Success of CRI: Bringing Hypervisor based Container to Kubernetes

Harry Zhang, @resouer





About Me

- ✓Previous:
  - ✓ VMware (Pivotal), Assistant Research Scientist @ZJU
- ✓HyperCrew:
  - ✓ https://hyper.sh
- ✓PM & feature maintainer of Kubernetes project

A survey about “boundary”

- ✓Are you comfortable with Linux containers as an effective boundary?
  - ✓**Yes**, I use containers in my private/safe environment
  - ✓**No**, I use containers to serve the public cloud

As long as we care security…

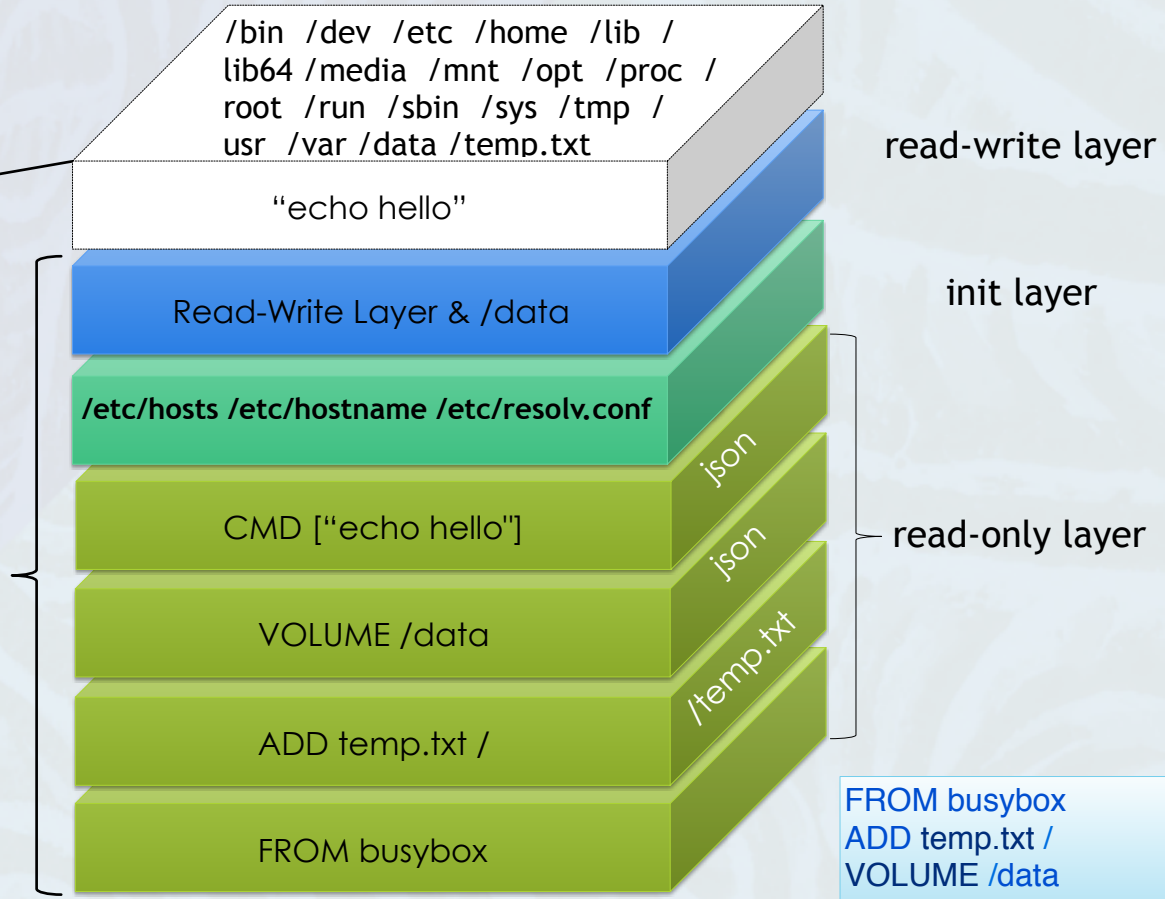
- ✓We have to wrap containers inside full-blown **virtual machines**
- ✓But we lose **C**loud **N**ative **D**eployment
  - ✓slow startup time
  - ✓huge resources wasting
  - ✓memory tax for every container
  - ✓…

HyperContainer

- ✓being **secure**
  - ✓while keep **Cloud Native**

Revisit container

- ✓ **Container Runtime**
  - ✓ The dynamic view and boundary of your running process
- ✓ **Container Image**
  - ✓ The static view of your program, data, dependencies, files and directories



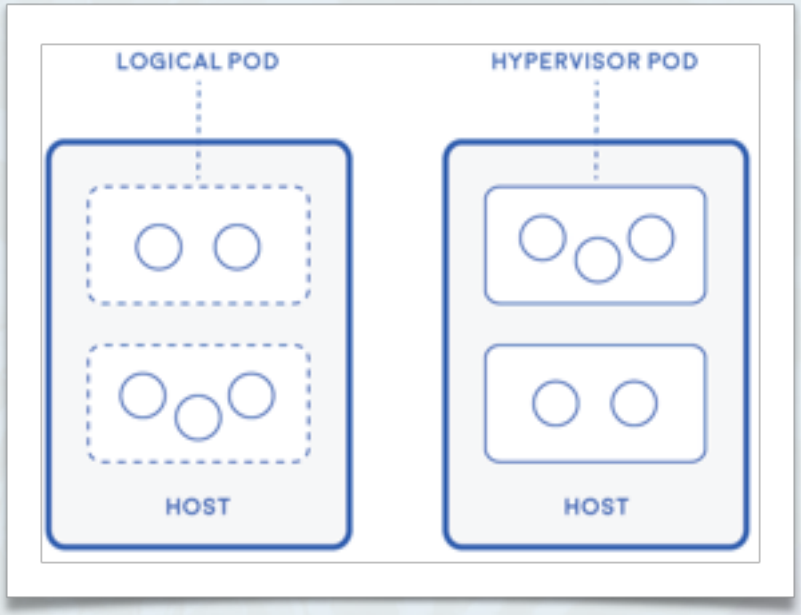
e.g. Docker Container

HyperContainer

- ✓Container runtime: hypervisor
  - ✓**RunV**
    - https://github.com/hyperhq/runv
    - The OCI compatible hypervisor based runtime implementation
  - ✓**Control daemon**
    - hyperd: https://github.com/hyperhq/hyperd
  - ✓ **Init service (PID=1)**
    - hyperstart: https://github.com/hyperhq/hyperstart/
- ✓Container image: docker image
  - ✓OCI Image Spec (next candidate)

