KubeCon | CloudNativeCon Europe 2019

# Let's Try Every CRI Runtime Available for Kubernetes

Phil Estes, Distinguished Engineer
IBM Cloud

# Background: OCI



Docker, containerd, cri-o, Kata, Firecracker, gVisor, Nabla, Singularity, ...

DockerHub, OSS distribution project, Cloud registries, JFrog, ...

# Background: CRI



**Monday, December 19, 2016**

## Introducing Container Runtime Interface (CRI) in Kubernetes

*Editor's note: this post is part of a series of in-depth articles on what's new in Kubernetes 1.5*

At the lowest layers of a Kubernetes node is the software that, among other things, starts and stops containers. We call this the "Container Runtime". The most widely known container runtime is Docker, but it is not alone in this space. In fact, the container runtime space has been rapidly evolving. As part of the effort to make Kubernetes more extensible, we've been working on a new plugin API for container runtimes in Kubernetes, called "CRI".

**What is the CRI and why does Kubernetes need it?**

Each container runtime has it own strengths, and many users have asked for Kubernetes to support more runtimes. In the Kubernetes 1.5 release, we are proud to introduce the Container Runtime Interface (CRI) -- a plugin interface which enables kubelet to use a wide variety of container runtimes, without the need to recompile. CRI consists of a protocol buffers and gRPC API, and libraries, with additional specifications and tools under active development. CRI is being released as Alpha in Kubernetes 1.5.

# K8s and CRI Responsibilities



## Kubernetes

- K8s API
- Storage
- Networking (CNI)
- Healthchecks
- Placement
- Custom resources

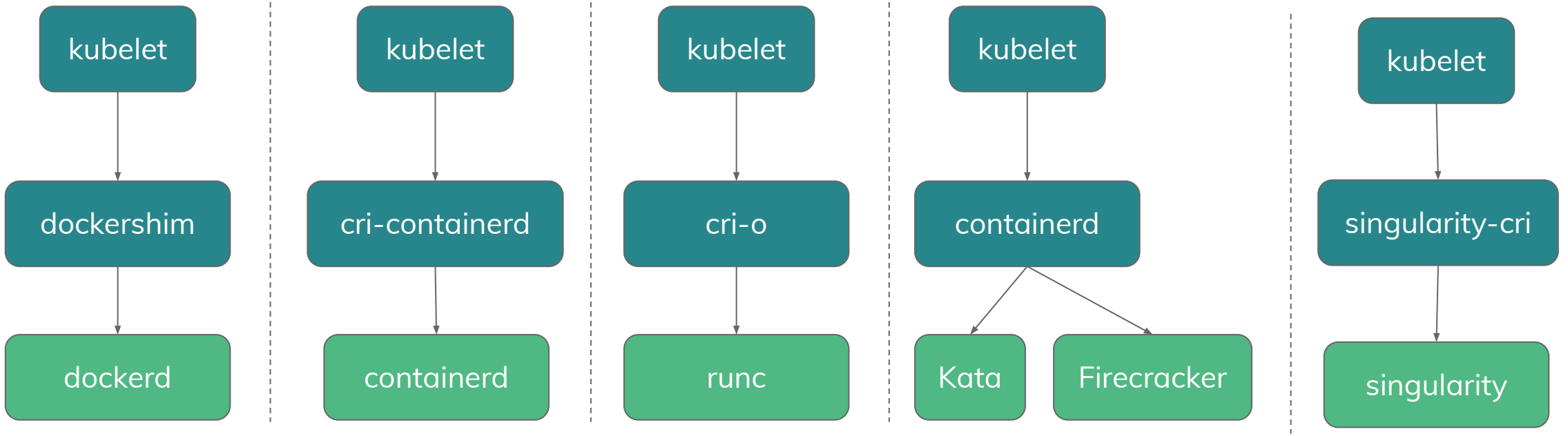
CRI

## Container Runtime

- Pod container lifecycle
  - Start/stop/delete
- Image management
  - Pull/status
- Status
- Container interactions
  - attach, exec, ports, log

# Background: CRI Runtimes






```
kubelet --container-runtime {string}
        --container-runtime-endpoint {string}
```

