KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

So what if I don’t want my Persistent Storage to be yet another Bind Mount

Deep Debroy and Feng Wang

So what if I don’t want my Persistent Storage to be yet another Bind Mount

KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

October 24-28, 2021

Deep Debroy
Software Engineer
Apple

Feng Wang
Software Engineer
Databricks

Introduction

- Current Lifecycle Flow for Pods with Persistent Storage
  - bind mounts

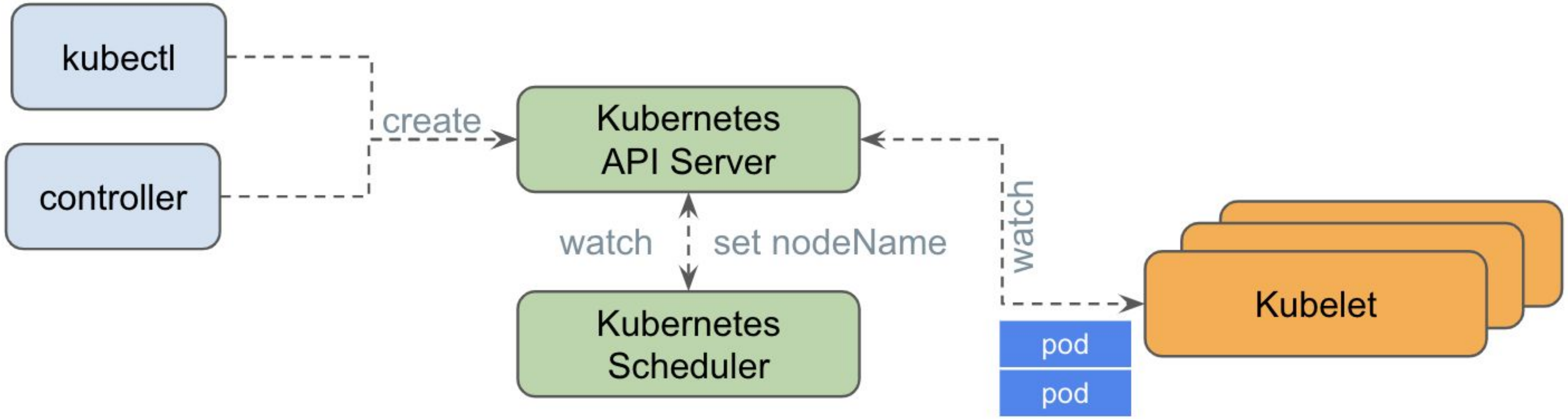
- Alternatives to the Standard Flow for Persistent Storage
  - from a MicroVM perspective
  - challenges

- Surfacing “Direct Assignment” Flows to CSI Plugins
  - with Kubernetes and Kata awareness
  - generic support

- Conclusion

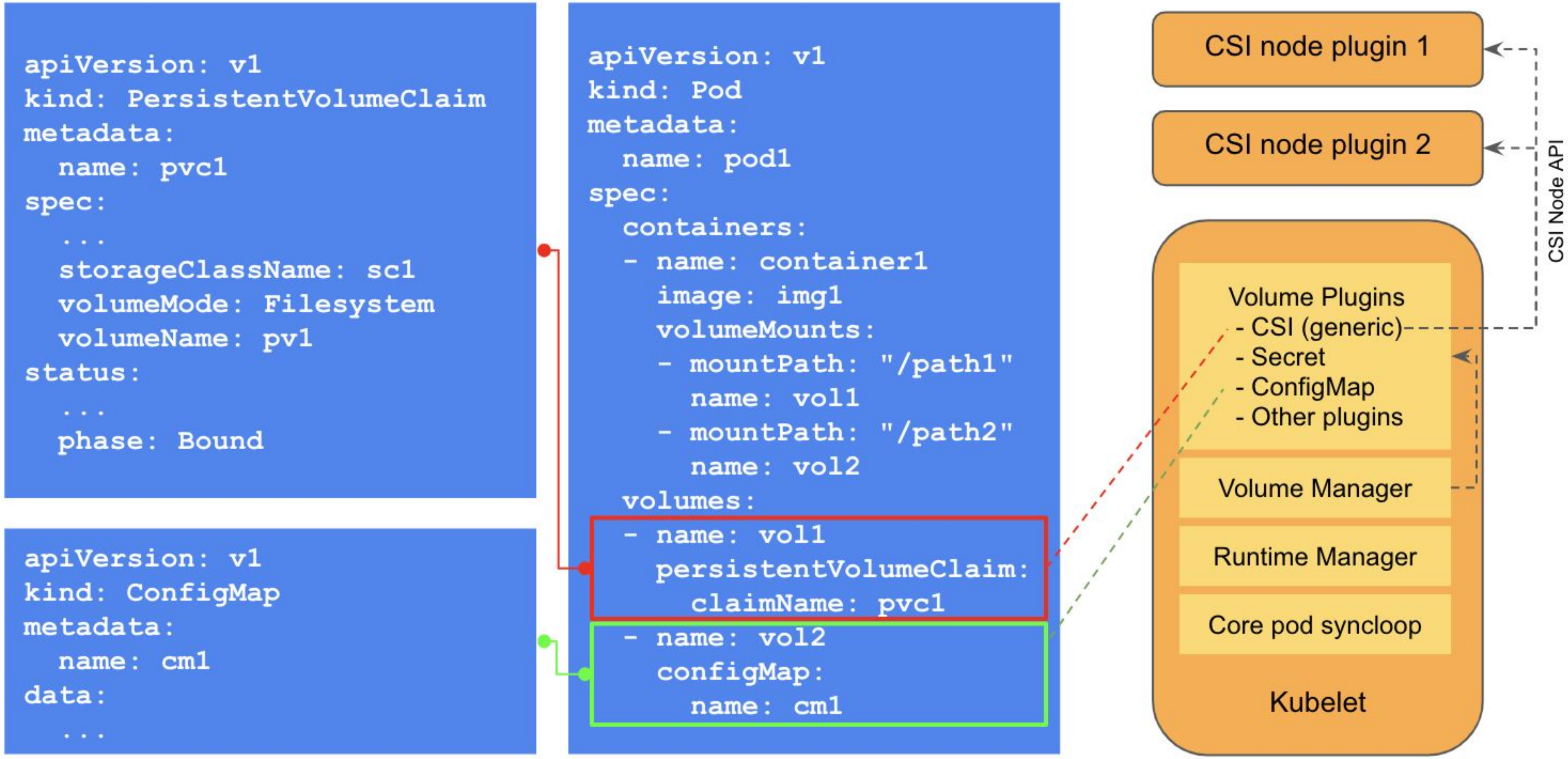
Pod Startup from Kubelet’s Perspective

- Kubelet gets notified about pods scheduled on the node



Pod Startup from Kubelet’s Perspective

- Kubelet mounts the specified volumes in the pod:
  - In-line volumes (configMap, secrets, etc)
  - Persistent Volumes bound to PVCs



