Introduction to
Container & Docker

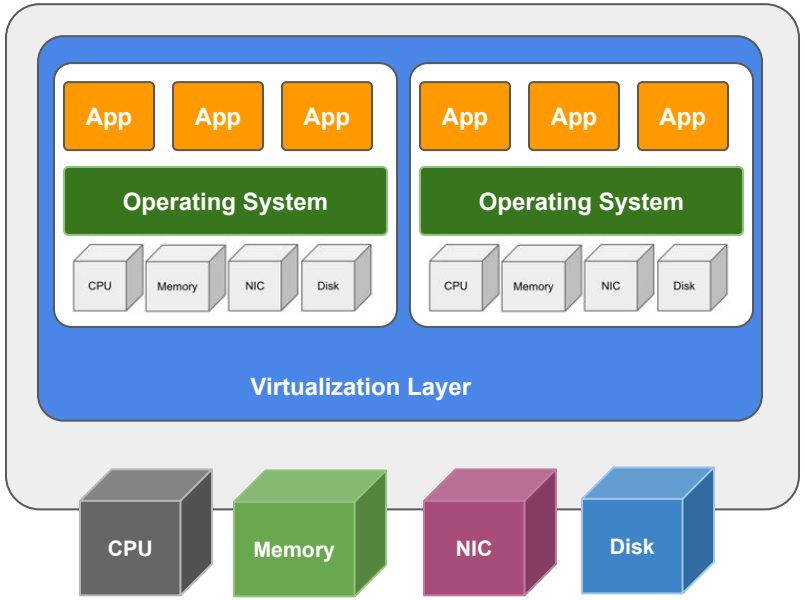
http://bit.ly/2ZyMU0A

ejlp12@gmail.com
Indonesia

Let’s start with Virtualization

Hardware/Platform Virtualization

The virtualization of computers as **complete hardware platforms**, certain logical abstractions of their componentry, or only the functionality required to run **various operating systems**.

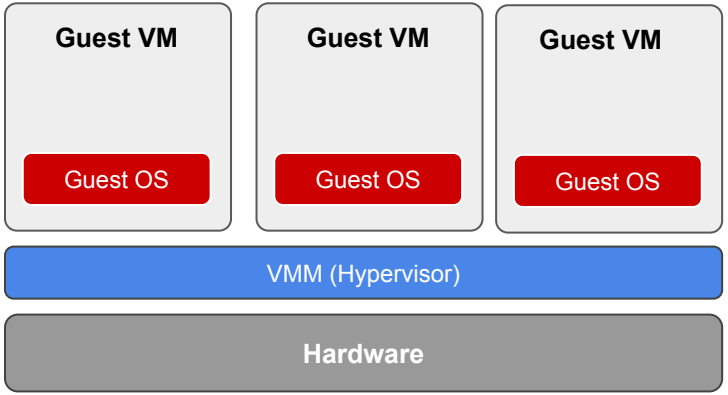


Virtualization

Giving illusion that each OS is running on real hardware

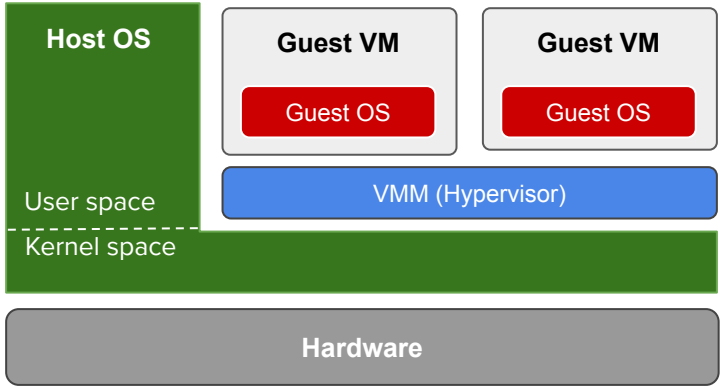
Type-1 vs. Type-2 virtualization

Depending on what sits right on Hardware



Bare metal architecture

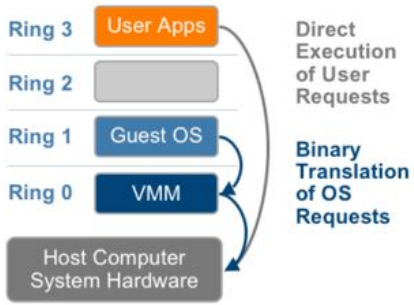
- Xen, VMware ESX server, Hyper-V
- Mostly for server, but not limited
- VMM by default
- OS-independent VMM



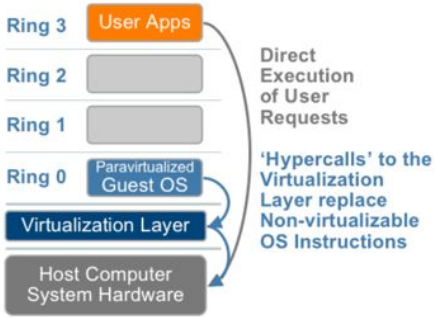
Hosted architecture.

- VMware Workstation, VirtualBox
- Mostly for client devices, but not limited
- VMM on demand
- OS-dependent VMM

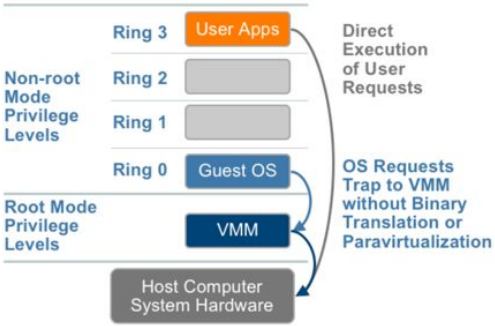
Hardware Virtualization Techniques



1



2



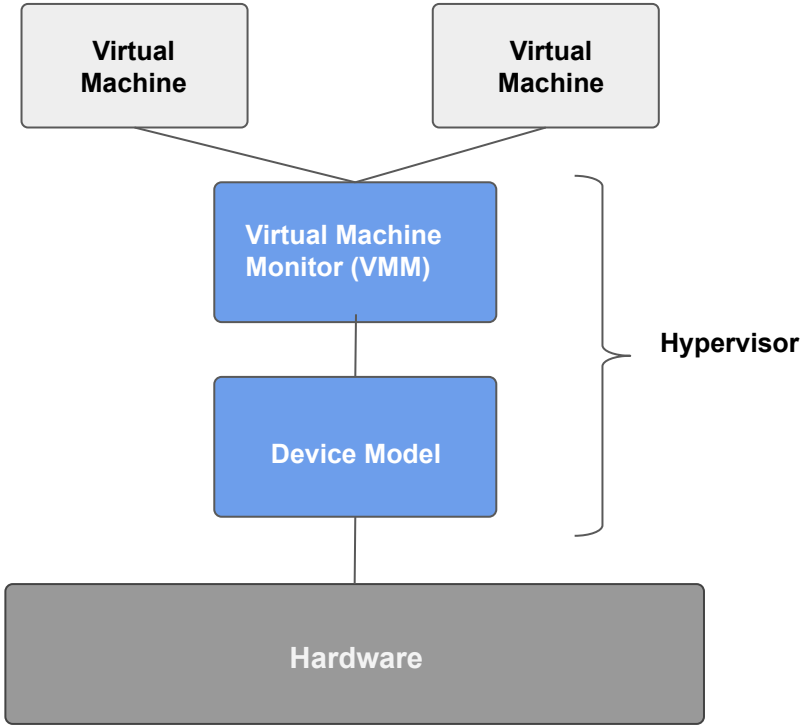
3

Virtualization Techniques:

1. Full Virtualization using Binary Translation
2. OS Assisted Virtualization or Paravirtualization
3. Hardware Assisted Virtualization

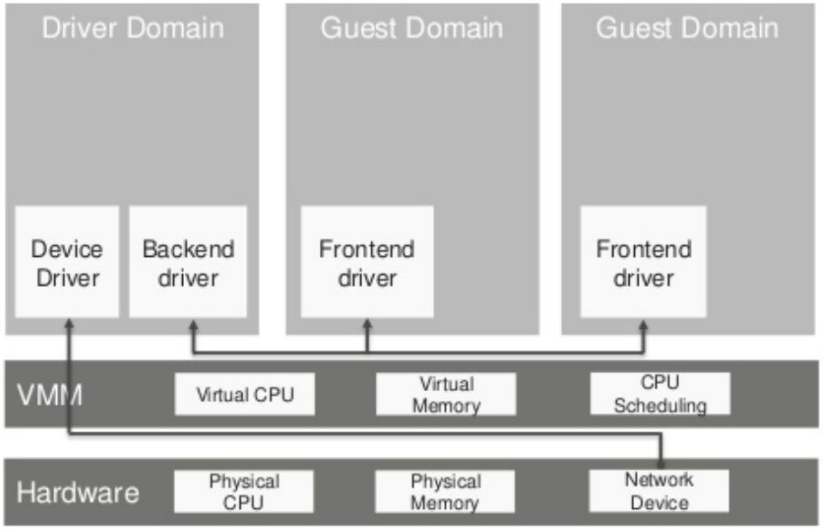
VMM = Virtual Machine Monitor = Hypervisor

Components of Hypervisor

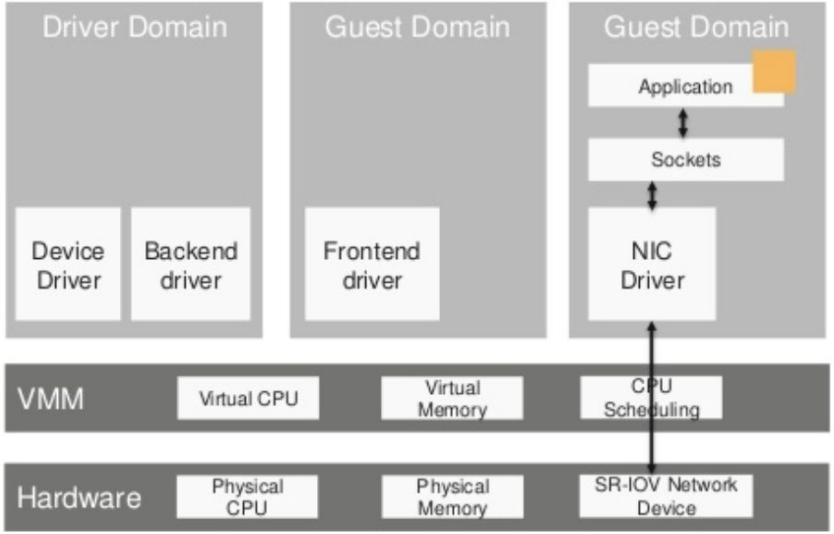


Device Model Operations

Split Driver of Emulation Model



Device Pass-through Model



Virtualization Products

Software:

- Commercial: VMware, Citix XenServer, Microsoft Hyper-V
- Opensource: KVM, Xen, VirtualBox, OpenVZ

Hardware virtualization assistance:

- Intel VT-x
- AMD-V

Virtualization used by Cloud Providers

- AWS EC2:
  - Xen based Paravirtualization (PV)
  - Xen based Hadware Vitual Machine (HVM),
  - Nitro (based on KVM)
  - Firecracker
- GCPGCE:
  - based on KVM + their own the user-space VMM and hardware emulation (does not use QEMU)
- Azure AVM:
  - Customized version of Hyper-V

Benefits of Virtualization

- Optimize hardware capacity (resource utilization)
- Run different systems (Operating Systems) on the same hardware
  - Multitenancy
  - Server consolidation
- Easy to move or replicate a system to different machine
  - Live VM migration or relocation
- Strong Isolation:
  - Protection from System Failures

What is Container?

A container is not a virtual machine.

What is Linux Container?

Short Answer

A container is a process...

...with file system isolation.

What is Linux Container?

Everything in Linux is a file.

- /dev/sda = hard disk
- /dev/proc = processes
- /dev/usb
- /dev/cpu
- /dev/std(in|out)
- /bin/bash... just a binary file

When you start a container,

you're just starting a process

on your machine.

Containers are not a new technology

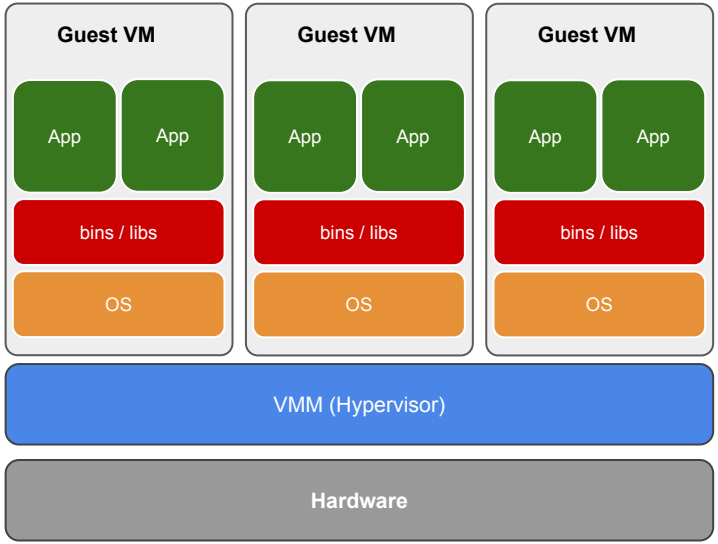
The earliest iterations of containers have been around in open source Linux code for decades.

What is Linux Container?

