IBM

Open by design™

IBM开源技术微讲堂
Kubernetes系列

第四讲

Kubernetes运行时

更多信息，请访问：http://ibm.biz/opentech-ma

Open by design™

“Kubenetes”系列公开课

每周四晚8点档

1. Kubernetes 初探
2. 上手 Kubernetes
3. Kubernetes 的资源调度
4. **Kubernetes 的运行时**
5. Kubernetes 的网络管理
6. Kubernetes 的存储管理
7. Kubernetes 的日志与监控
8. Kubernetes 的应用部署
9. 扩展 Kubernetes 生态
10. Kubernetes 的企业实践

课程Wiki：http://ibm.biz/opentech-ma

讲师简介

Michael Brown

目前就职于IBM Digital Business Group的开源与开放标准项目组，专注于开源云计算技术（例如Open Container Initiative, Docker与Kubernetes等）的开发与研究。在他的职业生涯中，曾参与过许多开源项目与开放标准的研发工作，包括操作系统、Java虚拟机、浏览器、中间件和金融产品等相关项目。

以下是他个人的自我描述：

“Engineer, IBMer, Inventor, Coach, PC Gamer, and Proud Father. Opinions are my own and not of my employer. ”

Kubernetes Runtime

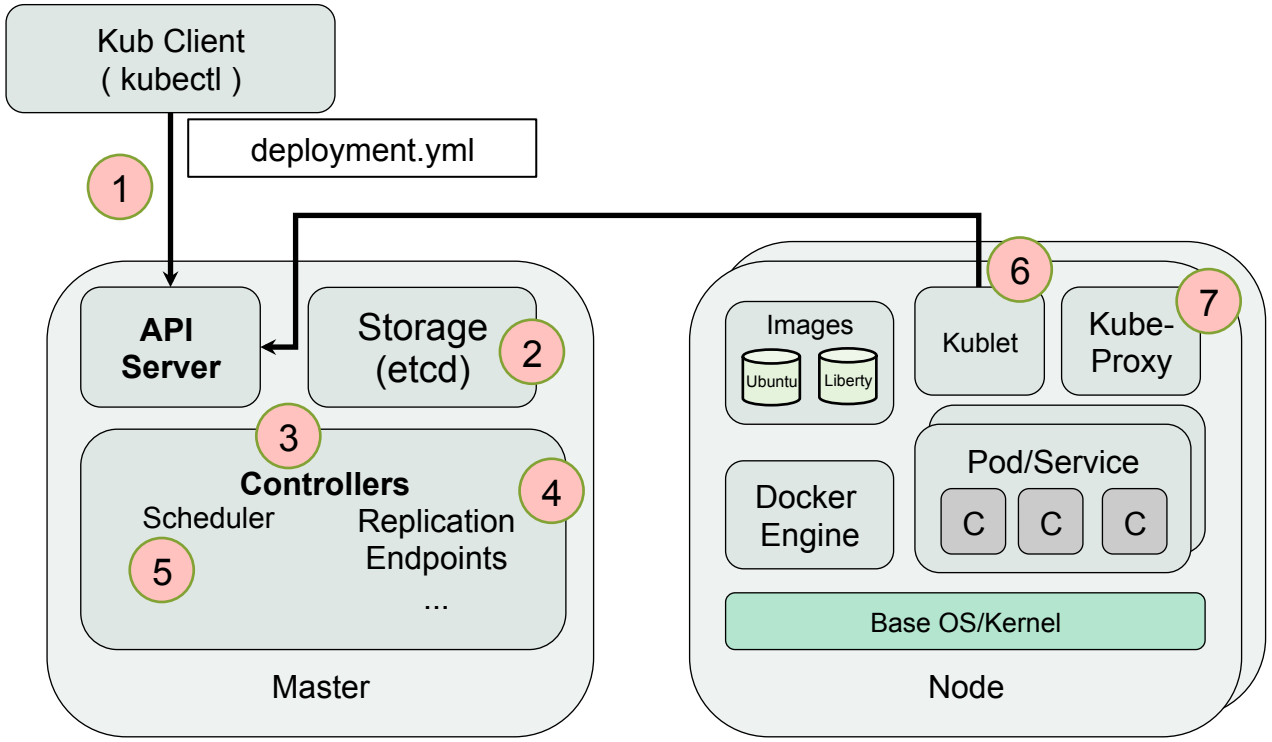
Michael W. Brown
Developer Advocate - Open Source Software Development
Maintainer – CRI-Containerd