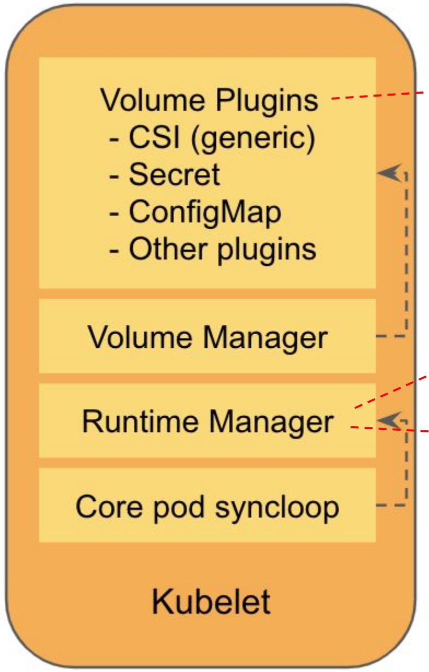
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc1
spec:
  ...
  storageClassName: sc1
  volumeMode: Filesystem
  volumeName: pv1
status:
  ...
  phase: Bound
```

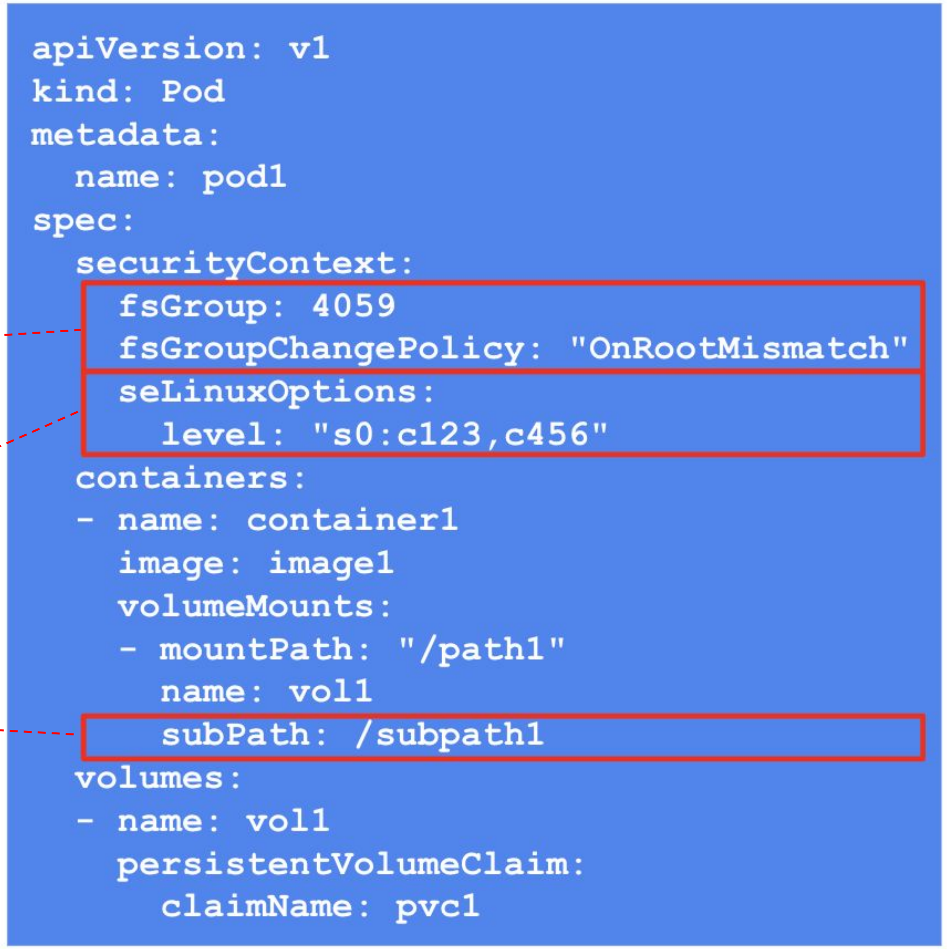
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cm1
data:
  ...
```

```
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
  - name: container1
    image: img1
    volumeMounts:
    - mountPath: "/path1"
      name: vol1
    - mountPath: "/path2"
      name: vol2
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc1
  - name: vol2
    configMap:
      name: cm1
```

Pod Startup from Kubelet’s Perspective

- Kubelet prepares the file system mount
  - Apply FsGroup based on FsGroupChangePolicy
  - Probe and prepare subpaths
  - Determine Security-Enhanced Linux (SELinux) labelling

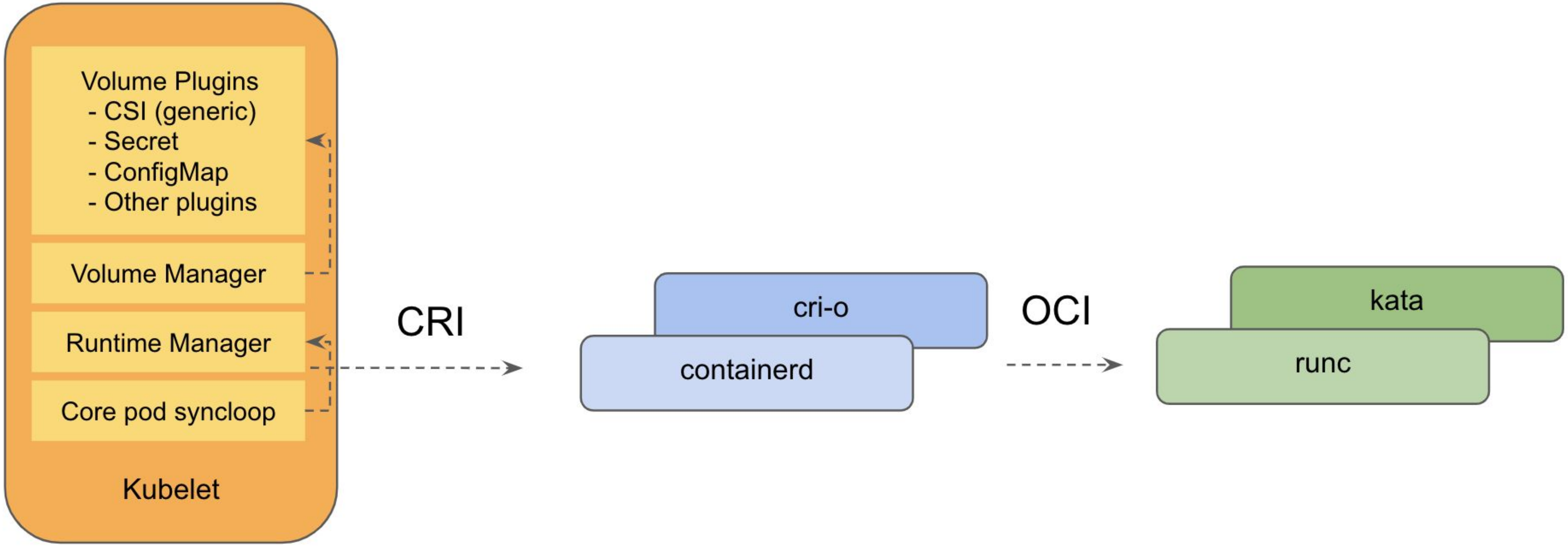




```
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  securityContext:
    fsGroup: 4059
    fsGroupChangePolicy: "OnRootMismatch"
    seLinuxOptions:
      level: "s0:c123,c456"
  containers:
  - name: container1
    image: image1
    volumeMounts:
    - mountPath: "/path1"
      name: vol1
      subPath: /subpath1
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc1
```

