@ewoutp @joerg_schad

ArangoDB
OASIS

Challenges in Building a Multi-Cloud-Provider Platform With Managed Kubernetes

Amazon EKS

Google Kubernetes Engine

tl;dr

- **Building a managed Database Service across multiple Cloud Vendors is challenging**
- Kubernetes is a great abstraction
- Managed K8s is removing a lot of operational efforts
- But supporting different Cloud Providers….
  - Security
  - Authentication/Authorization
  - Networking
  - Storage
  - Kubernetes Versions
  - Container Runtimes
  - Logging

Amazon
EKS

Google Kubernetes Engine

Ewout Prangsma

Teamlead ArangoDB Oasis
@ArangoDB

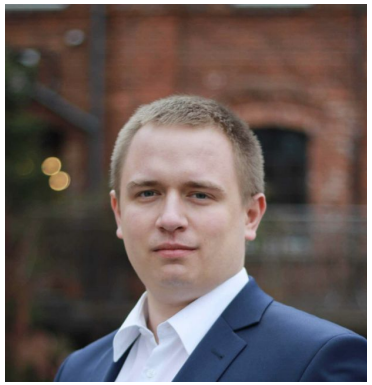
- **Cloud & Distributed Systems**
- **Kubernetes**
- **Make databases easy again**

- **Twitter: @ewoutp**
- **Slack: Ewoutp.ArangoDB**

ArangoDB
OASIS

ArangoDB
OASIS

Robert Stam



Adam Janikowski

Tomasz Mielech

Ewout Prangsma

Stanislav Filippov

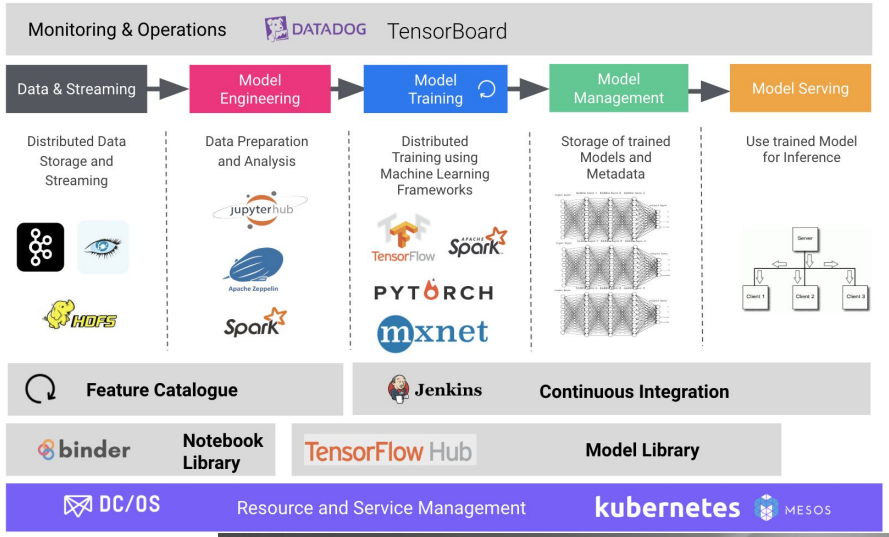
Gergely Bräutigam

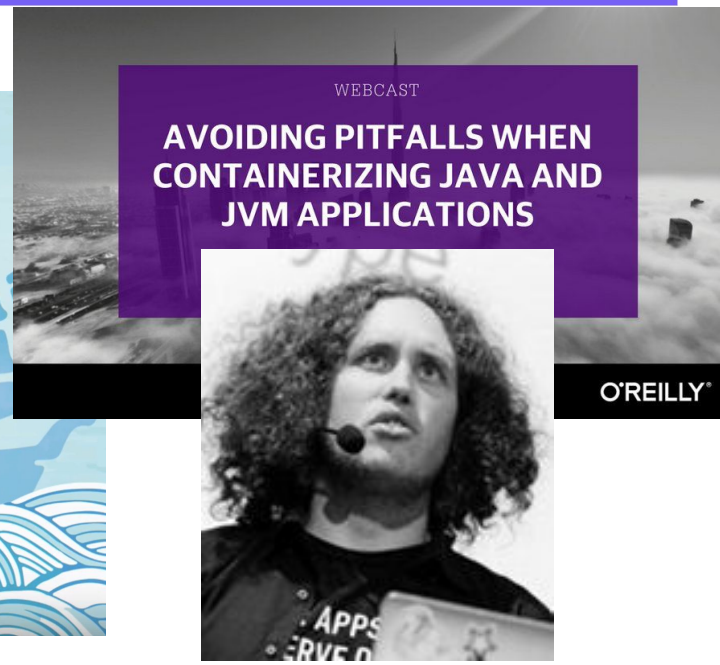
Jörg Schad, PhD

Head of Engineering and ML @ArangoDB

- **Suki.ai**
- **Mesosphere**
- **Architect @SAP Hana**
- **PhD Distributed DB Systems**

- **Twitter: @joerg_schad**

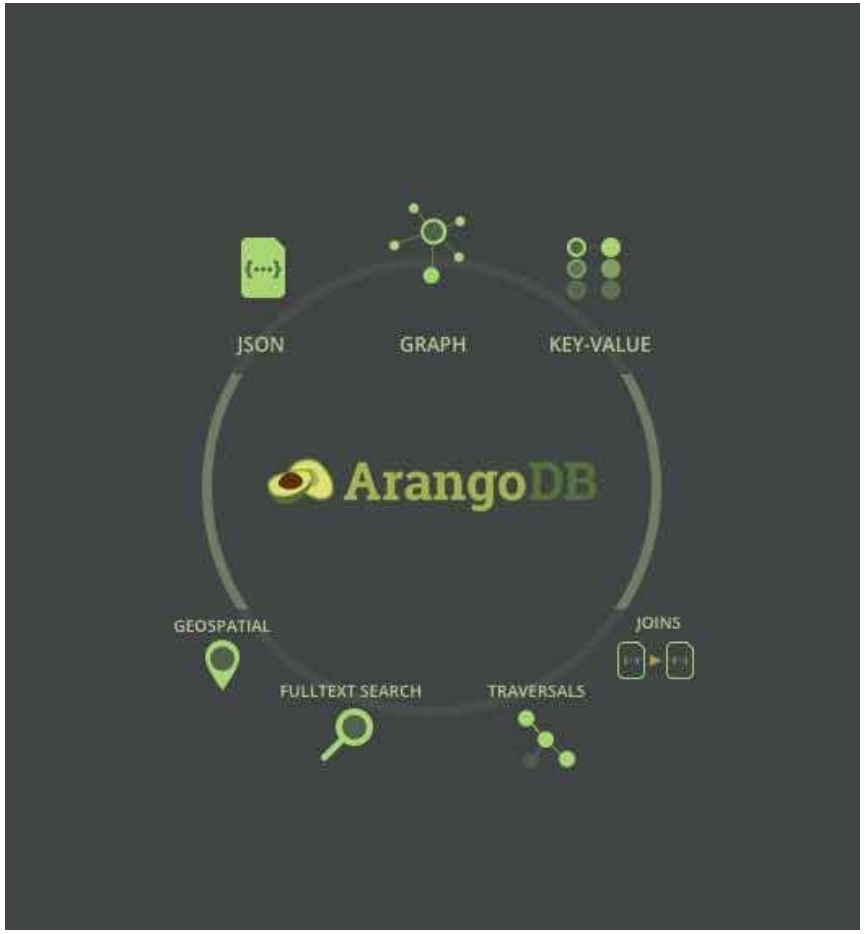




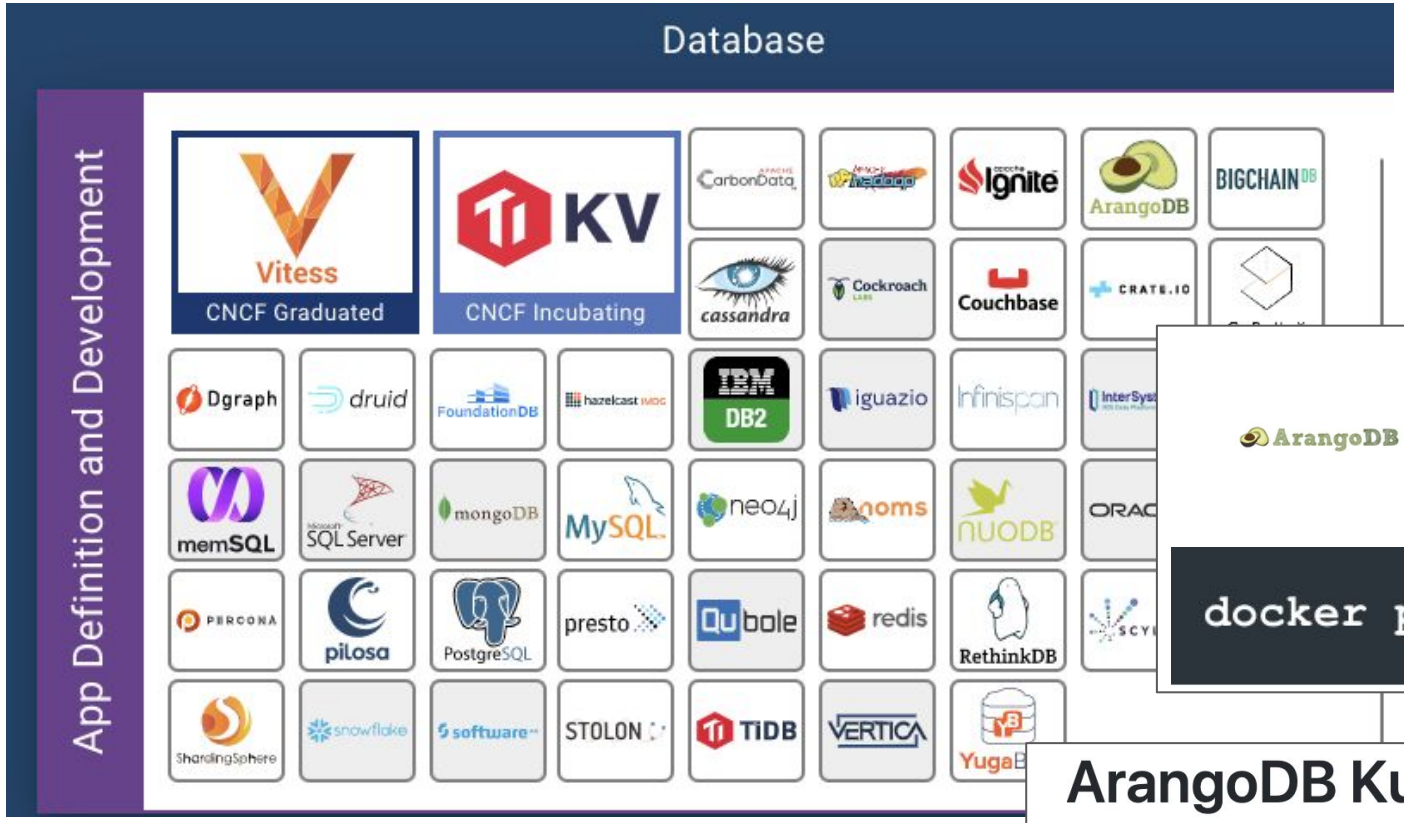
Why do we care?

ArangoDB

- Native Multi Model Database
  - Stores, K/V, Documents & Graphs
- Distributed
  - Graphs can span multiple nodes
- AQL - SQL-like multi-model query language
- ACID Transactions including Multi Collection Transactions



Databases & Container



ArangoDB
OASIS

arangodb ☆

ArangoDB

Docker Official Images

ArangoDB - a distributed database with a flexible data model for documents, graphs, and key-values.

```
docker pull arangodb
```

ArangoDB Kubernetes Operator

docker pulls 696k

ArangoDB Kubernetes Operator helps to run ArangoDB deployments on Kubernetes clusters.

https://landscape.cncf.io/

ArangoDB

OASIS

ArangoDB Oasis

- Managed Service for ArangoDB
- Run many ArangoDB deployments in the cloud
  - AWS, Google, Azure, ...
- Fully managed deployments
- Security for 2020 and beyond
- Highly scalable

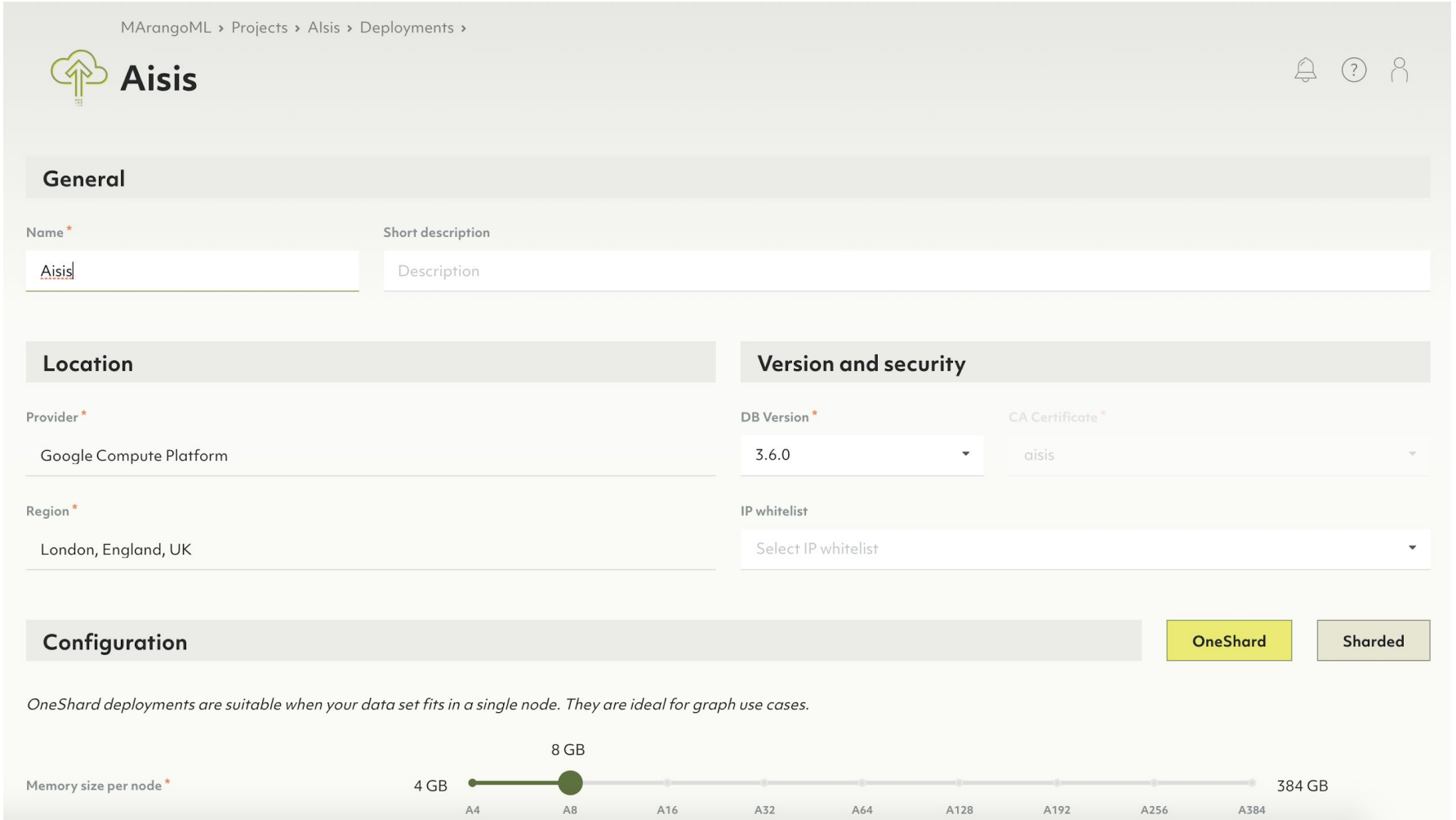
ArangoDB Oasis

- Manag
- Run m
  - AW
- Fully m
- Secur
- Highly

oDB

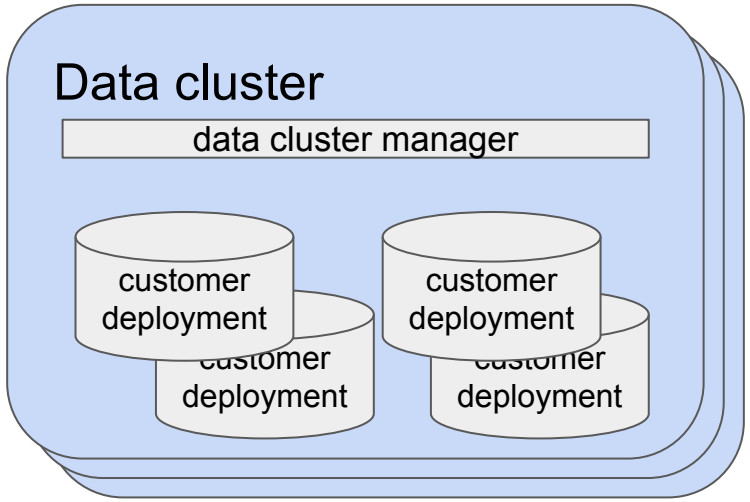
ASIS

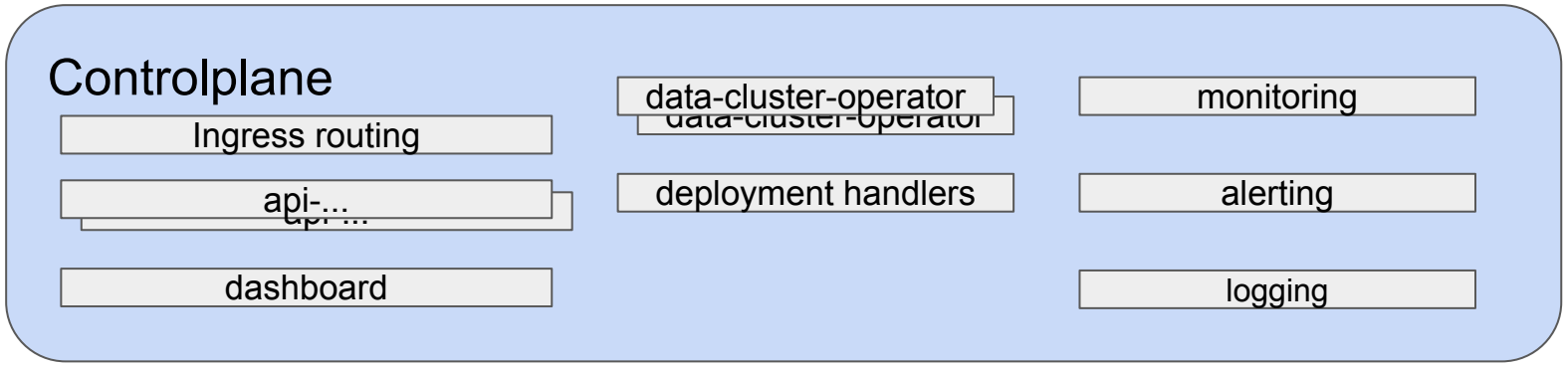
Deployment


MArangoML › Projects › AIsis › Deployments ›
Aisis
General
Name *
Aisis
Short description
Description
Location
Provider *
Google Compute Platform
Region *
London, England, UK
Version and security
DB Version *
3.6.0
CA Certificate *
aisis
IP whitelist
Select IP whitelist
Configuration
OneShard
Sharded
OneShard deployments are suitable when your data set fits in a single node. They are ideal for graph use cases.
Memory size per node *
8 GB
4 GB
384 GB
A4
A8
A16
A32
A64
A128
A192
A256
A384

