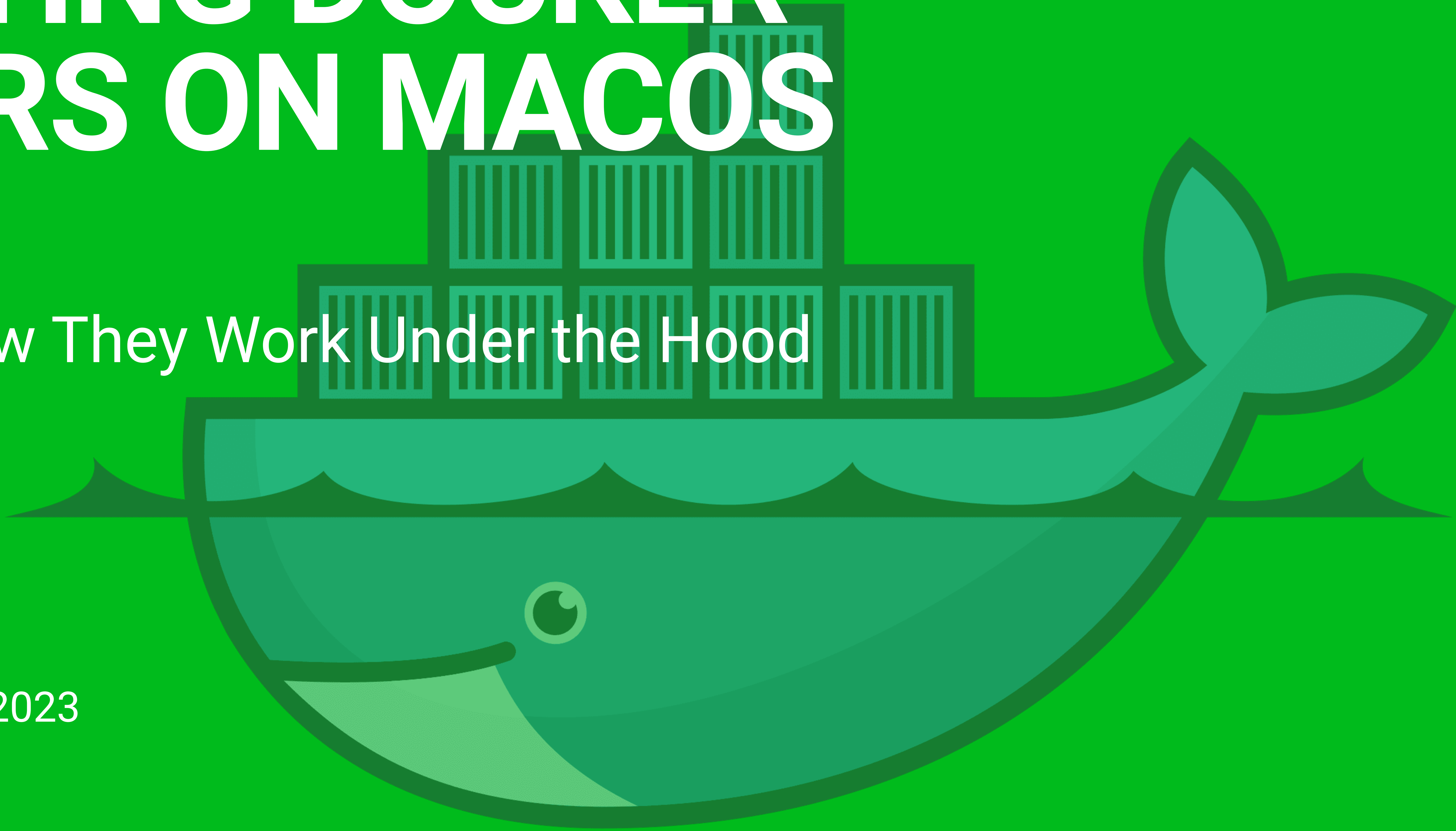
DEMYSTIFYING DOCKER CONTAINERS ON MACOS

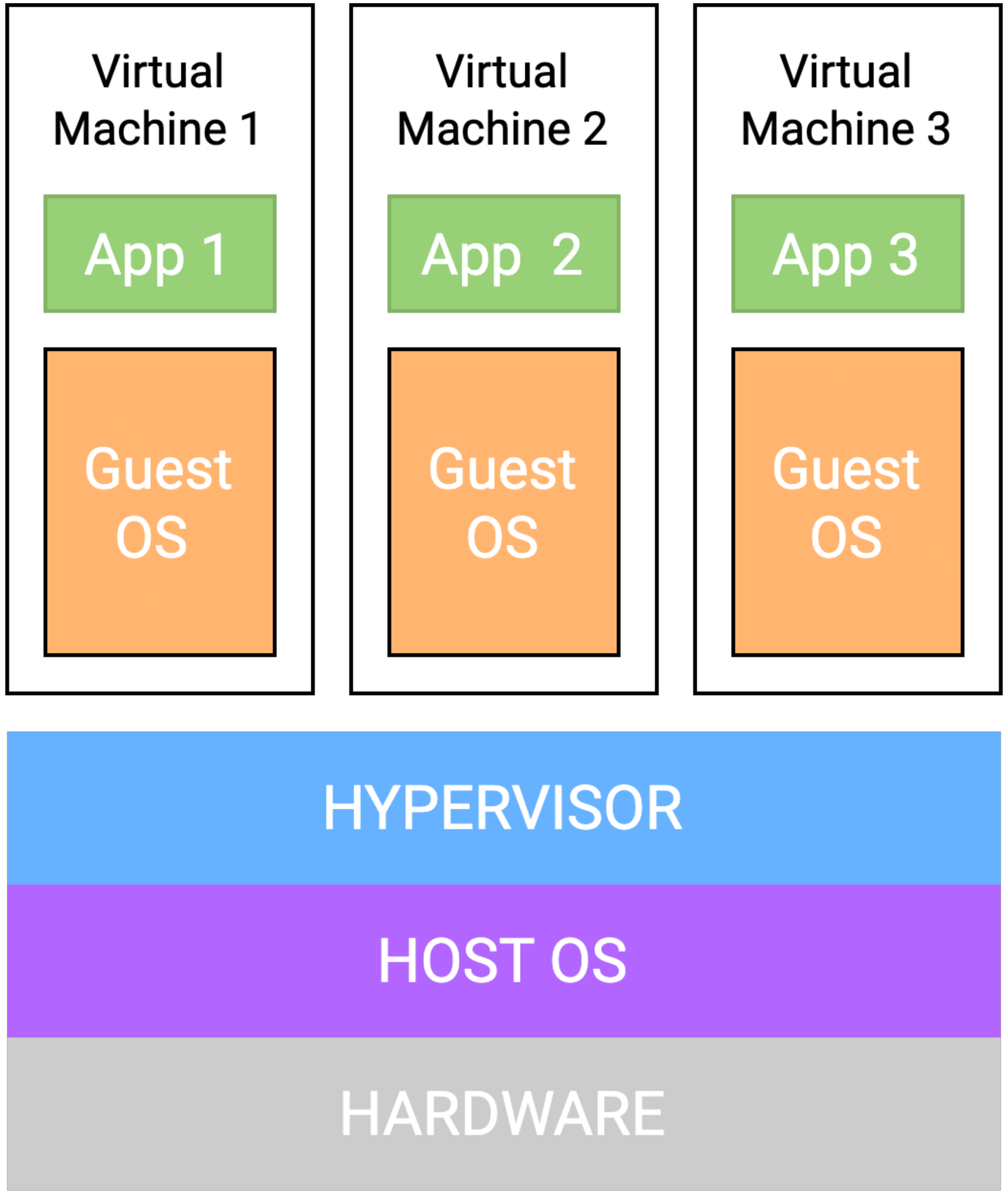
A Technical Look at How They Work Under the Hood