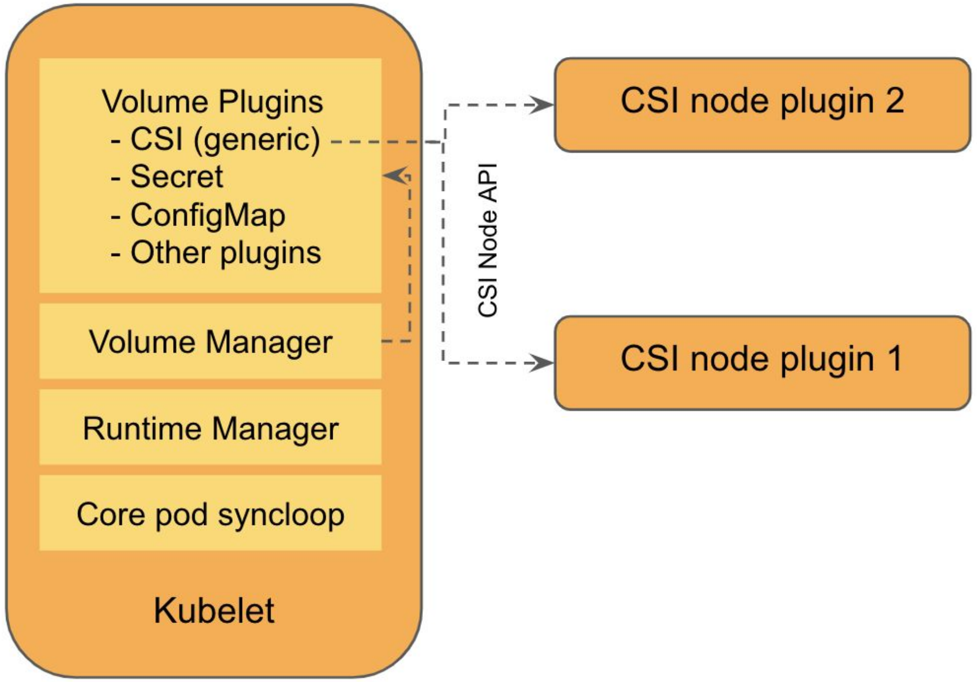
Pod Startup from Kubelet’s Perspective

- Kubelet invokes CRI implementation to
  - Create pod sandbox
  - Pull container images
  - Create containers