Combine the best parts

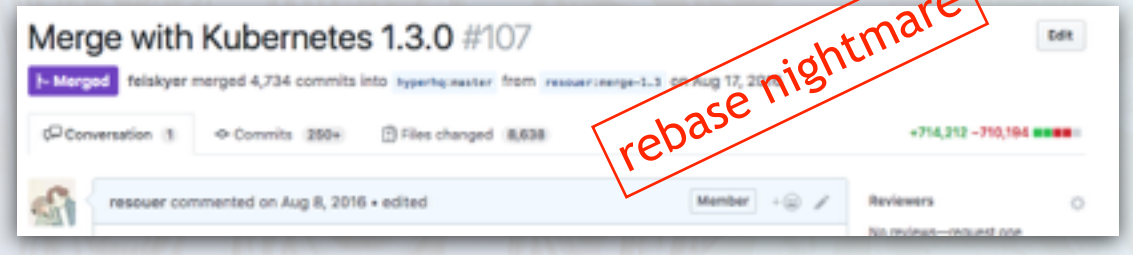
- ✓Portable and behaves like a Linux container
  - ✓`$ hyperctl run -t busybox echo helloworld`
    - sub-second startup time*
    - only cost ~12MB extra memory
- ✓Hardware level virtualization, with independent guest kernel
  - ✓`$ hyperctl exec -t busybox uname -r`
    - 4.4.12-hyper (or your provided kernel)
- ✓HyperContainer naturally match to the design of **Pod**

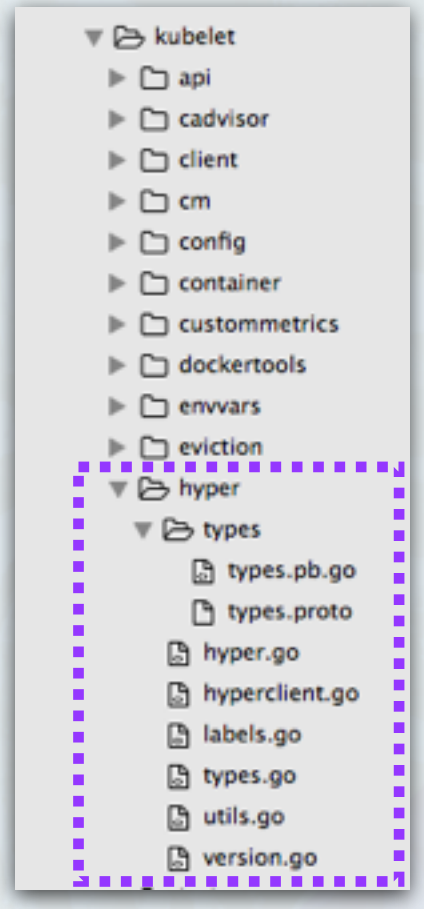


* More details: http://hypercontainer.io/why-hyper.html

Bring HyperContainer to Kubernetes?

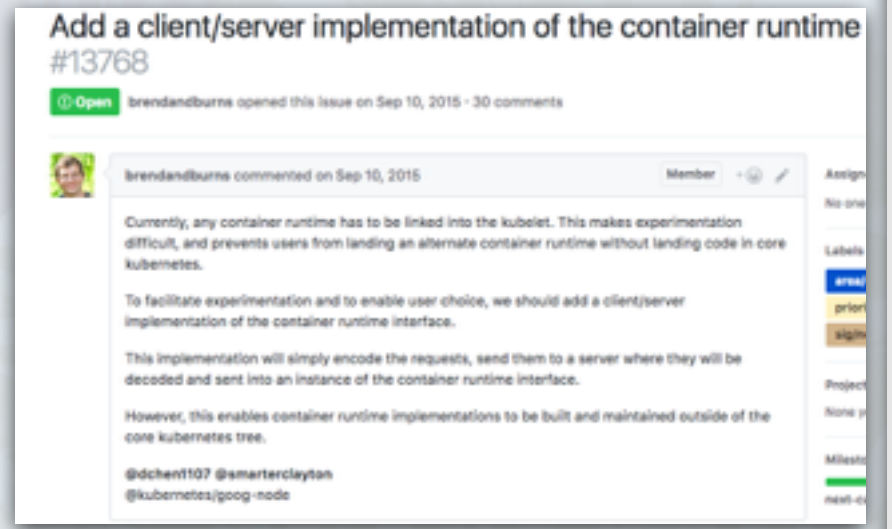
- ✓ hypernetes <= 1.5
  - ✓ a volatile internal interface (same as rkt)

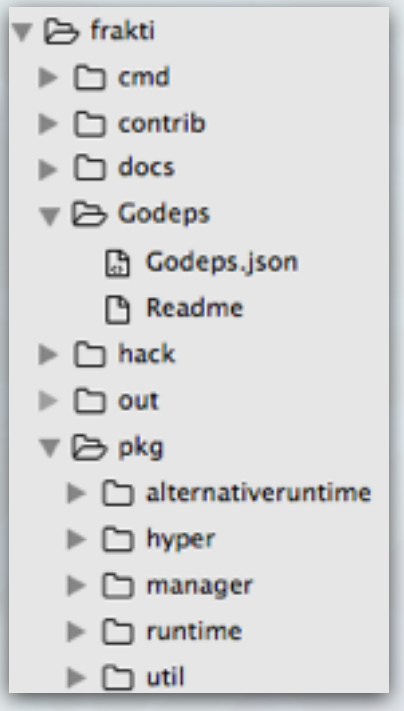




Bring HyperContainer to Kubernetes?