Oasis Architecture

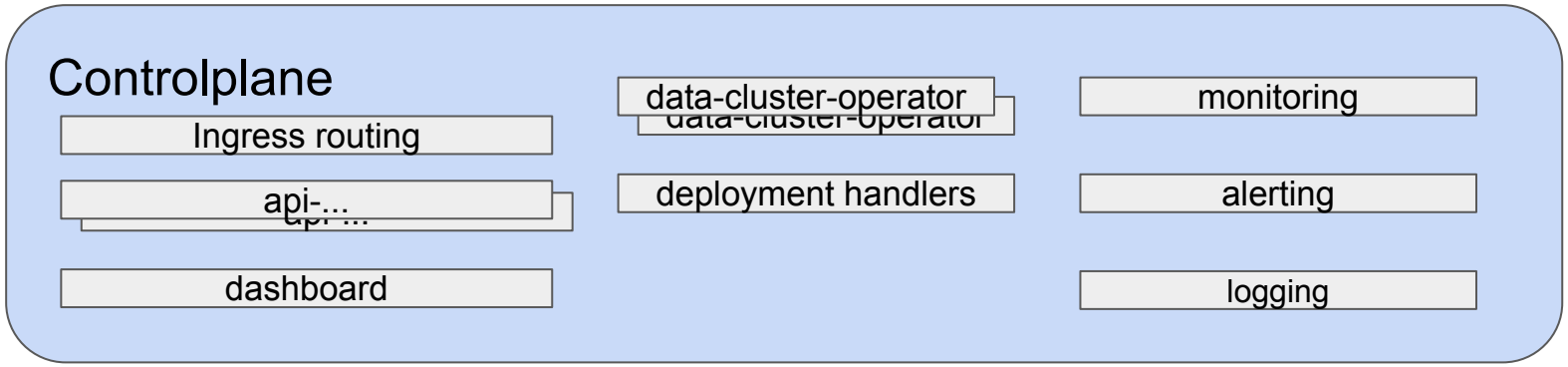
Architecture





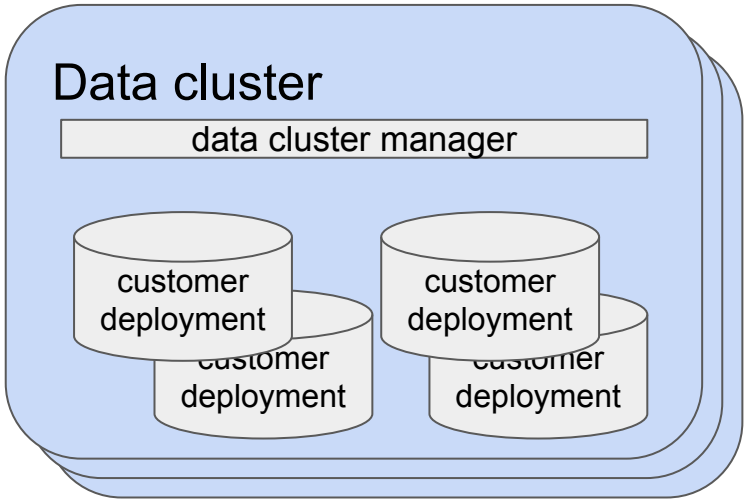
Controlplane

- Oasis Brain
  - Running as managed k8s in GKE
- Hosts dashboards, API handlers, ...
- Responsible for data cluster creation/removal
  - Data cluster operator per cloud provider
- Communication “hub” for data clusters



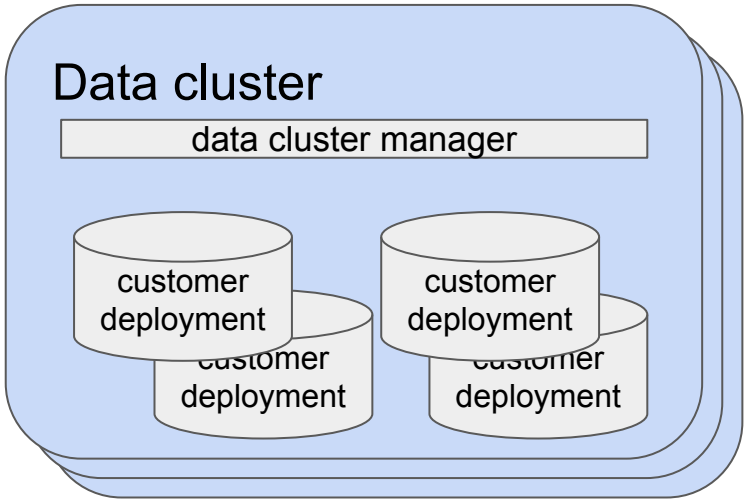
Data Cluster

- Used to run customer deployments
- Many Data cluster
  - Cloud provider
  - Zone
- Abstracts clusters details
- **Abstracts cloud provider details**
  - Mostly …
- Specialized k8s cluster
  - Few “system” components
- Cilium network layer



Data Cluster Creation

- Fully automatic
- Responsibility of data-cluster-operator-xyz
  - Different for each cloud provider .... really different
- Designed for “pull-only” setup
  - Future custom VPCs



Kubernetes as Abstraction

Container Abstractions

Container Orchestration

CONTAINER

RESOURCES

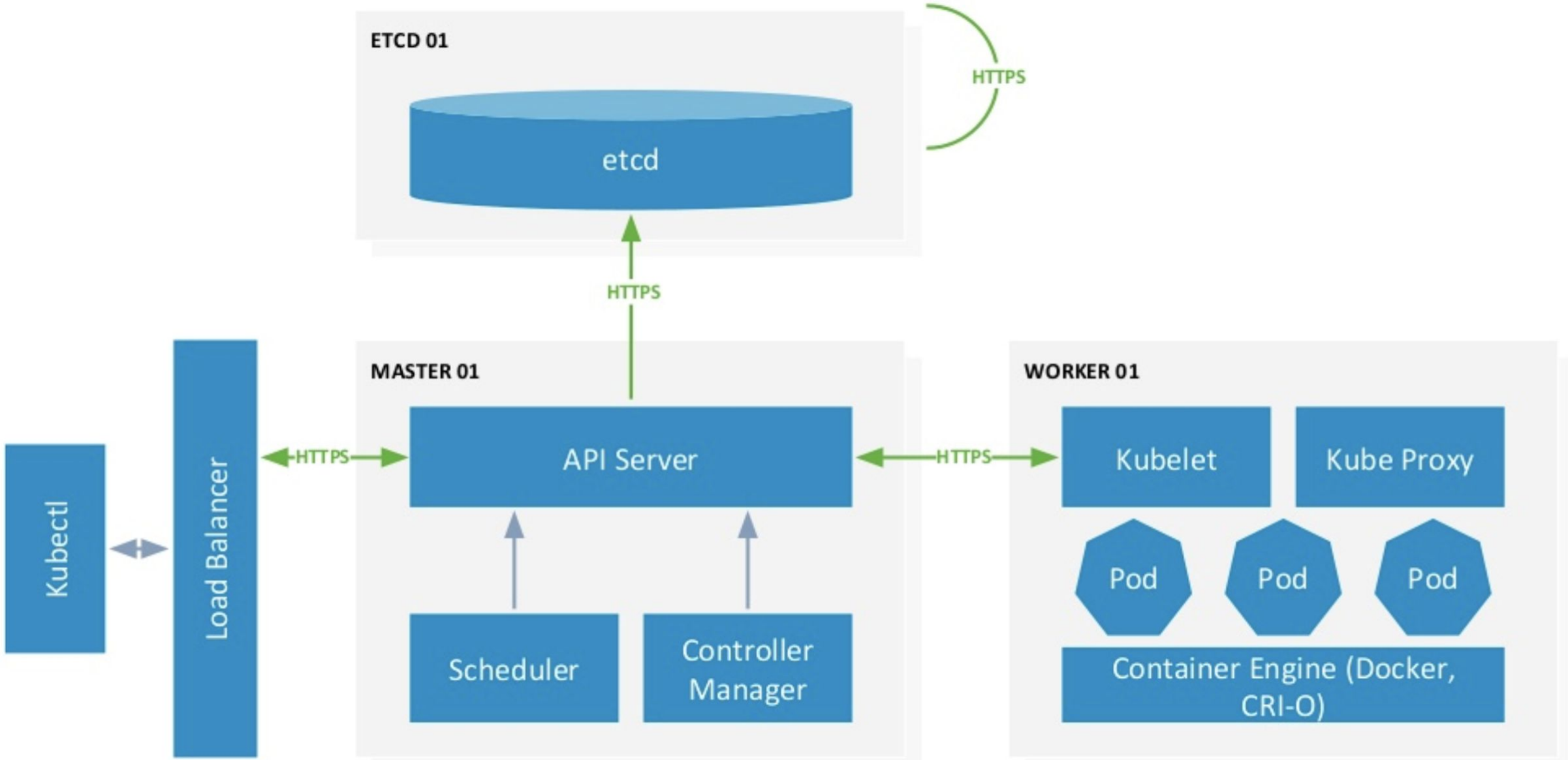
SERVICE MANAGEMENT

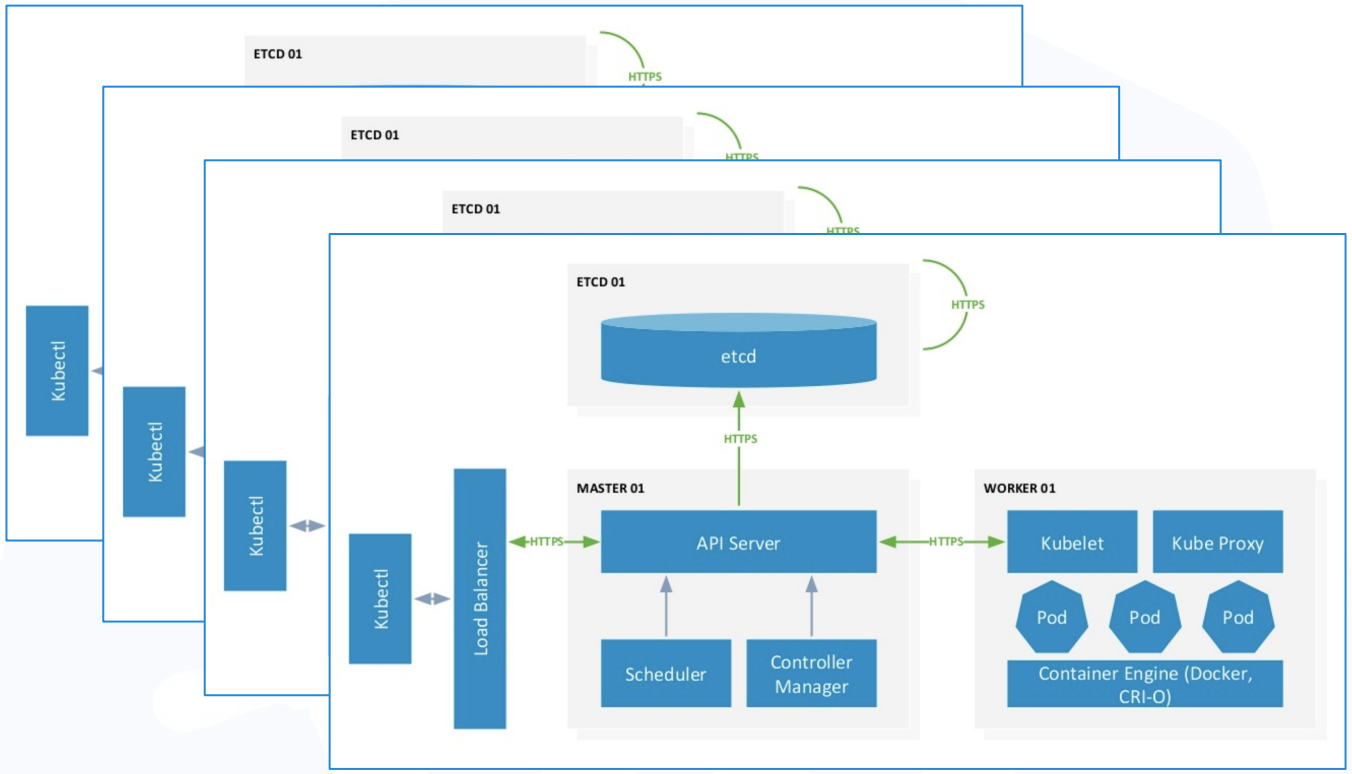
Abstraction II

aws

Google Kubernetes Engine

Azure


ETCD 01
etcd
HTTPS
HTTPS
MASTER 01
Kubectl
Load Balancer
HTTPS
API Server
Scheduler
Controller Manager
HTTPS
WORKER 01
Kubelet
Kube Proxy
Pod
Pod
Pod
Container Engine (Docker, CRI-O)



“Kubernetes Cluster are like Pringles - You can’t just have one”

Thomas Rampelberg

Abstraction II (managed)

Amazon EKS

Google Kubernetes Engine

aws

Google Kubernetes Engine

Azure

Why Multi-Cloud-Provider

Multi-Cloud-Provider

- Company Policy
  - Can be in/exclusive
  - Amazon might be a competitor...
  - GDPR/CCPA
- Avoid Vendor dependency
  - …
- Flexibility
- Buzzword

Multi-Cloud-Provider Challenges

- Resources
- Kubernetes Versions
- Security
- Authentication/Authorization
- Logging
- Networking
- Storage
- Container Runtimes

First: Blowing Steam off...

- AWS creates many resources on the fly
  - E.g. load-balancers, security group rules etc.
- Resources have dependencies … removal challenging
- Not all resources have tags
- Error handling is “not so” structured … string parsing needed

- Aggressive update policy

- Least mature … missing simple features
- VM Scalesets
- PVC Resizing is “broken”

Resource Creation

- VPC, InternetGateway, NAT Gateway, Subnets, Routing tables
- SecurityGroups
- AutoScalingGroups, AMI
- EKS Cluster

- AWS creates many resources on the fly
- Resources removal with dependencies
  - Not all resources are tagged

- VPC
- GKE Cluster, Nodepools

- Resource groups
- AKS Cluster, VMSSs
- StorageAccount

- VM Scalesets cannot be scaled to 0
- Number of VM Scalesets is limited (8)

Kubernetes Versions

- Moving quickly
  - Major and Minor
- Forced Upgrades

Kubernetes Cluster Options

- Managed Kubernetes Cluster are great!
- But...
  - Access to k8s API server options
  - Configuring command line options
  - e.g., webhooks for authz
    `--authorization-webhook-config-file=SOME_FILENAME`

Google Kubernetes Engine

Authentication/Authorization

- Each Cloud Provider has proprietary solution
- A number of OSS/proprietary solutions
  - Insecure
  - Not meeting our requirements
  - `--authorization-webhook-config-file=SOME_FILENAME`
- Solution

  k8s service accounts + oasis auth

Google Kubernetes Engine

Amazon
EKS

Logging/Audit Log

- Each Cloud Provider has proprietary solution
  - Grafana loki to the rescue!
- Audit Logging
  - see *storage*
  - Access to Kube-API-Server Logging
    - see *cluster options*...