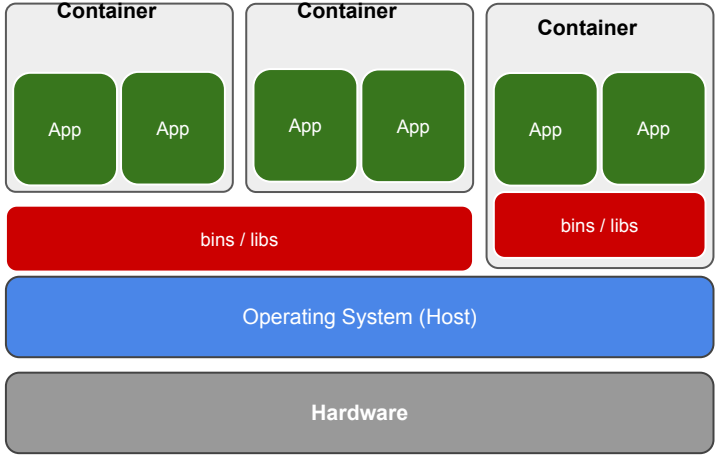
- Lightweight “virtualization”.
- OS-level “virtualization”
- Allow single host to operate multiple **isolated & resource-controlled** Linux Instances.
- included in the Linux kernel called LXC (Linux Container)

!!! LXC term can refer to a Linux container technology but in other context can refer to a tool for container management

Hypervisor vs Linux Container



Type 1 Hypervisor



Linux Container

OS-Level Virtualization

Not Only Linux

LXC

LXD

http://linux-vserver.org

lmctfy

https://github.com/google/lmctfy

solaris containers (zones)

AIX Workload partitions (WPARs)

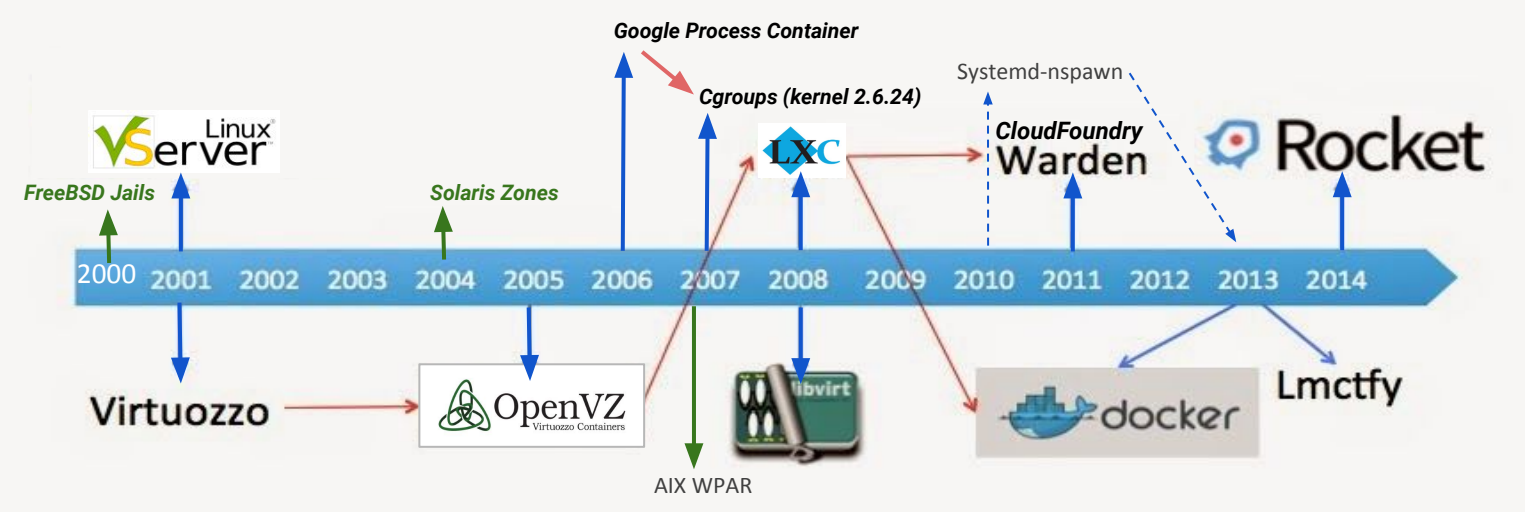
OpenBSD sysjail

FreeBSD jail

https://en.wikipedia.org/wiki/Operating-system-level_virtualization#IMPLEMENTATIONS

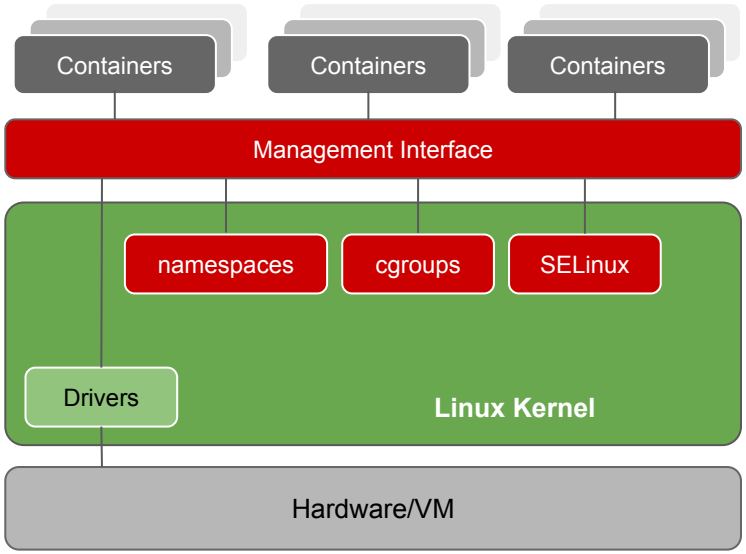
Container History

1979: chroot Unix7
1982: chroot BSD



http://kiwenlau.blogspot.co.id/2015/01/linux-container-technology-overview.html

Container Architecture (Example)



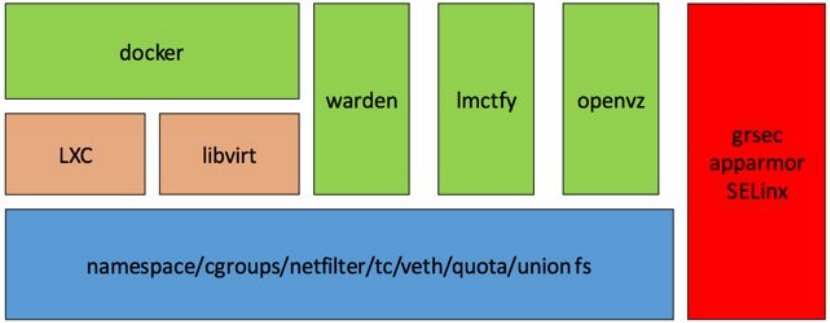
- **namespaces** allows complete **isolation of an applications' view of the operating environment**, including process trees, networking, user IDs and mounted file systems.
- **cgroups**: allows **limitation and prioritization of resources** (CPU, memory, block I/O, network, etc.)
- Security-Enhanced Linux (**SELinux**) provides **secure separation of containers** by applying SELinux policy and labels. It integrates with virtual devices by using the **sVirt** technology.

Other container technology might use different security component or use additional software components

Linux Container Technology

Underlying technology:

- namespace/cgroups
  - veth
  - union fs(AUFS)
  - netfilter/chroot/tc/quota
- Low-level container management
  - LXC/libvirt
- Security related
  - grsec/apparmor/SELinux
- High-level container/image management
  - docker/warden/garden/lmctfy/openVZ



Linux Container Technology

Container supports separation of various resources. They are internally realized with different technologies called "**namespace**."