- ✓ hypernetes 1.6+
  - ✓ C/S mode runtime
    - **CRI**
  - ✓ no fork
    - hypernetes repo will only contain plugins and TPRs





Container Runtime Interface (CRI)

- ✓Describe what kubelet expects from container runtimes
- ✓Imperative container-centric interface
  - ✓ **why not pod-centric?**
    - Every container runtime implementation needs to understand the concept of pod.
    - Interface has to be changed whenever new pod-level feature is proposed.
- ✓Extensibility
- ✓Feature Velocity
- ✓Code Maintainability

More details: kubernetes/kubernetes#17048 (by @feiskyer)

CRI Spec

- ✓ Sandbox
  - ✓ How to isolate Pod environment?
    - Docker: infra container + pod level cgroups
    - Hyper: light-weighted VM
- ✓ Container
  - ✓ Docker: docker container
  - ✓ Hyper: namespace containers controlled by hyperstart

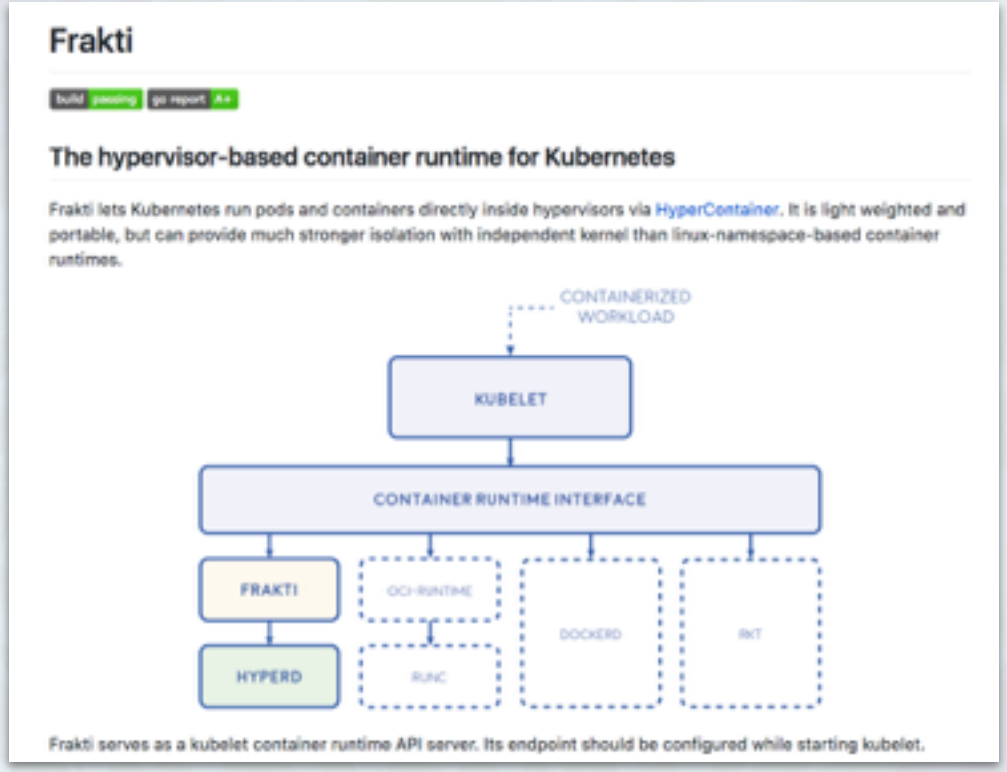
```
type RuntimeService interface {
    RunPodSandbox(config *kubeapi.PodSandboxConfig) (string, error)
    StopPodSandbox(podSandboxID string) error
    RemovePodSandbox(podSandboxID string) error
    PodSandboxStatus(podSandboxID string) (*kubeapi.PodSandboxStatus, error)
    ListPodSandbox(filter *kubeapi.PodSandboxFilter) ([]*kubeapi.PodSandbox, error)

    CreateContainer(podSandboxID string, config *kubeapi.ContainerConfig,
        sandboxConfig *kubeapi.PodSandboxConfig) (string, error)
    StartContainer(rawContainerID string) error
    StopContainer(rawContainerID string, timeout int64) error
    RemoveContainer(rawContainerID string) error
    ListContainers(filter *kubeapi.ContainerFilter) ([]*kubeapi.Container, error)
    ContainerStatus(rawContainerID string) (*kubeapi.ContainerStatus, error)

    ExecSync(rawContainerID string, cmd []string, timeout time.Duration) ([]byte, []byte, error)
    Exec(req *kubeapi.ExecRequest) (*kubeapi.ExecResponse, error)
    Attach(req *kubeapi.AttachRequest) (*kubeapi.AttachResponse, error)
    PortForward(req *kubeapi.PortForwardRequest) (*kubeapi.PortForwardResponse, error)
}

type ImageService interface {
    ListImages(filter *kubeapi.ImageFilter) ([]*kubeapi.Image, error)
    ImageStatus(image *kubeapi.ImageSpec) (*kubeapi.Image, error)
    PullImage(image *kubeapi.ImageSpec, auth *kubeapi.AuthConfig) (string, error)
    RemoveImage(image *kubeapi.ImageSpec) error
}
```

How CRI Works with HyperContainer?

