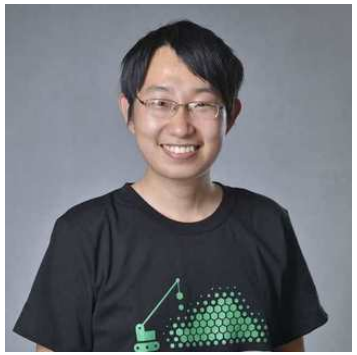
Alibaba Cloud | Worldwide Cloud Services Partner

Surviving Serverless Battle By Secure Runtime, CRI and RuntimeClass

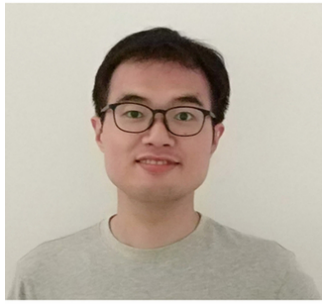
Lei Zhang & Xiaoyu Zhang, Alibaba Group

About US:



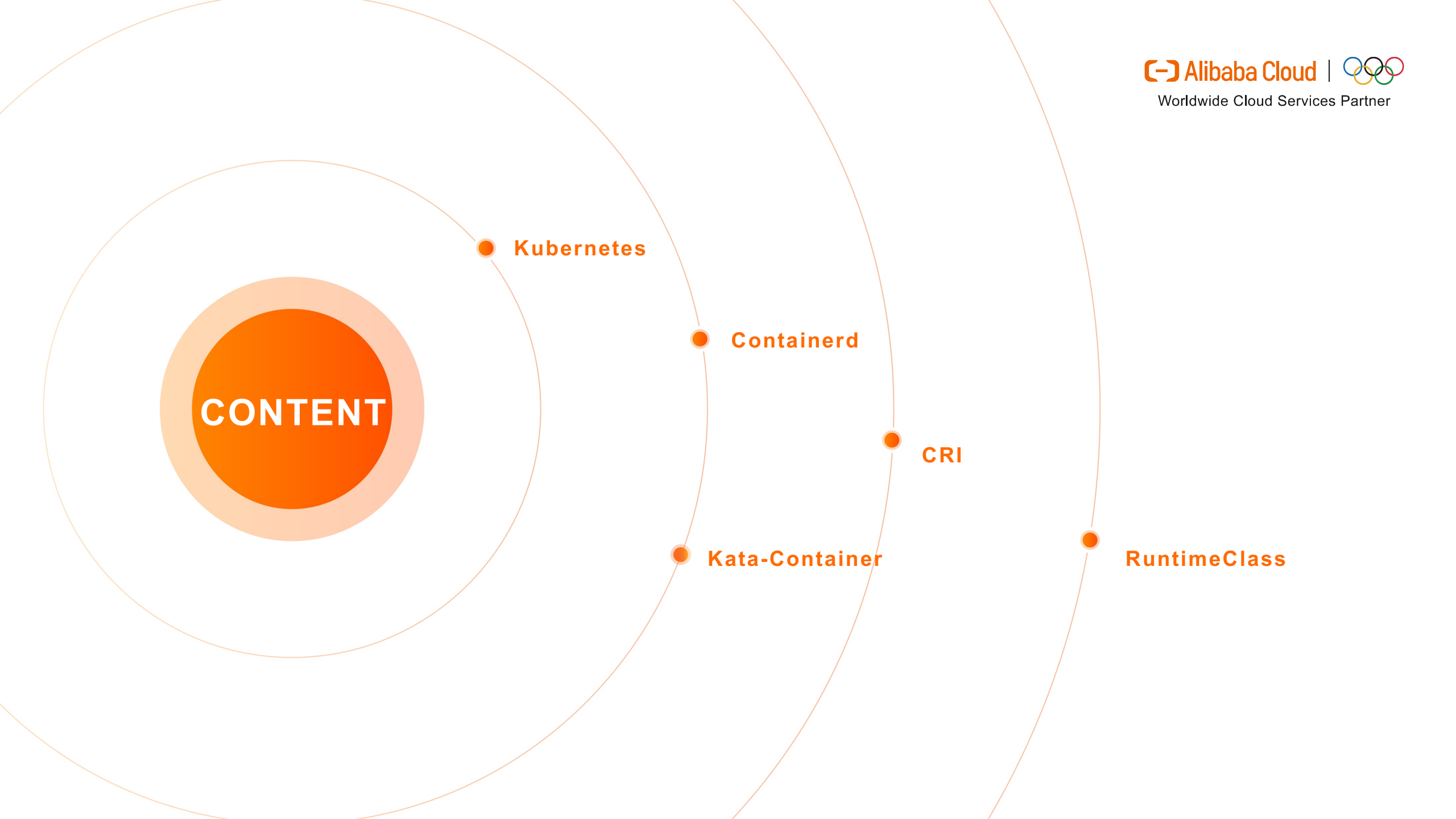
Zhang, Lei:

- Staff Engineer at Alibaba Group

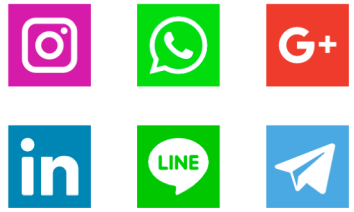


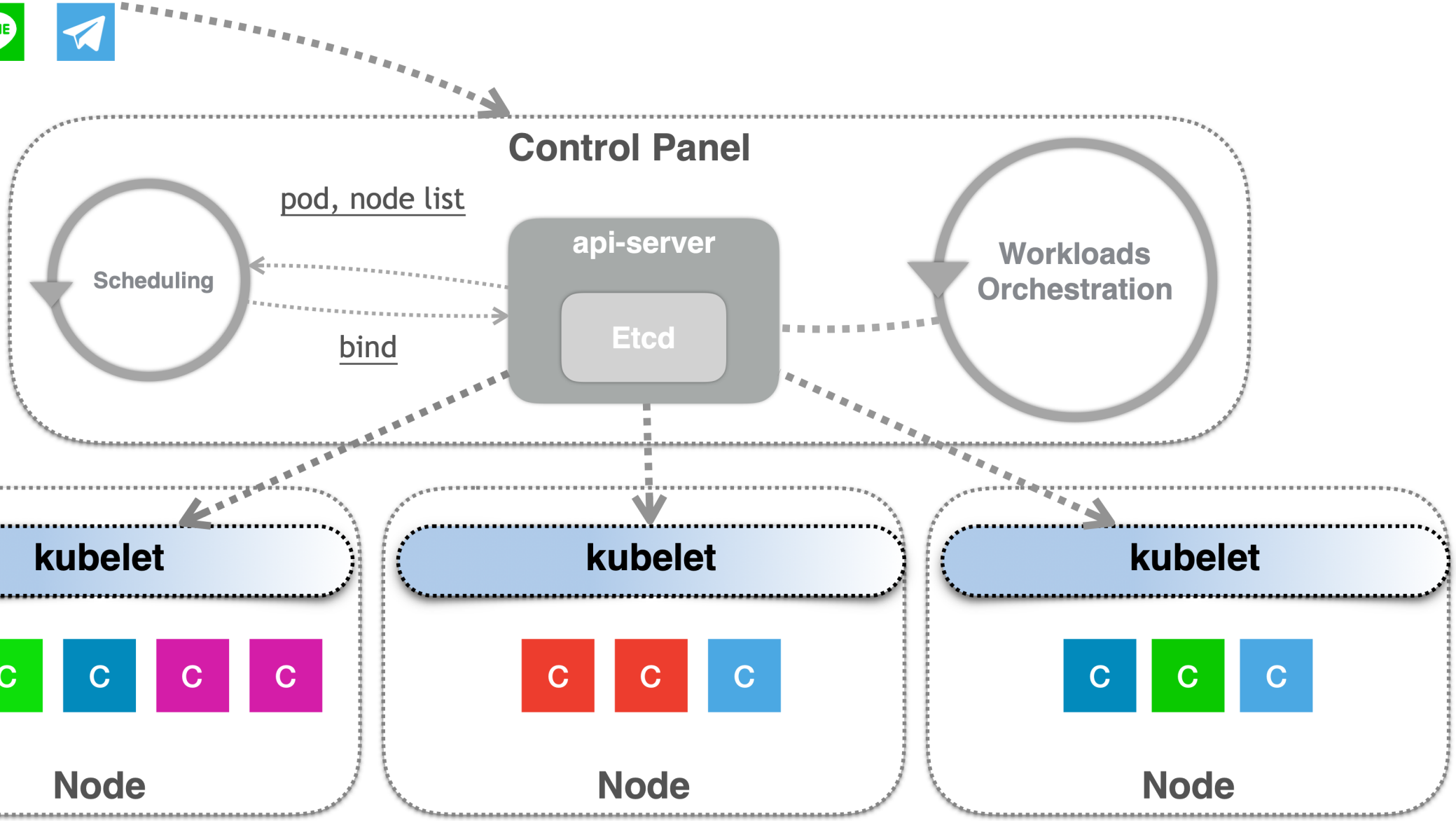
Zhang, Xiaoyu:

- Senior Engineer at Alibaba Group


CONTENT
Kubernetes
Containerd
CRI
Kata-Container
RuntimeClass

Kubernetes



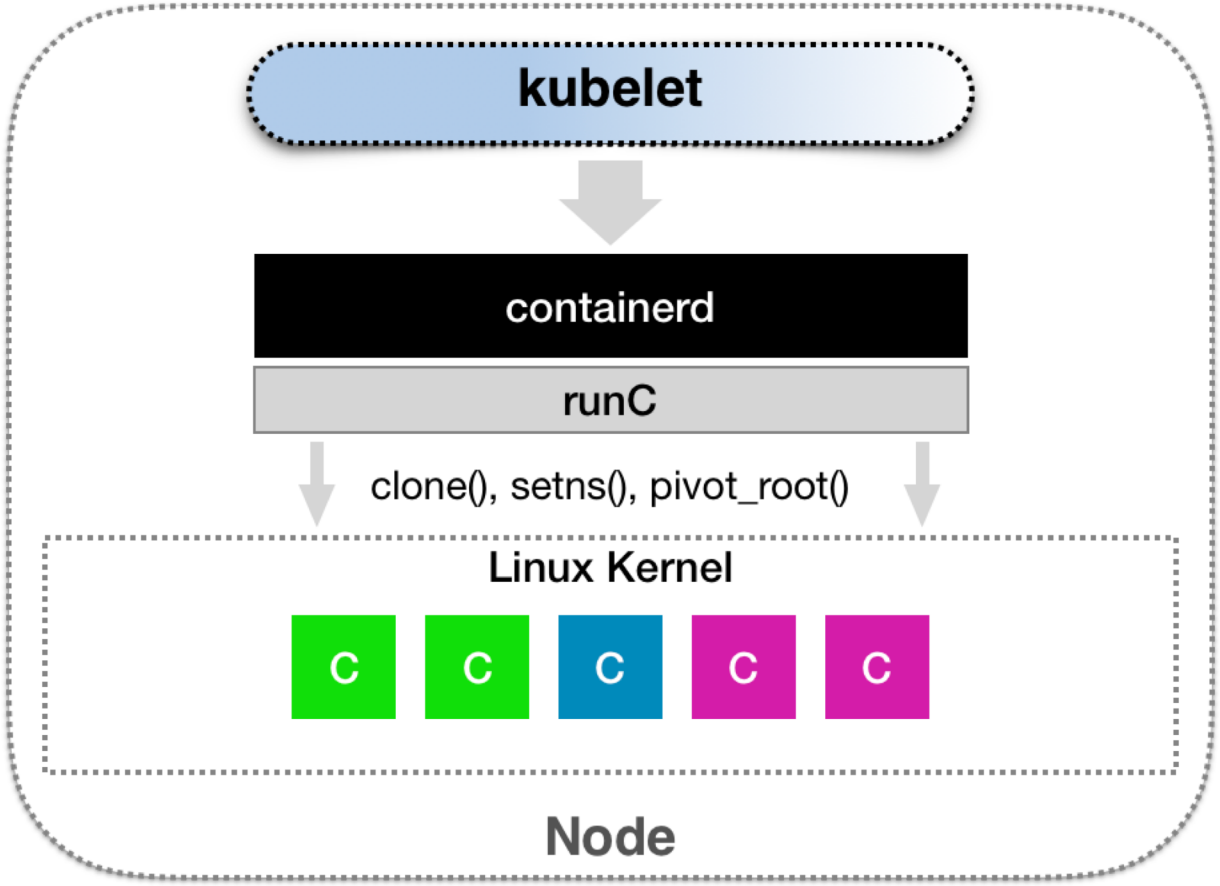


But what's the prerequisite when to deploy a Kubernetes cluster?

- Pick up one runtime

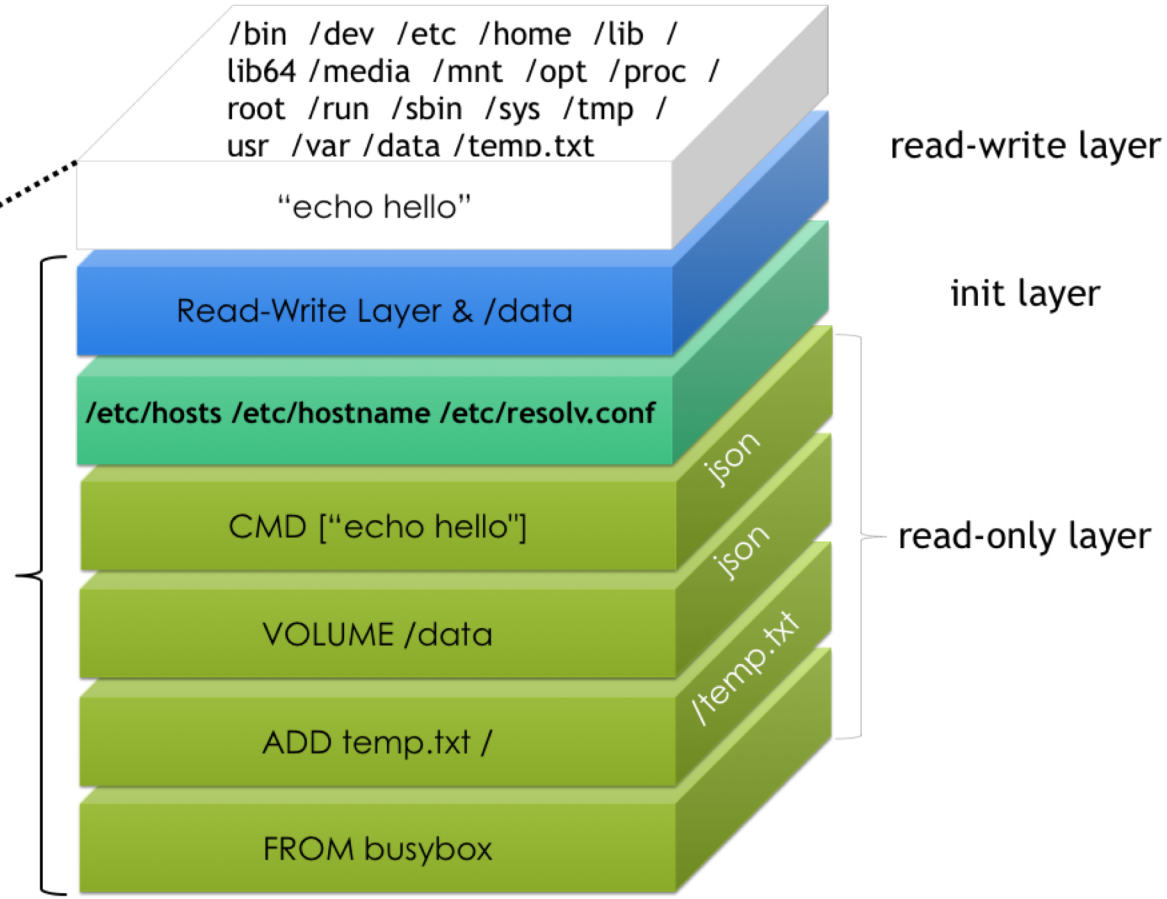
- Install it…

Kubernetes + containerd



Linux Container

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




```
FROM busybox
ADD temp.txt /
VOLUME /data
CMD ["echo hello"]
```

But, Serverless require **hard multi-tenancy**, how?