Artem Chernitsa, B20-AI-01, Spring 2023

Standard Virtual Machine Diagram

- Server underlying, some specific hardware
- Host OS aka Primary OS, interacting with hardware
- Hypervisor, first (on top of hardware) or second type, provides resources to VMs
- Guest OS, Applications, Binaries, Libraries



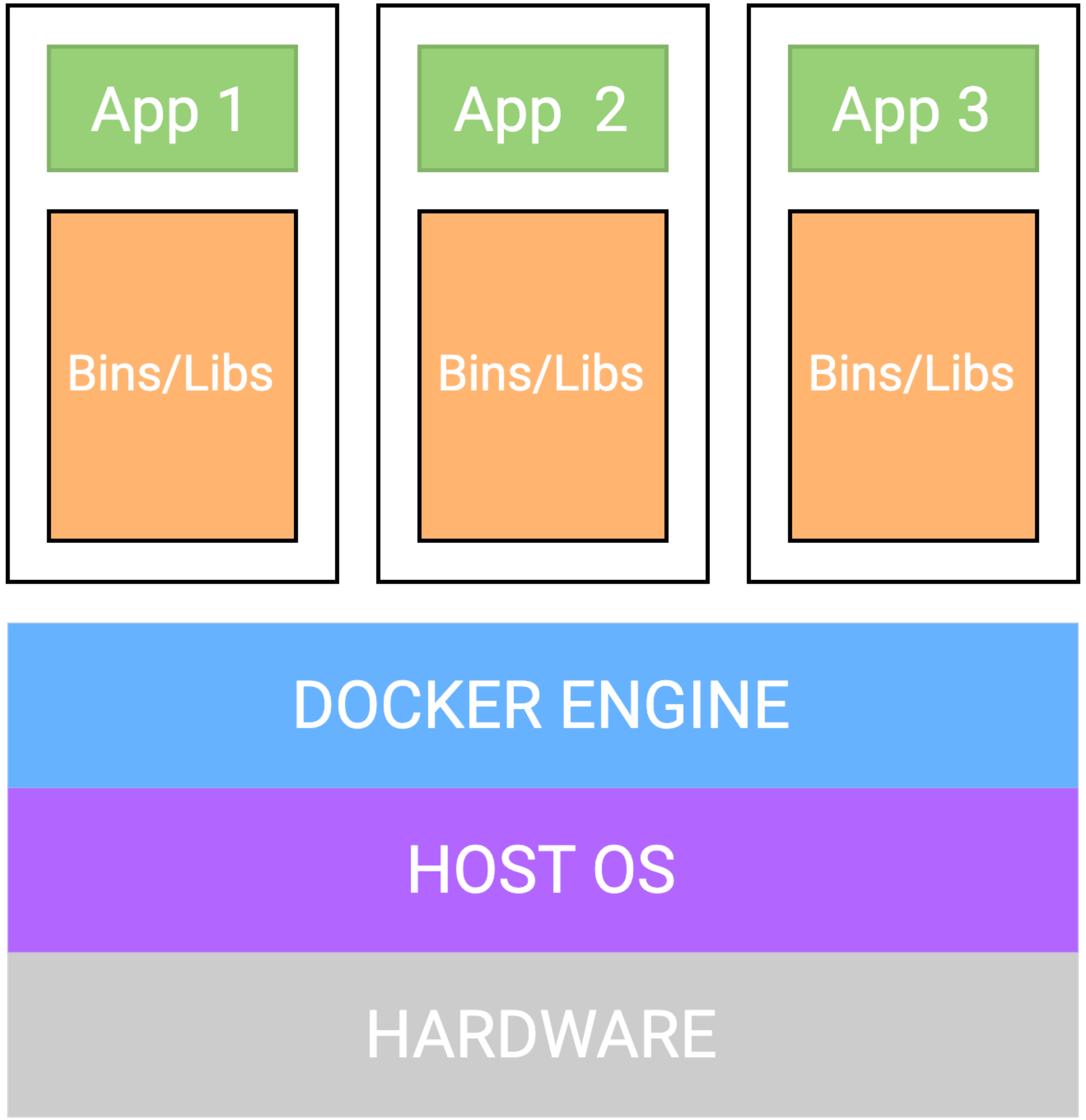
Docker aka Container Diagram

- Server underlying, some specific hardware

- Host OS aka Primary OS, interacting with hardware

- Docker Engine: Docker Daemon, Docker Client, REST API for Docker Client

- Applications, Binaries, Libraries



Docker Container

- Containers package up just the *user space*, and not the kernel or virtual hardware like a VM does.

- Each container gets its *own isolated user space* to allow multiple containers to run on a single host machine.

- Docker containers use Linux Kernel features like *namespaces* and *control groups* to create containers on top of an operating system.

Docker Container. Namespaces

- Namespaces provide containers with their own view of the underlying Linux system, limiting what the container can see and access.
  - NET – provides a container with its own view of the network stack of the system;
  - PID – gives containers their own scoped view of processes they can view and interact with, including an independent init (PID 1);
  - MNT – gives a container its own view of the “mounts” on the system

Docker Container. Namespaces (2)

- Namespaces provide containers with their own view of the underlying Linux system, limiting what the container can see and access.
  - UTS – UNIX Timesharing System. It allows a process to identify system identifiers (i.e. hostname, domainname, etc.);
  - IPC – responsible for isolating IPC resources between processes running inside each container;
  - USER – isolate users within each container. It functions by allowing containers to have a different view of the uid (user ID) and gid (group ID) ranges, as compared with the host system;

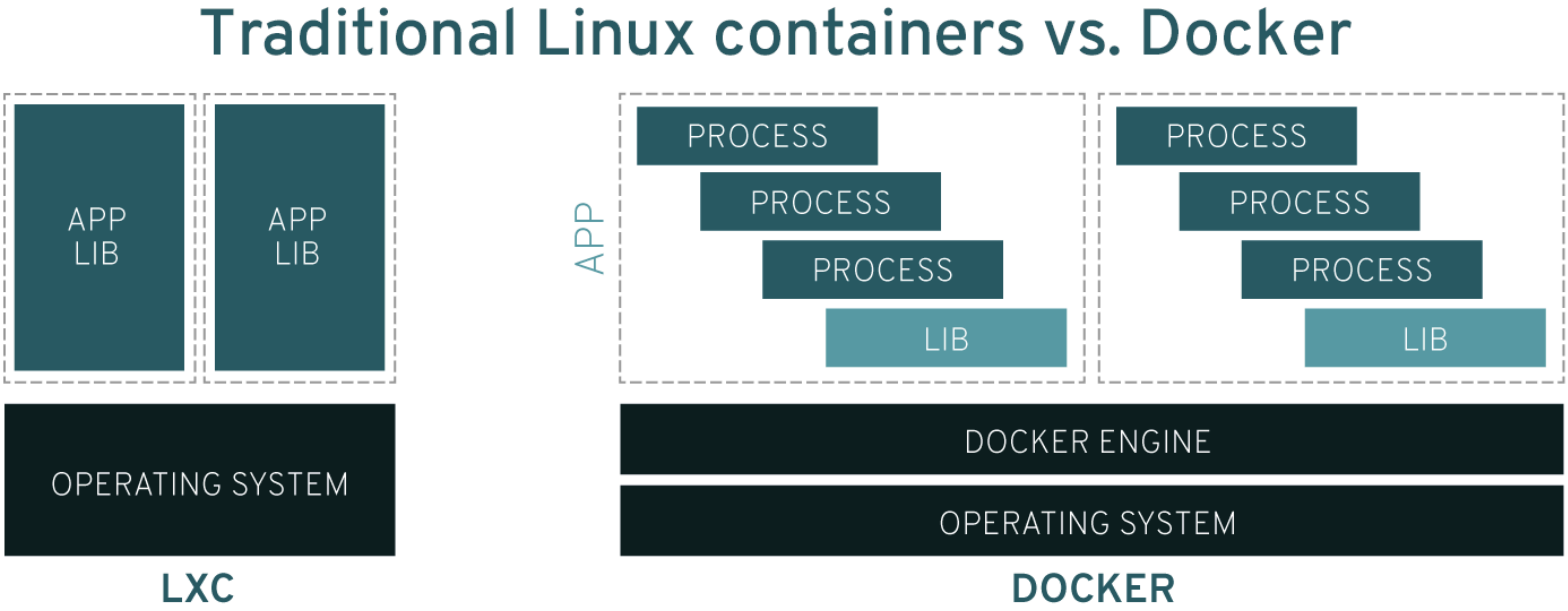
Docker Container. Control Groups (cgroups)