– Filesystem separation → Mount namespace (kernel 2.4.19)
– Hostname separation → UTS namespace (kernel 2.6.19)
– IPC separation → IPC namespace (kernel 2.6.19)
– User (UID/GID) separation → User namespace (kernel 2.6.23〜kernel 3.8)
– Processtable separation → PID namespace (kernel 2.6.24)
– Network separation → Network Namespace (kernel 2.6.24)
– Usage limit of CPU/Memory → Control groups

Benefit of Container over Virtualization

- Linux Containers are designed to support **isolation** of one or more applications.
- System-wide changes are visible in each container.

  For example, if you upgrade an application on the host machine, this change will apply to all sandboxes that run instances of this application.
- Since containers are lightweight, a large number of them can run simultaneously on a host machine.

  The theoretical maximum is 6000 containers and 12,000 bind mounts of root file system directories.

How big is the container image?

Top 10 image sizes (latest tag) on Docker Hub

IMAGE NAME	SIZE
busybox	1 MB
ubuntu	188 MB
swarm	17 MB
nginx	134 MB
registry	423 MB
redis	151 MB
mysql	360 MB
mongo	317 MB
node	643 MB
debian	125 MB

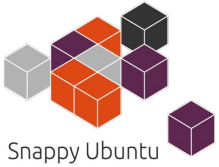
Some minimal Docker images built on top of Alpine:

IMAGE NAME	SIZE
Nginx	28 Mb
64 Bit Server JRE 8	124 Mb
64 bit JDK 8	165 Mb
Redis	12 Mb

Minimalistic OS

A tiny Linux distribution created for container

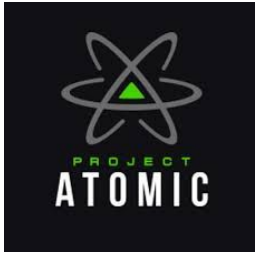
https://coreos.com/



https://developer.ubuntu.com/en/snappy/

http://osv.io/

- Alpine
- Busybox
- Cirros
- Ubuntu
- Centos
- Debian
- RHEL Atomic



http://www.projectatomic.io/

http://rancher.com/rancher-os/

https://vmware.github.io/photon/

http://boot2docker.io/

Minimalist OS

A common set of ideas:

- Stability is enhanced through transactional upgrade/rollback semantics.
- Traditional package managers are absent and may be replaced by new packaging systems (Snappy), or custom image builds (Atomic).
- Security is enhanced through various isolation mechanisms.
- systemd provides system/service management. In general, systemd has been adopted almost universally among Linux distributions, so this shouldn’t be a surprise.

Minimalistic OS Comparison

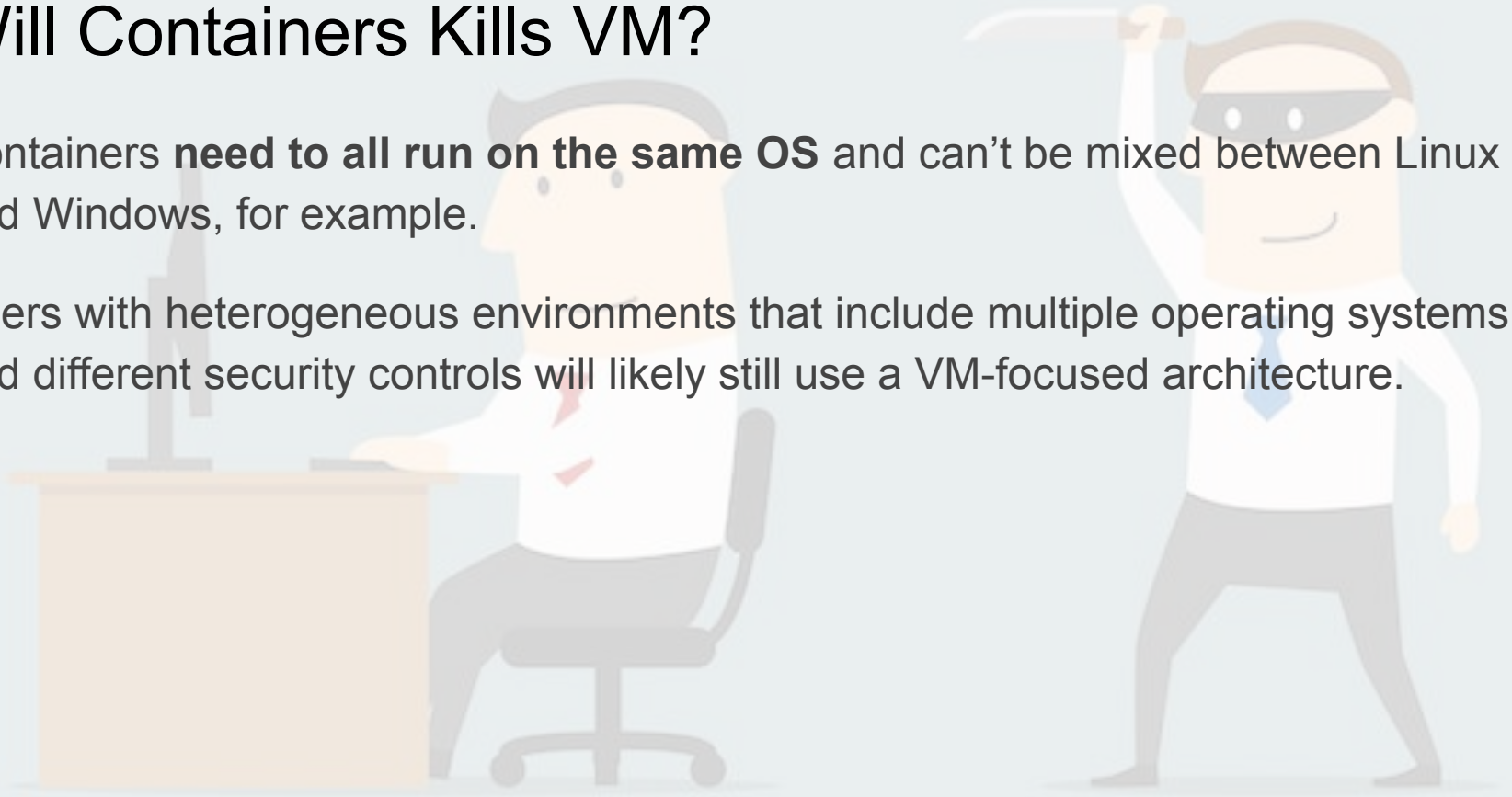
	CoreOS (647.0.0)	**RancherOS (0.23.0)**	**Atomic (F 22)**	**Photon**	**Snappy (edge – 145)**
Size	164MB	20MB	151/333MB	251MB	111MB
Kernel version	3.19.3	3.19.2	4.0.0	3.19.2	3.18.0
Docker version	1.5.0	1.6.0	1.6.0	1.5.0	1.5.0
Init system	systemd	Docker	systemd	systemd	systemd
Package manager	None (Docker/Rocket)	None (Docker)	Atomic	tdnf (tyum)	Snappy
Filesystem	ext4	ext4	xfs	ext4	ext4
Tools	Fleet, etcd	–	Cockpit (Anaconda, kickstart), atomic	–	

https://blog.inovex.de/docker-a-comparison-of-minimalistic-operating-systems/

Will Containers Kills VM?