Firecracker

katacontainers

gVisor

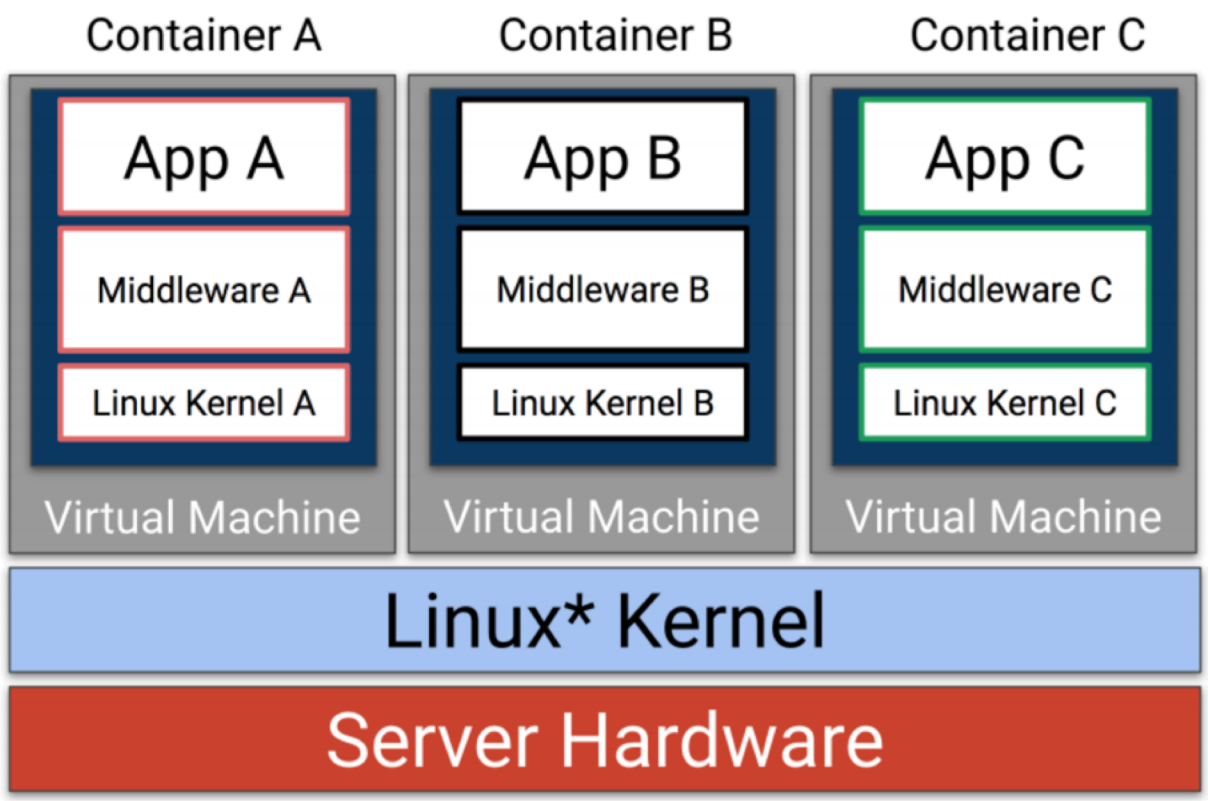
Multi-tenancy of Container Runtime layer

- Linux container
  - Dropping Linux capabilities
  - Read-only mount points
  - Mandatory access controls (MAC)
    - SELinux & AppArmor
  - Dropping syscalls
    - SECCOMP
  - In 99.99% cases
    - wrap containers in VMs

- KataContainers
  - Hardware virtualization
  - Independent Linux instance per Pod
    - e.g. run Linux 3.16 container on a Linux 4.0 host

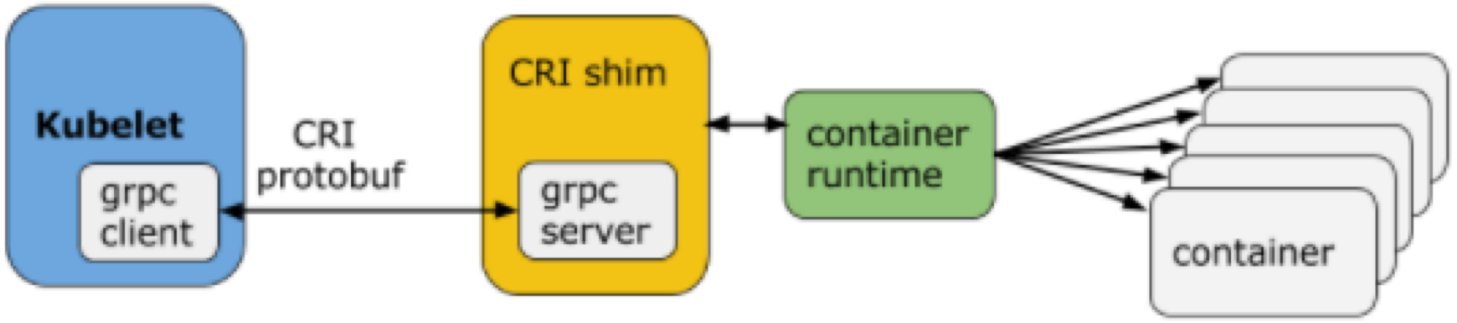
KataContainers

- **Container Runtime**
  - Each Pod is hypervisor isolated
    - Independent guest kernel
  - Secure as VM
  - Fast as container
- **Container Image**
  - Same as Linux container



What's the bridge between Kubernetes and Runtime?

CRI: Container Runtime Interface



Container Runtime Interface (CRI)