- Control groups (also called cgroups) is a Linux kernel feature that isolates, prioritizes, and accounts for the resource usage (CPU, memory, disk I/O, network, etc.) of a set of processes. cgroup ensures that Docker containers only use the resources they need — and, if needed, set up limits to what resources a container can use.

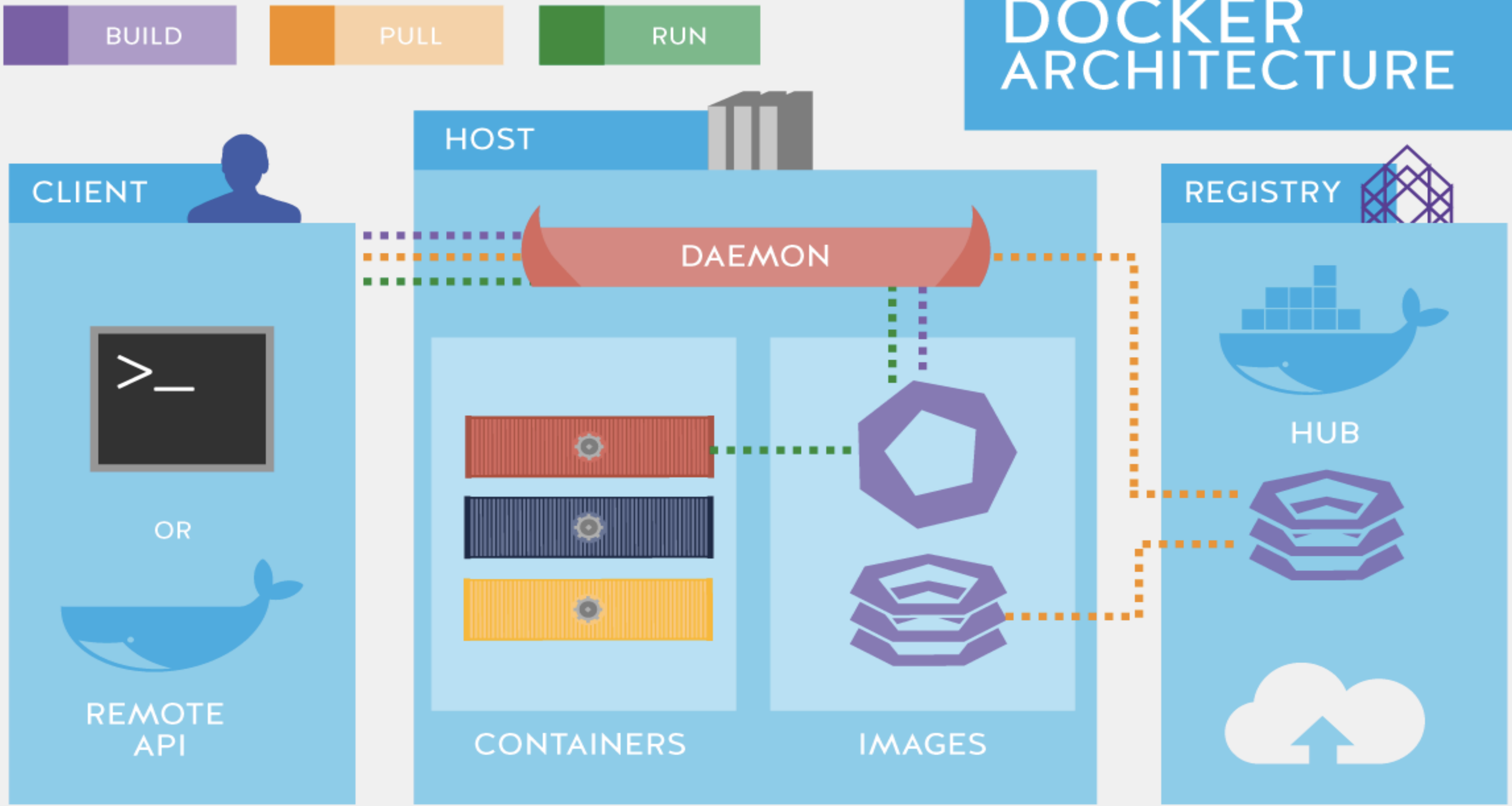
Docker Container. Union File System (UFS)

- Docker uses Union File Systems to build up an image. Union File System as a stackable file system, meaning files and directories of separate file systems (branches) can be transparently overlaid to form a single file system.
- “Copy-on-write” system. The contents of directories which have the same path within the overlaid branches are seen as a single merged directory