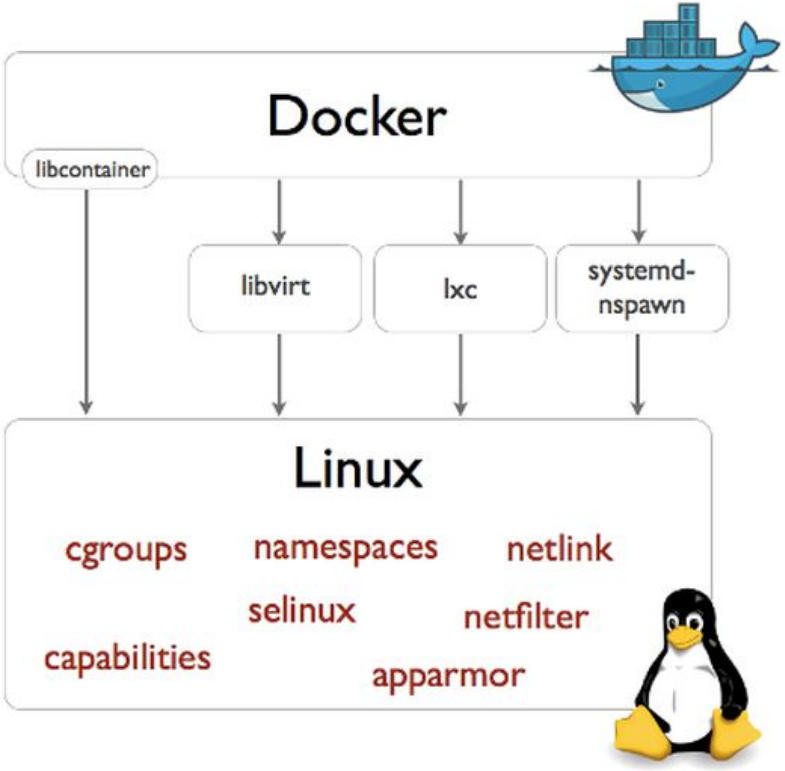
Containers **need to all run on the same OS** and can’t be mixed between Linux and Windows, for example.

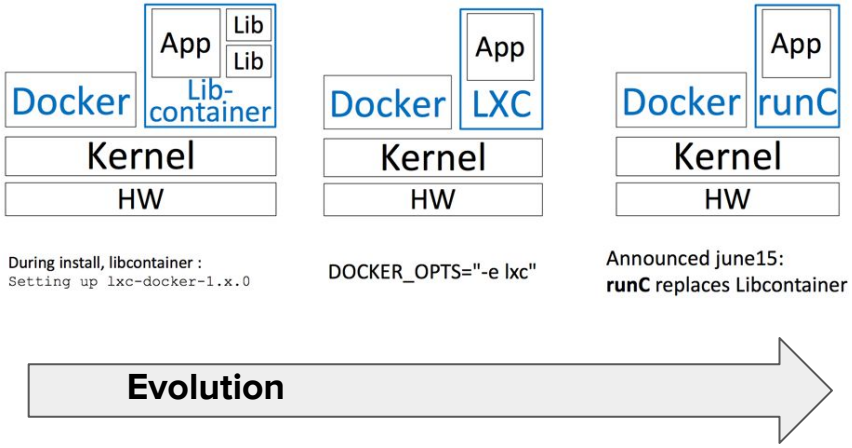
Users with heterogeneous environments that include multiple operating systems and different security controls will likely still use a VM-focused architecture.



DOCKER

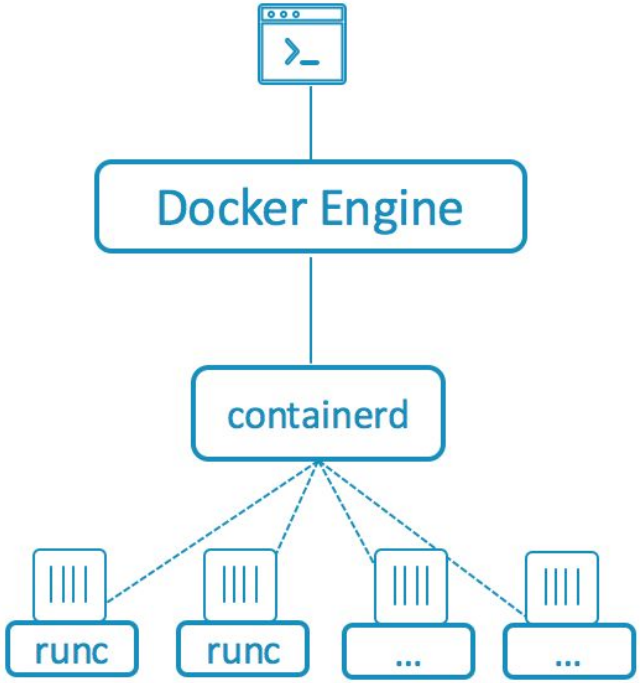
Docker Components (First Generation)





libcontainer - https://github.com/docker/libcontainer
LXC - https://linuxcontainers.org/
libvirt - http://libvirt.org/
systemd-nspawn - https://www.freedesktop.org/software/systemd/man/systemd-nspawn.html

Docker Components



Same Docker UI and commands

User interacts with the Docker Engine

Engine communicates with containerd

containerd spins up runc or other OCI compliant runtime to run containers

containerd

A daemon for Linux and Windows.

It manages the complete container lifecycle of its host system, from image transfer and storage to container execution and supervision to low-level storage to network attachments and beyond

https://containerd.io/

runC

Docker version > 0.9

CLI tool for spawning and running containers according to the OCI specification
https://www.opencontainers.org/