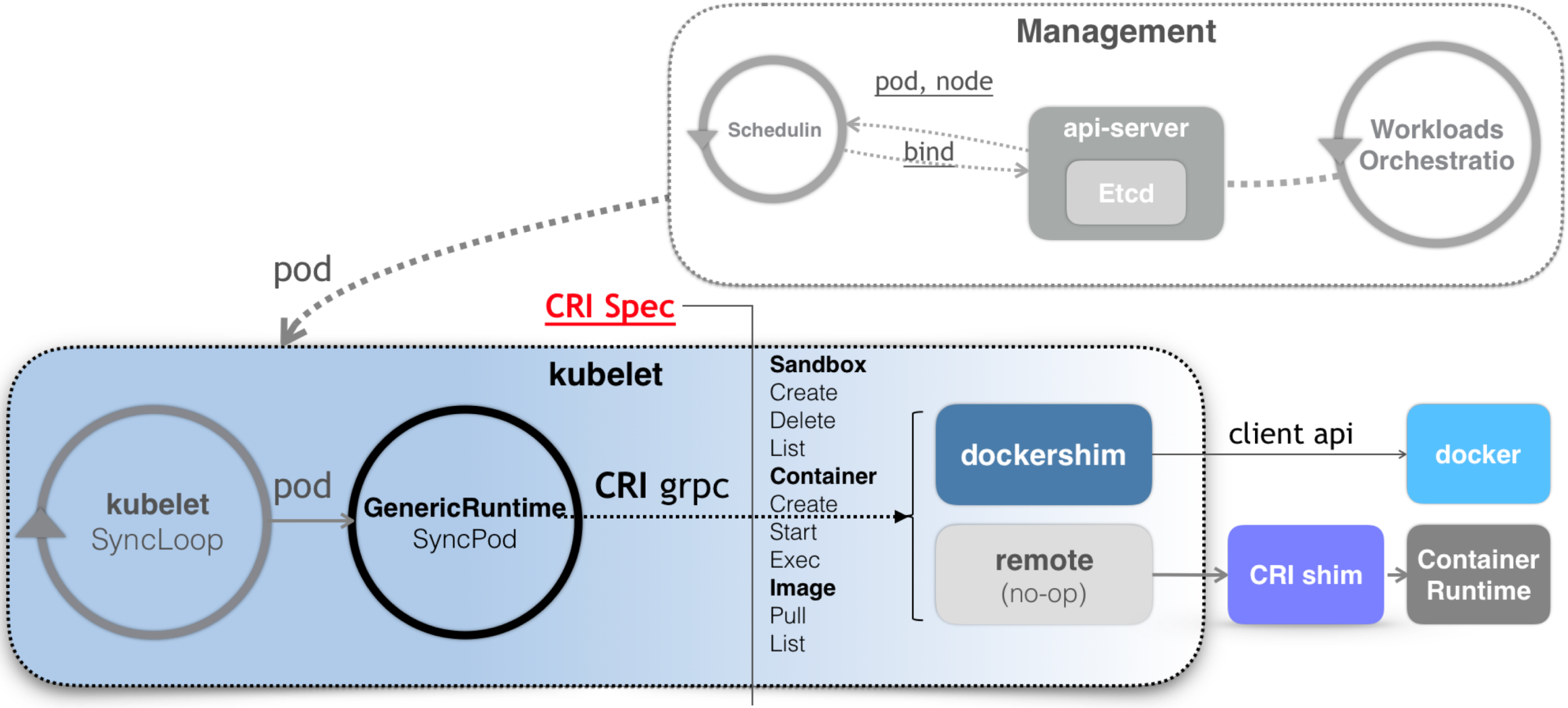
- Describe what kubelet expects from container runtimes
- Imperative container-centric interface
  - **why not pod-centric?**
    - Every container runtime implementation needs to understand the concept of pod.
    - Interface has to be changed whenever new pod-level feature is proposed.

CRI Spec

- **Sandbox**
  - How to isolate Pod environment?
    - Linux container: infra container + pod level cgroups
    - Kata: light-weighted VM
- **Container**
  - Linux container: namespace + cgroups
  - Kata: namespace containers controlled by hyperstart