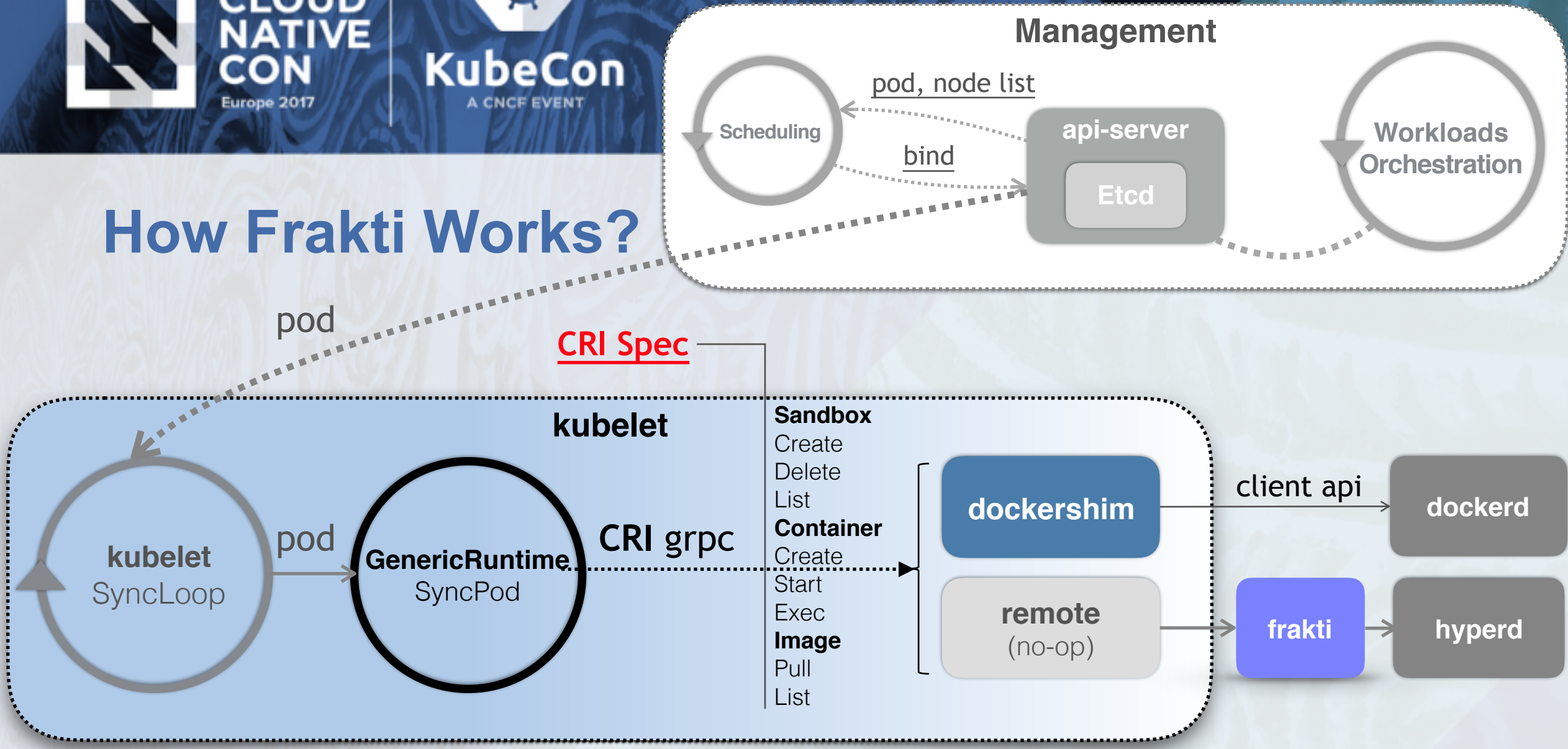
✓Just implement the interface!

Frakti

- ✓kubernetes/frakti project
  - ✓Released with Kubernetes 1.6
  - ✓Already passed 96% of node e2e conformance test
  - ✓Use CNI network
  - ✓Pod level resource management
  - ✓Mixed runtimes
  - ✓Can be used with kubeadm
  - ✓Unikernels Support (GSoC 2017)



How Frakti Works?

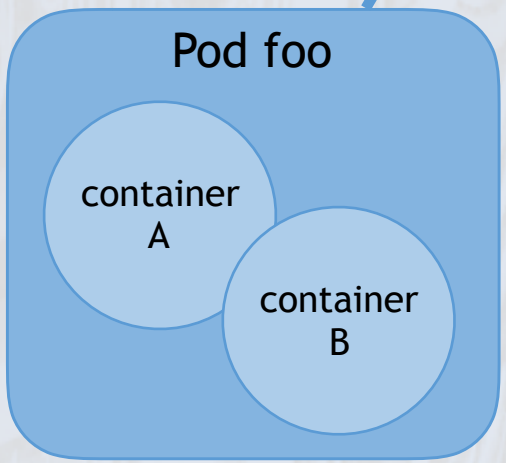


How to Write a Runtime Shim?

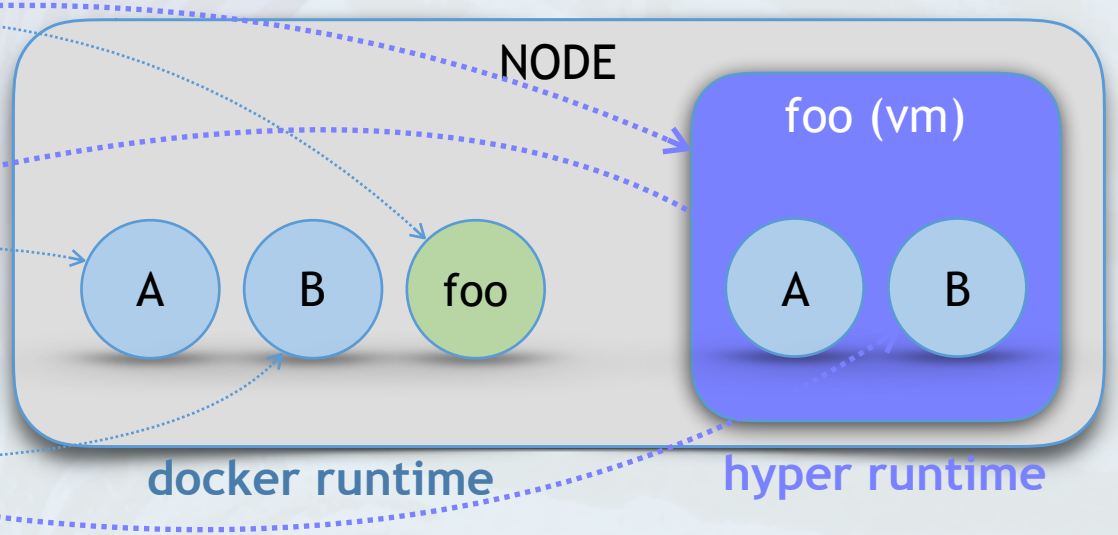
- ✓dockershim
- ✓frakti
- ✓cri-o
- ✓rktlet
- ✓…

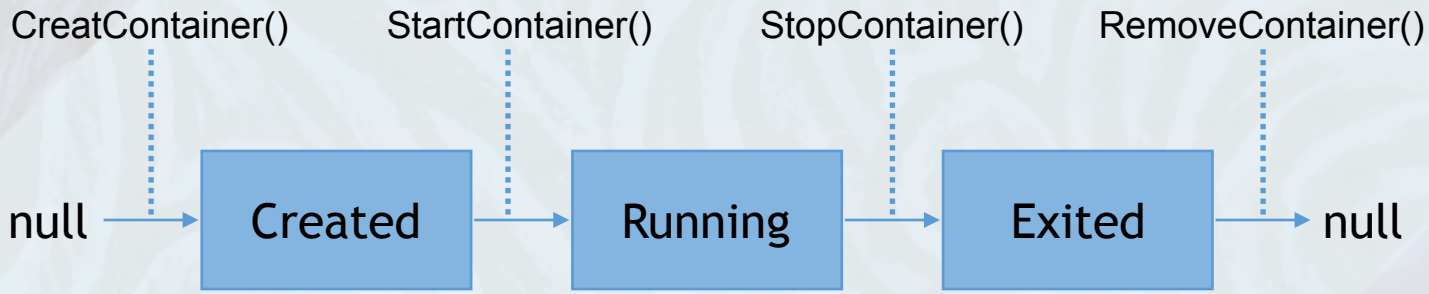
1. Lifecycle

$ kubectl run foo ...