Google Kubernetes Engine

Amazon
EKS

Storage

- **Volumes**
  - Different Performance
    - E.g., some offer configurable IOPs
    - Different performance characteristics
  - Resizing on AKS ->

- **Cluster-External Storage**
  - Each Cloud Provider has proprietary solution
    - needed for backup/restore
  - (Current solution) Multiple implementations

- Azure PVC Resizing is “broken”
  - Changes Metadata and Pricing but not filesystem
  - Manual workaround
  - Works fine on Azure, but AKS...
  - Great support by Azure Team!!!

Networking for Multi-tenancy

- Oasis data clusters are multi-tenant
  - Single VPC for entire data cluster
- Strong need for network separation
  - Separate deployments from each other
  - Separate deployments from platform

- Solution: Cilium

Cilium: The Promise

- Cilium enables Network Policy Rules
  - E.g. Pods in namespace A can talk to pods in namespace B
- Rules are directional
- Rules are implemented using BPF
  - Performance
- Cloud Provider independent
  - Except setup

Cilium: The Reality*

- When it works, it is great
- Cloud Provider independent
  - In reality not always
  - E.g. PodCIDR is set different on AWS/GCP.
    - Workaround: Currently manually drain node
  - Result; Almost no problems on GCP, relative frequent issues on AWS
    - Is improving with Cilium 1.7
    - Support for Azure is relatively new

* It is still great :-)

“Container Runtimes”

Some structure

Container Runtimes

“Secure” Container

Conclusions

Experiences

- AWS is clearly “older” than GCP (or Azure)
  - API is showing its age
- AWS is clearly more stable than GCP and Azure
  - e.g., network outages
  - But slower to bootstrap
  - except Managed k8s: GKE > EKS
- AWS has (much) more features than GCP
  - E.g. provisioned IOPS
- EKS is (much) less user friendly than GKE
- AKS is clearly less mature than GCP or AWS
-

Thanks for listening!

ArangoDB OASIS

Test-drive Oasis
14-days for free

ArangoDB

- @arangodb
- https://www.arangodb.com/

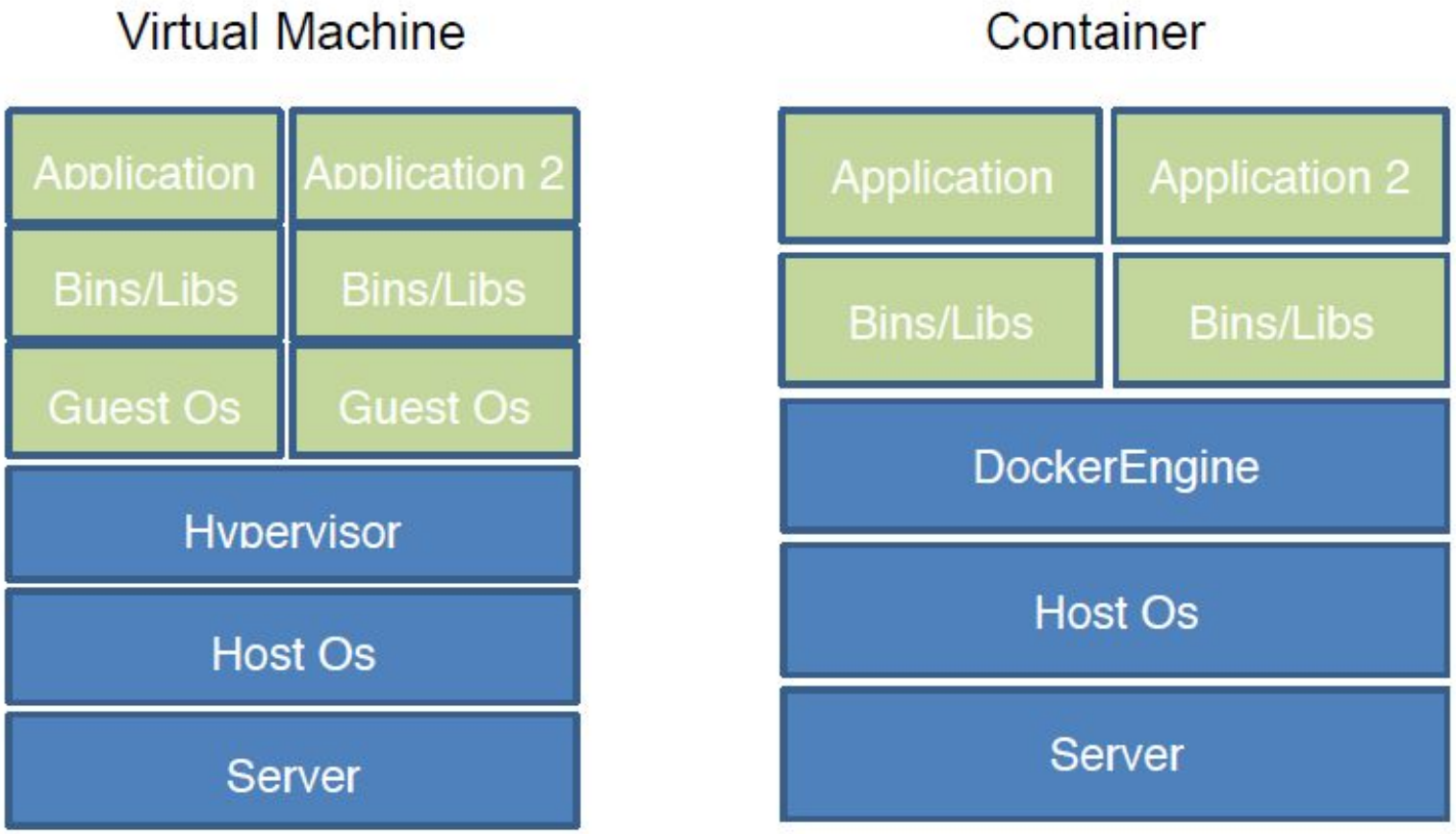
docker pull arangodb

Containers

© Gerard Julien/AFP

There are no containers...

Container vs VM



Container Internals

```
joerg@docker-test:~$ docker exec -it 737395b89b4a bash
root@737395b89b4a:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  32652  5240 ?        Ss   16:39   0:00 nginx: master process nginx -g daemon
off;
nginx        6  0.0  0.0  33108  2348 ?        S    16:40   0:00 nginx: worker process
root       255  1.0  0.0  18128  3168 pts/0    Ss   16:51   0:00 bash
root       260  0.0  0.0  36632  2868 pts/0    R+   16:51   0:00 ps aux

x
joerg@docker-test:~$ ps faux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
message+ 26869  0.0  0.1  45248  3816 ?        Ss   16:37   0:00 /usr/bin/containerd
root      4491  0.0  0.1  10728  5172 ?        Sl   16:39   0:00  \_ containerd-shim -namespace moby
-workdir /var/
root      4509  0.0  0.1  32652  5240 ?        Ss   16:39   0:00      \_ nginx: master process nginx -g
daemon off;
systemd+  4541  0.0  0.0  33108  2348 ?        S    16:40   0:00          \_ nginx: worker process
root      2140  0.2  1.9 550876 72948 ?        Ssl  16:38   0:02 /usr/bin/dockerd -H fd://
--containerd=/
joerg     4619  0.0  0.1  64844  6184 ?        Ss   16:42   0:00 /lib/systemd/systemd --user
```

Cgroups and Namespaces

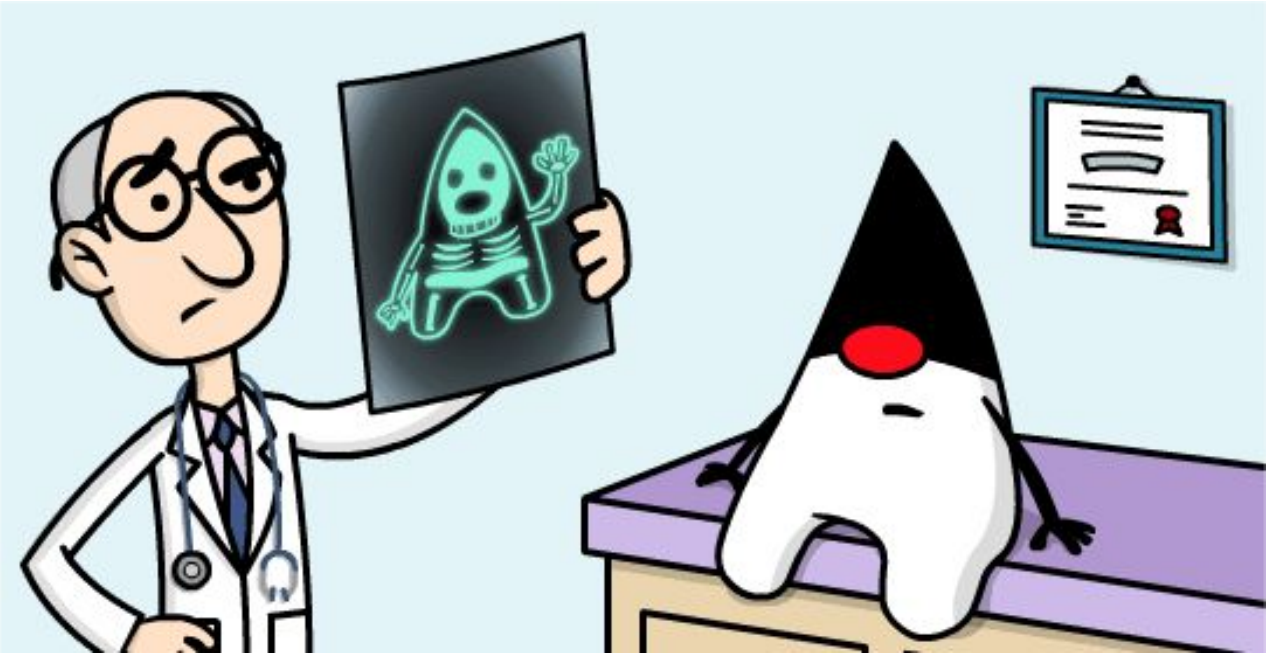
Namespaces provide isolation:

- pid (processes)
- net (network interfaces, routing...)
- ipc (System V IPC)
- mnt (mount points, filesystems)
- uts (hostname)
- user (UIDs)

Control groups control resources:

- cpu (CPU shares)
- cpuacct
- cpuset (limit processes to a CPU)
- memory (swap, dirty pages)
- blkio (throttle reads/writes)
- devices
- net_cls, net_prio: control packet class and priority
- freezer

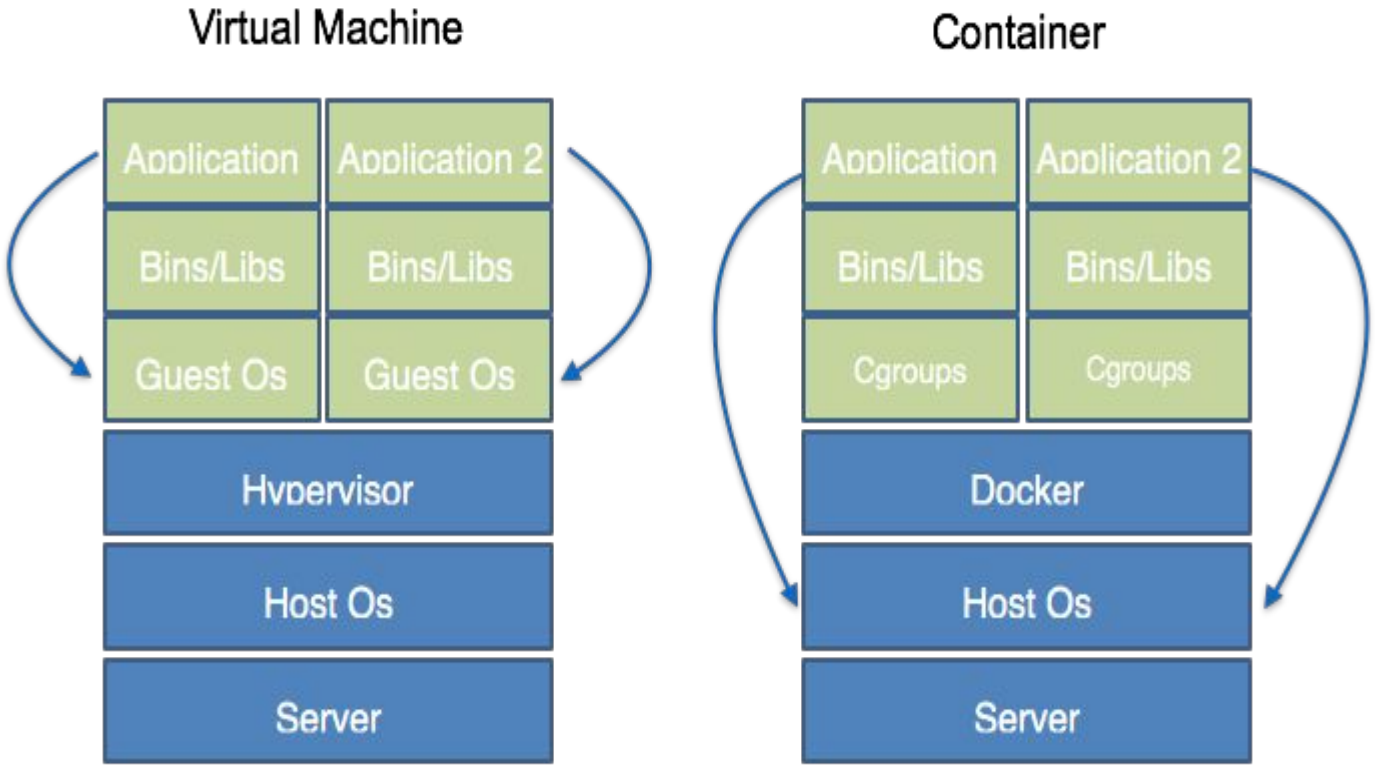
Implications - JRE 8



https://jaxenter.com/nobody-puts-java-container-139373.html

Implications - JRE 8

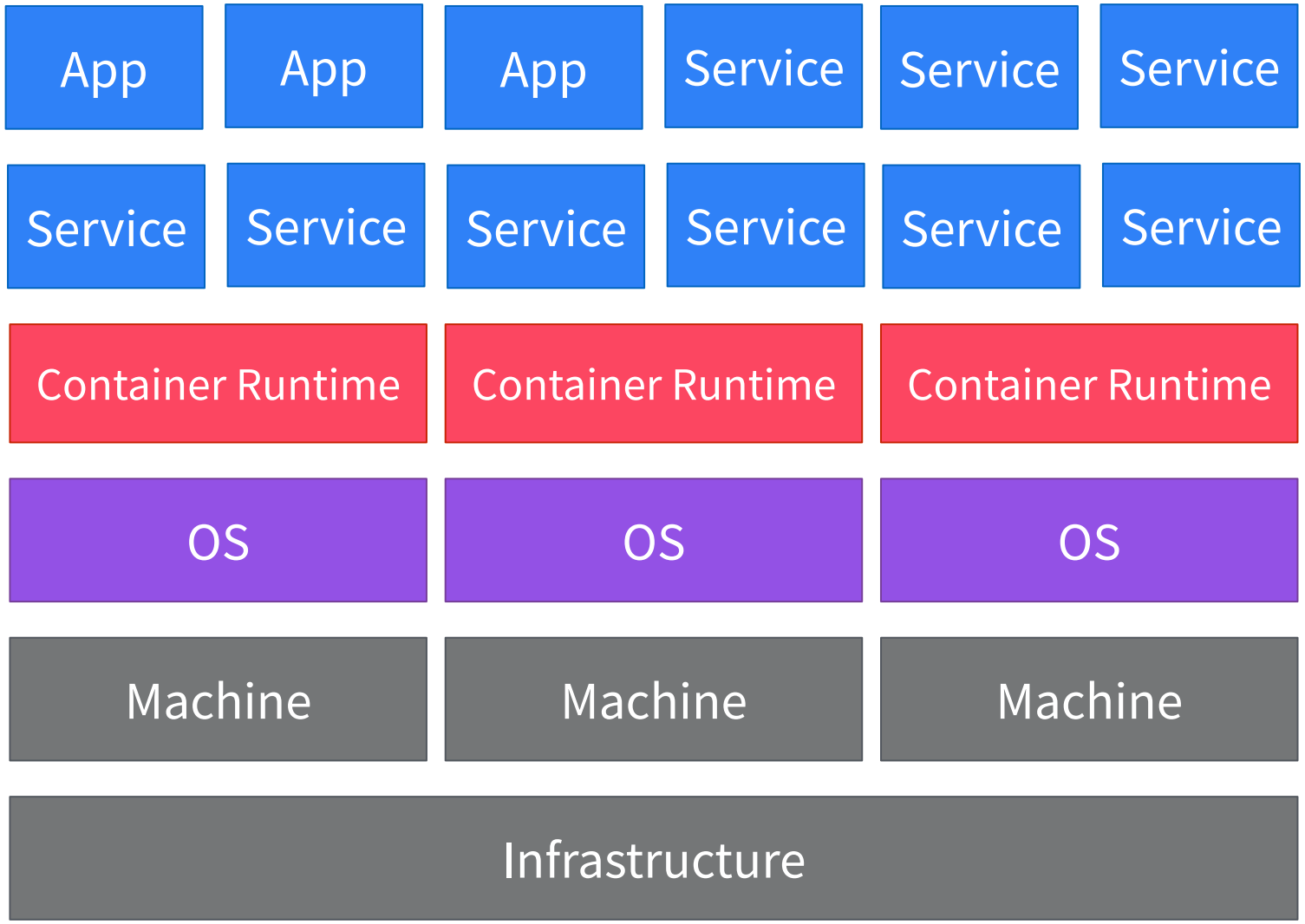
- # of JIT compiler threads
- # Garbage Collection threads
- # of thread in the common fork-join pool
- …



https://jaxenter.com/nobody-puts-java-container-139373.html

CONTAINERS

- Rapid deployment
- Dependency vendoring
- Container image repositories
- Spreadsheet scheduling