Agenda

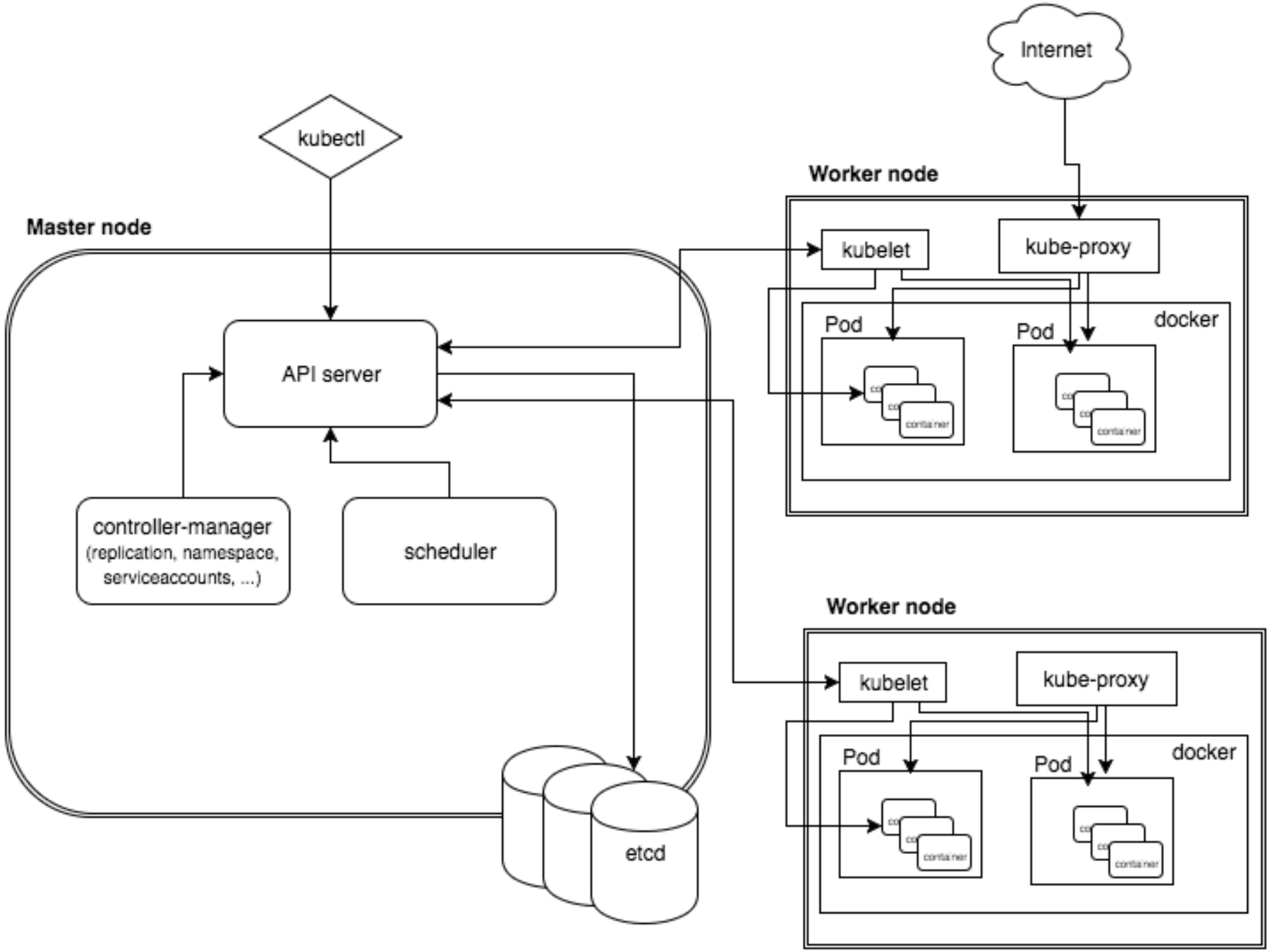
- **Kubelet Architecture - General Introduction**
- **Container runtimes in Kubernetes**
- **Open Containers Initiative (OCI) Introduction**