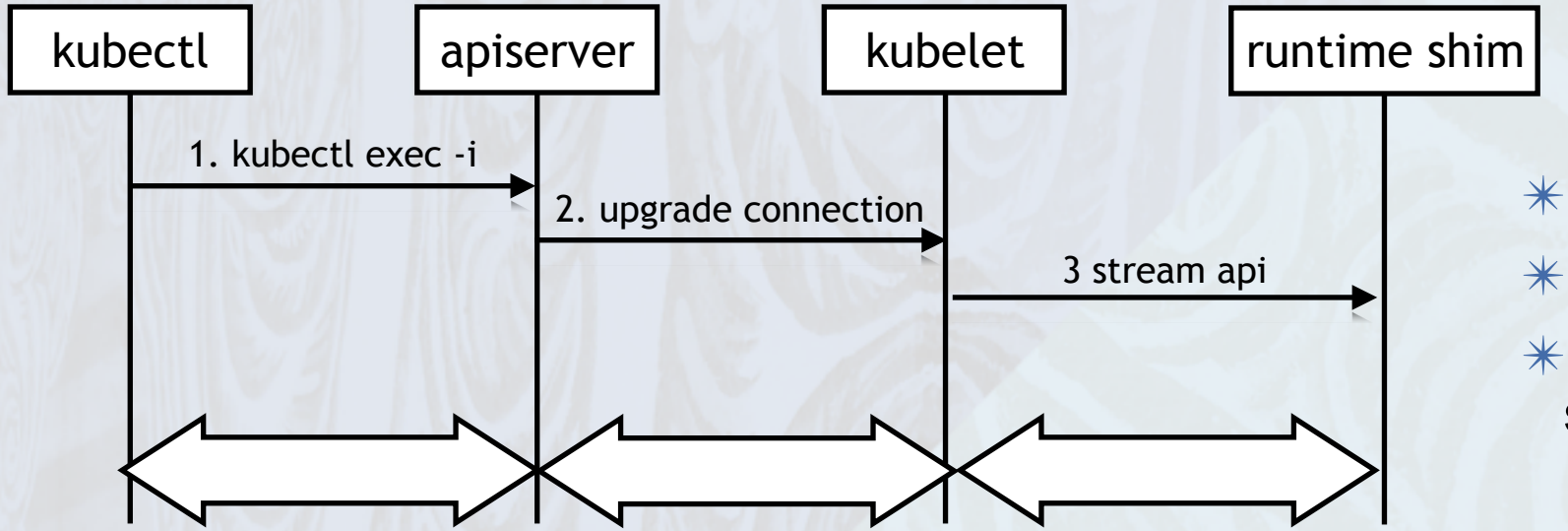


1. RunPodSandbox(foo)
2. CreatContainer(A)
3. StartContainert(A)
4. CreatContainer(B)
5. StartContainer(B)





2.1 Streaming (old version)

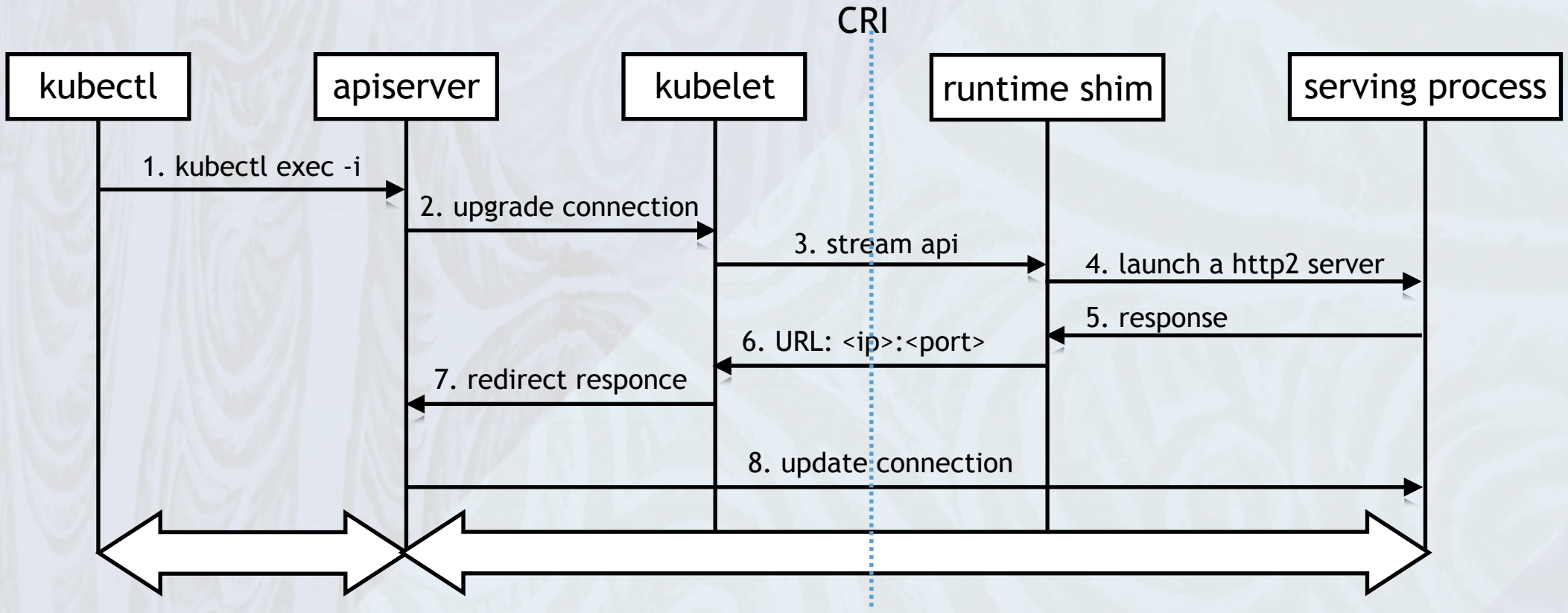


- kubelet becomes bottleneck
- runtime shim in critical path
- code duplication among runtimes/ shims