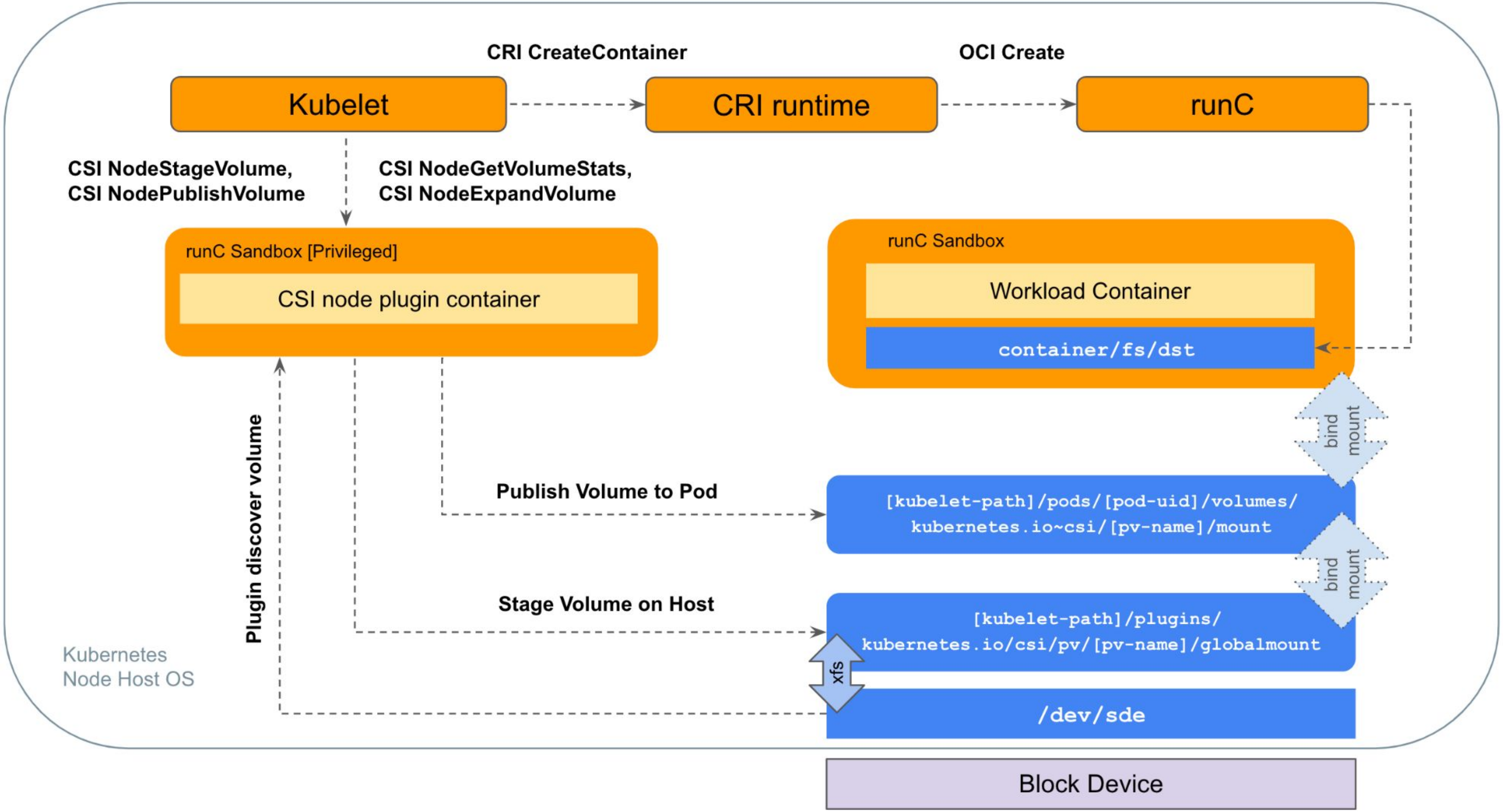


Volume Operations during Pod Lifetime

- Kubelet reports file system stats
- Kubelet resizes the mounted volume



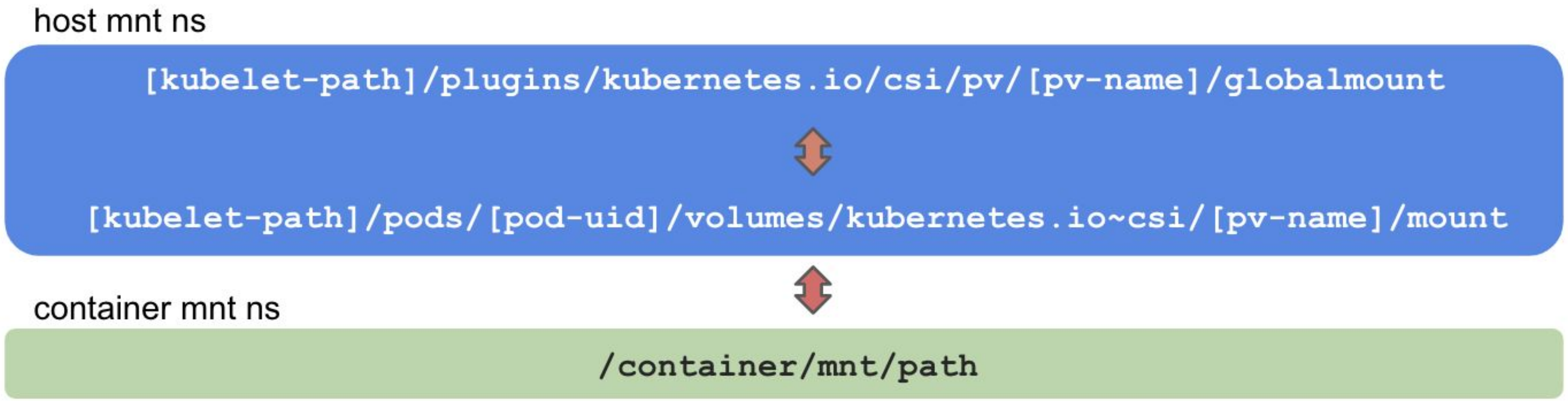
Regular Path for FS Operations on a PV



File System Bind Mounts Along the Way

- NodeStaging global path → NodePublish target path
  - Created during CSI NodePublish

- NodePublish target path → Path inside container
  - Created during OCI create container



host mnt ns

`[kubelet-path]/plugins/kubernetes.io/csi/pv/[pv-name]/globalmount`

`[kubelet-path]/pods/[pod-uid]/volumes/kubernetes.io~csi/[pv-name]/mount`

container mnt ns

`/container/mnt/path`

Current Assumptions in Kubelet

- All volumes have file-systems mounted before container bring-up
  - whenever PV VolumeMode is FileSystem
- Post mount actions *invoked on mounted file system* by Kubelet (and container runtime):
  - fsGroup ownership
  - subpath checks
  - SELinux labelling

Alternative Flow for Volume Mounts

How about a different sequence?

Alternative Flow for Volume Mounts

How about a different sequence?

mount volumes *after* container bringup *without* using raw block mode
[use case: microVM environments like Kata]

MicroVM-based Runtime

Kata Containers – A container runtime that runs workloads in a virtual machine.

- **Secure** – container workload runs on a dedicated kernel
- **Lightweight** – faster startup than a full-blown virtual machine
- **Compatible** – compliant to OCI container format and containerd/crio shim interface.

