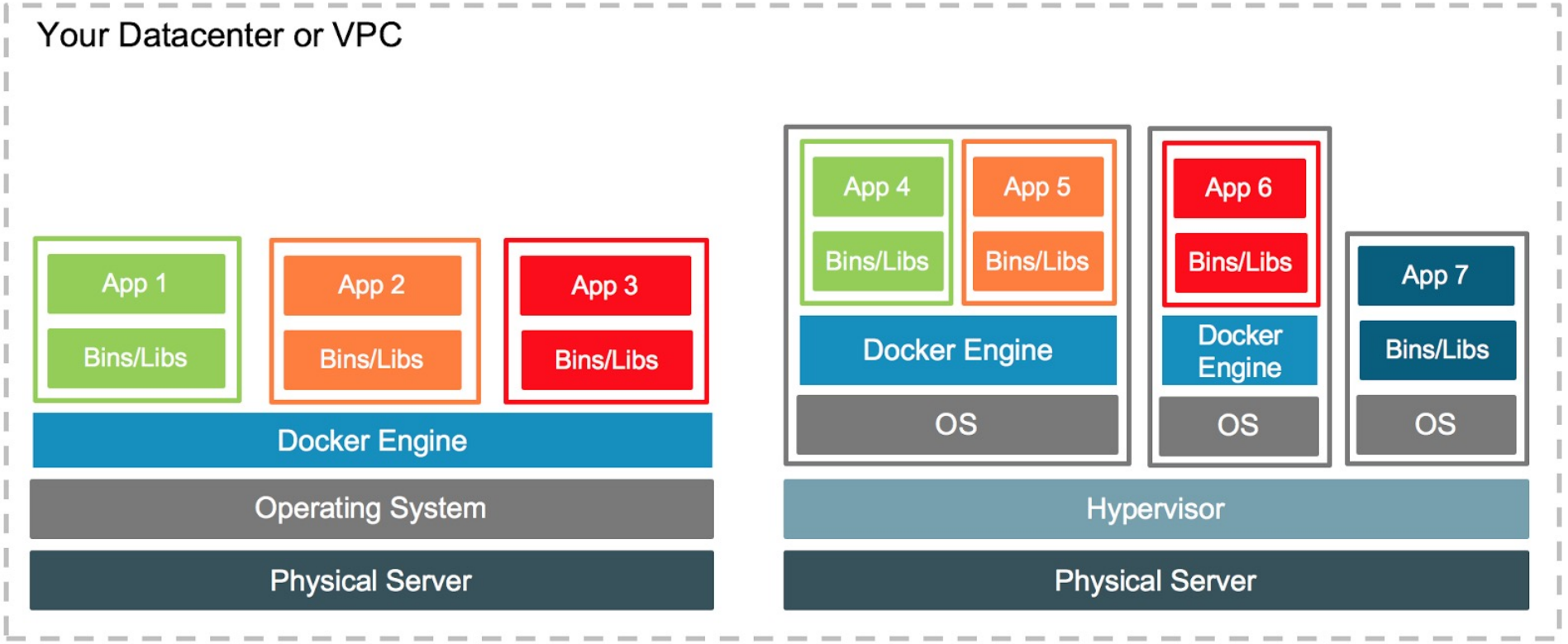
Server Virtualization -> Virtual Machines, *as is*
Operating System Virtualization -> Containers

- Overhead?
  - Containers have less overhead
    (VMs must virtualize the hardware + run full guest OS)
- Requirements?
  - Containers require no hardware support
- Flexibility?
  - VMs are more flexible (can run multiple guest operating systems of different types on the same host)

Virtual Machines -vs- Containers

- Security?
  - Containers are not fully isolated! ☹
    - Filesystems under /sys not virtualized
    - Devices not virtualized (/dev/mem, dev/sd*)
    - Kernel modules not virtualized
    - SELinux not virtualized
  - "Containers do not contain" https://opensource.com/business/14/7/docker-security-selinux
  - **Must apply same security practices as with applications running *outside* of a container**
    - Drop privileges as quickly as possible
    - Run services as non-root whenever possible
    - Treat root *within* container as if it was root *outside* of container

All of the Above: Containers and VMs



https://blog.docker.com/2016/04/containers-and-vms-together/

Wrap-Up

- Questions?
- Concerns?

- **This Week**
  - **Project Video** – 10/26
  - **Video Peer Review** – 11/2
  - **Lab 11** – Docker