# Caveats



## What I don't have time to cover/demo...

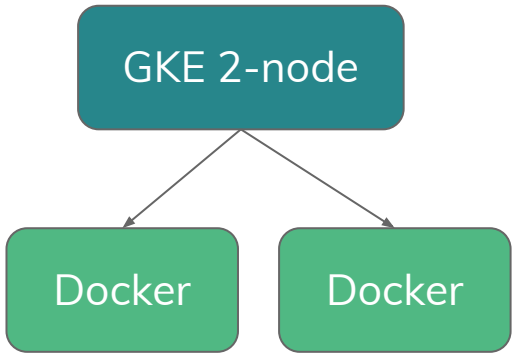
- Windows containers & runtimes
- rkt (CNCF)
- Virtual Kubelet (CRI implementation)
- Nabla containers (IBM)

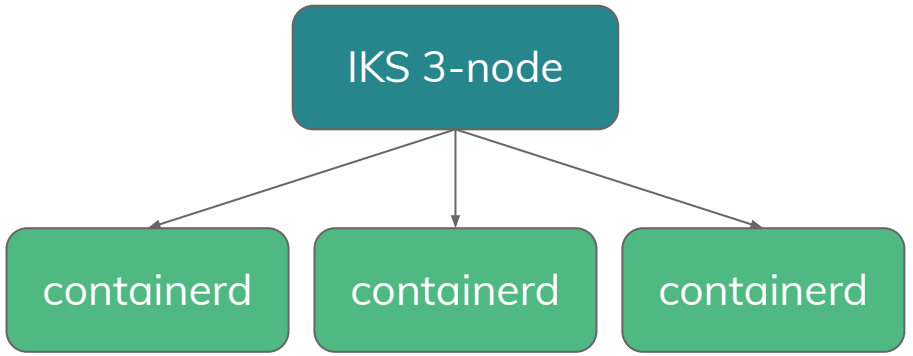
# Caveats

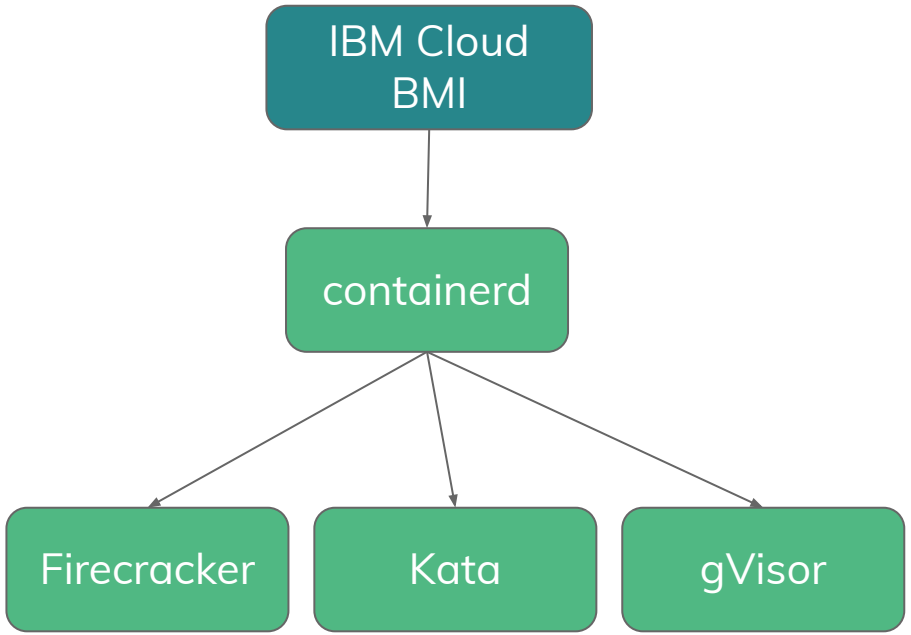


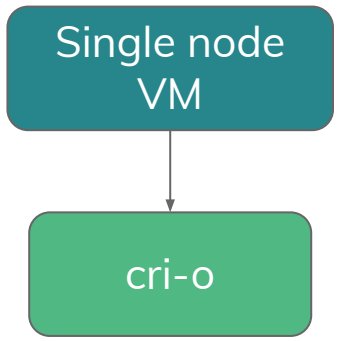

# Setup

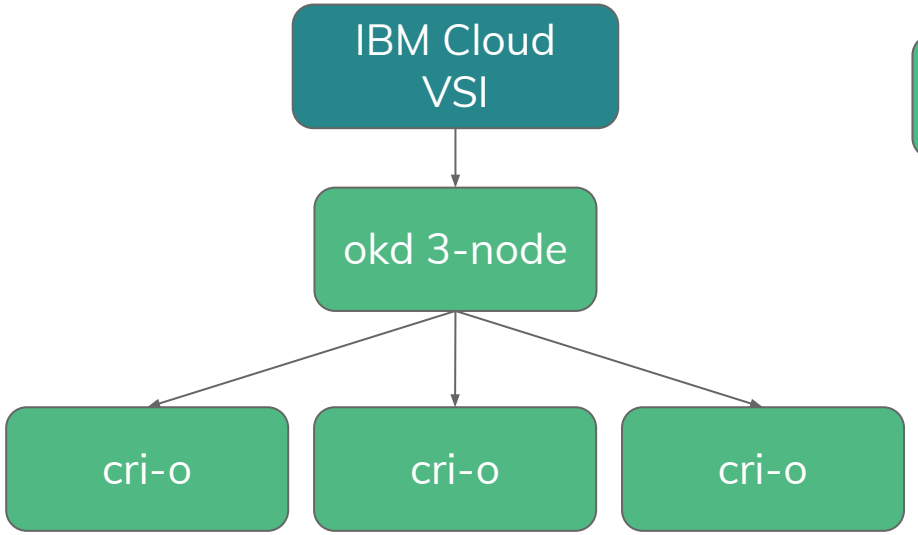



GKE 2-node
Docker
Docker


IKS 3-node
containerd
containerd
containerd

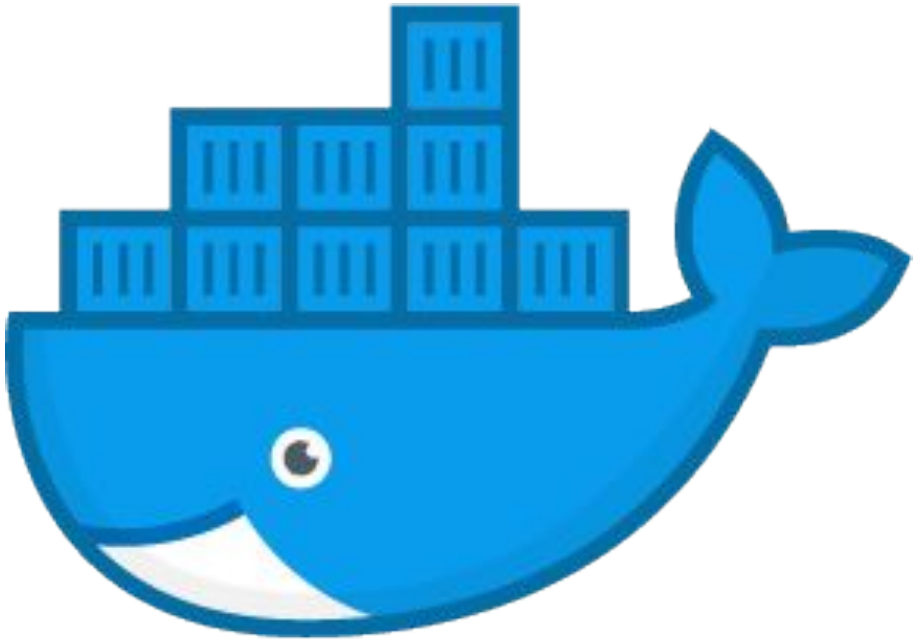

IBM Cloud
BMI
containerd
Firecracker
Kata
gVisor


Single node
VM
cri-o


IBM Cloud
VSI
okd 3-node
cri-o
cri-o
cri-o

# Docker





# Docker



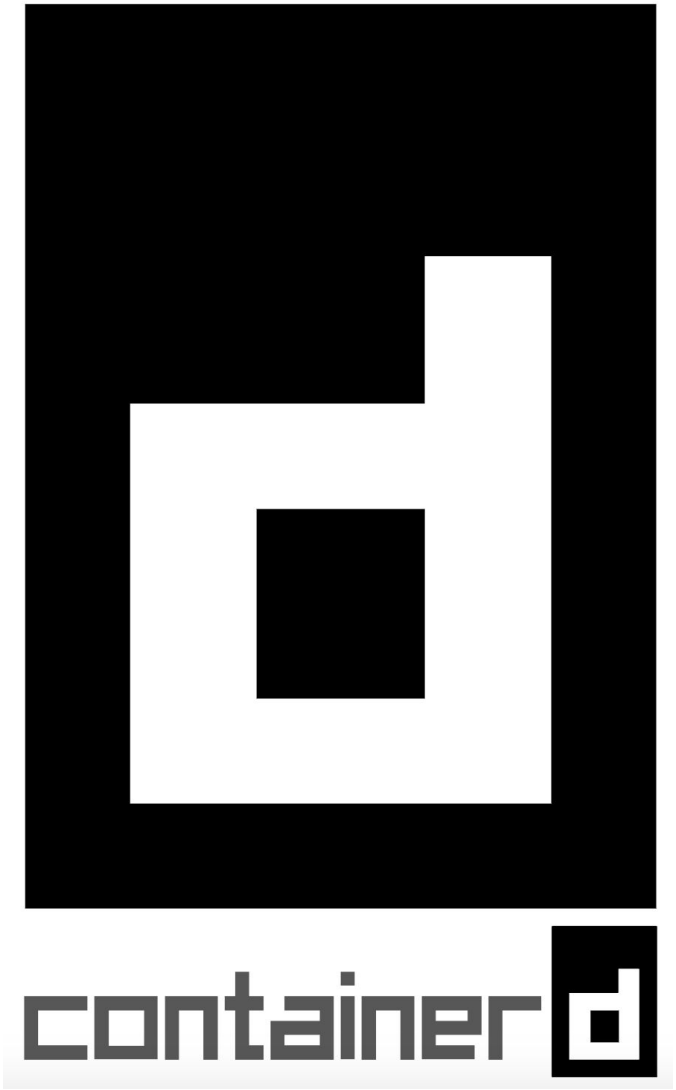
- Most common, original runtime for Kubernetes clusters
- Simplifies tooling for mixed use cluster node (e.g. applications relying on `docker …` commands “just work”
- Docker Enterprise customers get support and multi-orchestrator support (swarm + K8s in same cluster)

---

- “More than enough” engine for Kubernetes
- Concerns over mismatch/lack of release sync between Docker releases and Kubernetes releases (e.g. “certified” engine version)
- Extra memory/CPU use due to extra layer (docker->ctr->runc)

# Containerd





# Containerd



- Used in GKE (Google), IKS (IBM), & Alibaba public clouds
- Significant hardening/testing by nature of use in every Docker installation (tens of millions of engines)
- Lower memory/CPU use; clean API for extensibility/embedding

---

- No Docker API socket (tools/vendor support)
- Still growing in maturity/use
- Windows support in flight; soon at parity with Docker engine

# CRI-O



# CRI-O



- Used in RH OpenShift; SuSE CaaS; other customers/uses
- “all the runtime Kubernetes needs and nothing more”
- UNIX perspective on separating concerns (client, registry interactions, build)

---

- Not consumable apart from RH tools (design choice)
- Use/installation on non-RH distros can be complicated
- Extensibility limited (e.g. proposal from Kata to add containerd shim API to cri-o)