Kata Containers architecture

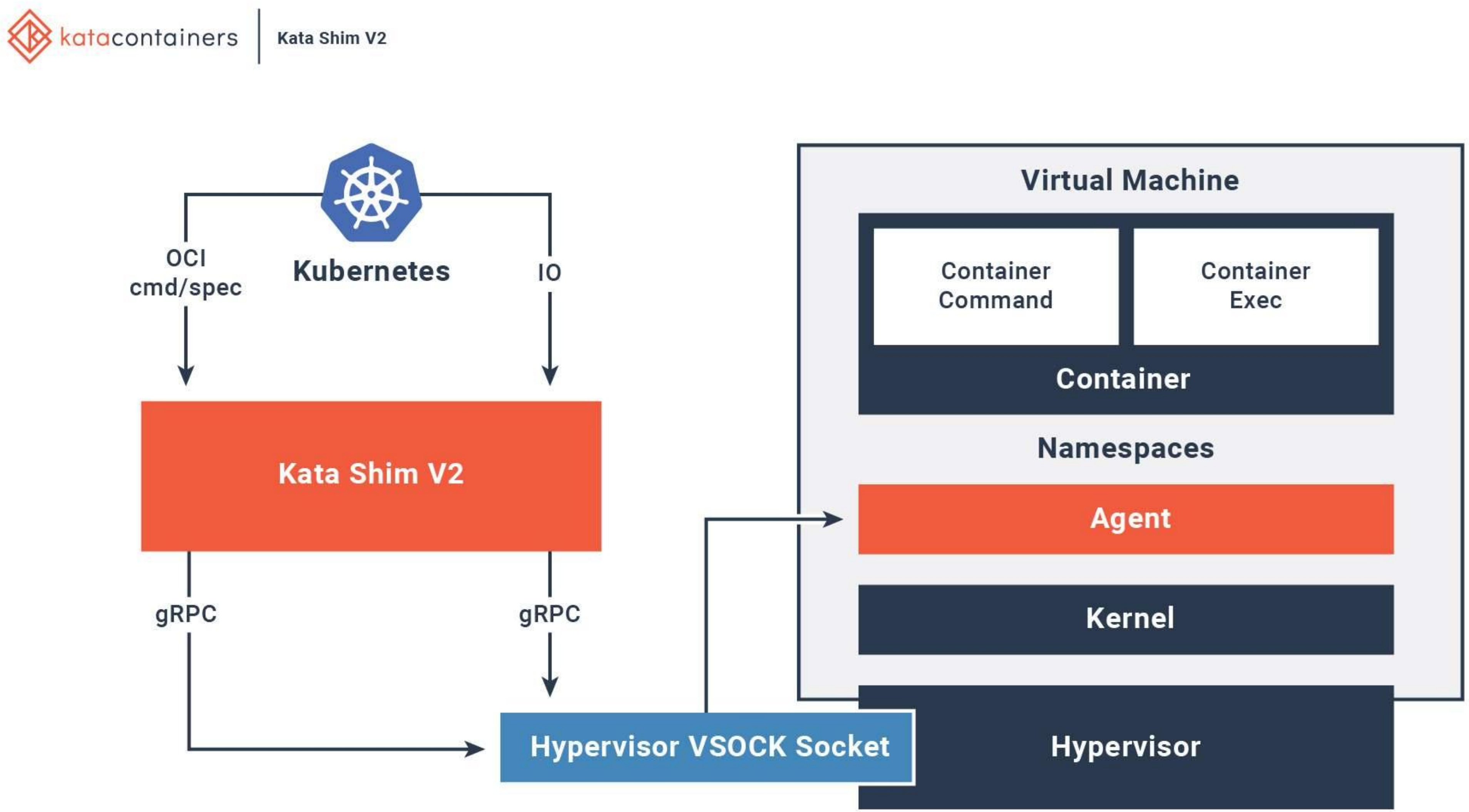
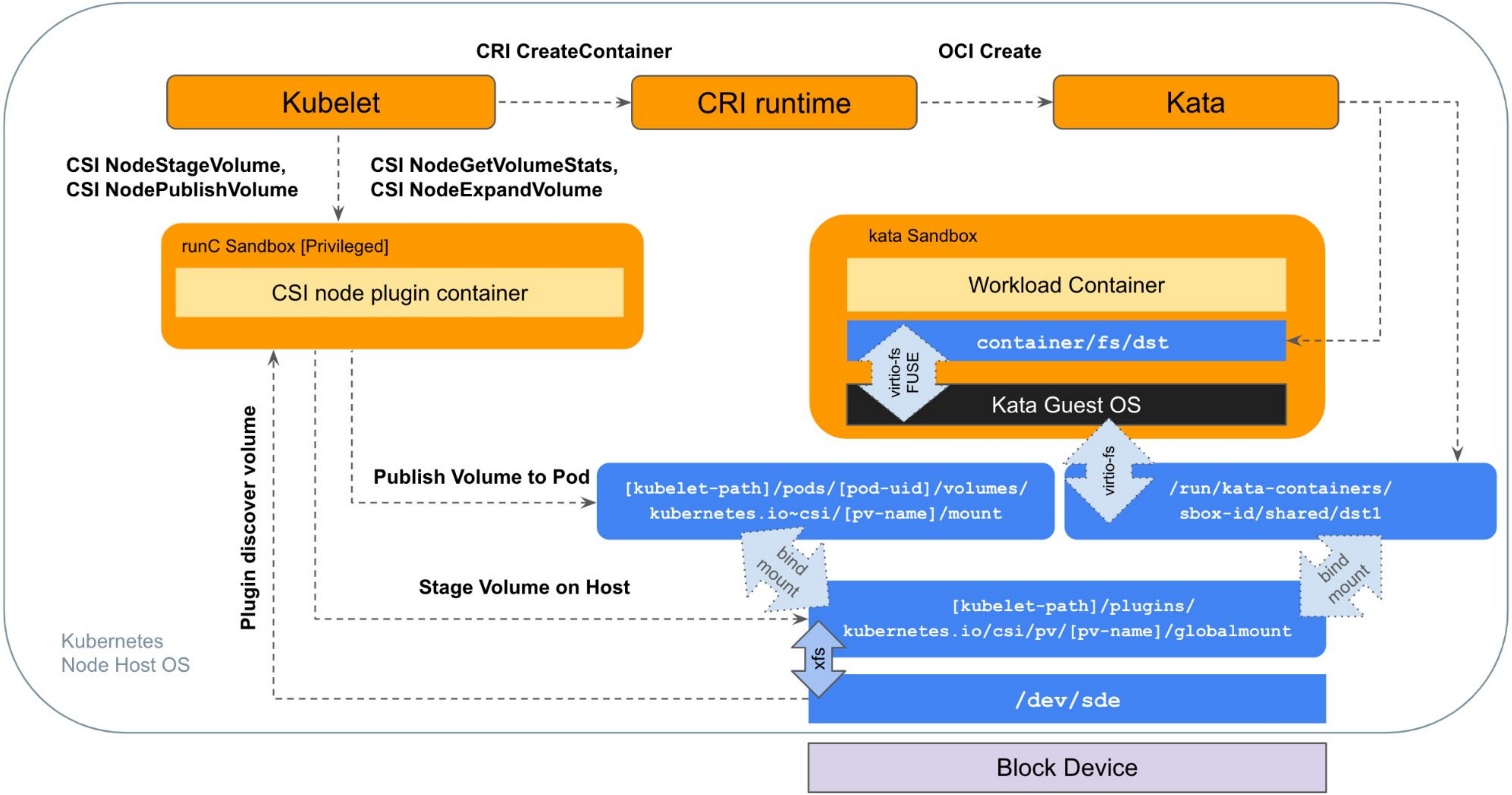