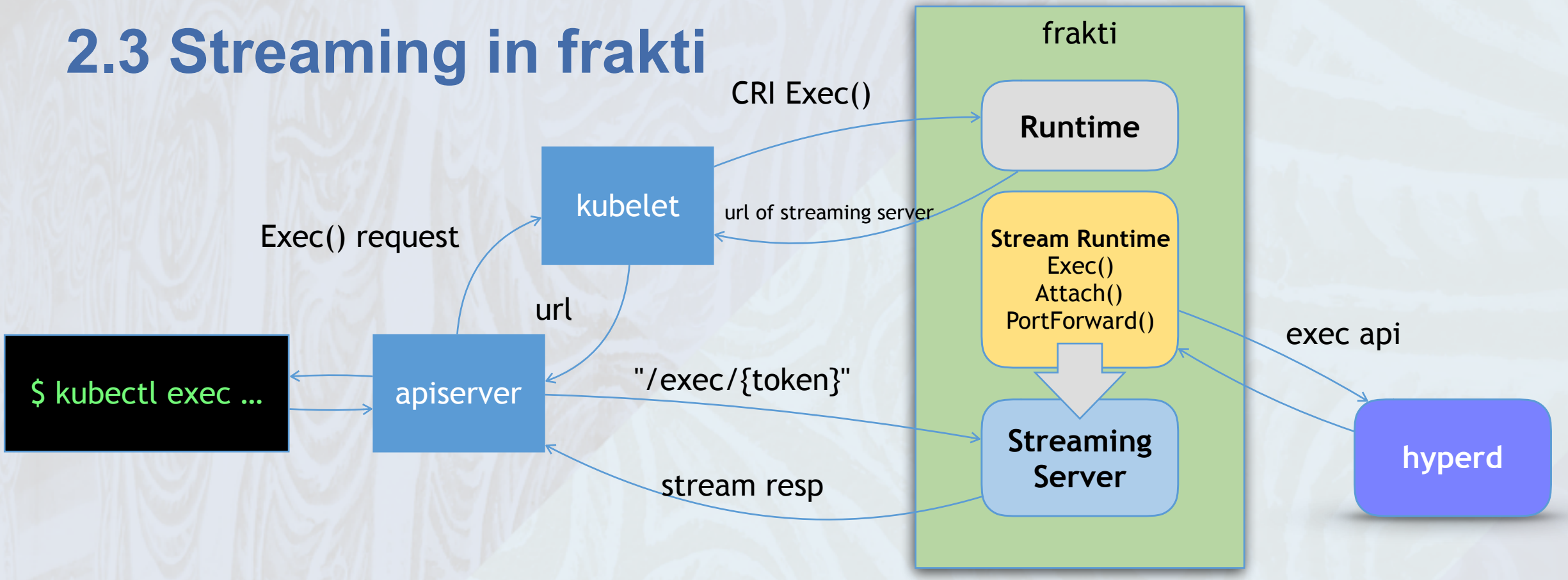
see: Design Doc

2.2 Streaming (CRI version)

see: Design Doc



2.3 Streaming in frakti



3.1 Pod Level Resource Management

```yaml
kind: Pod
metadata:
    name: Pod4
spec:
    containers:
        name: foo
            resources:
                limits:
                    cpu: 20m
                    memory: 2Gi
                requests:
                    cpu: 10m
                    memory: 1Gi
```

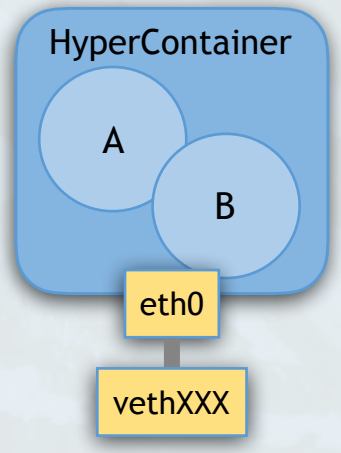
- ✓Enforce QoS classes and eviction
  - ✓Guaranteed
  - ✓Burstable
  - ✓BestEffort
- ✓Resource accounting
- ✓Charge container overhead to the pod instead of the node
  - streaming server , containerd-shim (per-container in docker)

```
/ROOT/burstable/Pod4/cpu.quota = 20m
/ROOT/burstable/Pod4/cpu.shares = 10m
/ROOT/burstable/Pod4/memory.limit_in_bytes = 2Gi
```

3.2 Pod Level Resource Management in Frakti

- ✓Pod sandbox expects resource limits been set **before** start
- ✓Pod level cgroups values are used for pod sandbox's resource spec
  - ✓/sys/fs/cgroup/memory/kubepods/burstable/podID/
    - **Memory of VM** = memory.limit_in_bytes
  - ✓/sys/fs/cgroup/cpu/kubepods/burstable/podID/
    - **vCPU** = cpu.cfs_quota_us/cpu.cfs_period_us
- ✓If not set:
  - ✓1 vCPU, 64MB memory

4. CNI Network in Frakti



- ✓Pod sandbox requires network been set **before** start
- ✓Workflow in frakti:
  1. Create a network NS for sandbox
  2. **plugin.SetUpPod(NS, podID)** to configure this NS
  3. Read the network info from the NS and cache it
  4. Also checkpoint the NS path for future usage (TearDown)
  5. Use cached network info to configure sandbox VM
  6. Keep scanning /etc/cni/net.d/xxx.conf to update cached info