# Sandboxes + RuntimeClass

KubeCon CloudNativeCon Europe 2019

kubernetes

Documentation Blog Partners Community Case

## Kubernetes v1.12: Introducing RuntimeClass

Wednesday, October 10, 2018

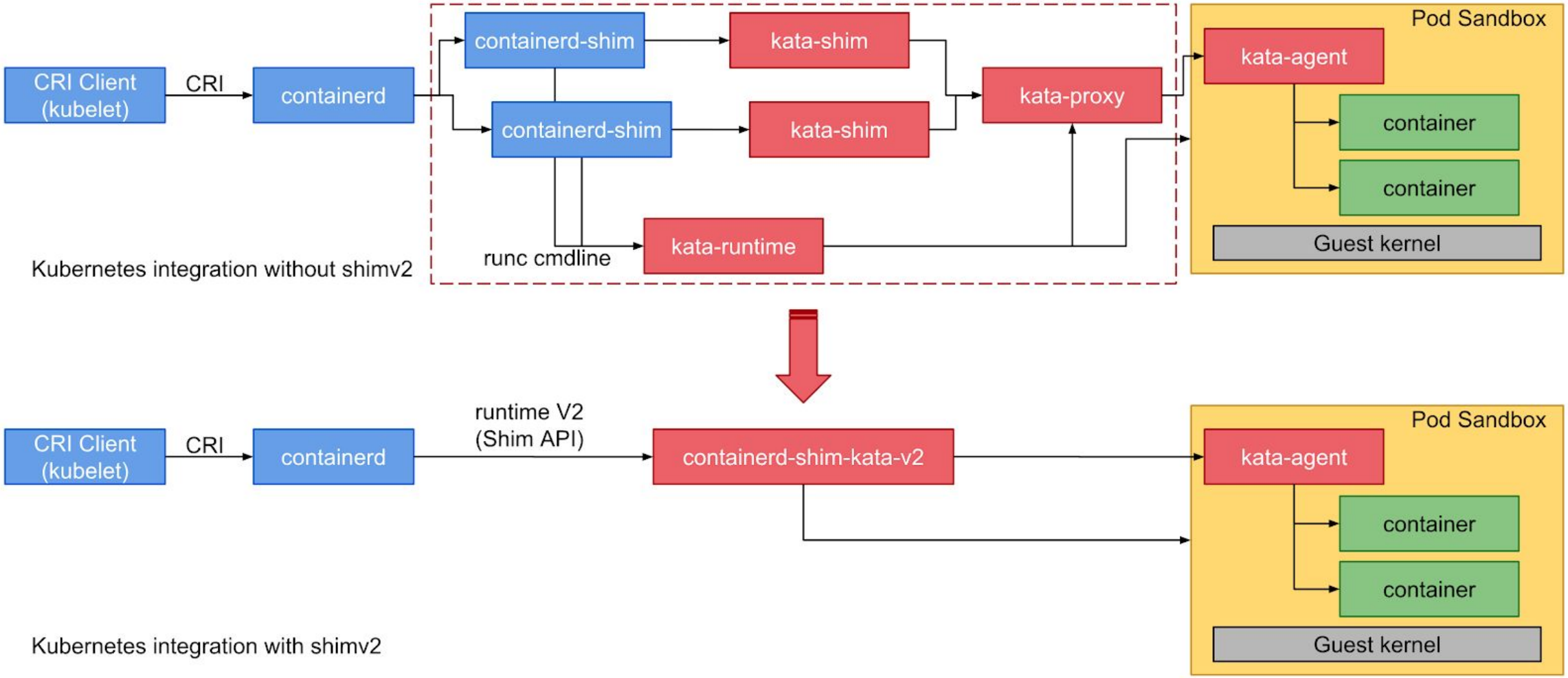
### Kubernetes v1.12: Introducing RuntimeClass

**Author**: Tim Allclair (Google)

Kubernetes originally launched with support for Docker containers running native applications on a Linux host. Starting with rkt in Kubernetes 1.3 more runtimes were coming, which lead to the development of the Container Runtime Interface (CRI). Since then, the set of alternative runtimes has only expanded: projects like Kata Containers and gVisor were announced for stronger workload isolation, and Kubernetes' Windows support has been steadily progressing.

# Containerd v2 Shim API





# Kata Containers



# Kata Containers



- Lightweight virtualization via Intel Clear Containers + Hyper.sh predecessors
- Implemented via KVM/qemu-based VM isolation
- Works with Docker, cri-o, & containerd

---

- Solid and maturing project with Intel and others leading; governance under OpenStack Foundation
- Have added ability to drive Firecracker VMM as well
- Supports ARM, x86_64, AMD64, and IBM p and zSeries

# AWS Firecracker



Firecracker

# AWS Firecracker



- Lightweight virtualization via Rust-written VMM, originating from Google's crosvm project; target serverless/functions area
- Open Sourced by Amazon in November 2018
- Works standalone via API or via containerd

---

- cgroup + seccomp "jailer" to tighten down kernel access
- Integrated with containerd via shim and external snapshotter implementation
- Quickly moving & young project; packaging and delivery still in flux and requires quite a few manual steps today


gVisor

# gVisor



- A kernel-in-userspace concept from Google; written in Golang
- Used in concert with GKE; for example with Google Cloud Run for increased isolation/security boundary
- Works standalone (OCI runc replacement) or via containerd shim implementation

---

- Reduced syscalls used against “real kernel”; applications run against gVisor syscall implementations
- Limited functionality; some applications may not work if syscall not implemented in gVisor
- Syscall overhead, network performance impacted (ref: KVM mode)

# Singularity



Sylabs.io

# Singularity



- An HPC/academic community focused container runtime
- Initially not implementing OCI, now has OCI compliant mode
- To meet HPC use model; not daemon-based, low privilege, user-oriented runtime (e.g. HPC end user workload scheduling)

---

- Sylabs, creator of Singularity have recently written a CRI implementation that drives Singularity runtime
- Uses OCI compliant mode; converts images to SIF, however
- Focused solely on the academic/HPC use case

# Nabla



# Nabla



- IBM Research created open source sandbox runtime
- Uses highly limited seccomp profile + unikernel implementation
- Similar to gVisor, but instead of user-mode kernel, uses unikernel+application approach