Diagram source: https://katacontainers.io/learn/

Regular Path to Mount a PV in Kata



Performance and Security Trade-offs

- **Performance**

  virtio-fs has worse performance compared to virtio-blk/virtio-scsi

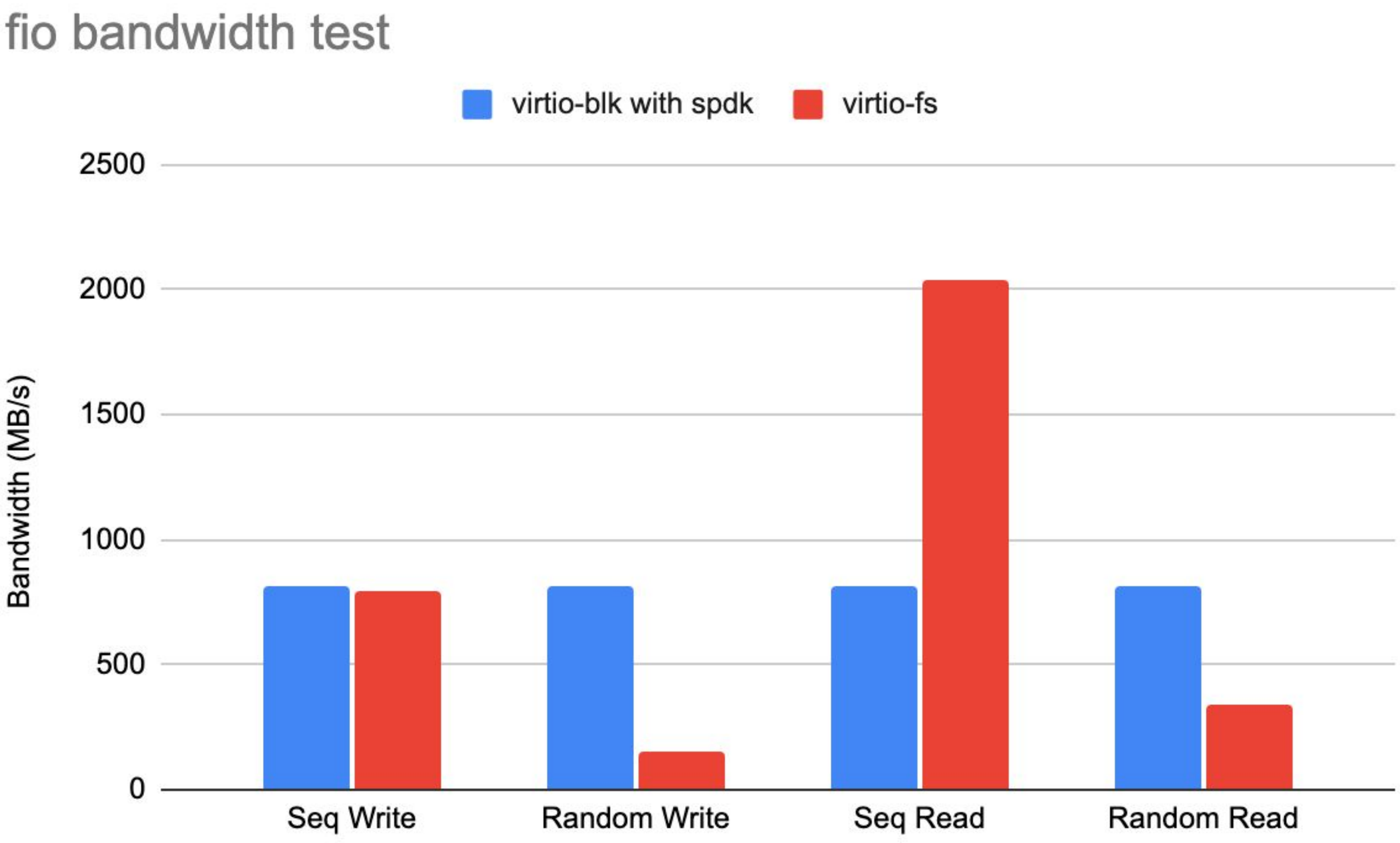
- **Isolation**

  virtio-fs requires file system to be mounted in the host

- **Other Gaps**

  virtio-fs may not surface native file system features

Performance Comparison of virtio fs vs blk



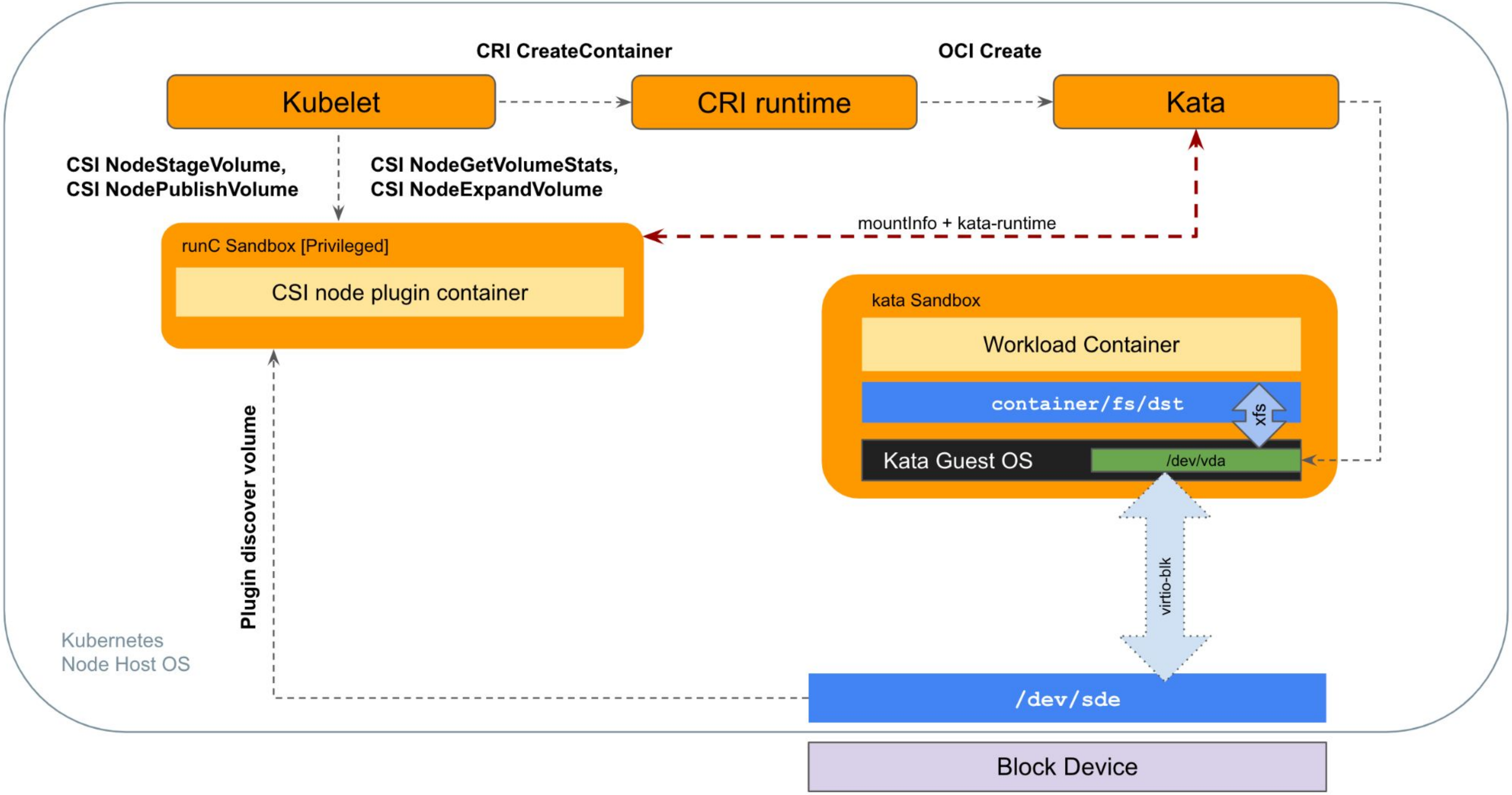
test details: https://gist.github.com/fengwang666/a624c11e26fb7f9035af00fad6cea467