Container Orchestration

CONTAINER

RESOURCE

SERVICE MANAGEMENT

Container Orchestration

CONTAINER SCHEDULING

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

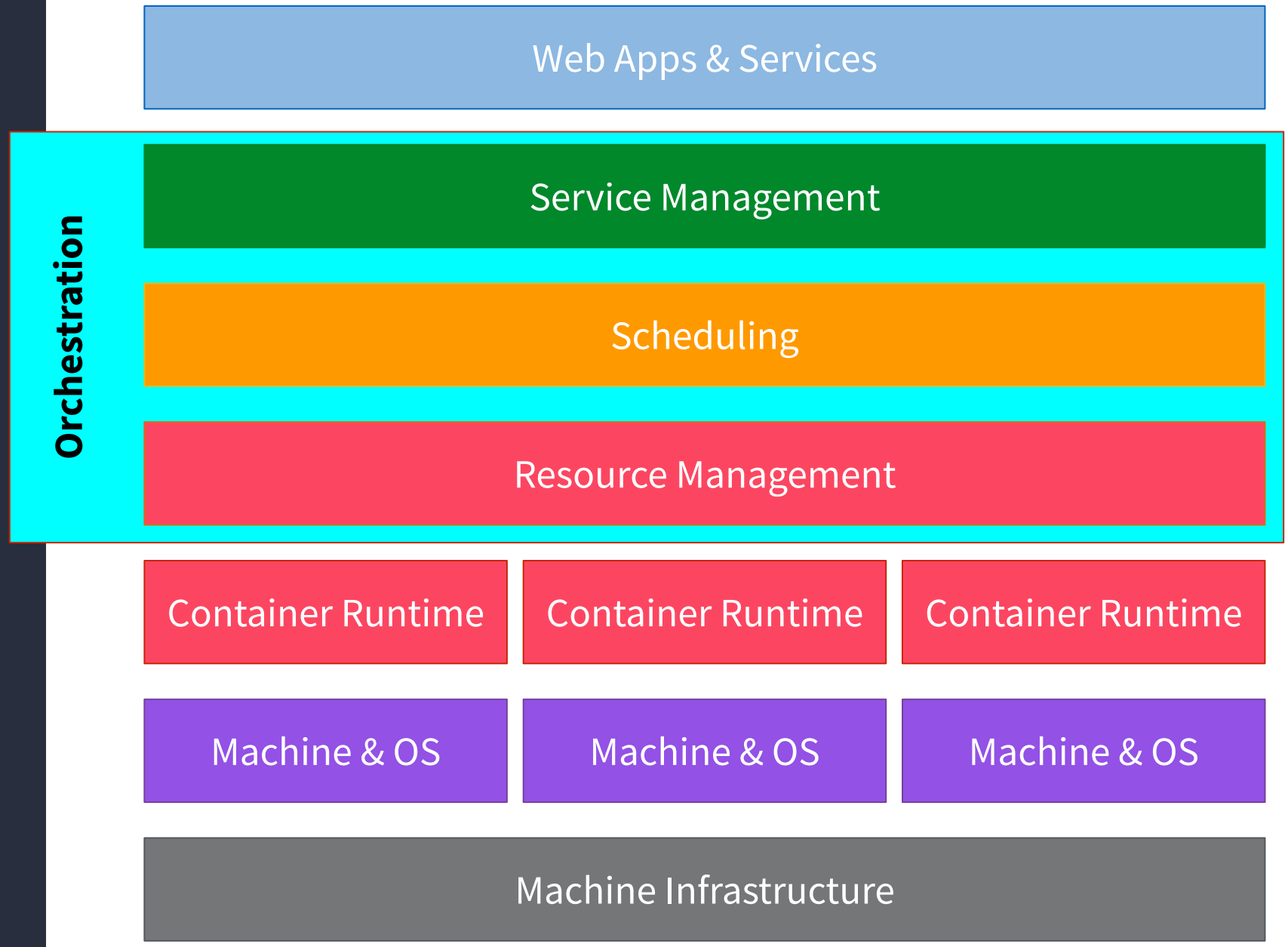
RESOURCE MANAGEMENT

- Memory
- CPU
- GPU
- Volumes
- Ports
- IPs
- Images/Artifacts

SERVICE MANAGEMENT

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

CONTAINER ORCHESTRATION



Kubernetes

CONTAINER SCHEDULING

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

RESOURCE MANAGEMENT

- Memory
- CPU
- GPU
- Volumes
- Ports
- IPs
- Images/Artifacts

SERVICE MANAGEMENT

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

What Does “Kubernetes” Mean?

Greek for “pilot” or
“Helmsman of a ship”

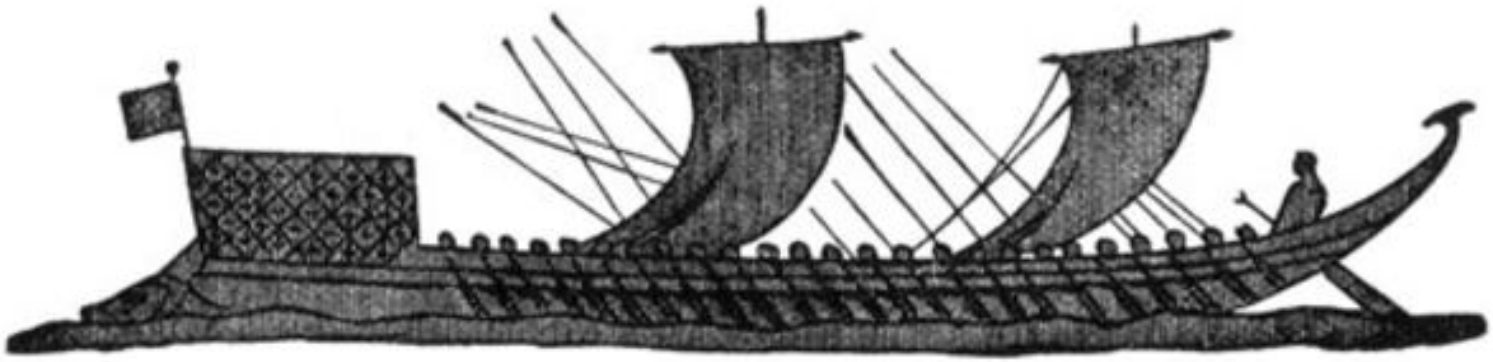
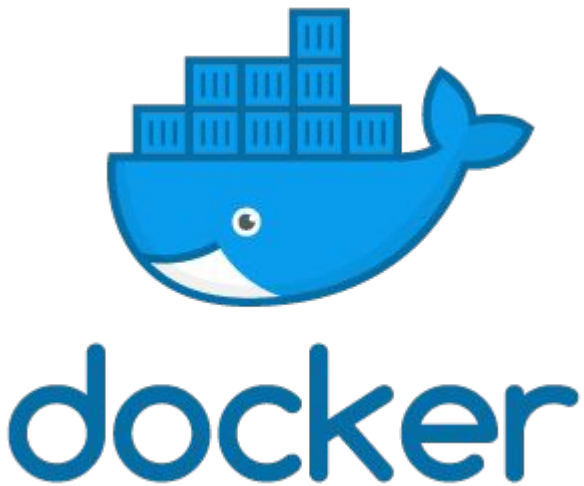


Image Source

Container



How did we end up here?

Container Runtimes

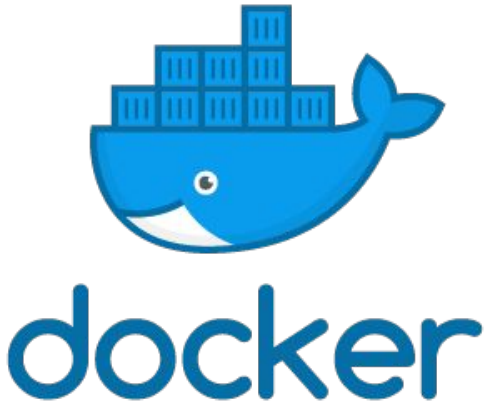
podman

“Secure Container

Firecracker

gVisor

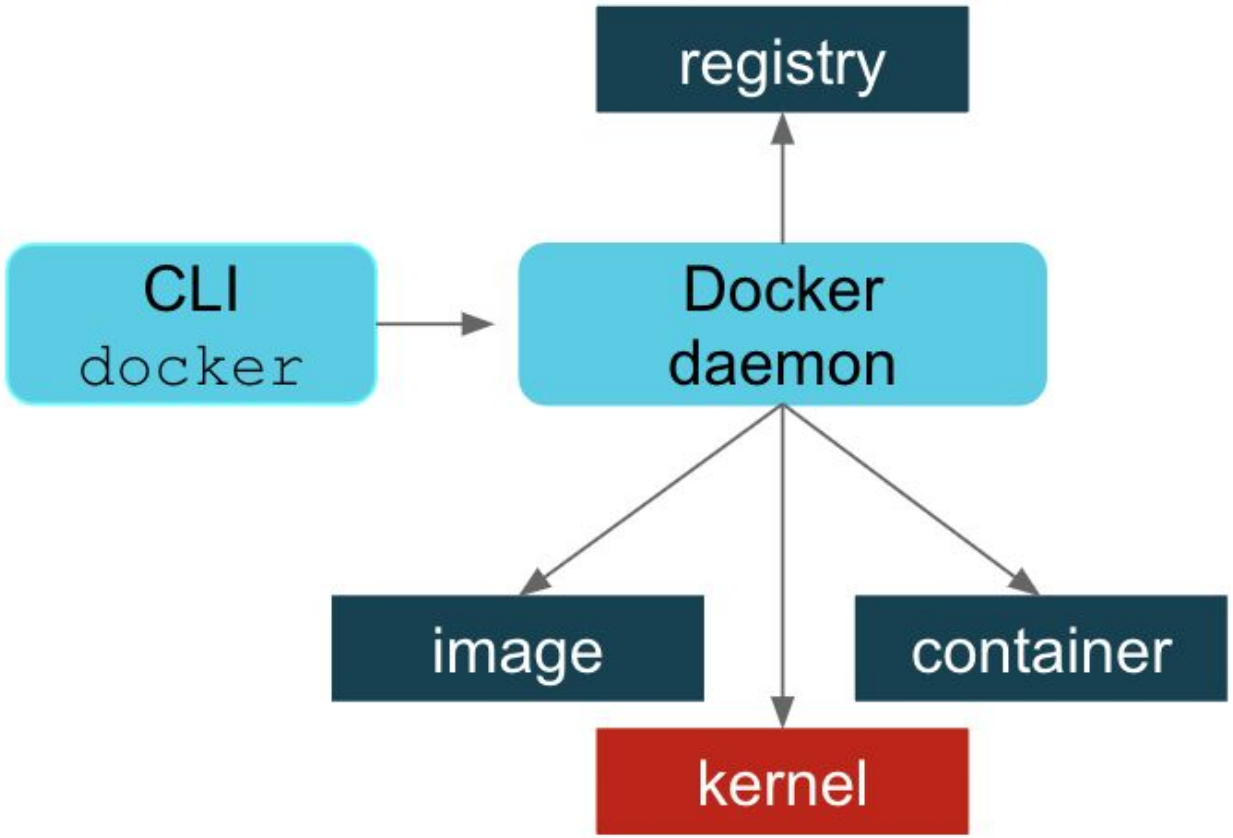
Docker



All-In-One Solution

- A container image format
- A method for building container images (Dockerfile/docker build)
- A way to manage container images (docker images, docker rm , etc.)
- A way to manage instances of containers (docker ps, docker rm , etc.)
- A way to share container images (docker push/pull)
- A way to run containers (docker run)

Docker Daemon



rkt

rkt - the pod-native container engine

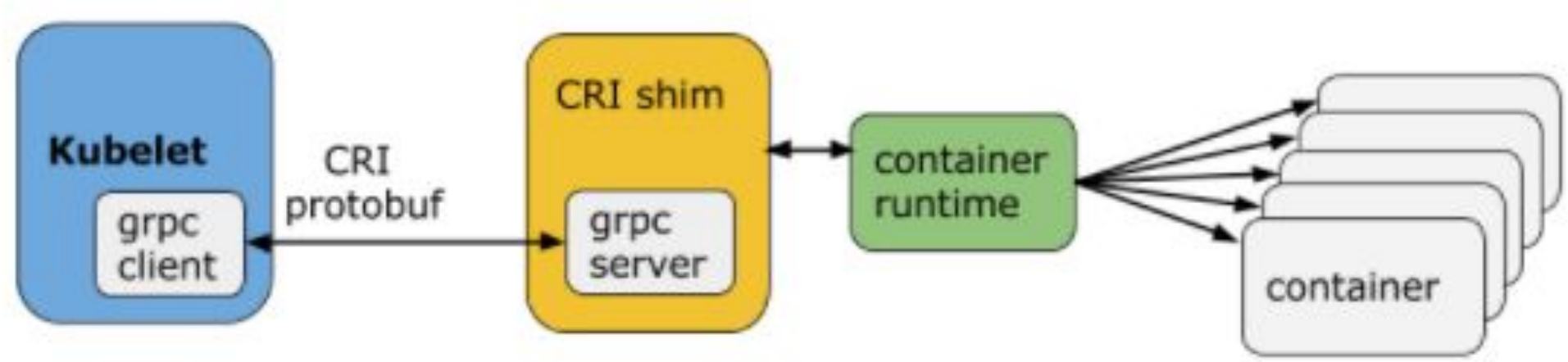
rkt (pronounced like a *"rocket"*) is a CLI for running application containers on Linux. rkt is designed to be secure, composable, and standards-based.

Some of rkt's key features and goals include:

- *Pod-native*: rkt's basic unit of execution is a pod, linking together resources and user applications in a self-contained environment.