https://github.com/opencontainers/runc

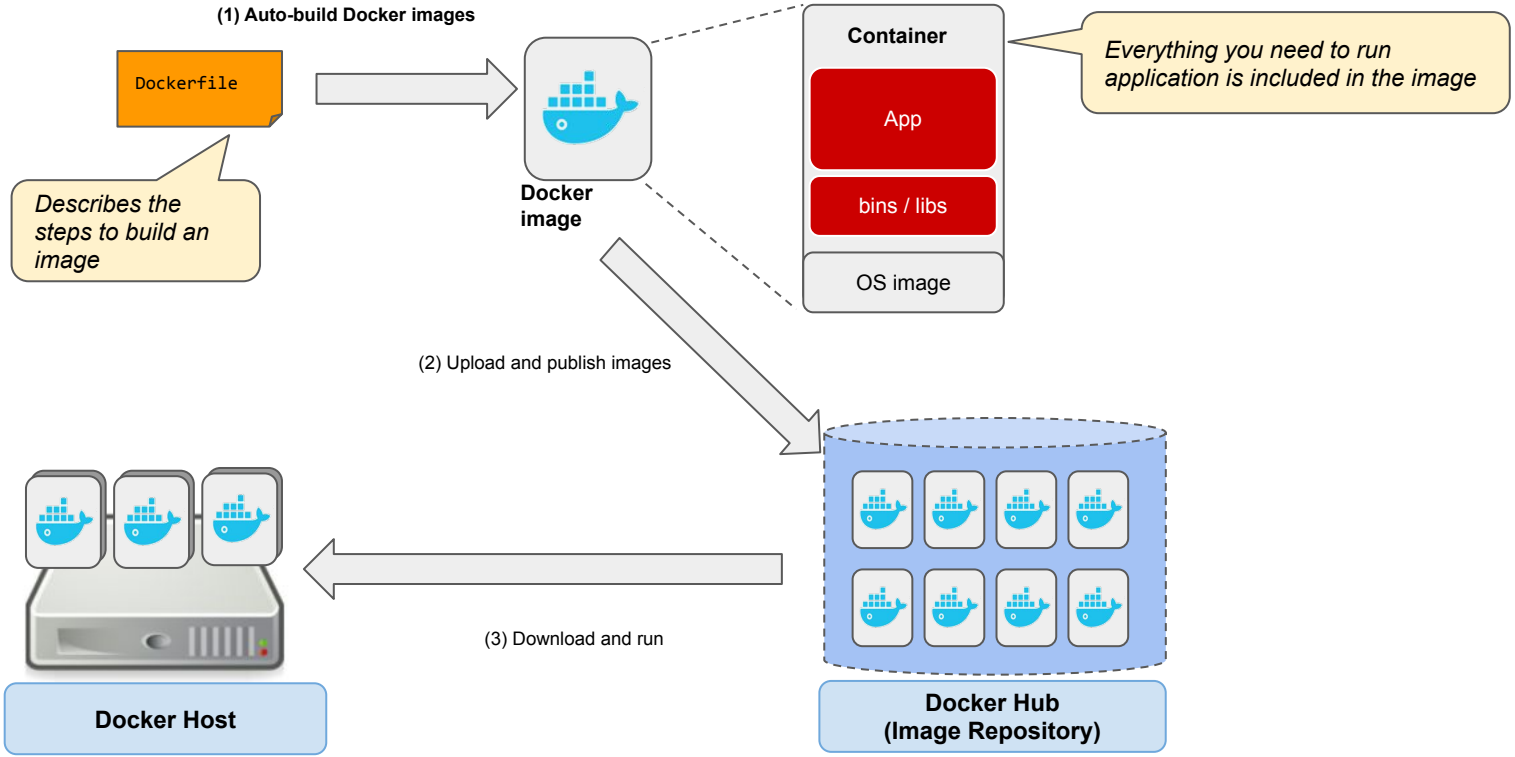
Docker in Linux Distributions

- Docker natively included starting with Ubuntu 14.04 LTS
- Red Hat supporting Docker since Red Hat Enterprise Linux 6.5, integrated since RHEL version 7

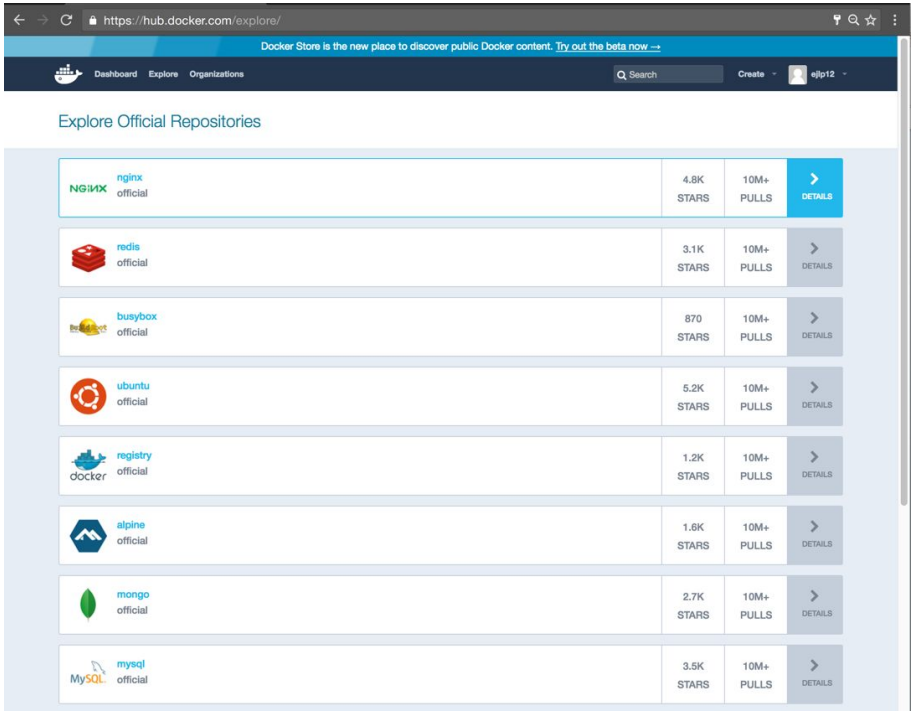
Containerization standard?

https://www.opencontainers.org/

What you can do with Docker



Docker Image Registry



Hosted:

- Docker Hub
- Quay.io
- AWS EC2 Container Registry (ECR)
- Azure Container Registry (ACR)
- Google Container Registry (GCR)

Non hosted:

- Artifactory by JFrog
- Sonatype Nexus
- Harbor (CNCF project)

INSTALL DOCKER ON Amazon Linux

```
sudo yum update -y
sudo amazon-linux-extras install docker
sudo service docker start
sudo systemctl enable docker
sudo usermod -a -G docker ec2-user
```

Best practices for writing Dockerfiles

- Create ephemeral containers
- Understand build context
- Pipe Dockerfile through stdin
- **Exclude with .dockerignore**
- Use multi-stage builds
- Don’t install unnecessary packages
- Decouple applications
- Minimize the number of layers
- Sort multi-line arguments
- Leverage build cache