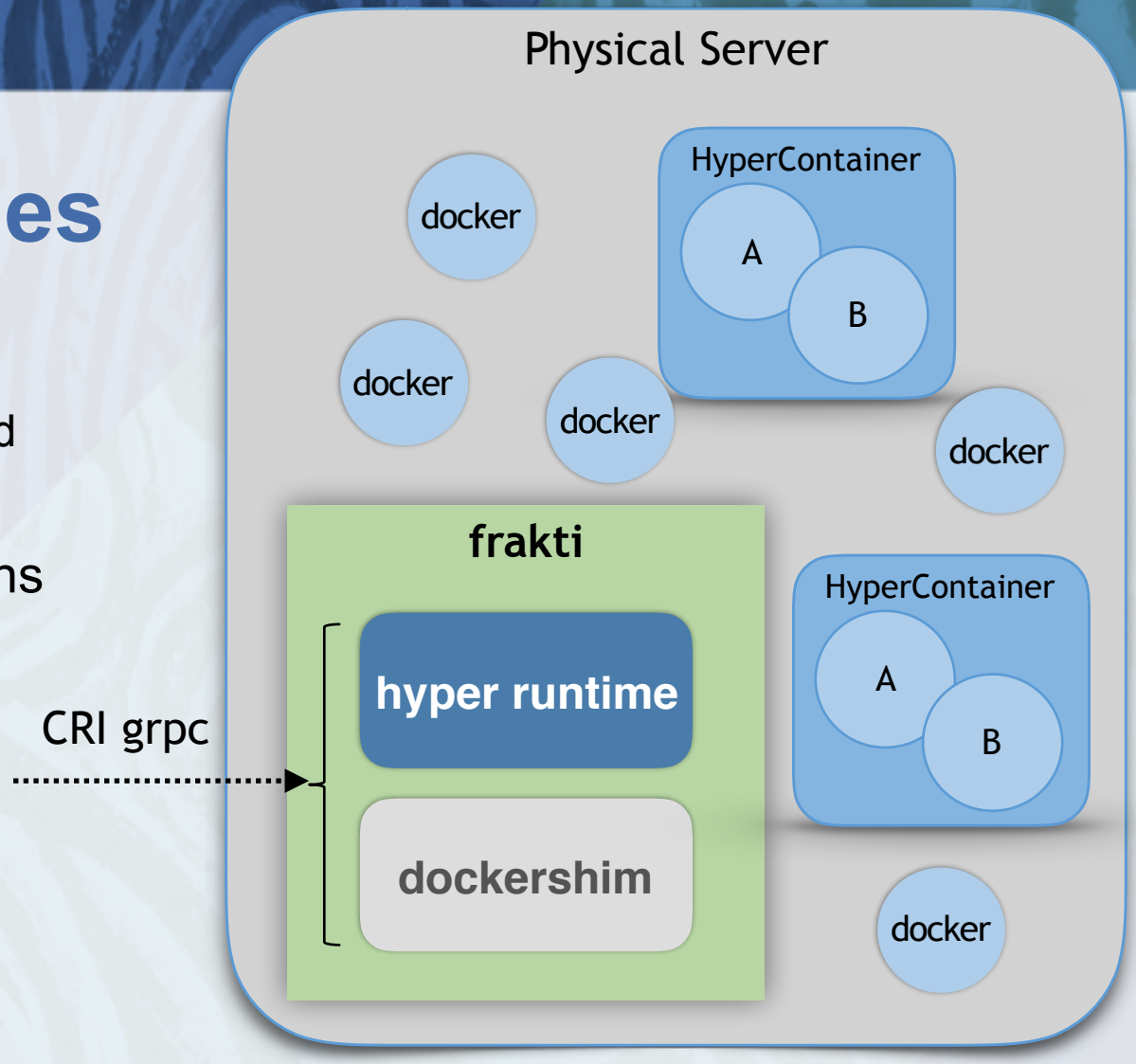
```
type NetworkInfo struct {
    IfName      string
    Mac         net.HardwareAddr
    Ip          *net.IPNet
    Gateway     string
    BridgeName string
}
```

5.1 More Than Hypervisor

- ✓There's are some workload can not be handled by hypervisor …
  - ✓privileged
  - ✓host namespace (network, pid, ipc)
  - ✓user prefer to run them in Linux containers
- ✓And kubelet does not want deal with multiple runtimes on same node
  - ✳complicated
  - ✳break the current model

5.2 Frakti: Mixed Runtimes

- Handled by build-in dockershim
  - host namespace, privileged, specially annotated
- Use the same CNI network
- Mixed run micro-services & legacy applications
  - hyper: independent kernel
- High resource efficiency
  - Remember the core idea of Borg?
  - When workload classes meet QoS tiers
    - Guaranteed VS Best-Effort job