Kubelet Architecture - Reminder chart from Kubernetes 101

1. User via "kubectl" deploys a new application
2. API server receives the request and stores it in the DB (etcd)
3. Watchers/controllers detect the resource changes and act upon it
4. ReplicaSet watcher/controller detects the new app and creates new pods to match the desired # of instances
5. Scheduler assigns new pods to a kubelet
6. Kubelet detects pods and deploys them via the container runing (e.g. Docker)
7. Kubeproxy manages network traffic for the pods – including service discovery and load-balancing



Kubelet Architecture



Kubelet Architecture - Kubelet

1. Kublet runs on each node.
2. Kublet is the primary node agent.
3. Kublet manages isolated application containers.
4. Kublet creates and manages pods, and the application containers requested for each pod.
5. Kublet processes pod specs to identify the configuration for the pod and application containers.
6. Kublet receives pod specs from the apiserver, command line, HTTP endpoint, and HTTP server.
7. Kubelet also links in the cAdvisor resource monitoring agent.

Kubelet Architecture - Kube Proxy

1. The Kubernetes service abstraction provides a way to group pods.
   - The group of pods of a service have a common access policy (e.g. load-balanced)
   - A virtual IP is created for the service
2. Each node runs a kube-proxy, acting as a network proxy and load balancer
3. Service endpoints are found primarily via DNS.

Kubelet Architecture - Container Runtime

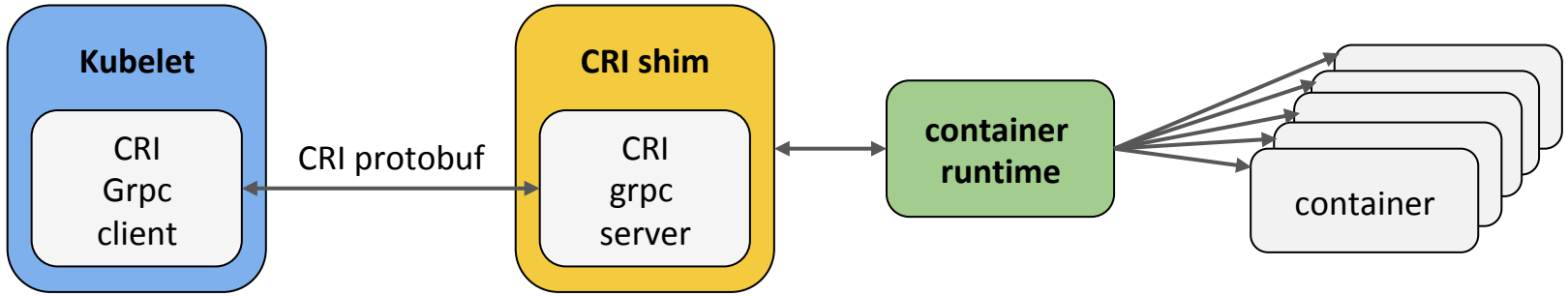
1. Each node includes at least one container runtime.

2. The container runtime is responsible for downloading images and running containers.

3. Kubelet does not link in the base container runtime.
4. The Kubernetes Container Runtime Interface (CRI) has been defined to decouple and c

Container Runtimes in Kubernetes

- What is Container Runtime Interface (CRI) ?
  - A GRPC interface and a group of libraries
  - Enables Kubernetes to use a wide variety of container runtimes
  - Introduced in Kubernetes 1.5+



Container Runtime Interface

- CRI Integrations
  - containerd (with cri-containerd): https://github.com/kubernetes-incubator/cri-containerd
  - cri-o: https://github.com/kubernetes-incubator/cri-o
  - Docker (Upstream): https://github.com/kubernetes/kubernetes/tree/master/pkg/kubelet/dockershim
  - frakti: https://github.com/kubernetes/frakti
  - rktlet: https://github.com/kubernetes-incubator/rktlet
  - virtlet: https://github.com/Mirantis/virtlet