A New Model for Mounting PVs

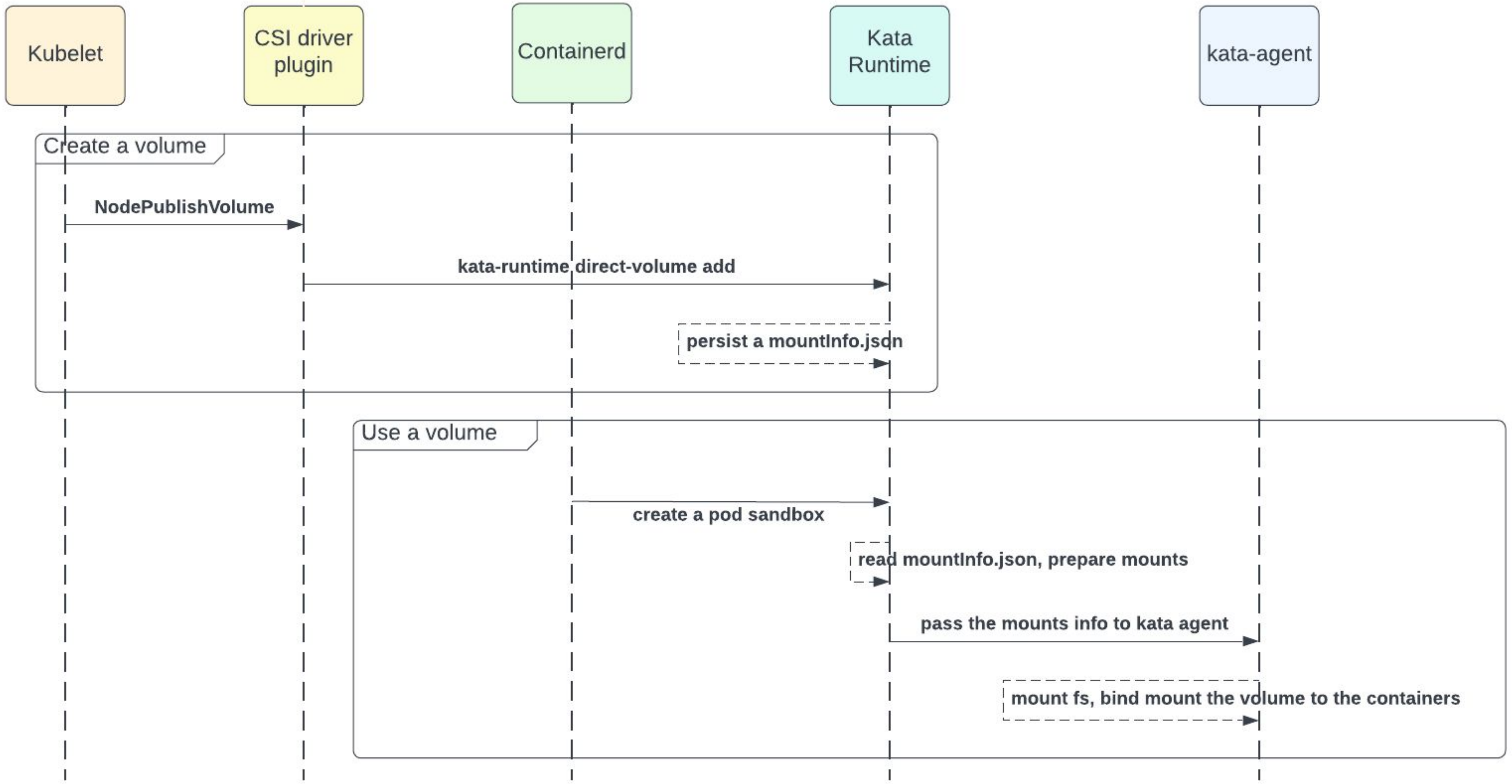
Direct Assigned Storage

CSI plugin delegates PV mount and preparation to container runtime

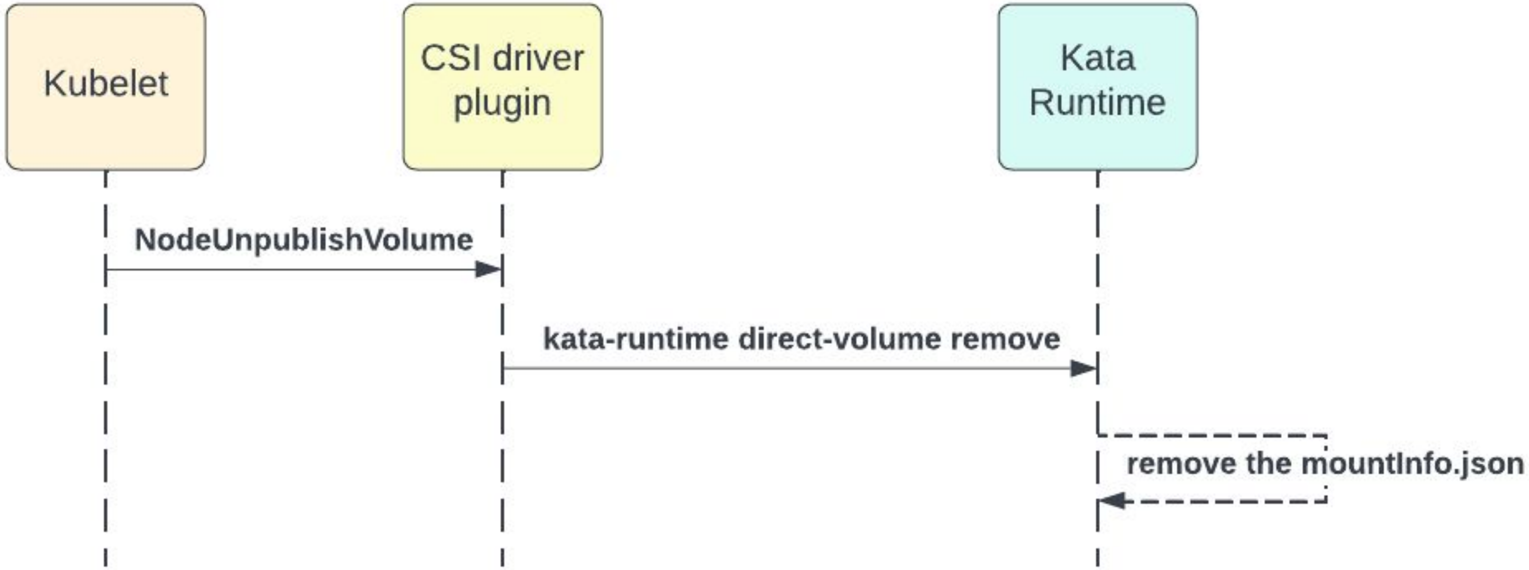
Direct Assign PV to Container Runtime



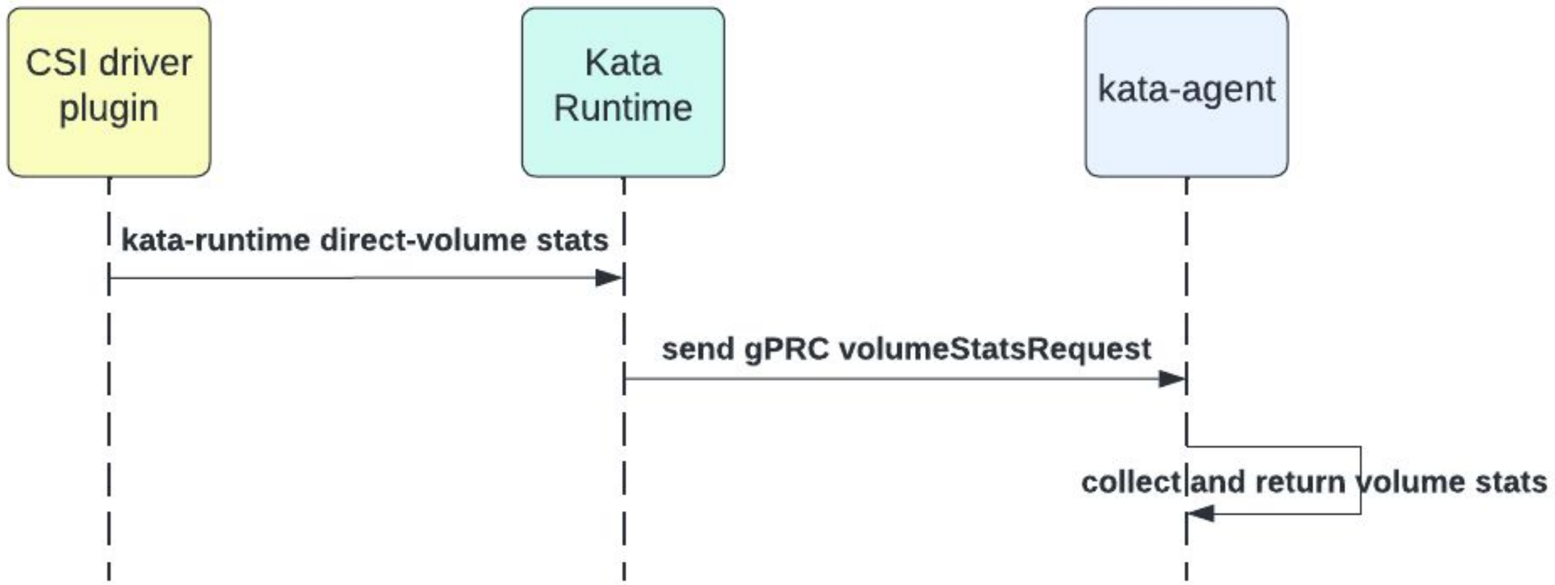
Direct Assigned PV: Mount Volume


Kubelet
CSI driver plugin
Containerd
Kata Runtime
kata-agent
Create a volume