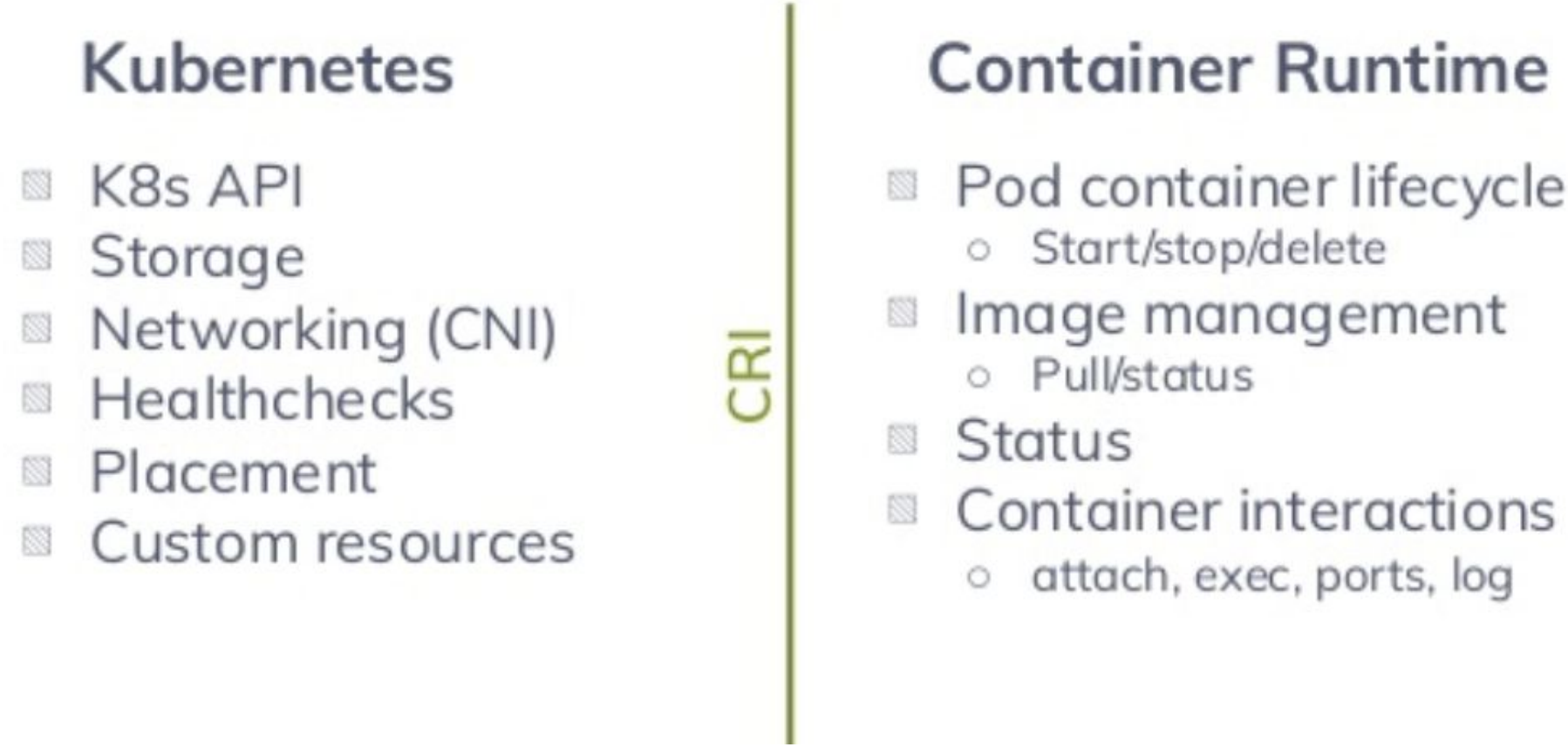
https://github.com/rkt/rkt

CRI (Container Runtime Interface)



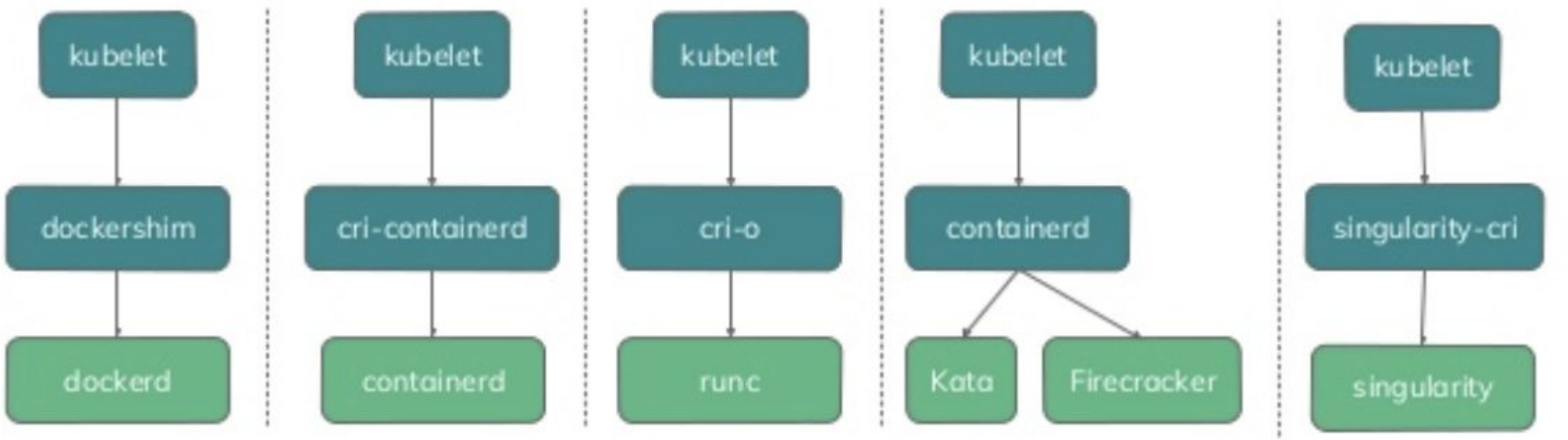
https://github.com/kubernetes/cri-api/

CRI (Container Runtime Interface)

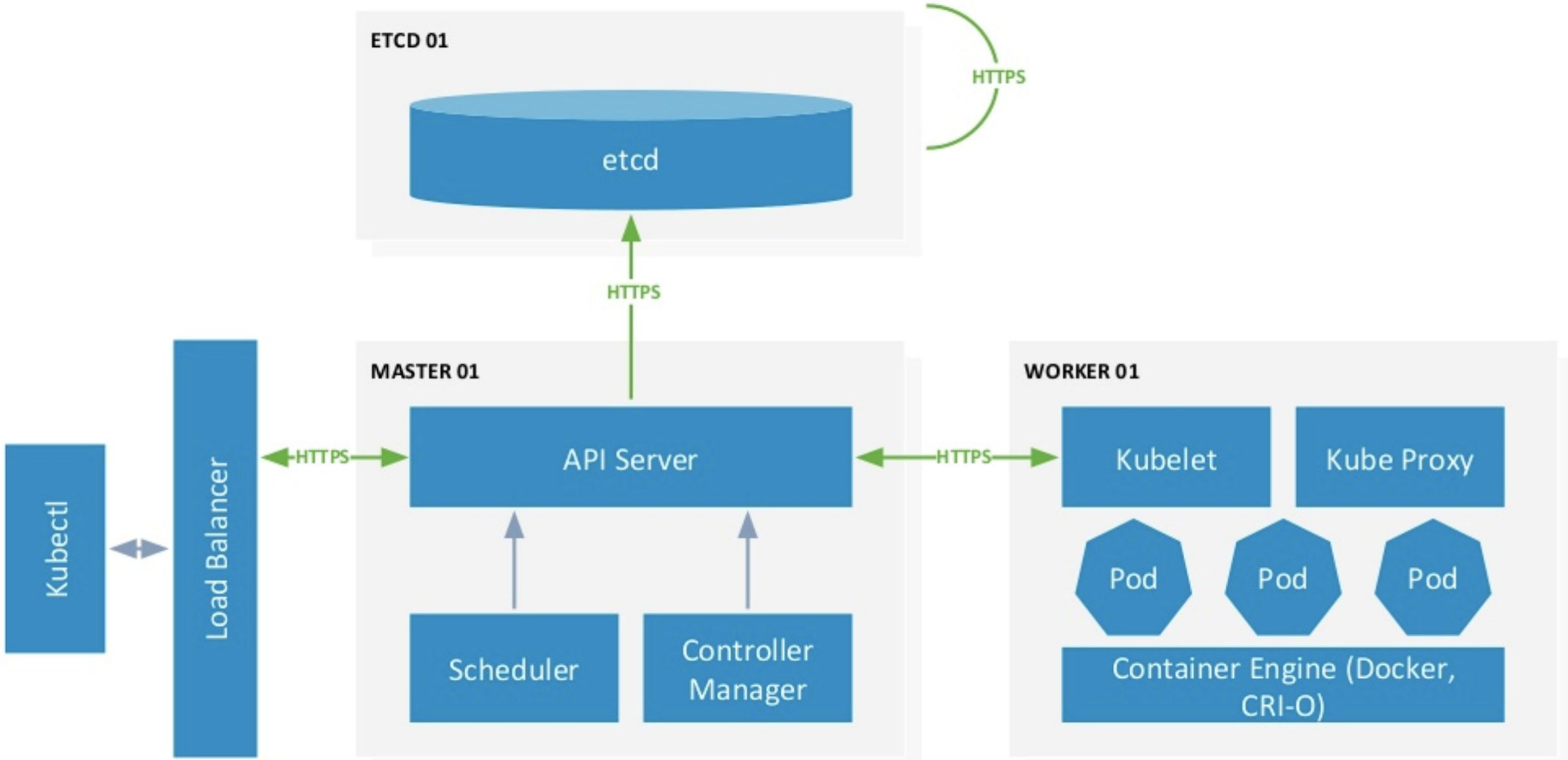


https://www.slideshare.net/PhilEstes/lets-try-every-cri-runtime-available-for-kubernetes

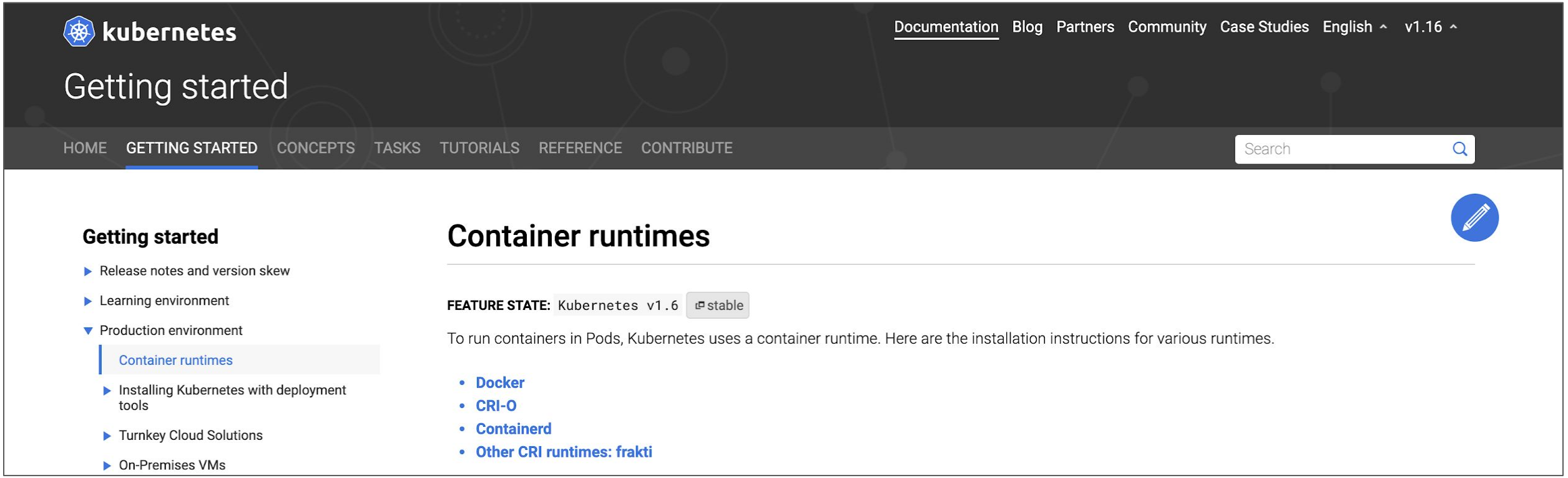
CRI (Container Runtime Interface)



https://www.slideshare.net/PhilEstes/lets-try-every-cri-runtime-available-for-kubernetes

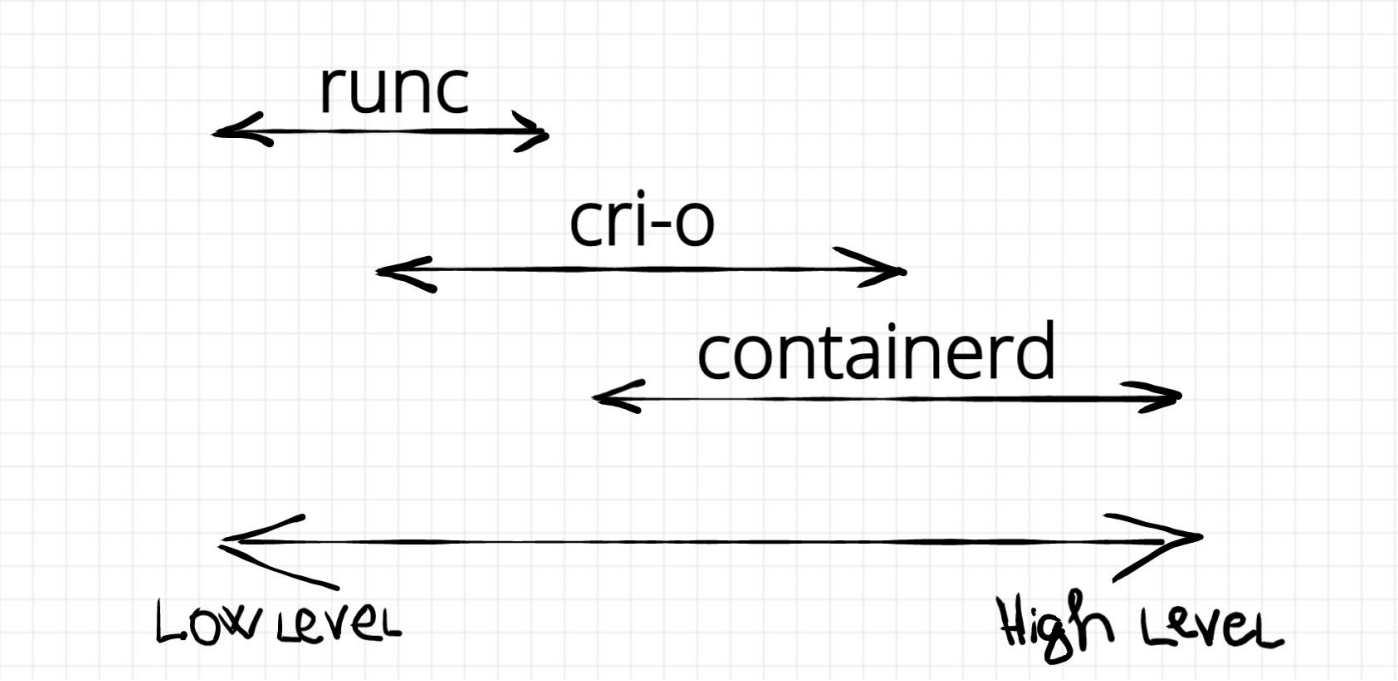

ETCD 01
etcd
HTTPS
HTTPS
MASTER 01
Kubectl
Load Balancer
HTTPS
API Server
Scheduler
Controller Manager
HTTPS
WORKER 01
Kubelet
Kube Proxy
Pod
Pod
Pod
Container Engine (Docker, CRI-O)

Container Runtimes



https://kubernetes.io/docs/setup/production-environment/container-runtimes/

Container Runtimes

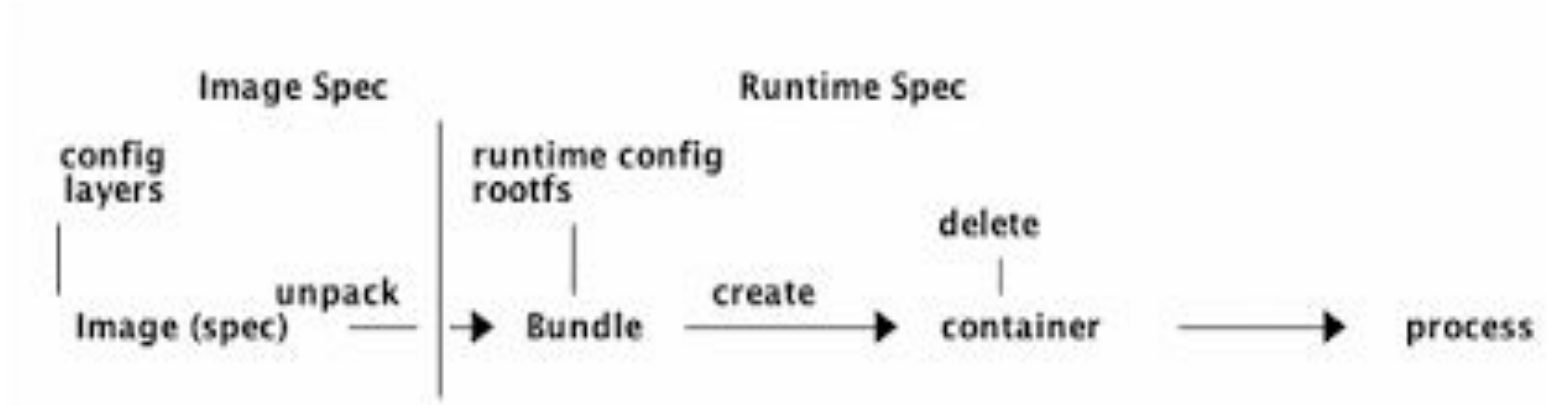


https://iximiuz.com/en/posts/journey-from-containerization-to-orchestration-and-beyond

OCI (Open Container Initiative)