- CRI Tools https://github.com/kubernetes-incubator/cri-tools
  - `critest`: CRI Validation Test Suite
  - `crictl`: CRI Command Line Tool

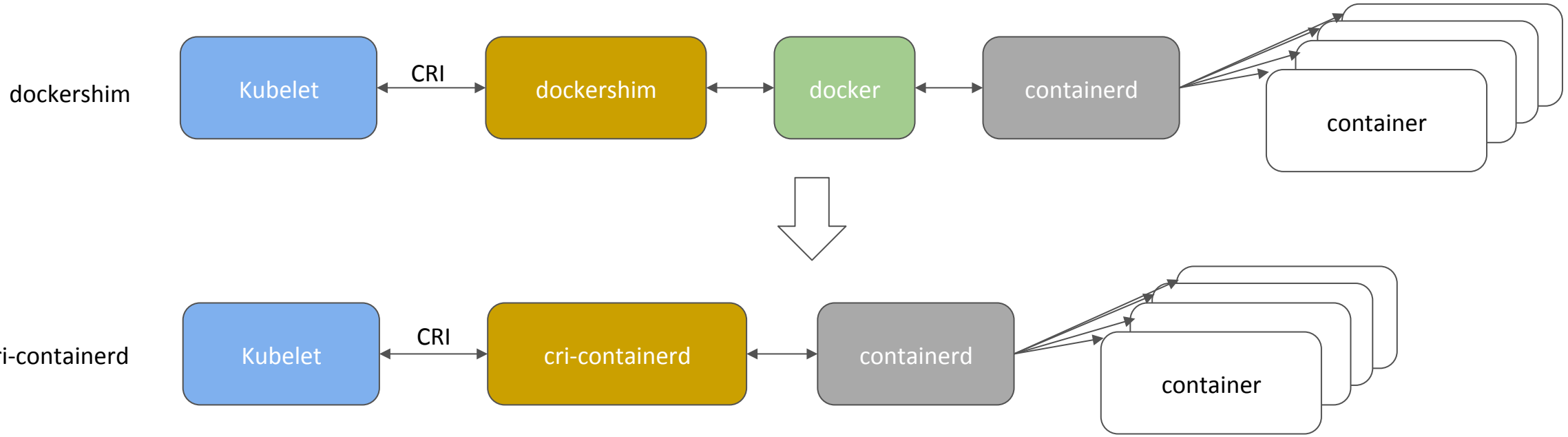
Kubernetes & Containerd

- The scope of containerd aligns with the requirement of Kubernetes.