Docker Container vs LXC



DOCKER ARCHITECTURE



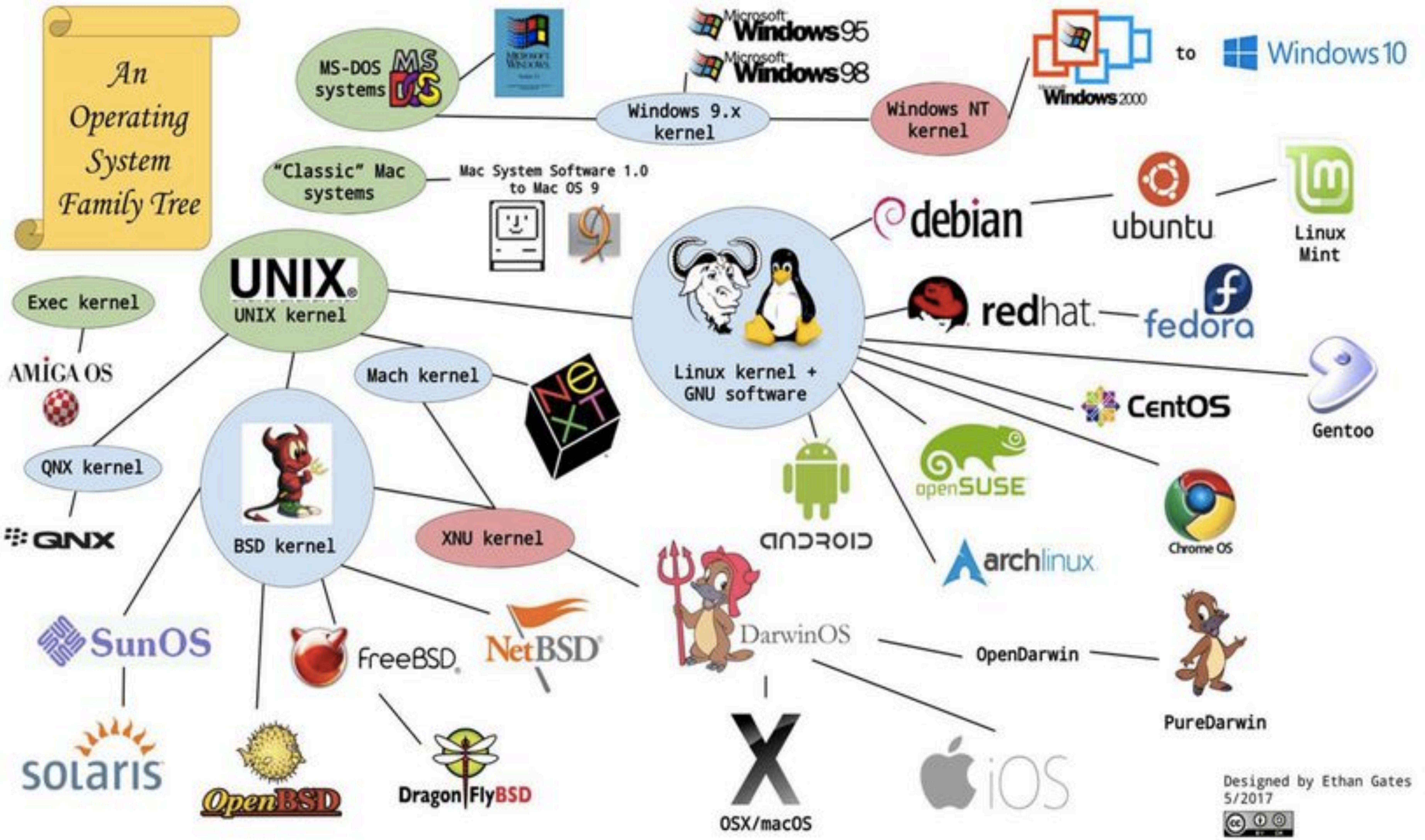
NORDICAPIS.COM

So, what about macOS?

So, what about macOS?

Here is a trick!