Open industry standards around container formats and runtime

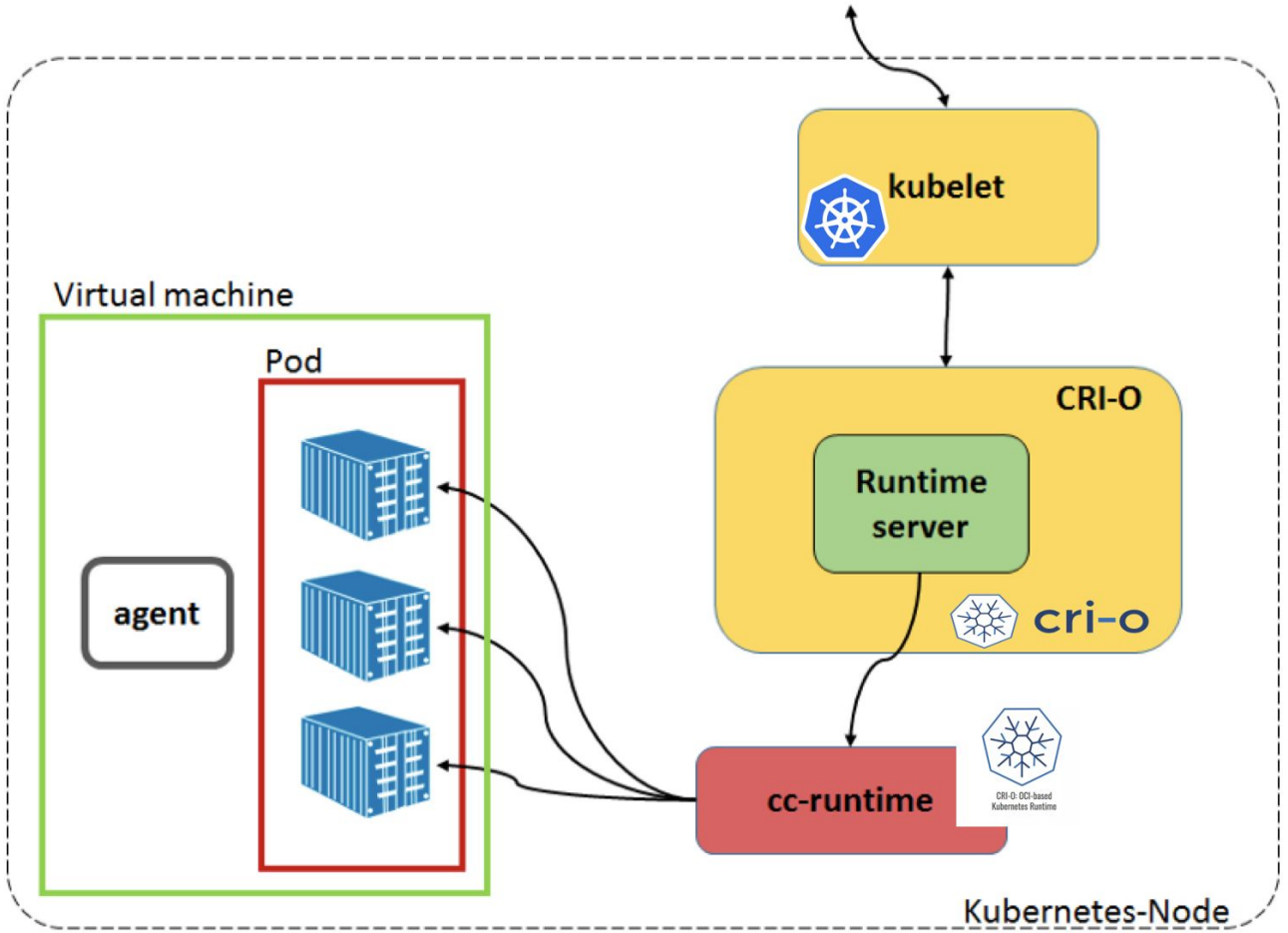
- docker run example.com/org/app:v1.0.0
- rkt run example.com/org/app,version=v1.0.0



https://www.opencontainers.org/

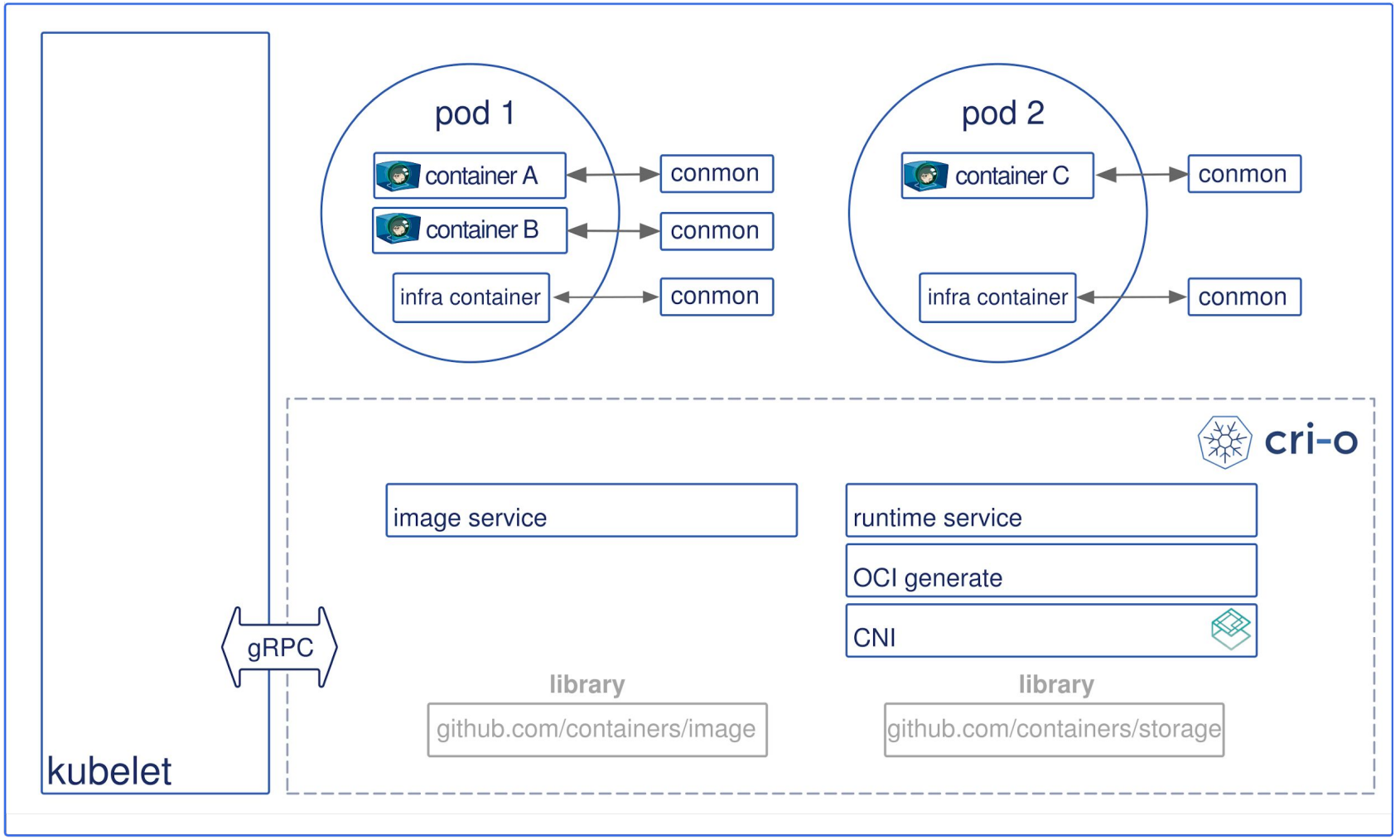
CRI-O

OCI-based implementation
of CRI



https://cri-o.io/

CRI-0

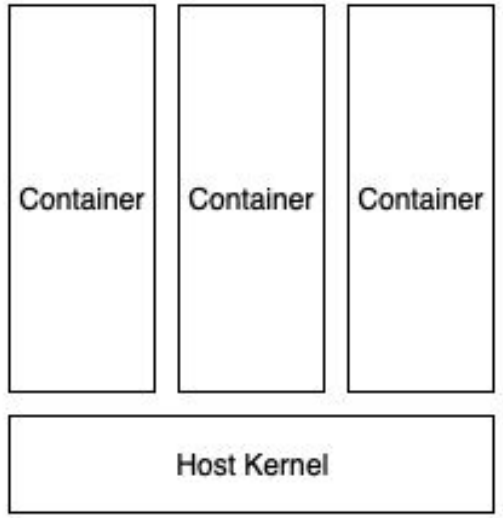


https://cri-o.io/

runc (libcontainer)

CLI tool for spawning and **running** containers

- Default (low-level) runtime
- Original Docker runtime
- OCI project

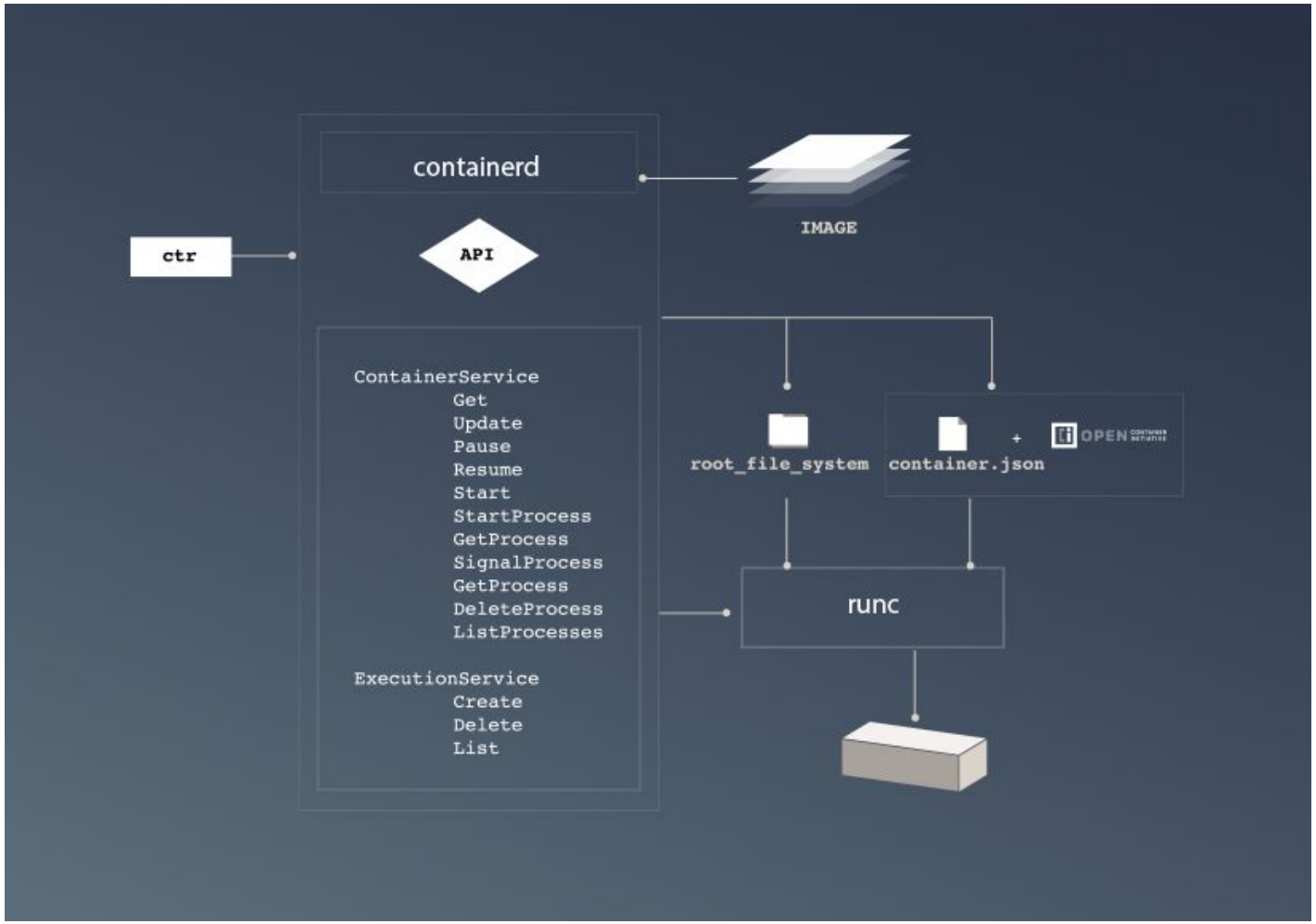


https://github.com/opencontainers/runc

Container-d

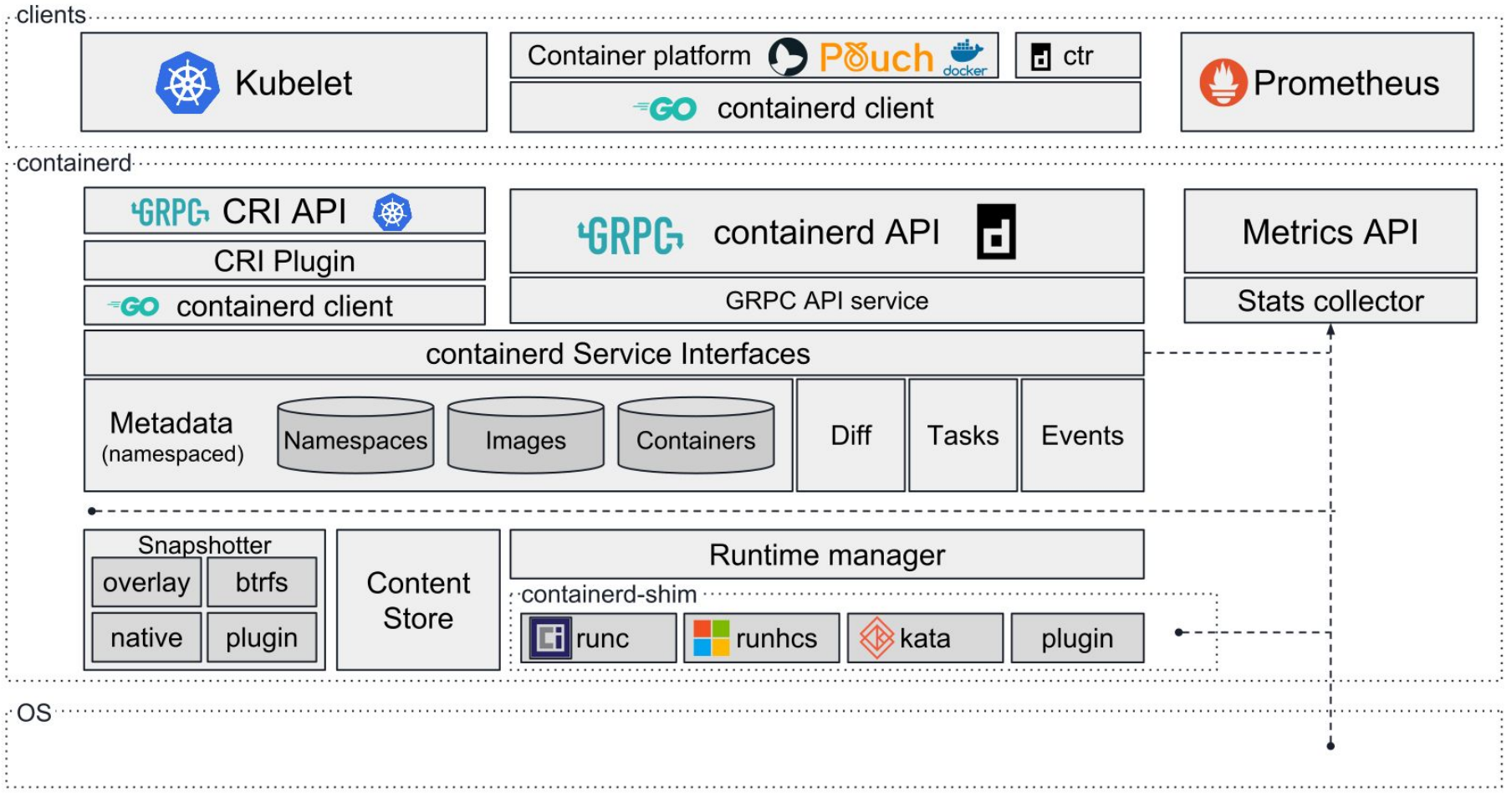
Daemon managing the complete container lifecycle

- image transfer
- storage
- container execution
- supervision
- storage to network attachments and beyond.