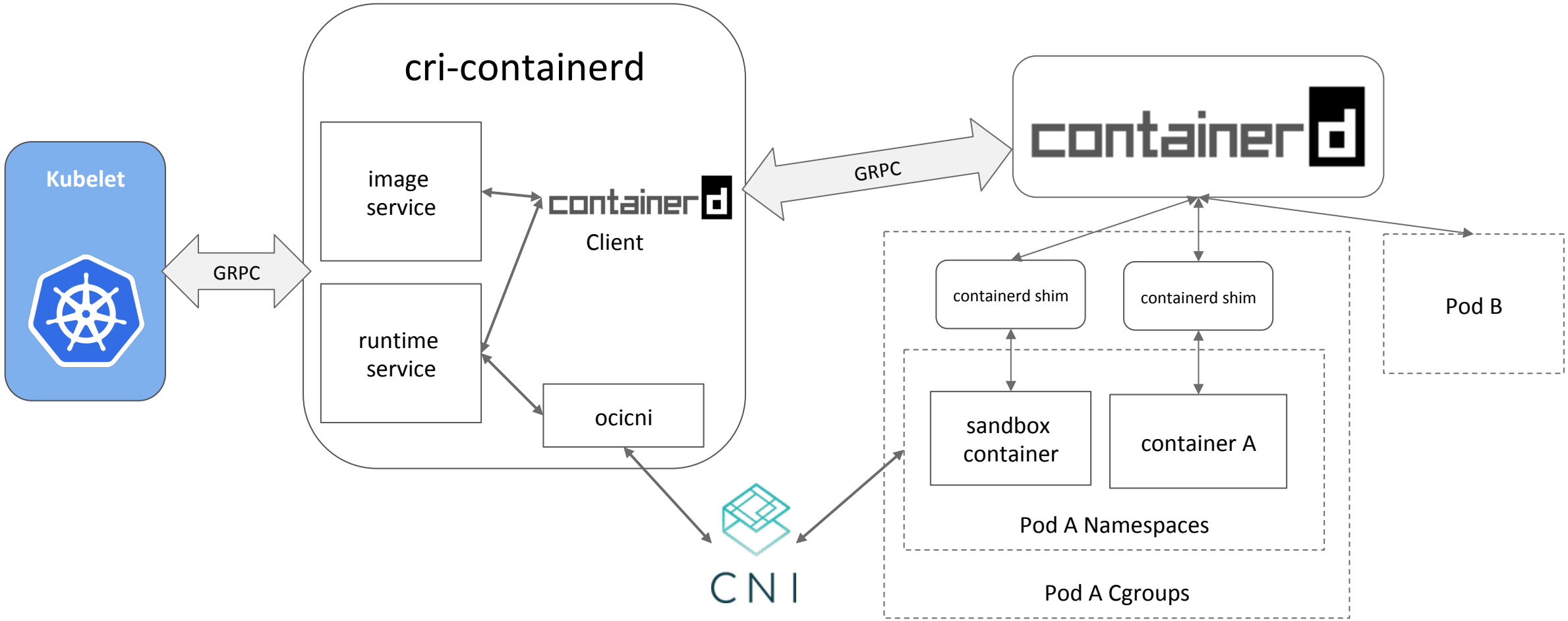
	Containerd Scope (In/Out)	Kubernetes Requirement
Container Lifecycle Management	In	Container Create/Start/Stop/Delete/List/Inspect (✔)
Image Management	In	Pull/List/Inspect (✔)
Networking	Out No concrete network solution. User can setup network namespace and put containers into it.	Kubernetes networking deals with pods, rather than containers, so container runtimes should not provide complex networking solutions that don't satisfy requirements. (✔)
Volumes	Out No volume management. User can setup host path, and mount it into container.	Kubernetes manages volumes. Container runtimes should not provide internal volume management that may conflict with Kubernetes. (✔)
Persistent Container Logging	Out No persistent container log. Container STDIO is provided as FIFOs, which can be redirected/ decorated as is required.	Kubernetes has specific requirements for persistent container logs, such as format and path etc. Container runtimes should not persist an unmanageable container log. (✔)
Metrics	In Containerd provides container and snapshot metrics as part of the API.	Kubernetes expects container runtime to provide container metrics (CPU, Memory, writable layer size, etc.) and image filesystem usage (disk, inode usage, etc.). (✔)

CRI-Containerd

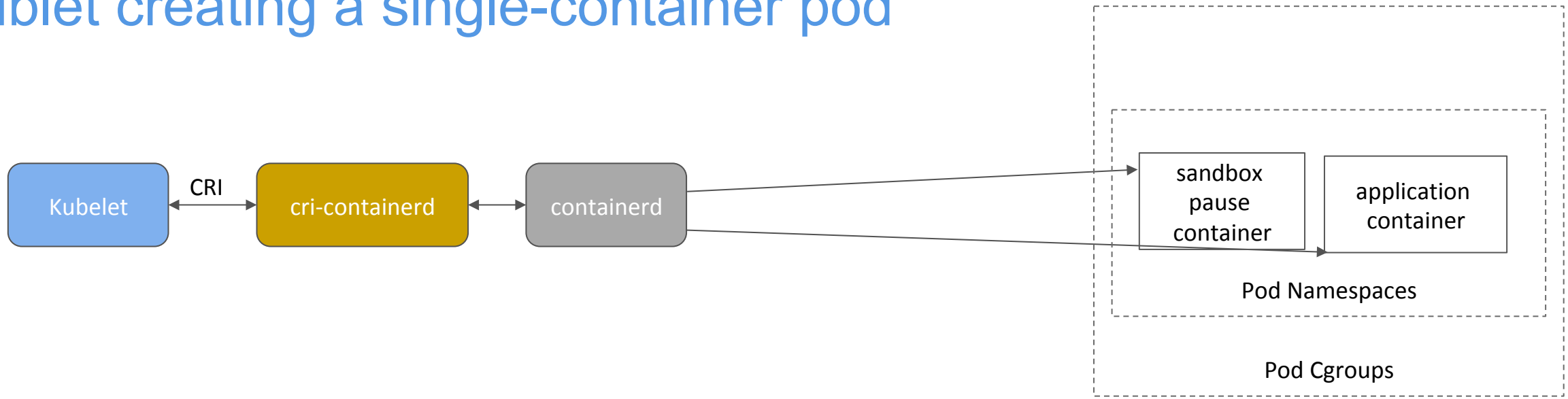
- `cri-containerd`: A containerd based implementation of CRI.
  - https://github.com/kubernetes-incubator/cri-containerd
  - Kubernetes incubator project.
  - Started in April 2017.