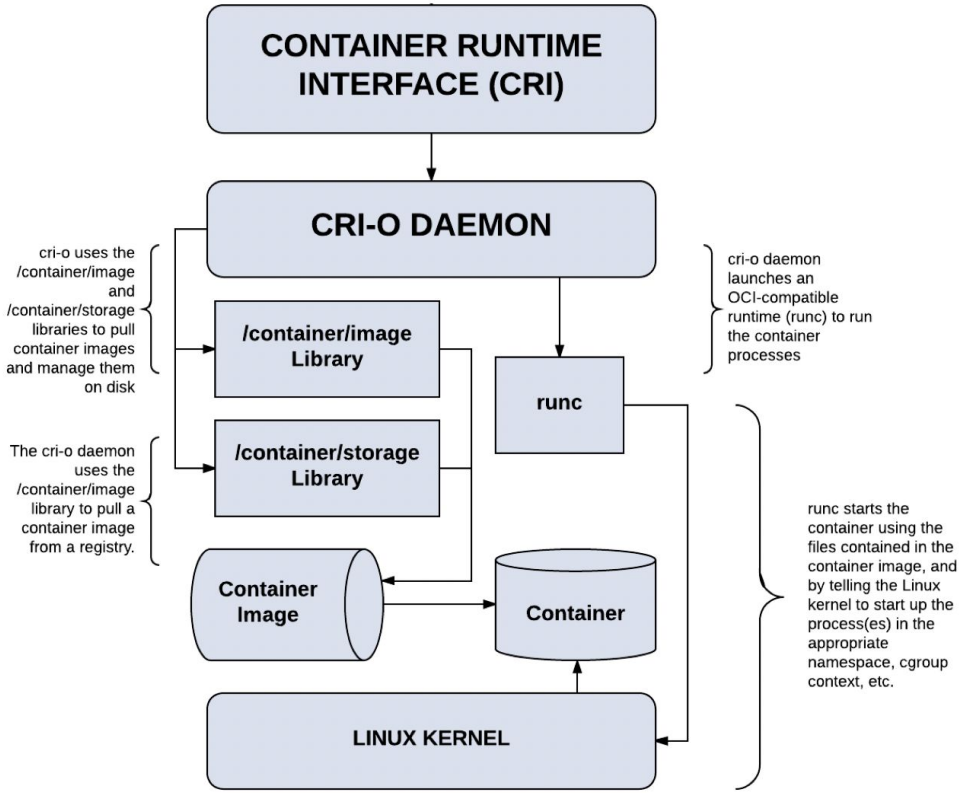
Labs

1. Docker Beginner: http://bit.ly/2L1YOx3
2. Porting NodeJS App to Docker: http://bit.ly/2PsuWZa
3. CI/CD using Docker: http://bit.ly/2Gvuup0

CRI-O: an alternative to Containerd

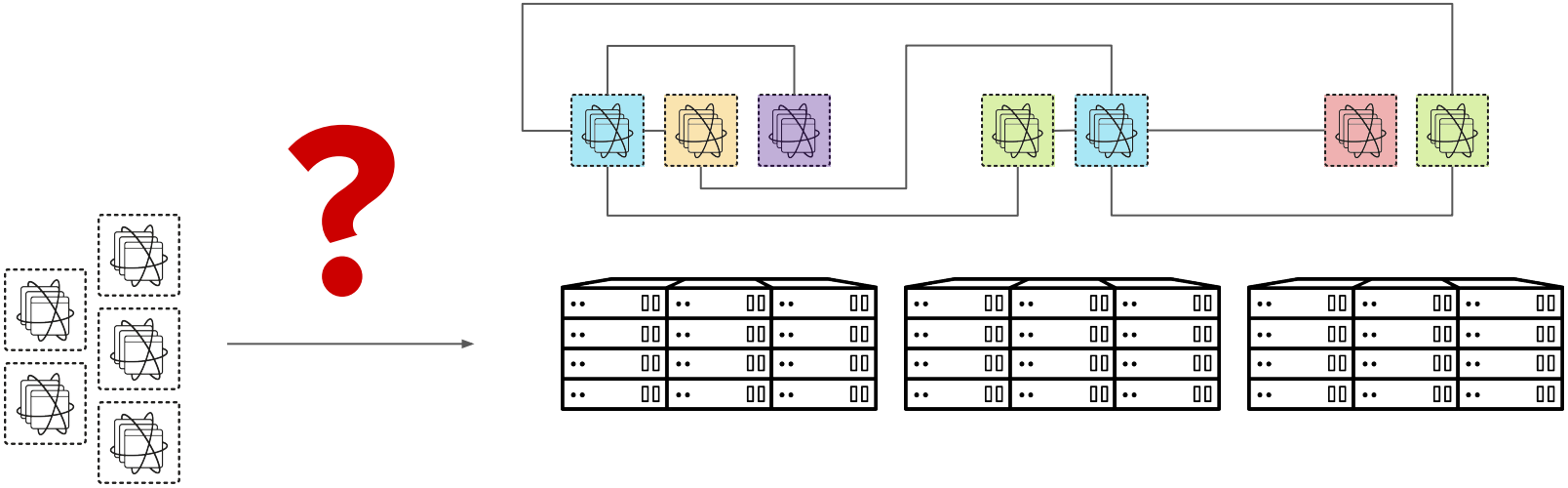
It is a container Engine

1. Provide API/User Interface
2. Pulling/Expanding images to disk
3. Building a config.json



Container Orchestration

How to deploy complex app to multiple servers, data centers?



WE NEED MORE THAN JUST CONTAINERS

Scheduling
Decide where to deploy containers

Lifecycle and health
Keep containers running despite failures

Discovery
Find other containers on the network

Monitoring
Visibility into running containers

Security
Control who can do what

Scaling
Scale containers up and down

Persistence
Survive data beyond container lifecycle

Aggregation
Compose apps from multiple containers

Infrastructure as code (IaC)

The process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

Immutable infrastructure

An approach to managing services and software deployments on IT resources wherein **components are replaced rather than changed**. An application or services is effectively redeployed each time any change occurs.

Containers Orchestrations

Functional Capabilities

SCHEDULING

- Placement
- Replication/Scaling
- Resurrection
- Rescheduling
- Rolling Deployment
- Upgrades
- Downgrades
- Collocation

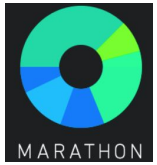
RESOURCE MANAGEMENT

- Memory
- CPU
- GPU
- Volumes
- Ports
- IPs

SERVICE MANAGEMENT

- Labels
- Groups/Namespaces
- Dependencies
- Load Balancing
- Readiness Checking

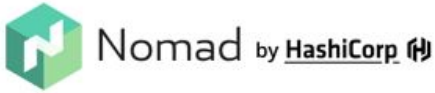
Containers Orchestrations



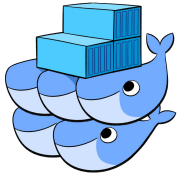
(lightweight)

https://clusterhq.com/flocker/

an open-source container
data volume manager



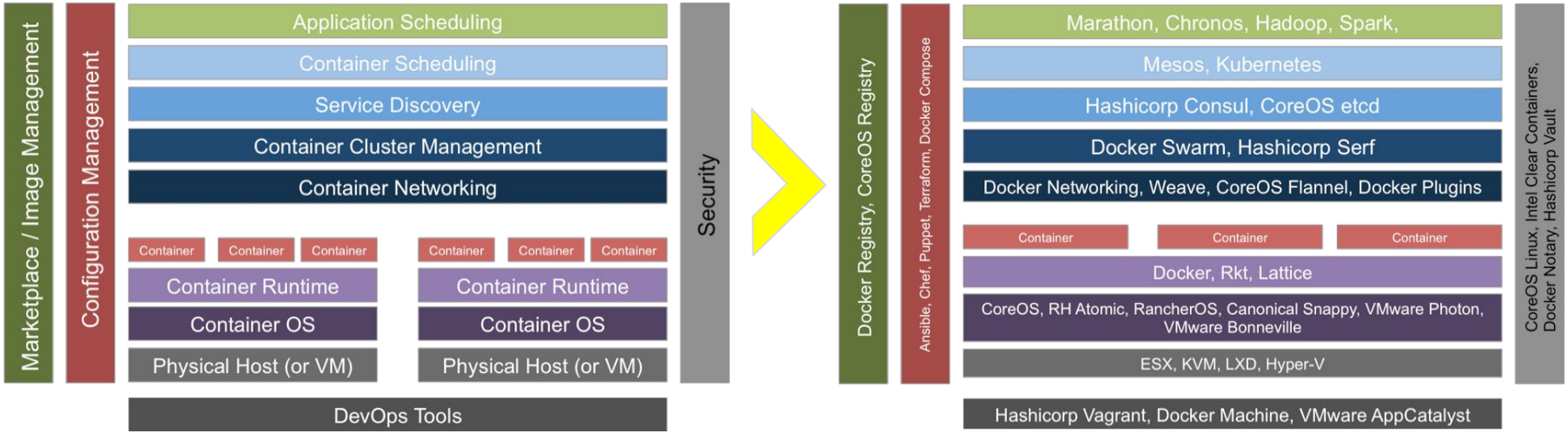
https://coreos.com/fleet/
https://github.com/coreos/fleet