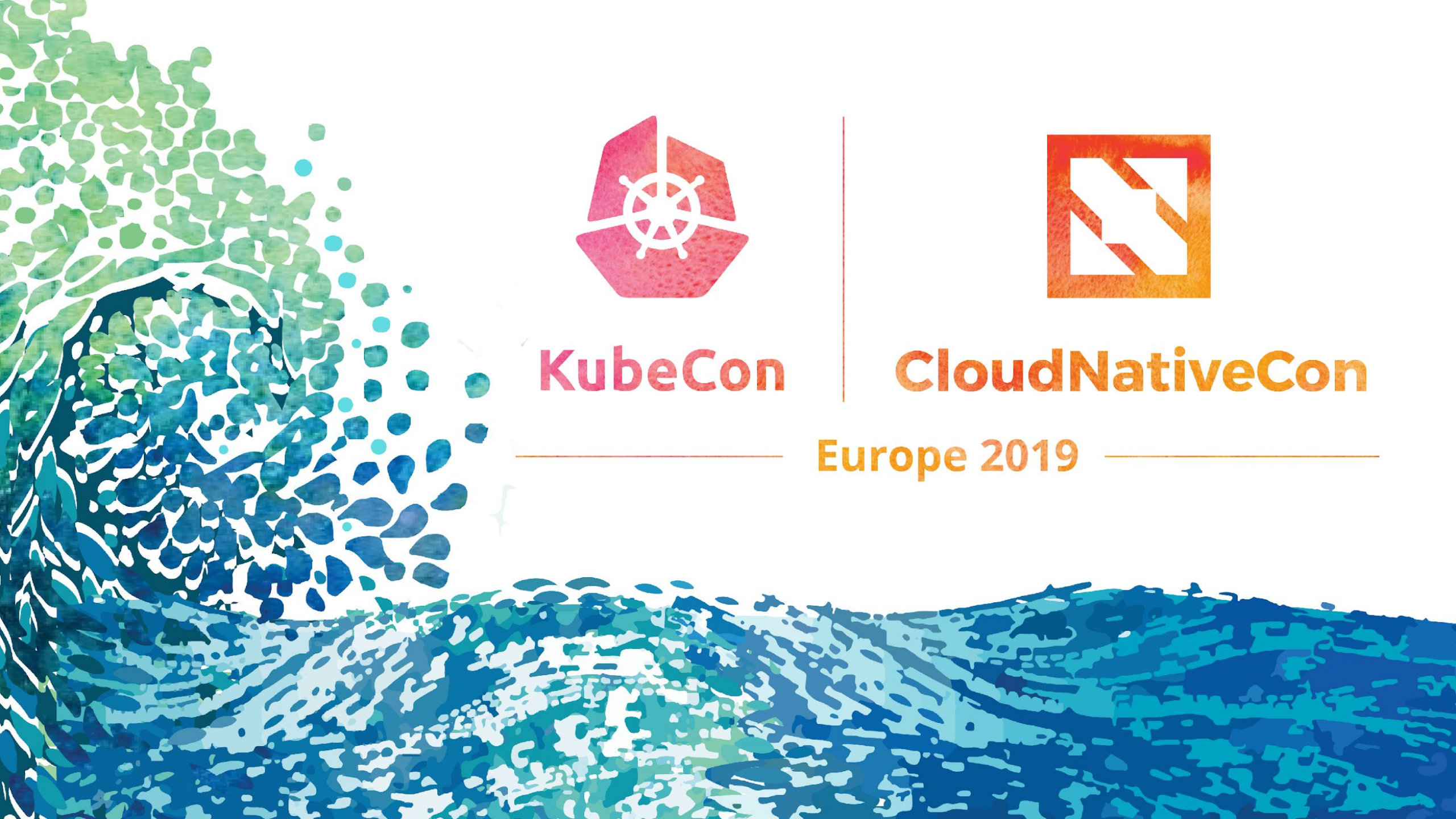
---

- Currently requires building images against special set of unikernel-linked runtimes (Node, Python, Java, etc.)
- IBM Research pursuing ways to remove this limitation; only runtime which doesn't allow generic use of any container image

# Summary



- OCI specs (runtime, image, distribution) have enabled a common underpinning for **innovation** that maintains **interoperability**
- CRI has enabled a "**pluggable**" model for container runtimes underneath Kubernetes
- Options are growing; most innovation is around sandboxes and enabled for easier use with **RuntimeClass** in K8s


KubeCon
CloudNativeCon
Europe 2019