Architecture



Kublet creating a single-container pod



Example to demonstrate how cri-containerd works for the case when Kubelet creates a single-container pod:

1. Kubelet calls cri-containerd, via the CRI runtime service API, to create a pod;
2. cri-containerd uses containerd to create and start a special pause container (the sandbox container) and put that container inside the pod's cgroups and namespace (steps omitted for brevity);
3. cri-containerd configures the pod's network namespace using CNI;
4. Kubelet subsequently calls cri-containerd, via the CRI image service API, to pull the application container image;
5. cri-containerd further uses containerd to pull the image if the image is not present on the node;
6. Kubelet then calls cri-containerd, via the CRI runtime service API, to create and start the application container inside the pod using the pulled container image;
7. cri-containerd finally calls containerd to create the application container, put it inside the pod's cgroups and namespace, then to start the pod's new application container.

After these steps, a pod and its corresponding application container is created and running.

Status

Cri-containerd v1.0.0-alpha.0 was released on Sep. 25, 2017.

It is feature complete. All Kubernetes features are supported.

All CRI validation tests have passed. (A CRI validation is a test framework for validating whether a CRI implem

All regular node e2e tests have passed. (The Kubernetes test framework for testing Kubernetes node level func

For more current status, see the github project repository, links provided on last slide.