OS Family Tree



An Operating System Family Tree

MS-DOS systems

Windows 9.x kernel

Microsoft Windows 95

Microsoft Windows 98

Windows NT kernel

Windows 2000

to

Windows 10

"Classic" Mac systems

Mac System Software 1.0 to Mac OS 9

UNIX

UNIX kernel

Exec kernel

AMIGA OS

QNX kernel

QNX

Mach kernel

BSD kernel

XNU kernel

Linux kernel + GNU software

debian

ubuntu

Linux Mint

redhat

fedora

Gentoo

CentOS

openSUSE

android

archlinux

Chrome OS

DarwinOS

OpenDarwin

PureDarwin

OSX/macOS

iOS

SunOS

solaris

OpenBSD

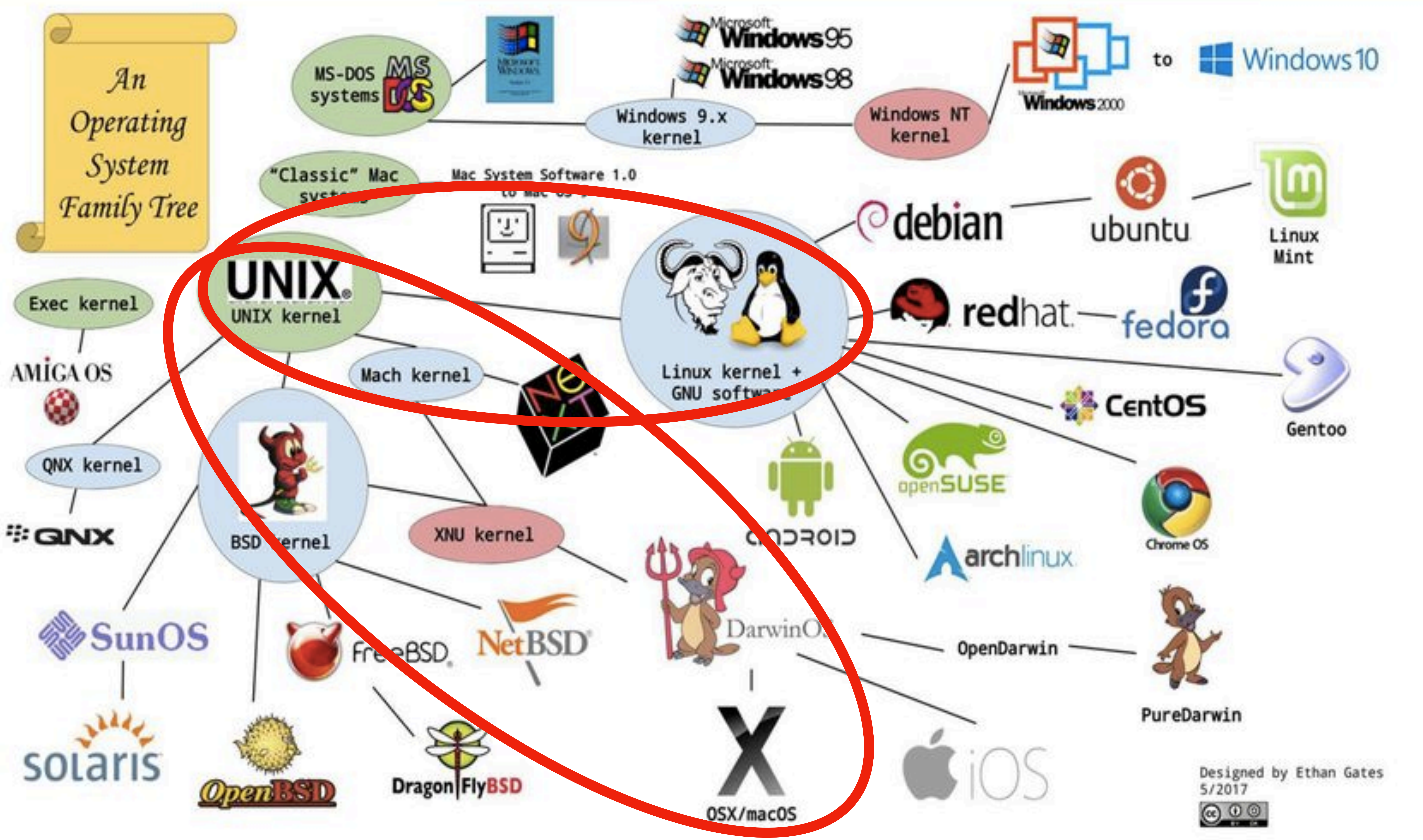
FreeBSD

NetBSD

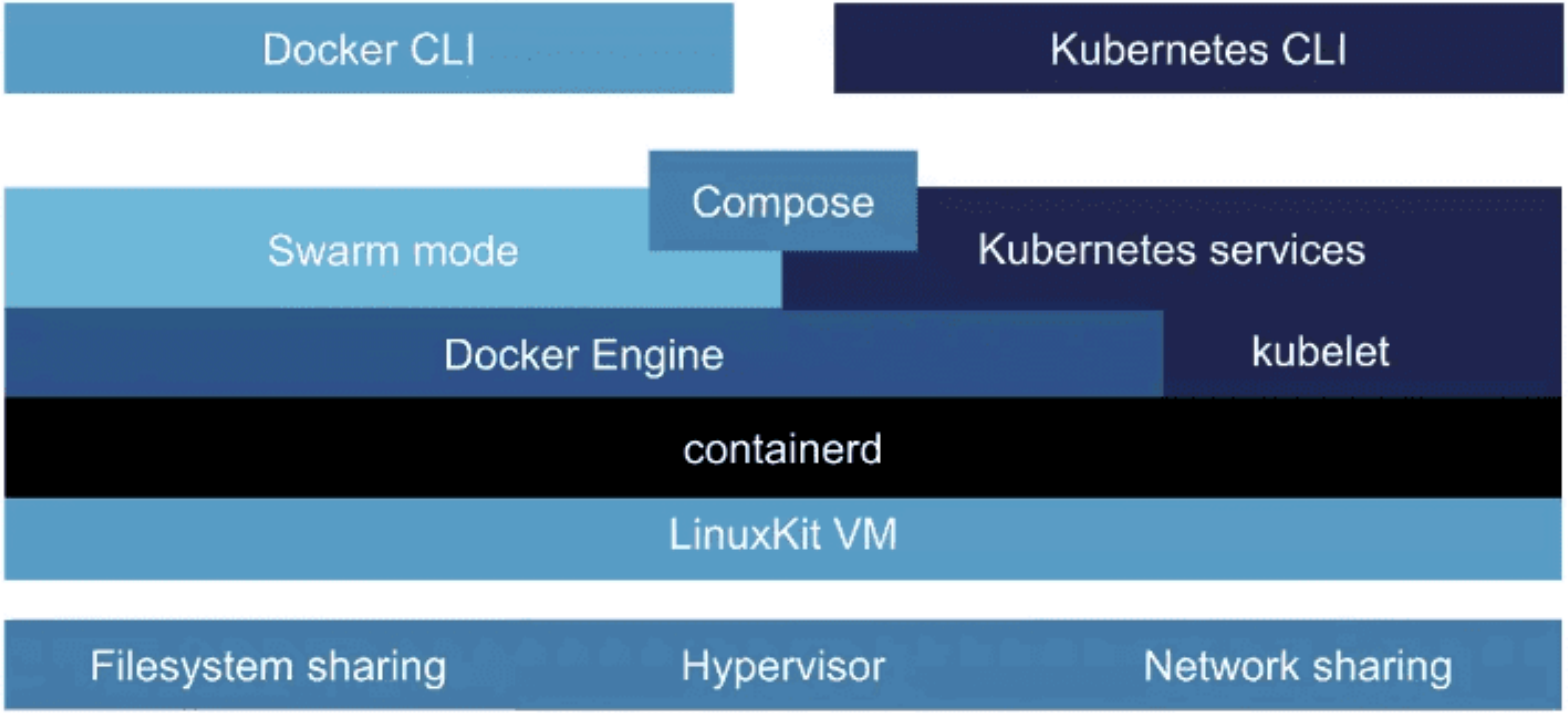
DragonFlyBSD

Designed by Ethan Gates 5/2017

OS Family Tree



Docker on macOS



Docker CLI

Kubernetes CLI

Compose

Swarm mode

Kubernetes services

Docker Engine

kubelet

containerd

LinuxKit VM

Filesystem sharing

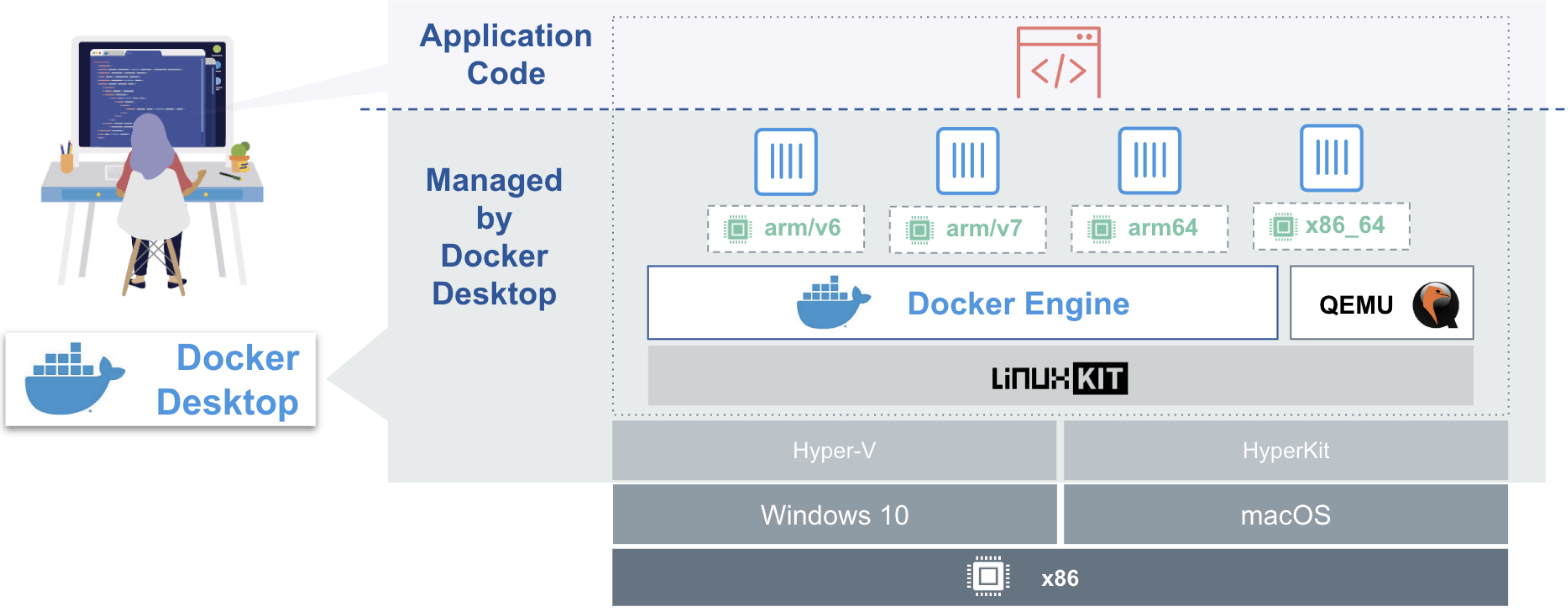
Hypervisor

Network sharing

Docker on macOS (2)

- Docker Engine needs a *Linux Kernel*;
- Instead Docker CLI and docker-compose are *native binaries;*