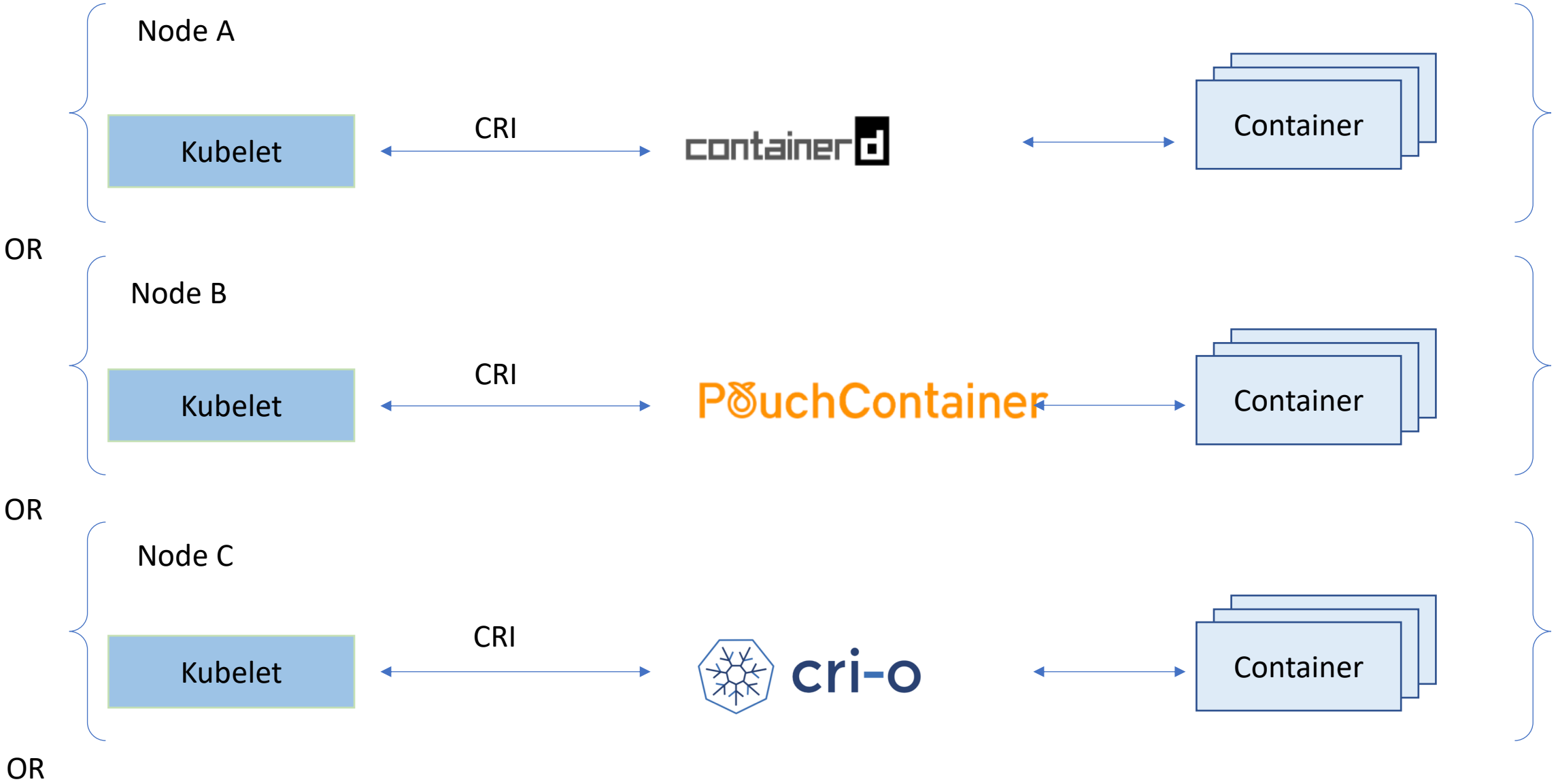
```
type RuntimeService interface {
    RunPodSandbox(config *kubeapi.PodSandboxConfig) (string, error)
    StopPodSandbox(podSandboxID string) error
    RemovePodSandbox(podSandboxID string) error
    PodSandboxStatus(podSandboxID string) (*kubeapi.PodSandboxStatus, error)
    ListPodSandbox(filter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

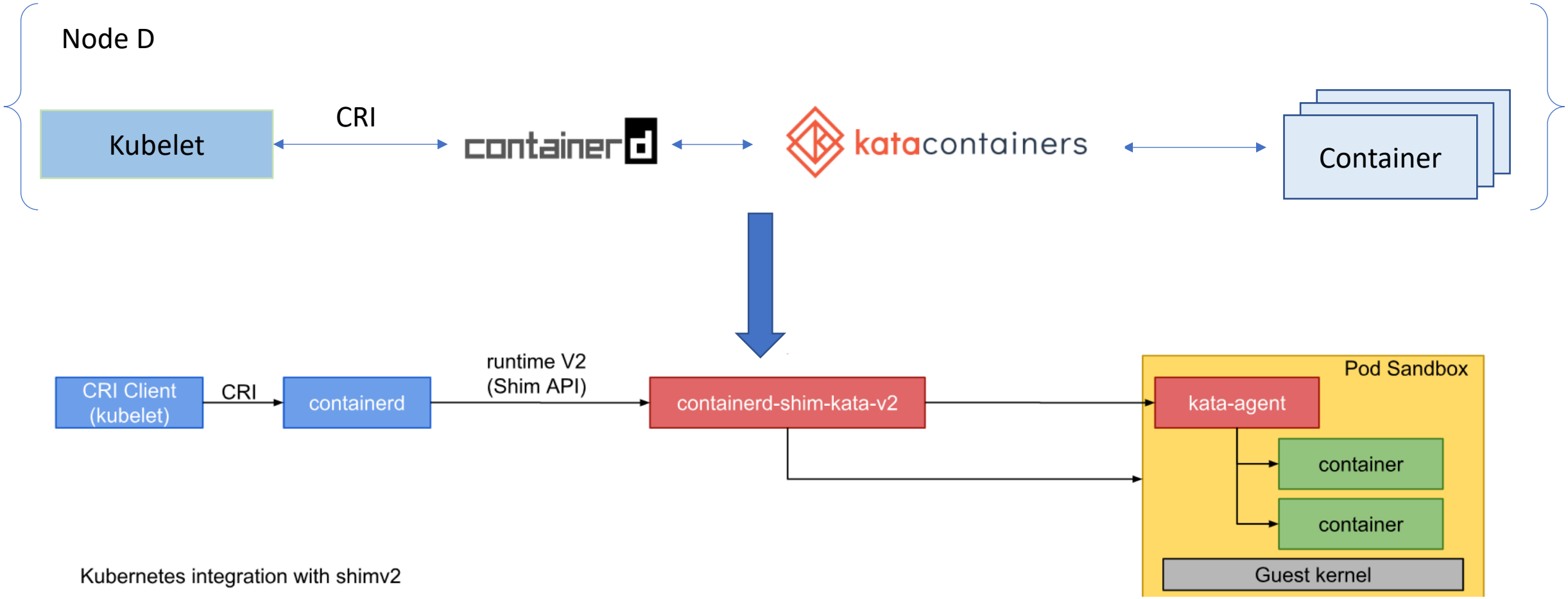


Alibaba Cloud | Worldwide Cloud Services Partner

Make a choice for your Cluster



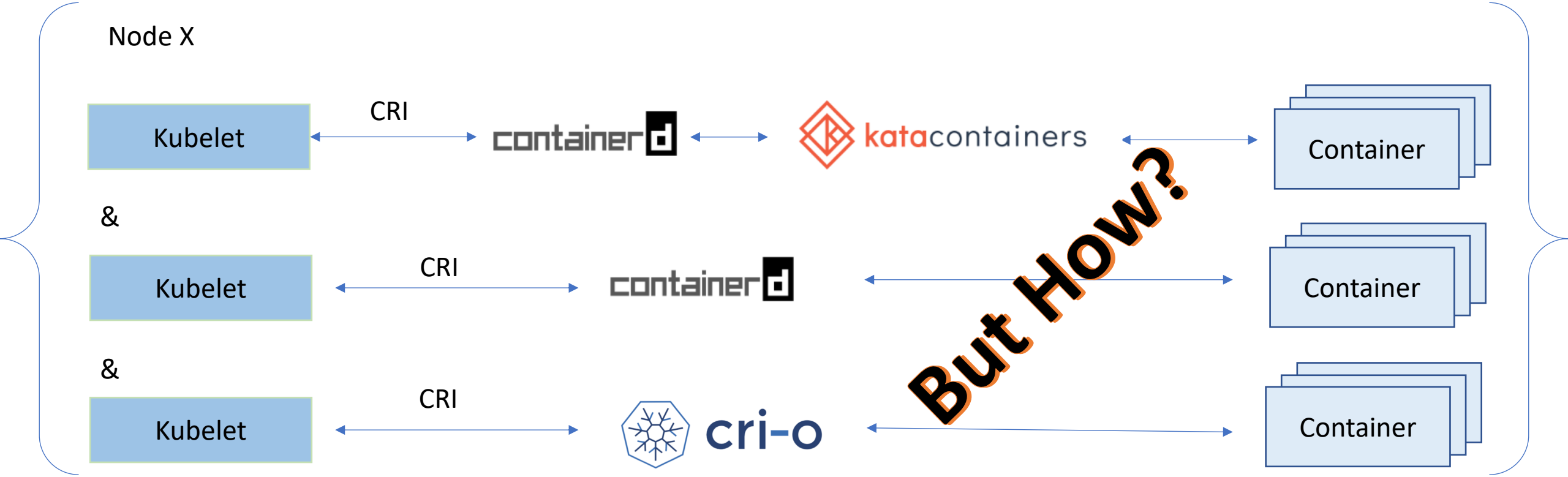
Make a choice for your Cluster



Kubernetes integration with shimv2

Serverless

- Application has no sense of infrastructure, so a Kubernetes cluster should look like below:



RuntimeClass

- How can we use different runtimes in a single Kubernetes cluster?
  - RuntimeClass
- What can RuntimeClass do for us?
  - More choices
  - Better abstract
- Why RuntimeClass is significant important to CNCF and the whole OpenSource ecosystem?
  - Provide a mechanism for surfacing container runtime properties to the control plane
  - Support multiple runtimes per-cluster, and provide a mechanism for users to select the desired runtime

K8s + RuntimeClass + X

Alibaba Cloud | Worldwide Cloud Services Partner

- RuntimeClass Config