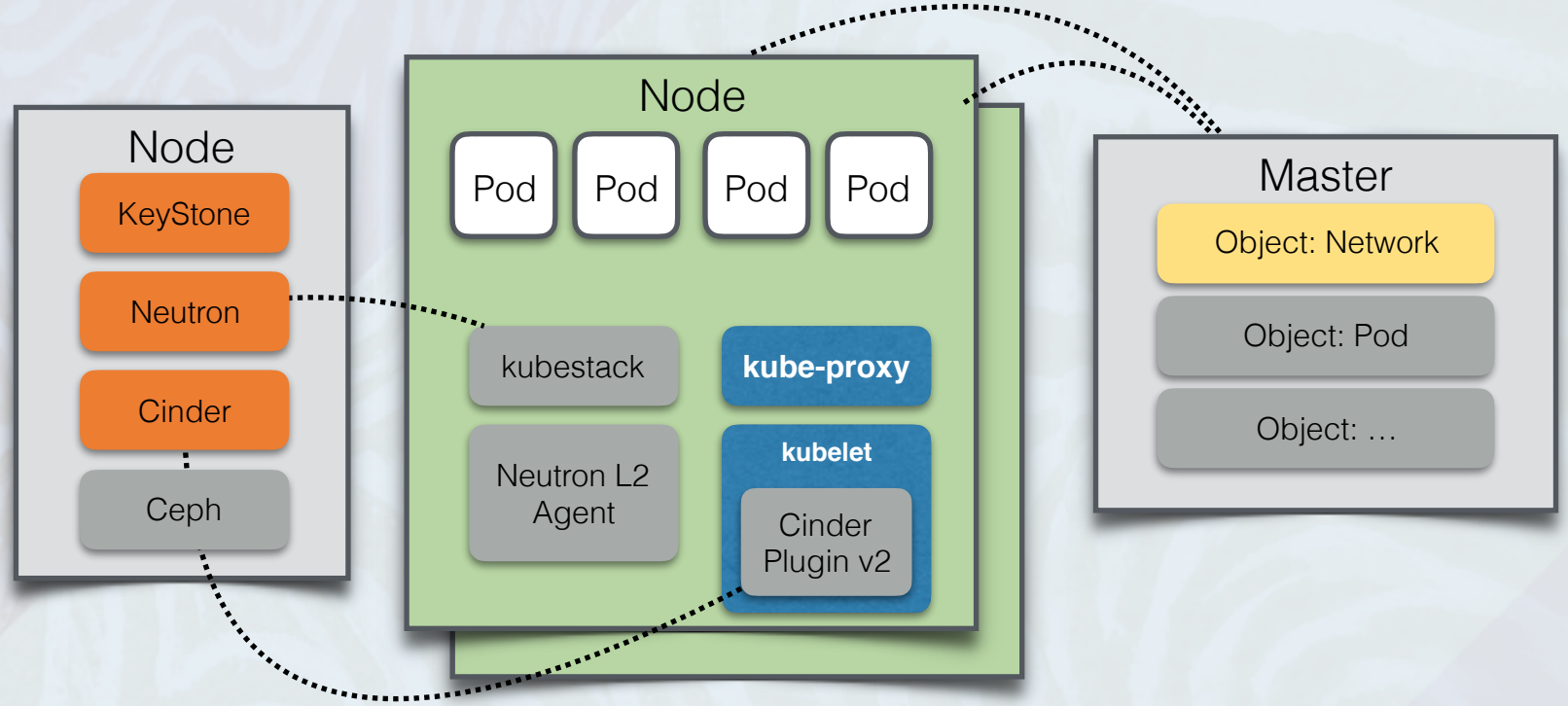


But frakti is Only Part of the Whole Picture

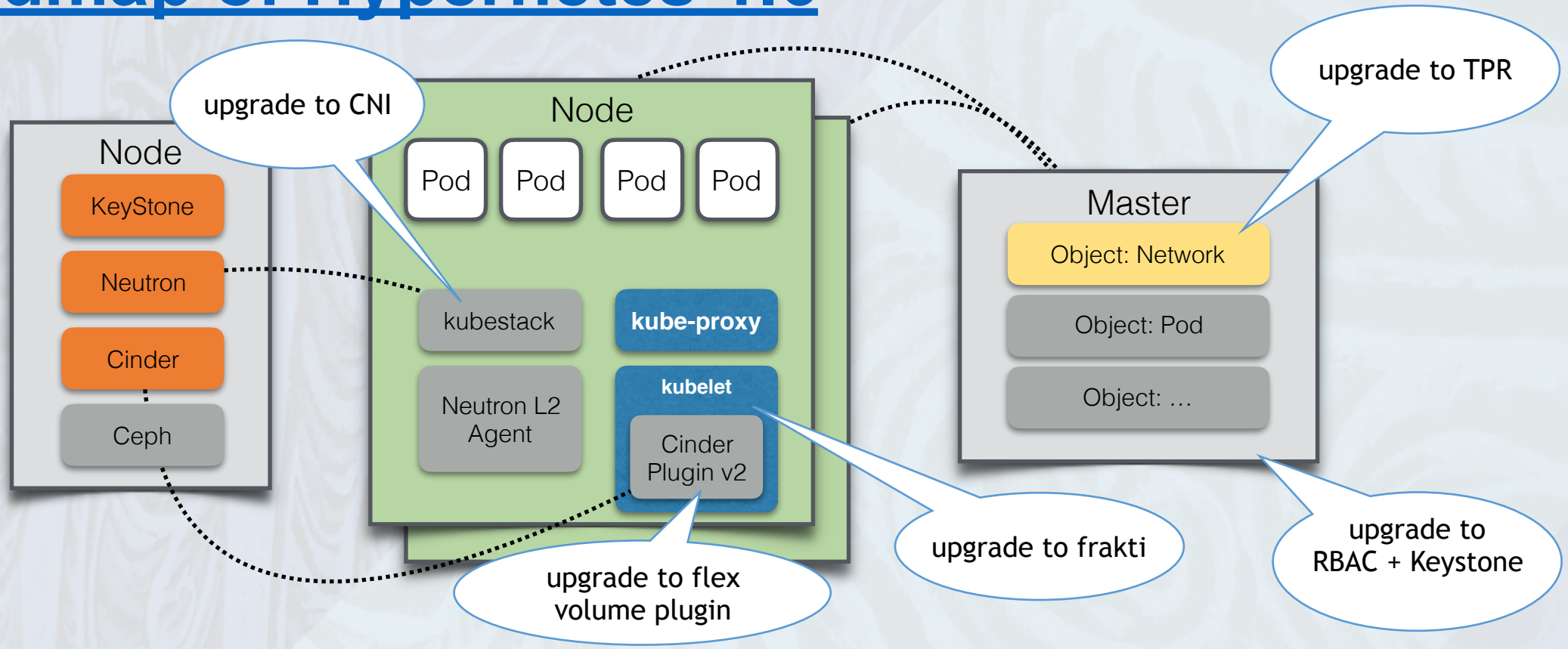
- ✓Hypernetes
  - ✓HyperContainer
  - ➡multi-tenancy
  - ➡isolated network
  - ➡persistent volume

Architecture of Hypernetes < v1.3

- ✓Multi-tenant
  - ✓Top level resource: Network
  - ✓tenant 1: N Network
- ✓Network
  - ✓Network -> Neutron “Port”
  - ✓kubelet -SetUpPod() -> kubestack -> Neutron
  - ✓build-in ipvs based kube-proxy
- ✓Persistent Volume
  - ✓**Directly** attach block devices to Pod
- ✓https://hyper.sh



Roadmap of Hypernetes 1.6



Summary

- ✓CRI simplified the most tricky parts of container runtime integration work
  - ✓eliminate pod centric runtime API
  - ✓runtime lifecycle
    - PodSandbox & Container & Image API
  - ✓Checkpoint
    - store the auxiliary data in runtime shim
  - ✓streaming
    - leave to implementation to runtime shim
    - common streaming server library

- ✓Kubernetes plugins make re-innovation possible
  - ✓Third Party Resource
    - for Network object management
  - ✓CNI network
    - simple but powerful
    - while CNM is impossible to be used in runtime other than Docker
- ✓Enable more possibilities
- ✓Success of CRI is the success of orchestration project itself
  - ✓think about containerd

END

Harry Zhang, @resouer, HyperHQ

Most of these CRI efforts owe to my co-worker @feiskyer and the #sig-node!

Thank you!