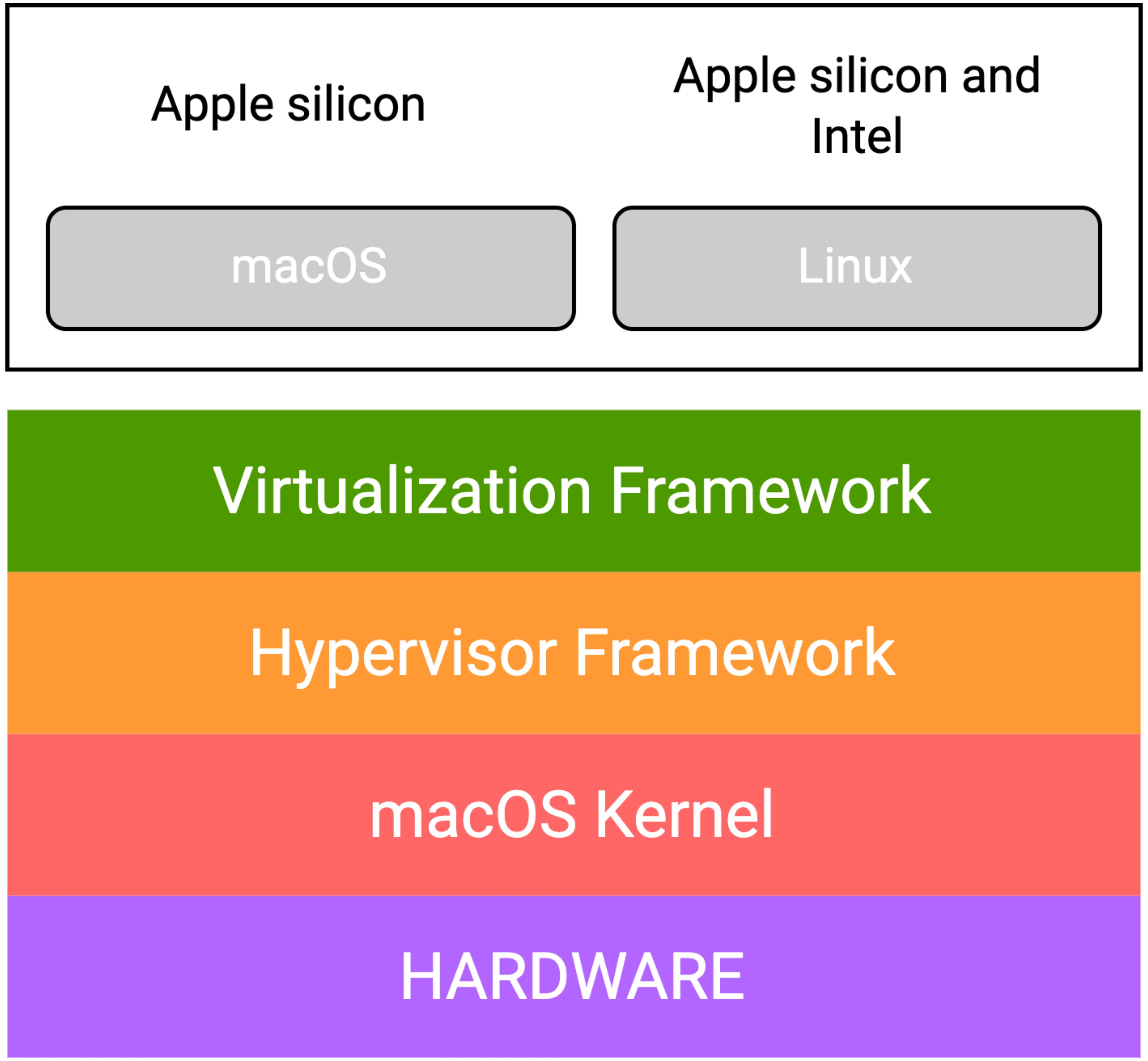
Docker Desktop



Docker Desktop

- It used *HyperKit* as Hypervisor, but for now it uses new *Virtualization Framework* instead;
- As Filesystem Sharing it uses either *gRPC FUSE*, *VirtioFS* (newer one) or *OSXFS* (deprecated).
- Networking based on *VPNKit*.