```
apiVersion: node.k8s.io/v1beta1
kind: RuntimeClass
metadata:
  name: katacontainer
handler: kata-runtime
```

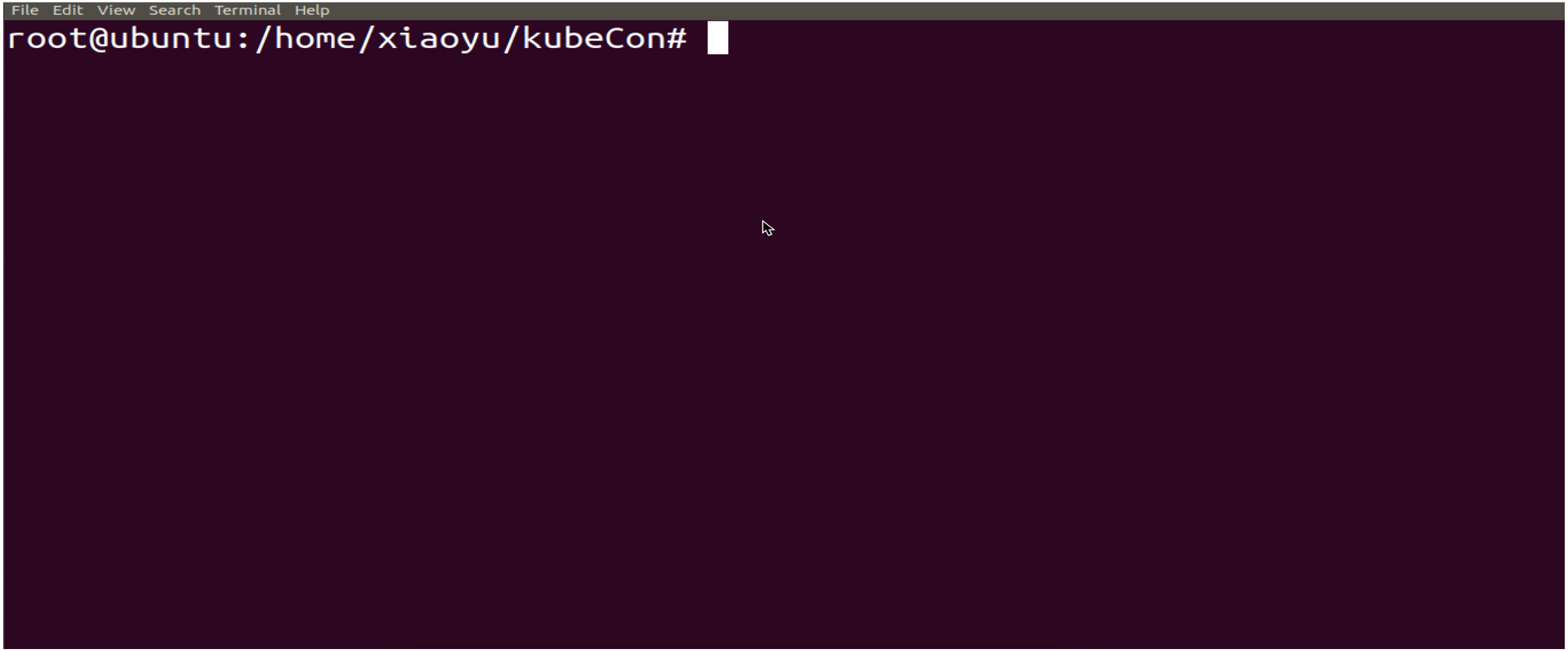
- RuntimeClass Use Demo

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
spec:
  runtimeClassName: katacontainer
  containers:
  - name: busybox
    image: busybox
    command: ['sh', '-c', 'echo The app is running! && sleep 3600']
```

Live Demo

Alibaba Cloud | Worldwide Cloud Services Partner

Here is a demo based on Kubernetes V1.14.s showing RuntimeClass.



KEP

Kubernetes enhancement proposal：

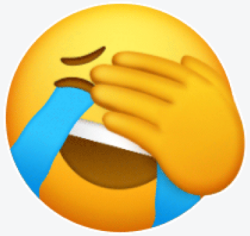
- https://github.com/kubernetes/enhancements/blob/master/keps/sig-node/runtime-class.md
- https://github.com/kubernetes/enhancements/blob/master/keps/sig-node/runtime-class-scheduling.md

Serverless Project Reinforcement

- **Knative**

- **Kubeless**

- **…**



Unfortunately no serverless project supports RuntimeClass currently.