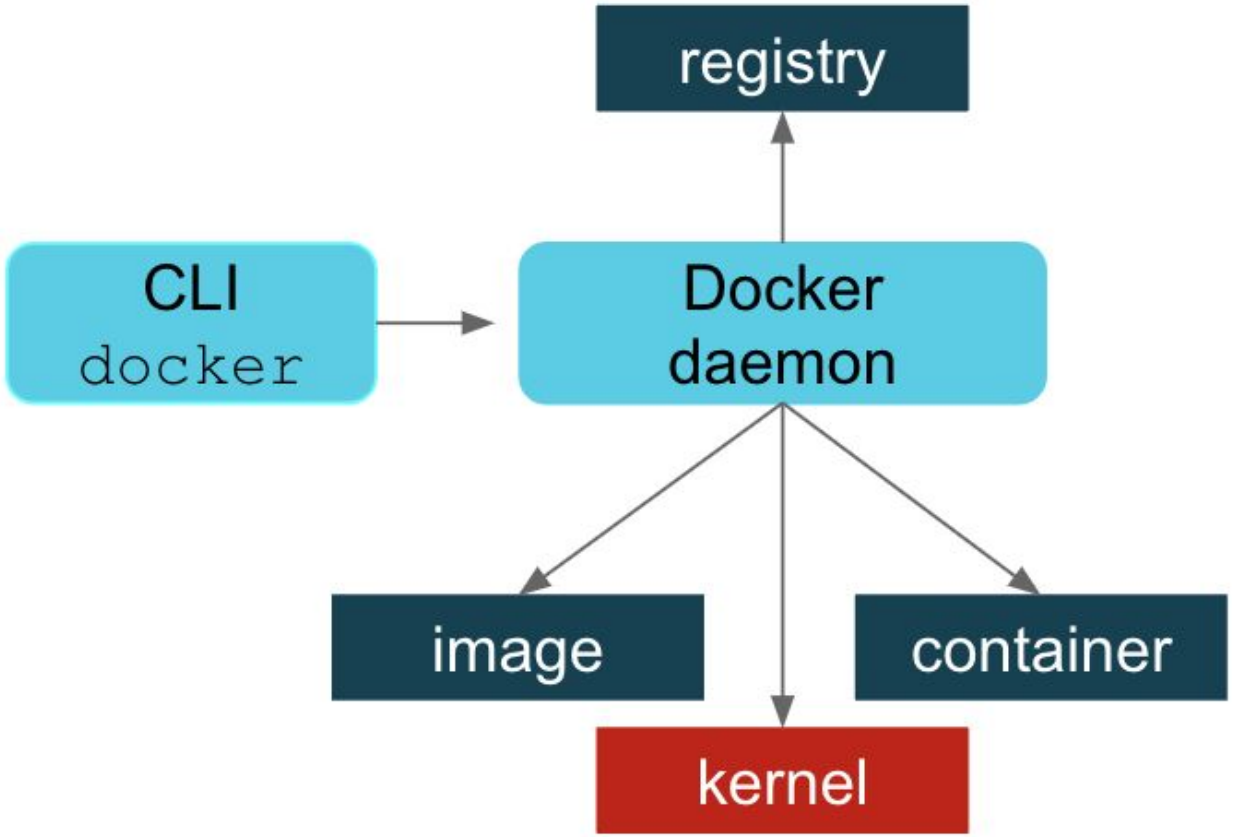
https://containerd.io/

Container-d



https://containerd.io/

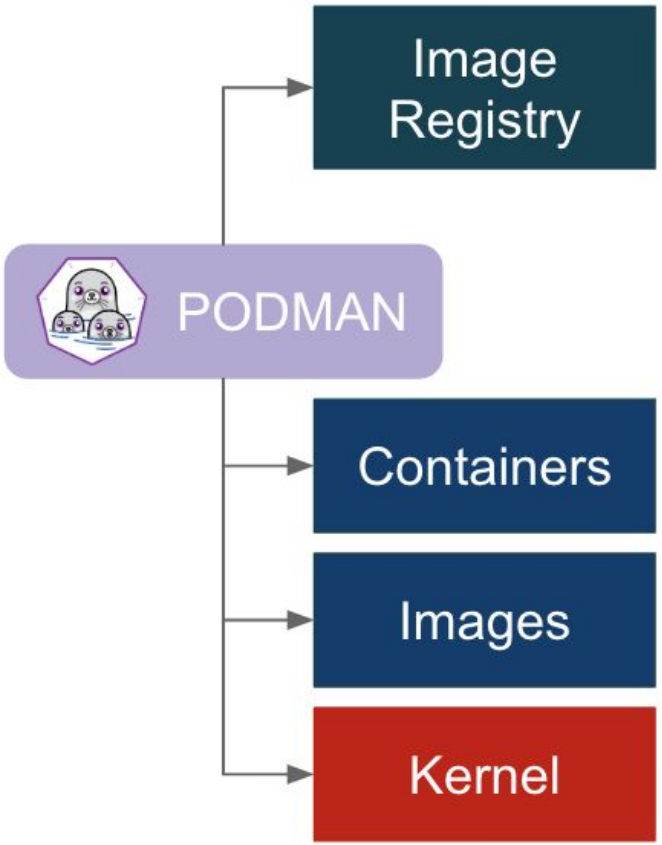
Recall: Docker Daemon



Podman

Daemonless container engine for running OCI containers

- docker compatible
  alias docker=podman



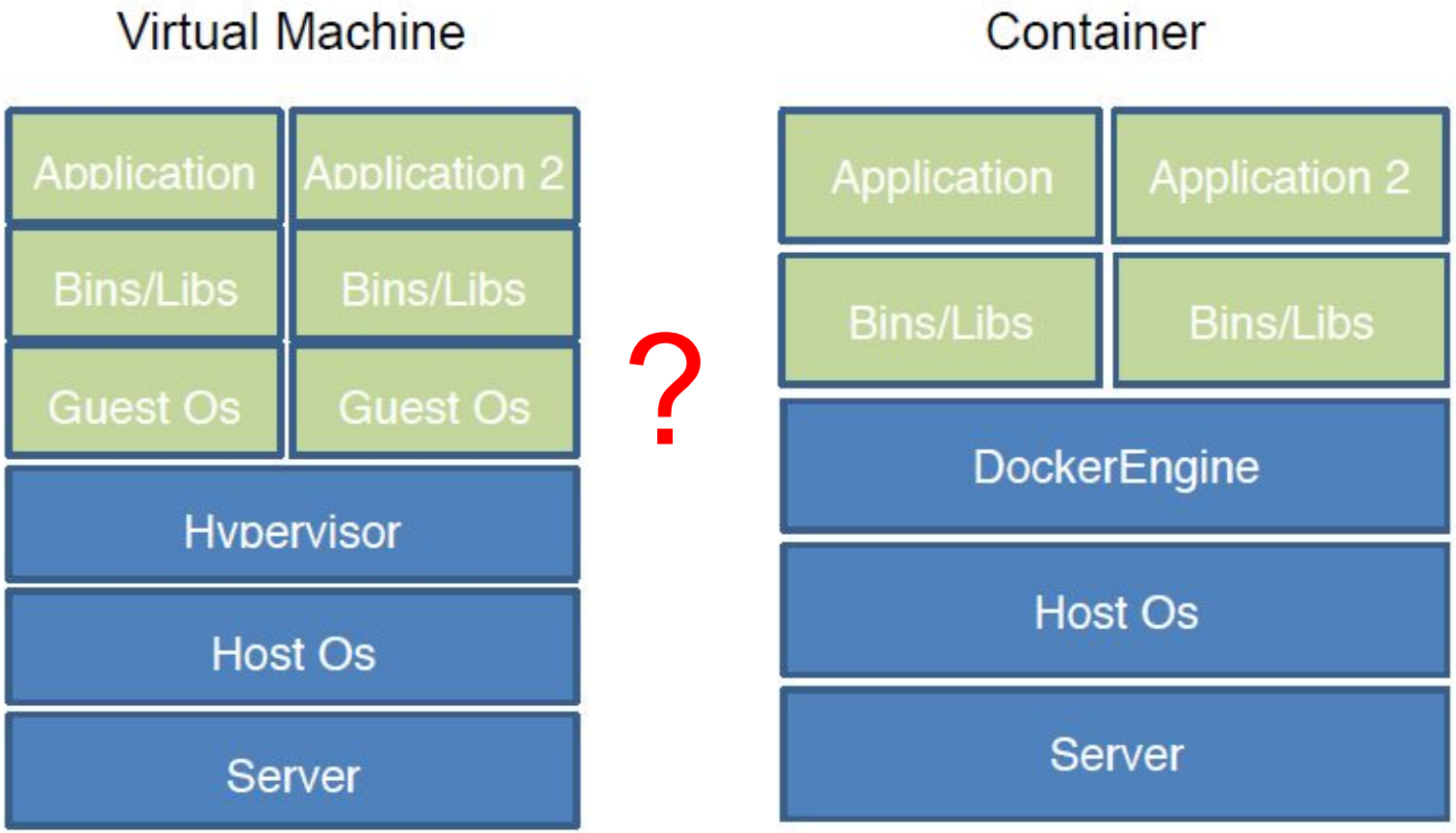
https://podman.io/

Security

CVE-ID	
CVE-2016-5195	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
Race condition in mm/gup.c in the Linux kernel 2.x through 4.x before 4.8.3 allows local users to gain privileges by leveraging incorrect handling of a copy-on-write (COW) feature to write to a read-only memory mapping, as exploited in the wild in October 2016, aka "Dirty COW."	

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-5195
https://blog.paranoidsoftware.com/dirty-cow-cve-2016-5195-docker-container-escape/

Container vs VM (revisited)



How did we end up here?

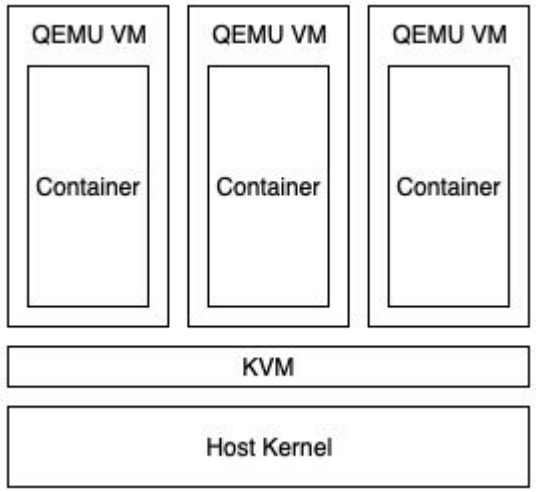
Container Runtimes

“Secure Container

Kata Containers

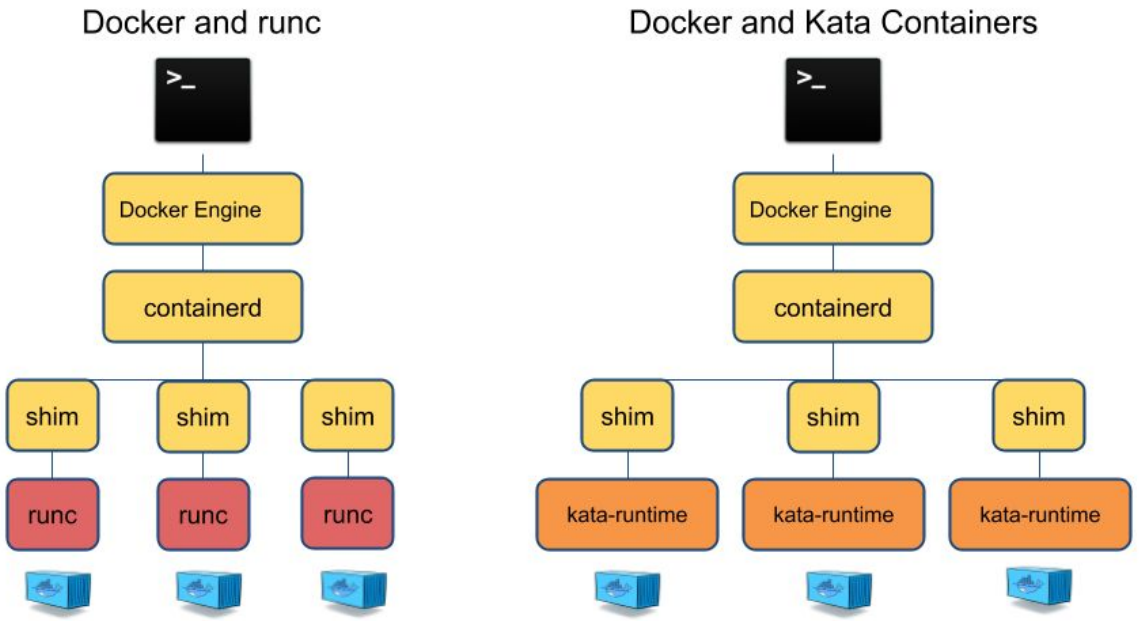
Lightweight virtual machines

- KVM Hypervisor
- QEMU* VM
- Kernel Samepage Merging
- Overhead
  - Disk
  - Memory
  - CPU



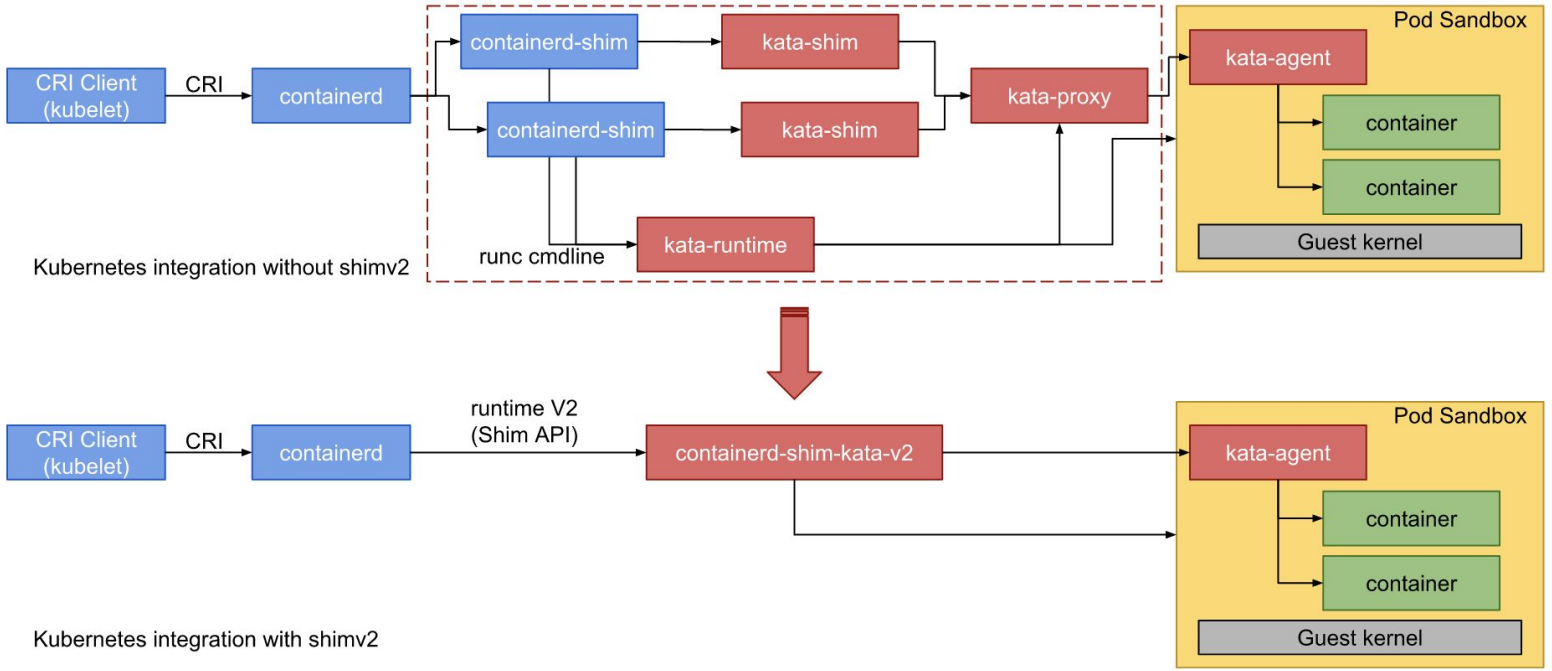
https://katacontainers.io/

Kata Containers



https://katacontainers.io/

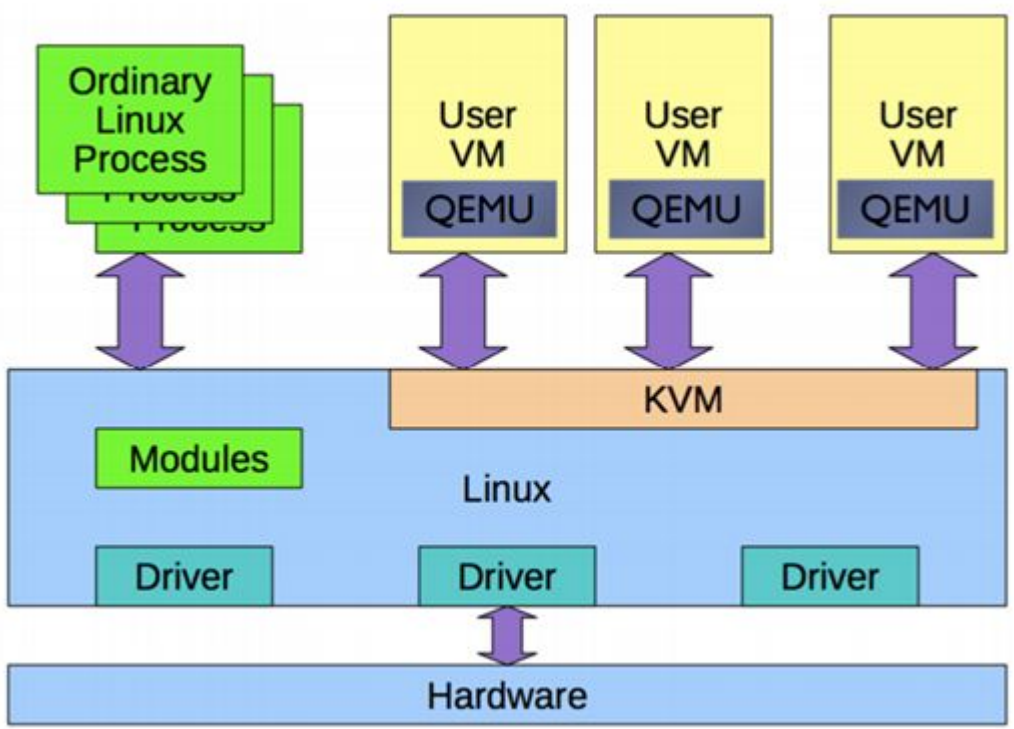
Kata Containers & container-d



Firecracker

Firecracker is an open source virtualization technology that is purpose-built for creating and managing secure, multi-tenant container and function-based services.