macOS Virtualization



Apple silicon

Apple silicon and Intel

macOS

Linux

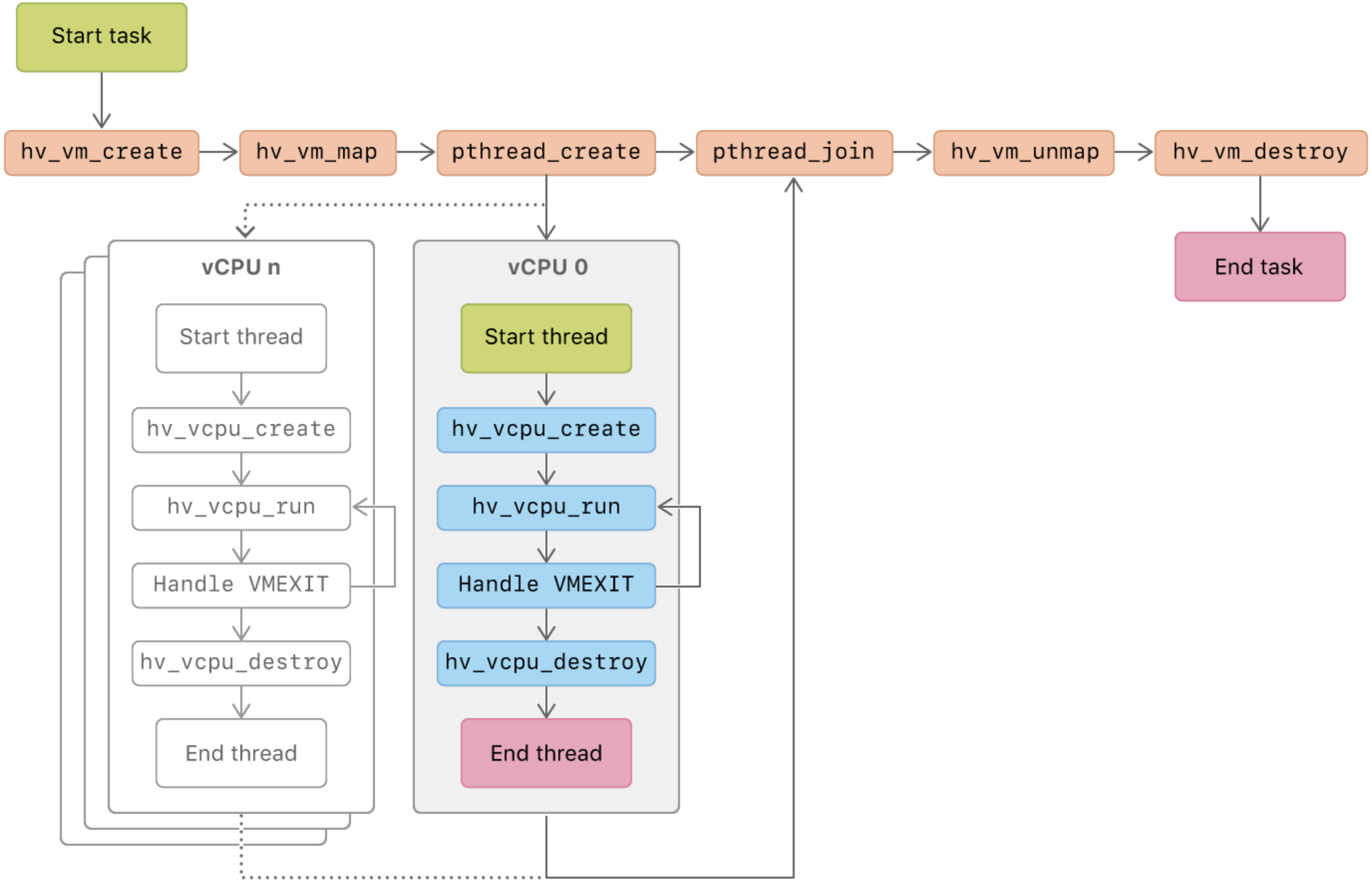
Virtualization Framework

Hypervisor Framework

macOS Kernel

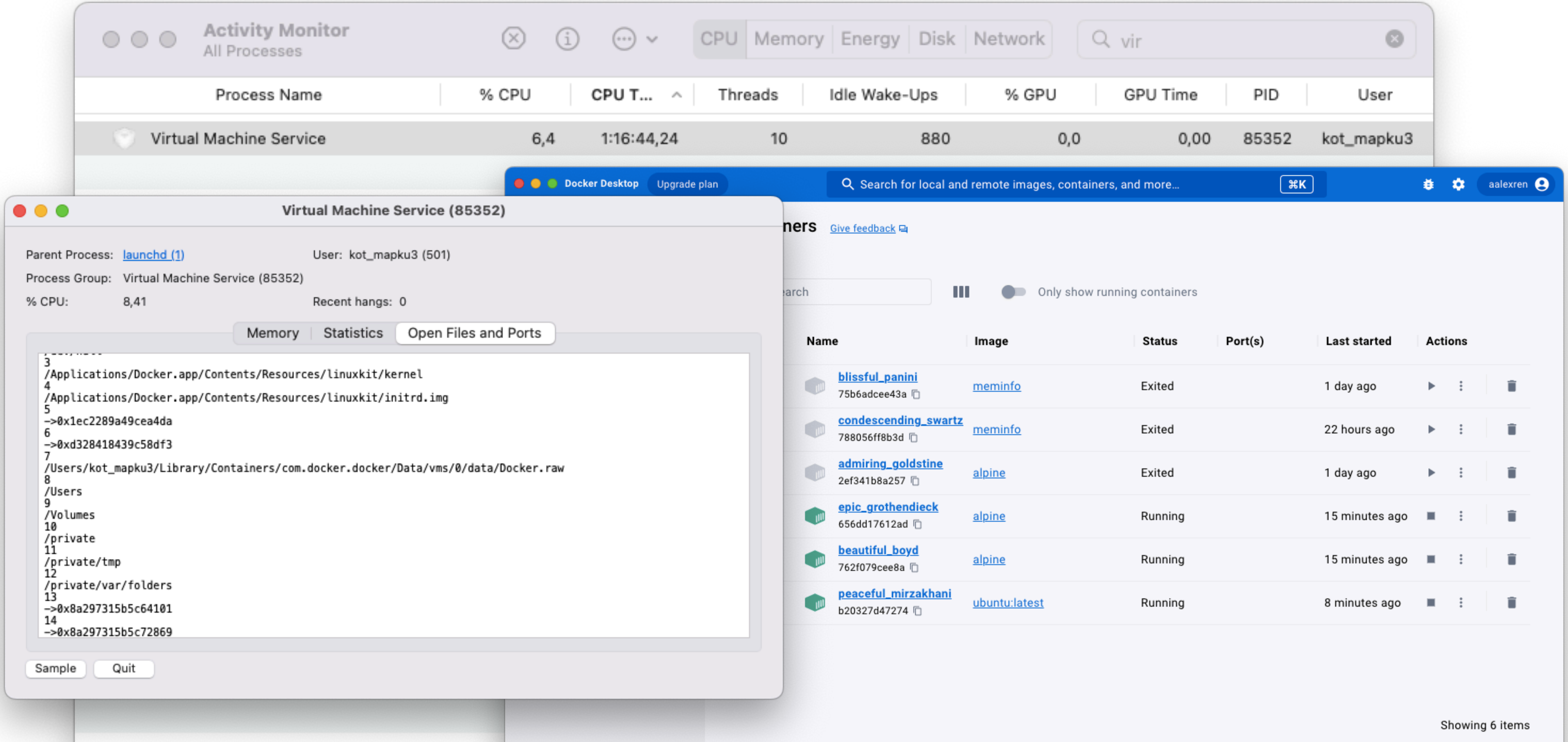
HARDWARE

Hypervisor Framework. VM Life Cycle



Many people do not realize that containers are really Linux. As such, Linux containers cannot run natively on macOS.

One VM per container?



Process Name	% CPU	CPU T...	Threads	Idle Wake-Ups	% GPU	GPU Time	PID	User
Virtual Machine Service	6,4	1:16:44,24	10	880	0,0	0,00	85352	kot_mapku3

Virtual Machine Service (85352)

Parent Process: launchd (1)
User: kot_mapku3 (501)
Process Group: Virtual Machine Service (85352)
% CPU: 8,41
Recent hangs: 0

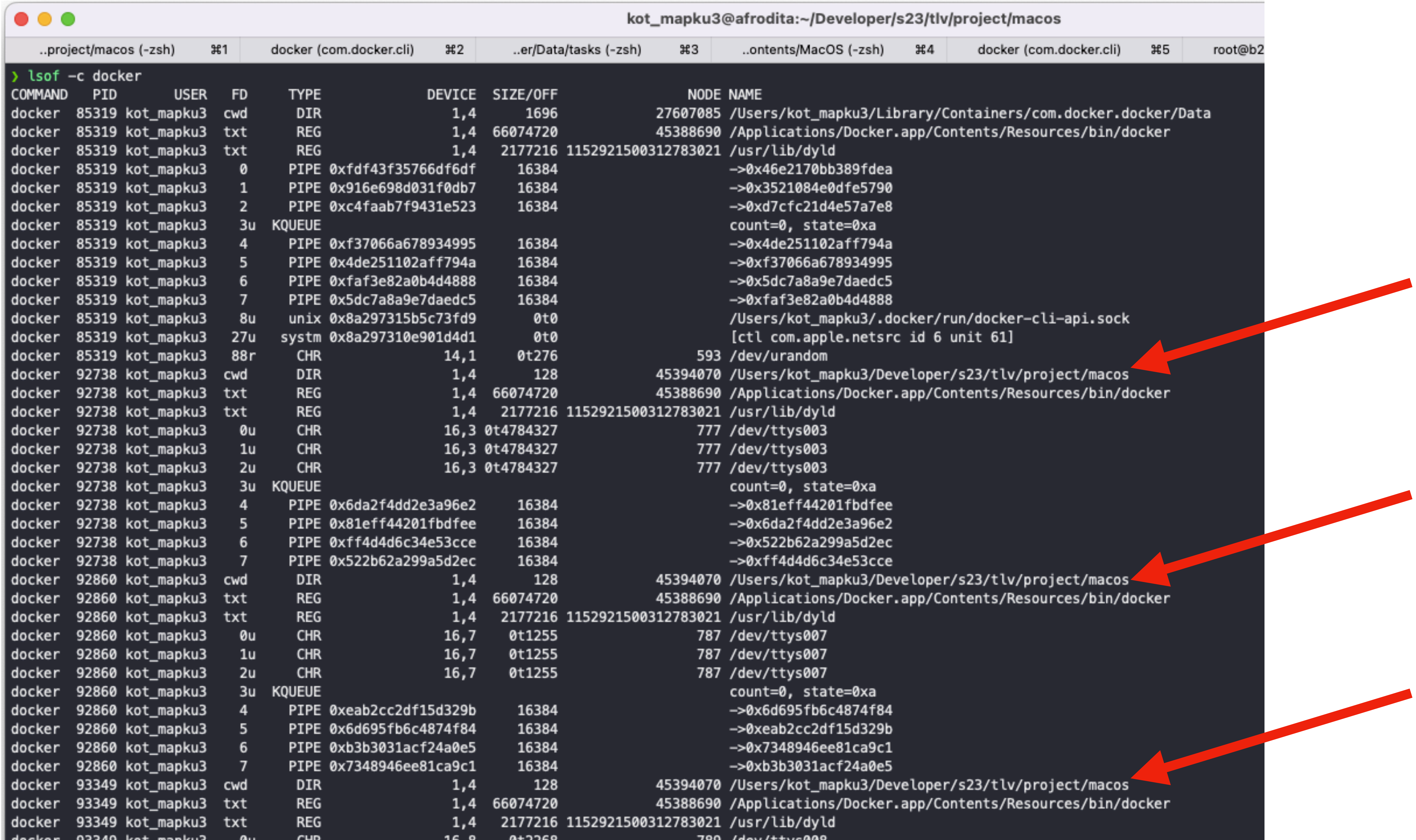
```
3
/Applications/Docker.app/Contents/Resources/linuxkit/kernel
4
/Applications/Docker.app/Contents/Resources/linuxkit/initrd.img
5
->0x1ec2289a49cea4da
6
->0xd328418439c58df3
7
/Users/kot_mapku3/Library/Containers/com.docker.docker/Data/vms/0/data/Docker.raw
8
/Users
9
/Volumes
10
/private
11
/private/tmp
12
/private/var/folders
13
->0x8a297315b5c64101
14
->0x8a297315b5c72869
```