DOCKER SWARM

Emerging Container Stack

Source: Wikibon 2015



http://wikibon.com/evolving-container-architectures/

PaaS products based on Container

https://www.openshift.org/

https://www.cloudfoundry.org/

http://stratos.apache.org/

http://getcloudify.org/

http://deis.io/

https://www.kontena.io/

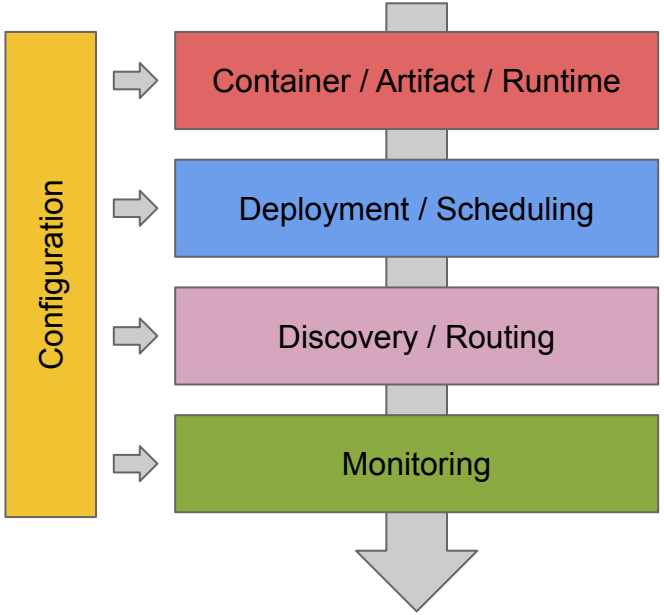
https://flynn.io/
https://github.com/dawn/dawn

https://github.com/Yelp/paasta

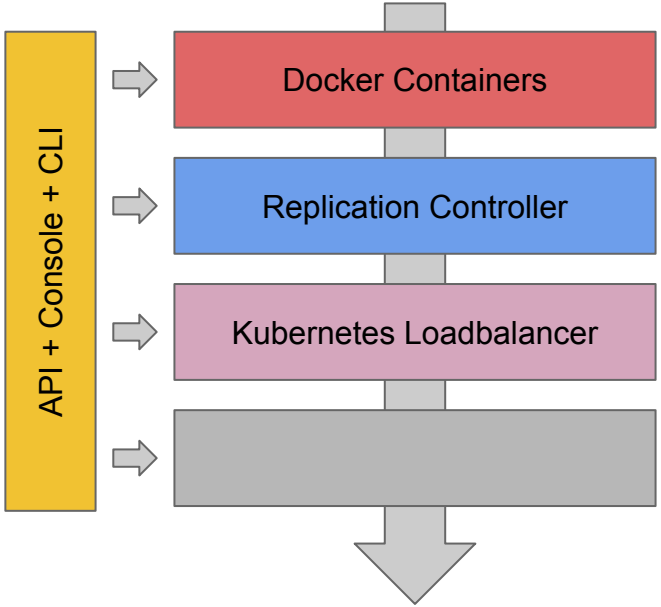
https://tsuru.io/

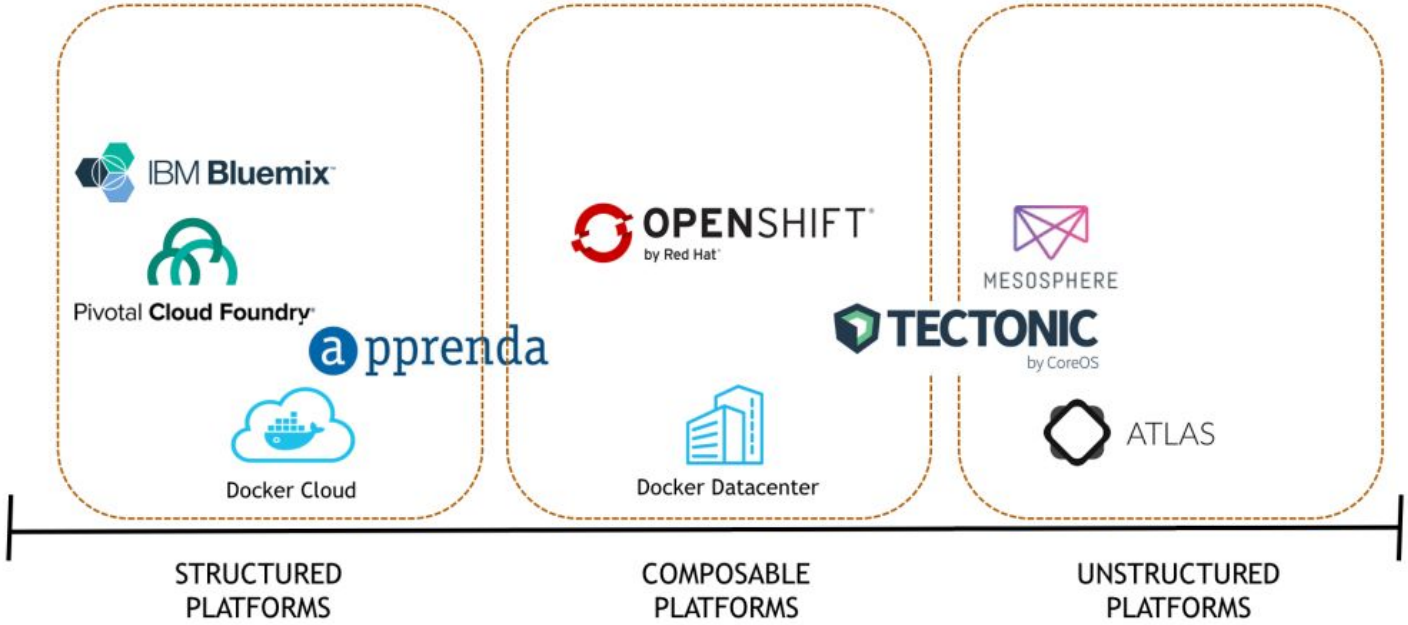
http://www.octohost.io/

PaaS Model



Example




IBM Bluemix
Pivotal Cloud Foundry
apprenda
Docker Cloud
OPENSHIFT
by Red Hat
Docker Datacenter
MESOSPHERE
TECTONIC
by CoreOS
ATLAS
STRUCTURED PLATFORMS
COMPOSABLE PLATFORMS
UNSTRUCTURED PLATFORMS

Thanks!