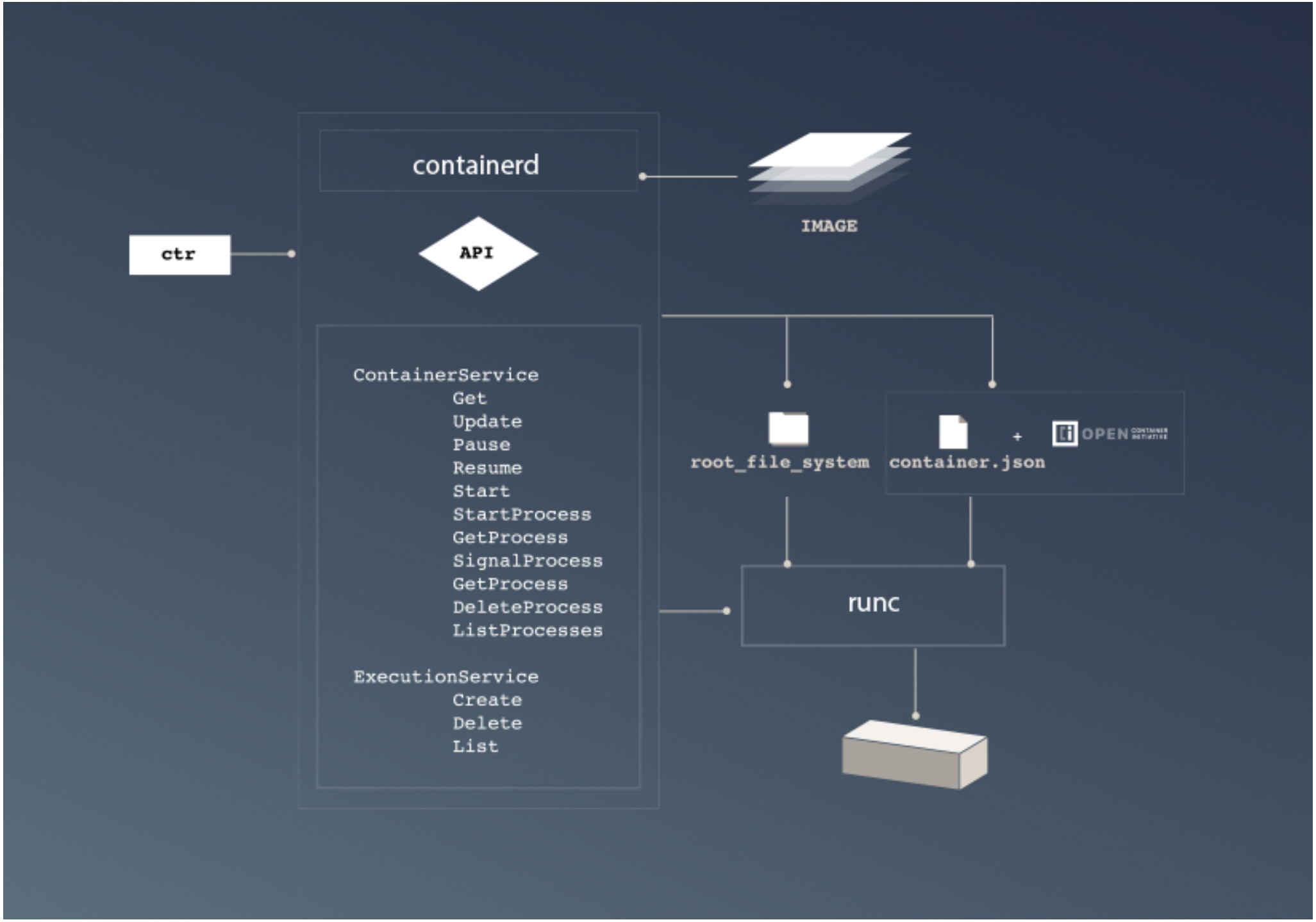
Next Steps for CRI-Containerd

- Stability:
  - Set up a full set of Kubernetes integration test in the Kubernetes test infrastructure on various OS distros such as Ubuntu, COS (Container-Optimized OS) etc.
  - Actively fix any test failures and other issues reported by users.