QEMU

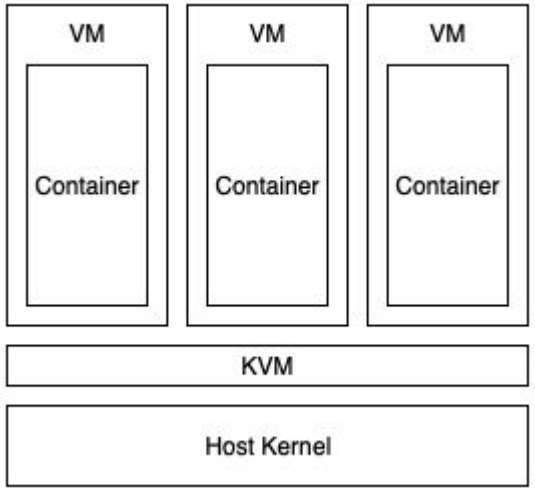


https://medium.com/@jain.sm/kvm-and-qemu-as-linux-hypervisor-18271376449

Firecracker

Minimal VM

- Alternative to QEMU
- only 5 emulated devices virtio-net, virtio-block, virtio-vsock, serial console, minimal keyboard controller
- < 125 ms startup time
- < 5 MiB memory footprint.
- Needs to be specifically compiled



Kata meets Firecracker

kata-containers / documentation

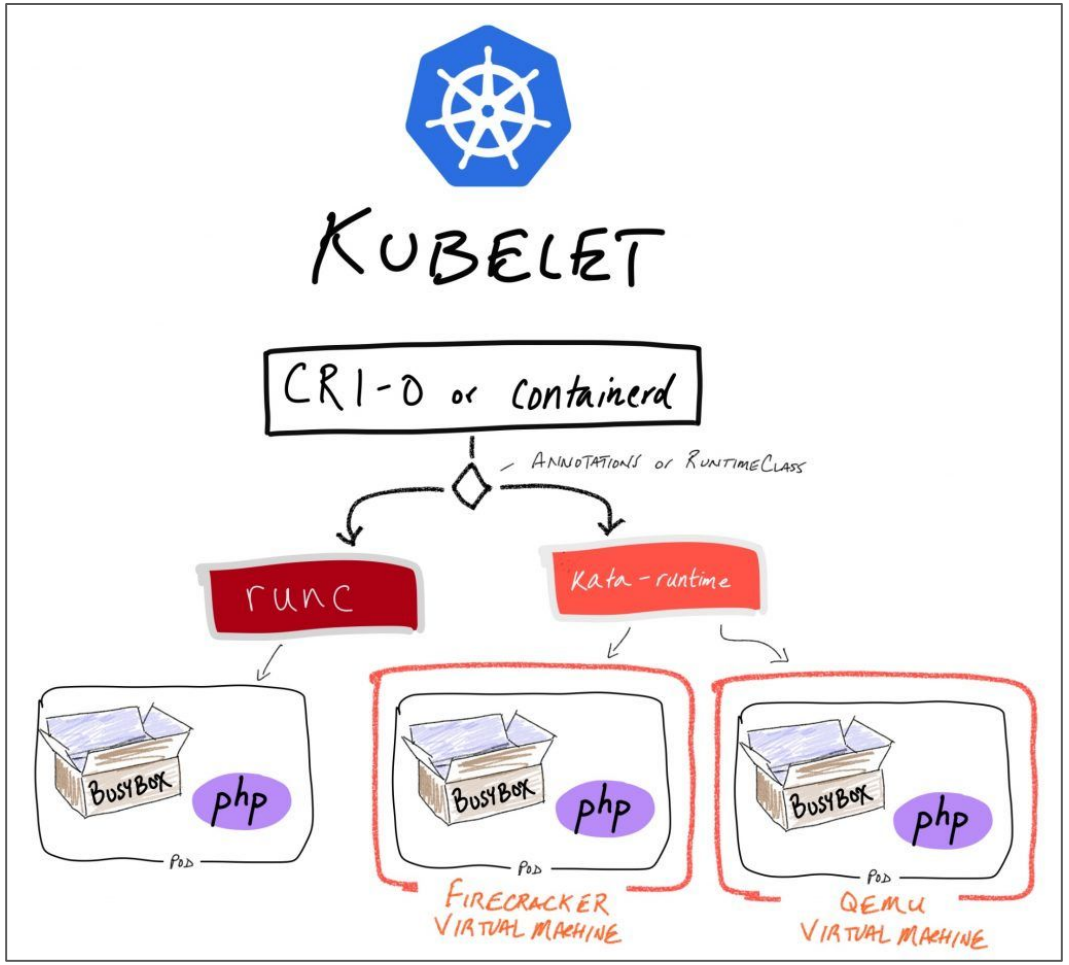
Watch 51 | Star 298 | Fork 177

Code | Issues 75 | Pull requests 6 | Actions | Projects 0 | Wiki | Security | Insights

Initial release of Kata Containers with Firecracker support

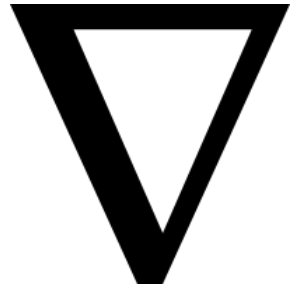
Graham Whaley edited this page on May 14 · 19 revisions

Kata meets Firecracker



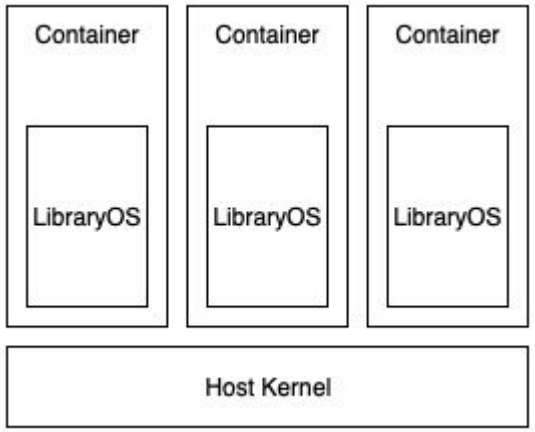
https://aws.amazon.com/blogs/opensource/kata-containers-1-5-firecracker-support/

Nabla Containers



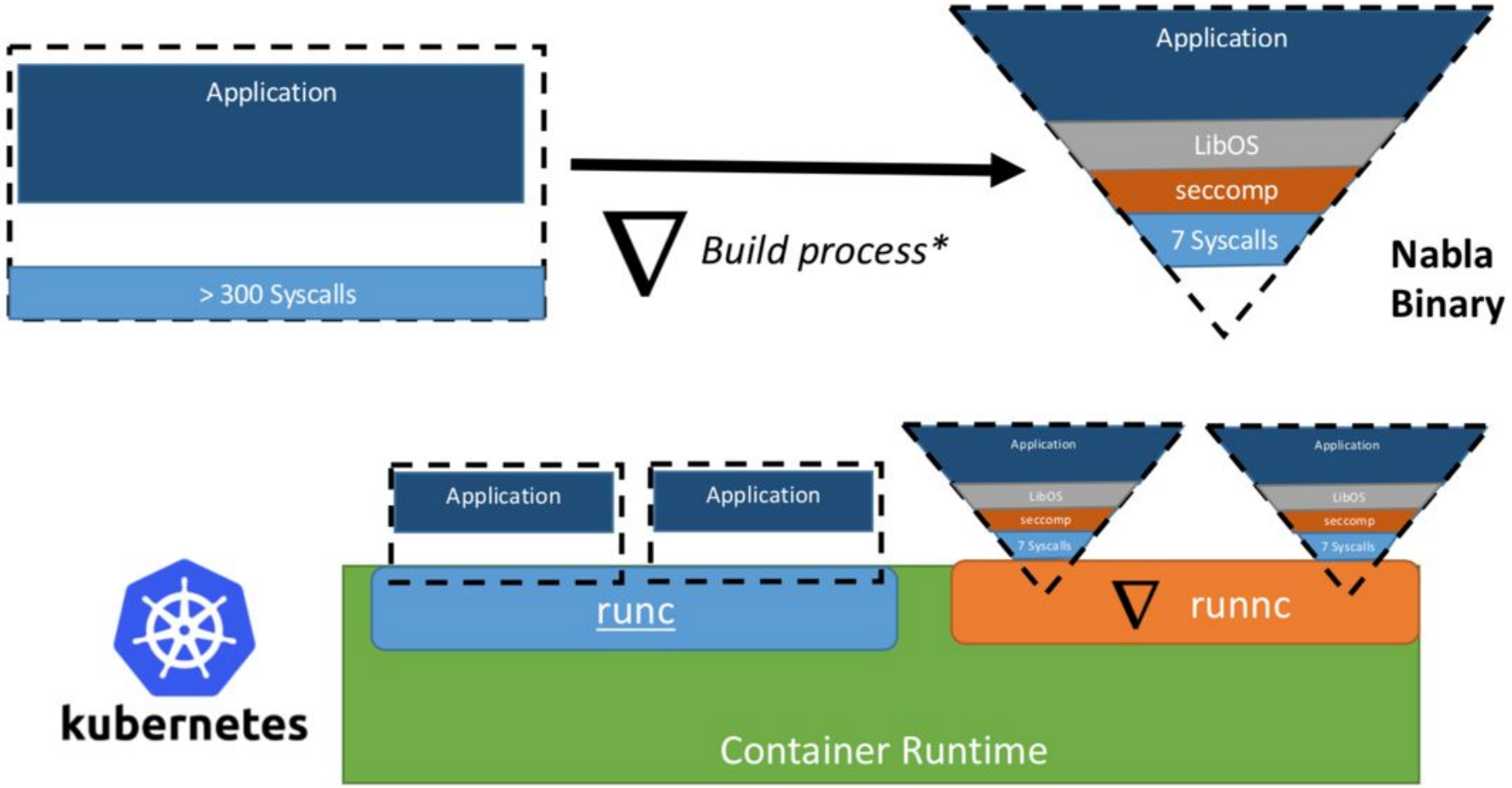
Reimplement OS in Userspace

- Library OS
- Unikernel
- Few native syscalls (secomp)
  - read, write, exit_group, clock_gettime, ppoll, pwrite64, and pread64



https://github.com/google/gvisor

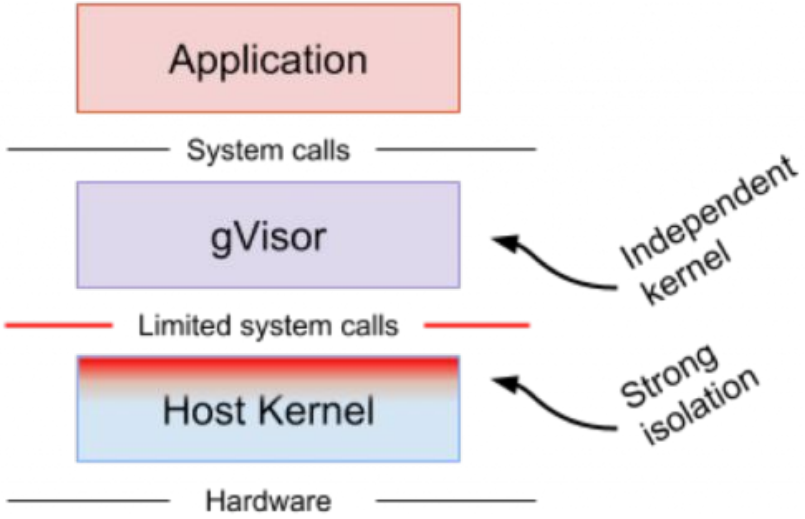
Nabla Containers

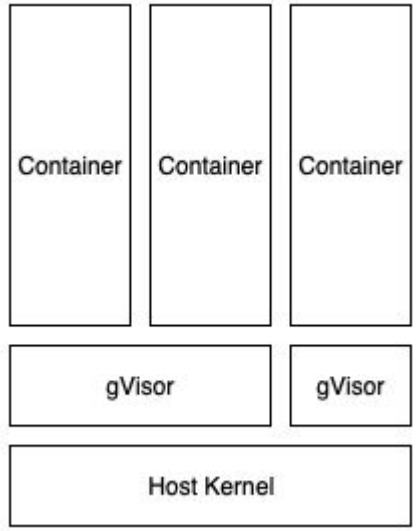


https://nabla-containers.github.io/

GVisor

- **User Space Kernel**
  - Reimplement kernel in Golang
  - Limited sycalls to actual kernel ~ 80
  - Performance impact
    - Network
    - Storage
  - Hardware Passthrough
    - GPU





https://github.com/google/gvisor

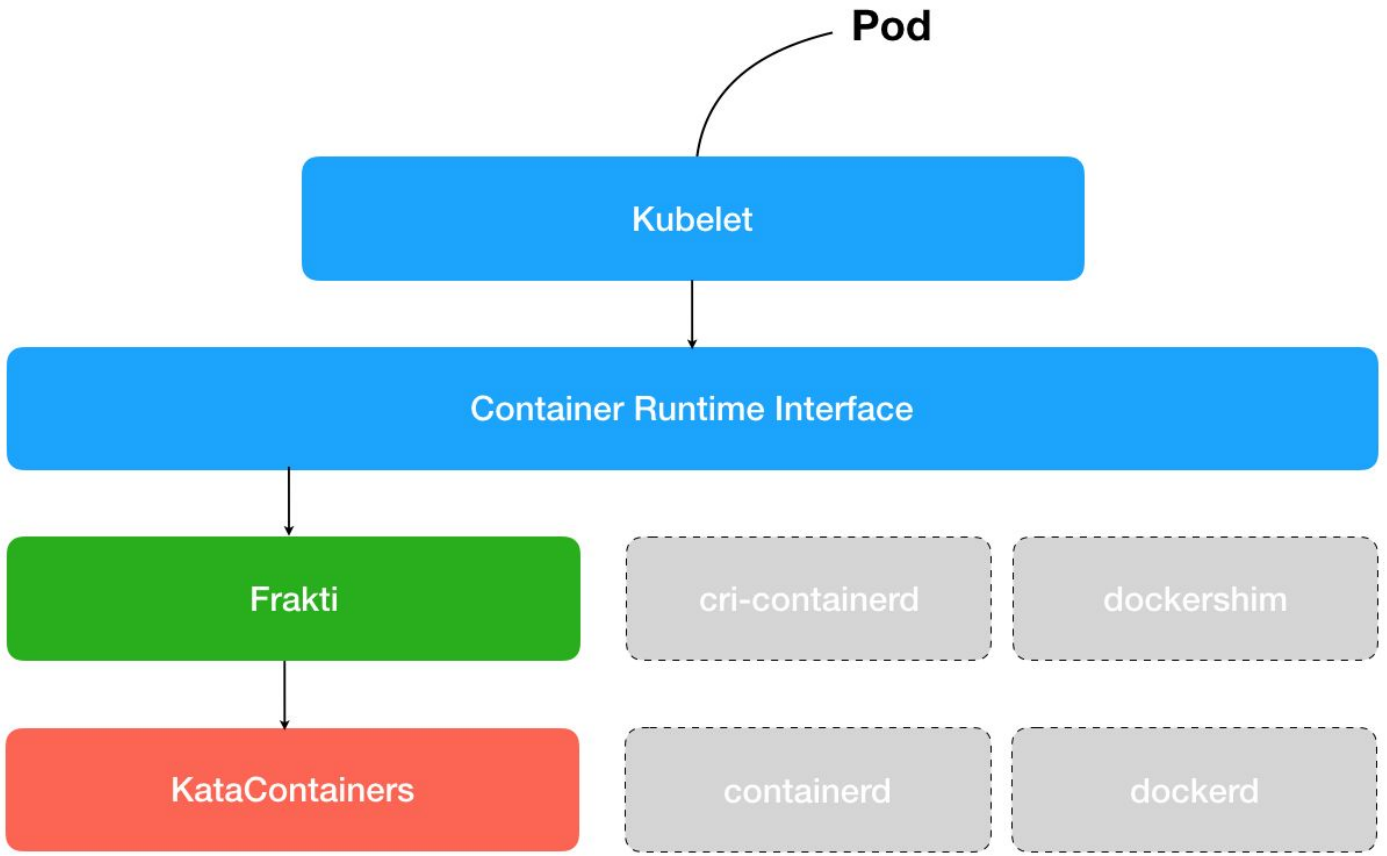
runV

Hypervisor based OCI runtime

- Ignores certain OCI spec fields
  - Namespace
  - Capability
  - Device
  - linux and mount fields

https://github.com/hyperhq/runv

Frakti



https://github.com/kubernetes/frakti

Comparison

Native Container	**User Space Kernel**	**Virtual Machine**
Fast, low overhead	Reduced attack interface	Safest
Little isolation	Limited Syscalls, special build	Performance & Resource overhead

What now?

Container Runtime

- What does your Cloud Provider Support?
- What does your k8s support
- Mostly Standards

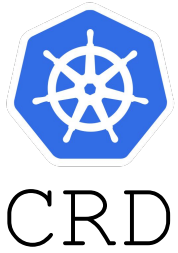
“Secure” Container

- What environment are you running?
- Multi-Tenant?
- User-defined input/code?
- Risk
  - User-data?
  - ….

What else?

https://github.com/containernetworking/cni

https://github.com/container-storage-interface/spec



https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/