Sample
Quit
Docker Desktop
Upgrade plan
Search for local and remote images, containers, and more...
⌘K
aalexren
ners
Give feedback
arch
Only show running containers

Name	Image	Status	Port(s)	Last started	Actions
blissful_panini 75b6adcee43a	meminfo	Exited		1 day ago	
condescending_swartz 788056ff8b3d	meminfo	Exited		22 hours ago	
admiring_goldstine 2ef341b8a257	alpine	Exited		1 day ago	
epic_grothendieck 656dd17612ad	alpine	Running		15 minutes ago	
beautiful_boyd 762f079cee8a	alpine	Running		15 minutes ago	
peaceful_mirzakhani b20327d47274	ubuntu:latest	Running		8 minutes ago	

Showing 6 items

One VM per container? (2)


kot_mapku3@afrodita:~/Developer/s23/tlv/project/macos
..project/macos (-zsh) ⌘1
docker (com.docker.cli) ⌘2
..er/Data/tasks (-zsh) ⌘3
..ontents/MacOS (-zsh) ⌘4
docker (com.docker.cli) ⌘5
root@b2

```
❯ lsof -c docker
COMMAND   PID      USER   FD     TYPE             DEVICE  SIZE/OFF                NODE NAME
docker  85319 kot_mapku3  cwd     DIR                1,4      1696            27607085 /Users/kot_mapku3/Library/Containers/com.docker.docker/Data
docker  85319 kot_mapku3  txt     REG                1,4  66074720            45388690 /Applications/Docker.app/Contents/Resources/bin/docker
docker  85319 kot_mapku3  txt     REG                1,4   2177216 1152921500312783021 /usr/lib/dyld
docker  85319 kot_mapku3    0    PIPE 0xfdf43f35766df6df     16384                     ->0x46e2170bb389fdea
docker  85319 kot_mapku3    1    PIPE 0x916e698d031f0db7     16384                     ->0x3521084e0dfe5790
docker  85319 kot_mapku3    2    PIPE 0xc4faab7f9431e523     16384                     ->0xd7cfc21d4e57a7e8
docker  85319 kot_mapku3   3u  KQUEUE                                                  count=0, state=0xa
docker  85319 kot_mapku3    4    PIPE 0xf37066a678934995     16384                     ->0x4de251102aff794a
docker  85319 kot_mapku3    5    PIPE 0x4de251102aff794a     16384                     ->0xf37066a678934995
docker  85319 kot_mapku3    6    PIPE 0xfaf3e82a0b4d4888     16384                     ->0x5dc7a8a9e7daedc5
docker  85319 kot_mapku3    7    PIPE 0x5dc7a8a9e7daedc5     16384                     ->0xfaf3e82a0b4d4888
docker  85319 kot_mapku3   8u    unix 0x8a297315b5c73fd9       0t0                     /Users/kot_mapku3/.docker/run/docker-cli-api.sock
docker  85319 kot_mapku3  27u   systm 0x8a297310e901d4d1       0t0                     [ctl com.apple.netsrc id 6 unit 61]
docker  85319 kot_mapku3  88r     CHR               14,1     0t276                 593 /dev/urandom
docker  92738 kot_mapku3  cwd     DIR                1,4       128            45394070 /Users/kot_mapku3/Developer/s23/tlv/project/macos
docker  92738 kot_mapku3  txt     REG                1,4  66074720            45388690 /Applications/Docker.app/Contents/Resources/bin/docker
docker  92738 kot_mapku3  txt     REG                1,4   2177216 1152921500312783021 /usr/lib/dyld
docker  92738 kot_mapku3   0u     CHR               16,3 0t4784327                 777 /dev/ttys003
docker  92738 kot_mapku3   1u     CHR               16,3 0t4784327                 777 /dev/ttys003
docker  92738 kot_mapku3   2u     CHR               16,3 0t4784327                 777 /dev/ttys003
docker  92738 kot_mapku3   3u  KQUEUE                                                  count=0, state=0xa
docker  92738 kot_mapku3    4    PIPE 0x6da2f4dd2e3a96e2     16384                     ->0x81eff44201fbdfee
docker  92738 kot_mapku3    5    PIPE 0x81eff44201fbdfee     16384                     ->0x6da2f4dd2e3a96e2
docker  92738 kot_mapku3    6    PIPE 0xff4d4d6c34e53cce     16384                     ->0x522b62a299a5d2ec
docker  92738 kot_mapku3    7    PIPE 0x522b62a299a5d2ec     16384                     ->0xff4d4d6c34e53cce
docker  92860 kot_mapku3  cwd     DIR                1,4       128            45394070 /Users/kot_mapku3/Developer/s23/tlv/project/macos
docker  92860 kot_mapku3  txt     REG                1,4  66074720            45388690 /Applications/Docker.app/Contents/Resources/bin/docker
docker  92860 kot_mapku3  txt     REG                1,4   2177216 1152921500312783021 /usr/lib/dyld
docker  92860 kot_mapku3   0u     CHR               16,7    0t1255                 787 /dev/ttys007
docker  92860 kot_mapku3   1u     CHR               16,7    0t1255                 787 /dev/ttys007
docker  92860 kot_mapku3   2u     CHR               16,7    0t1255                 787 /dev/ttys007
docker  92860 kot_mapku3   3u  KQUEUE                                                  count=0, state=0xa
docker  92860 kot_mapku3    4    PIPE 0xeab2cc2df15d329b     16384                     ->0x6d695fb6c4874f84
docker  92860 kot_mapku3    5    PIPE 0x6d695fb6c4874f84     16384                     ->0xeab2cc2df15d329b
docker  92860 kot_mapku3    6    PIPE 0xb3b3031acf24a0e5     16384                     ->0x7348946ee81ca9c1
docker  92860 kot_mapku3    7    PIPE 0x7348946ee81ca9c1     16384                     ->0xb3b3031acf24a0e5
docker  93349 kot_mapku3  cwd     DIR                1,4       128            45394070 /Users/kot_mapku3/Developer/s23/tlv/project/macos
docker  93349 kot_mapku3  txt     REG                1,4  66074720            45388690 /Applications/Docker.app/Contents/Resources/bin/docker
docker  93349 kot_mapku3  txt     REG                1,4   2177216 1152921500312783021 /usr/lib/dyld
docker  93349 kot_mapku3   0u     CHR               16,8    0t2268                 789 /dev/ttys008
```

Are Docker containers really secure?

Are Docker containers really secure?

By default not really

Need to be sure to drop privileges as quickly as possible

No Hypervisor

Run your services as non-root whenever possible

Treat root within a container as if it is root outside of the container

Open Source Docker Desktop Alternatives

- Rancher Desktop
- Colima
- Podman instead of Docker Engine *

*https://www.redhat.com/sysadmin/podman-mac-machine-architecture

References on Pictures

- https://slideplayer.com/slide/13937531/
- https://devopedia.org/docker
- https://collabnix.com/how-docker-for-mac-works-under-the-hood
- https://www.infoq.com/news/2022/06/apple-virtualization-framework/
- https://developer.apple.com/documentation/hypervisor

References

- https://pastebin.com/rVhaspra