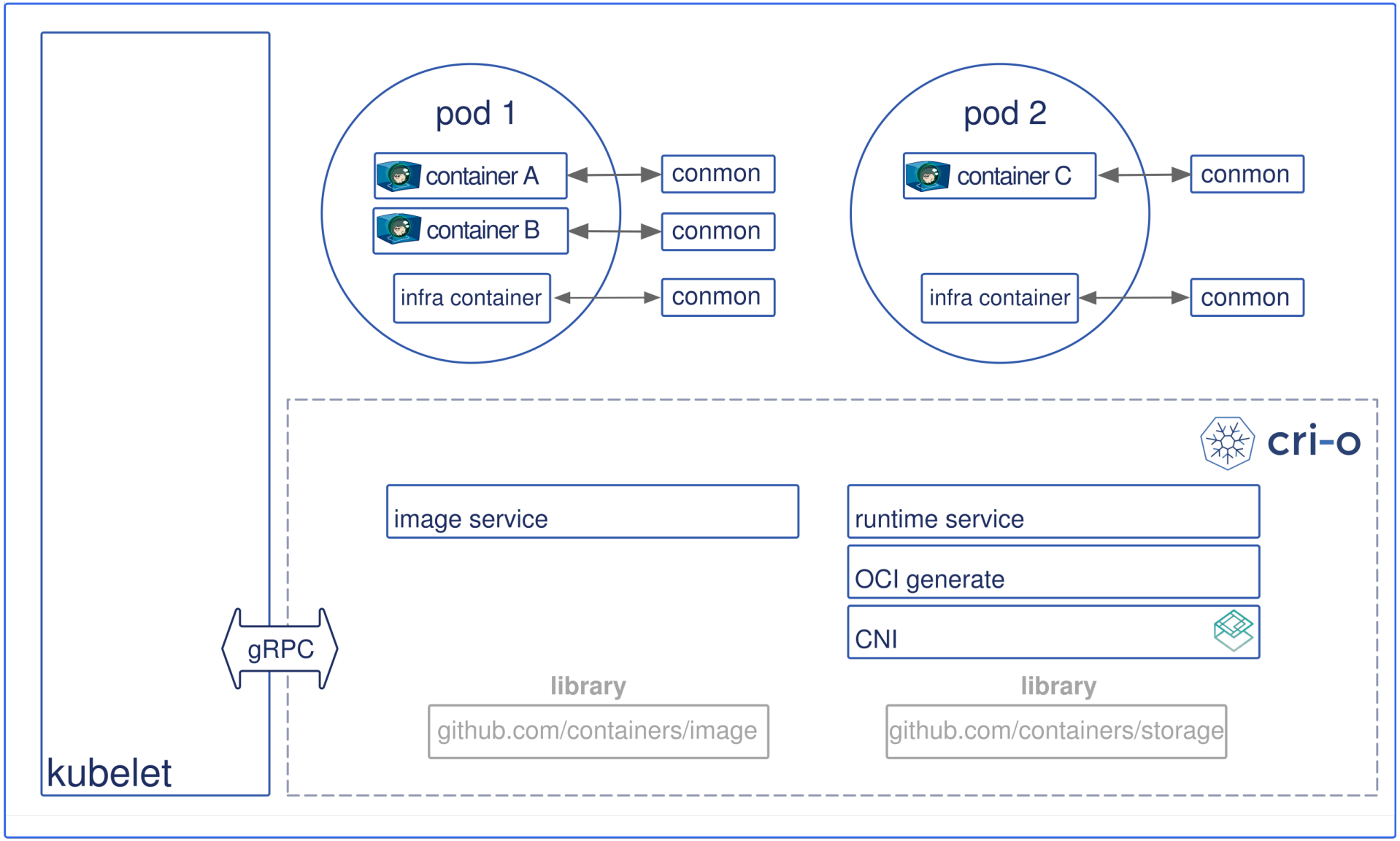
- Usability:
  - Improve the user experience of crictl. Crictl is a portable command line tool for all CRI container runtimes. The goal here is to make it easy to use for debug and development scenarios.
  - Integrate cri-containerd with kube-up.sh, to help users bring up a production quality Kubernetes cluster using cri-containerd and containerd.
  - Improve our documentation for users and admins alike.

We plan to release our v1.0.0-beta.0 by the end of 2017.

Open Container Initiative (OCI) - Containerd



Open Container Initiative (OCI) with CRI-O



Thank You! Q&A

About OCI:
https://www.opencontainers.org

runc on github:
https://github.com/opencontainers/runc

OCI testing:
https://github.com/opencontainers/runtime-tools

ContainerD
https://github.com/containerd/containerd

CRI-O:
http://cri-o.io
https://github.com/kubernetes-incubator/cri-o

More on CRI-Containerd

Cri-containerd is a Kubernetes incubator project located at https://github.com/kubernetes-incubator/cri-containerd. Any contril

Try it Out:

For a multi-node cluster installer and bring up steps using ansible and kubeadm, see:
https://github.com/kubernetes-incubator/cri-containerd/blob/master/contrib/ansible/README.md

For creating a cluster from scratch on Google Cloud, see Kubernetes the Hard Way:
https://github.com/kelseyhightower/kubernetes-the-hard-way

For a custom installation from release tarball, see:
https://github.com/kubernetes-incubator/cri-containerd/blob/master/docs/installation.md.

For a installation with LinuxKit on a local VM, see:
https://github.com/linuxkit/linuxkit/tree/master/projects/kubernetes.

CRI-Containerd is developed and maintained by the Kubernetes SIG-Node community. We'd love to hear feedback from you. To join the community:

- sig-node community site: https://github.com/kubernetes/community/tree/master/sig-node
- Slack: #sig-node channel in Kubernetes (kubernetes.slack.com)
- Mailing List: https://groups.google.com/forum/#!forum/kubernetes-sig-node