Knative

```
apiVersion: serving.knative.dev/v1alpha1 # Current version of Knative
kind: Service
metadata:
  name: helloworld-go # The name of the app
  namespace: default # The namespace the app will use
spec:
  template:
    spec:
      containers:
        - image: gcr.io/knative-samples/helloworld-go # The URL to the image of the app
          env:
            - name: TARGET # The environment variable printed out by the sample app
              value: "Go Sample v1"
```

Copy from:
https://knative.dev/docs/install/getting-started-knative-app/

Knative

```
apiVersion: serving.knative.dev/v1alpha1 # Current version of Knative
kind: Service
metadata:
  name: helloworld-go # The name of the app
  namespace: default # The namespace the app will use
spec:
  template:
    spec:
      runtimeClassName: katacontainer
      containers:
        - image: gcr.io/knative-samples/helloworld-go # The URL to the image of the app
          env:
            - name: TARGET # The environment variable printed out by the sample app
              value: "Go Sample v1"
```

```
root@ubuntu:/home/xiaoyu/kubeCon/yaml# kubectl apply -f knative-case.yaml
Error from server (InternalError): error when creating "knative-case.yaml": Internal error occurred: admission webhook "webhook.serving.knative.dev" de
nied the request: mutation failed: cannot decode incoming new object: json: unknown field "runtimeClassName"
```

Kubeless

---------- test.py---------
```
def hello(event, context):
  print event
  return event['data']
```

```
$ kubeless function deploy hello --runtime python2.7 \
                                  --from-file test.py \
                                  --handler test.hello
```

```
kubeless function call hello --data 'Hello world!'
```

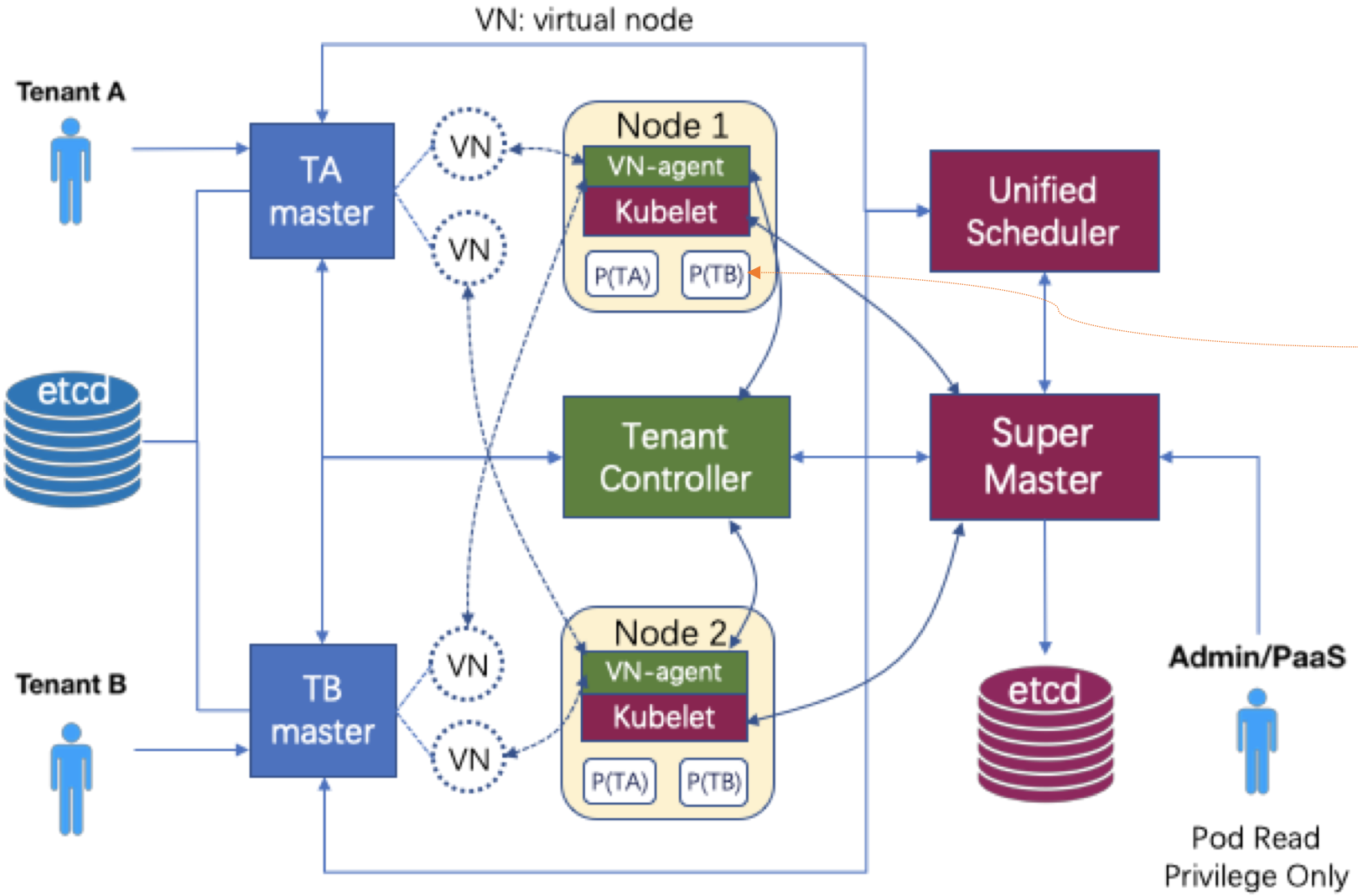
There is even no place to edit...

Copy from:
https://kubeless.io/docs/quick-start/

There is still a long journal to go…

Alibaba Cloud | Worldwide Cloud Services Partner

Case Study: Alibaba Serverless Infra



1. Per tenant per K8s control plane
2. Tenant only see Virtual Nodes
3. Tenant Pods are handled by KataContainers

The Virtual Cluster based hard multi-tenancy model **will be open sourced in KubeCon NA 2019**

See upstream proposal: https://docs.google.com/document/d/1EELeVaduYZ65j4AXg9bp3Kyn38GKDU5fAJ5LFcxt2ZU/edit#