NodePublishVolume
kata-runtime direct-volume add
persist a mountInfo.json
Use a volume
create a pod sandbox
read mountInfo.json, prepare mounts
pass the mounts info to kata agent
mount fs, bind mount the volume to the containers

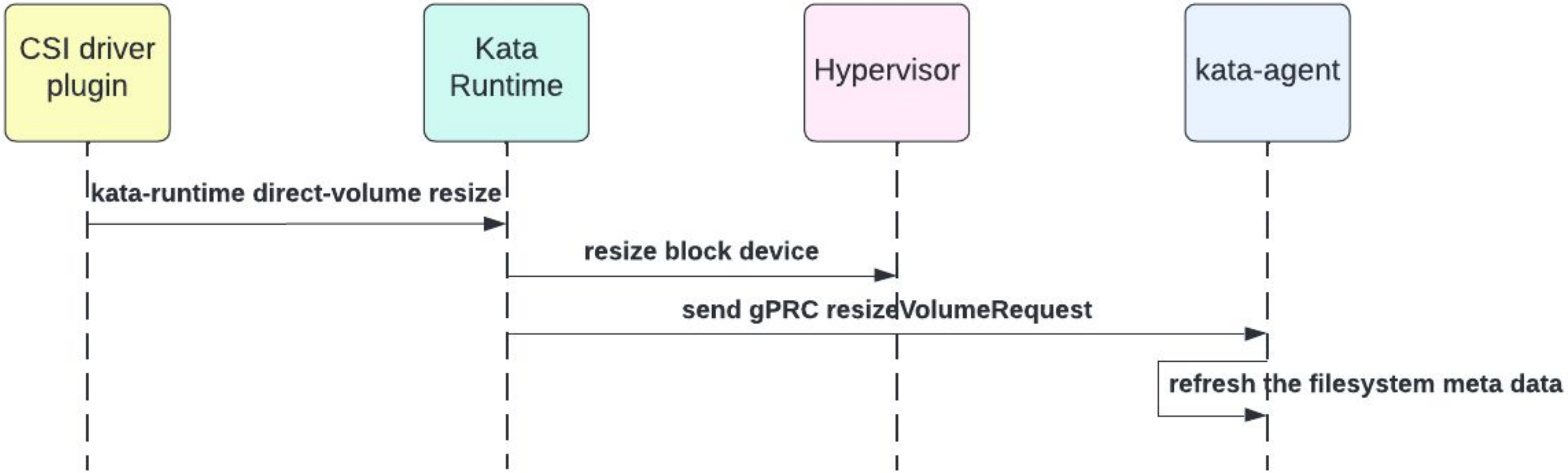
Direct Assigned PV: Unmount Volume



Direct Assigned PV: Volume Stats


CSI driver plugin
Kata Runtime
kata-agent
kata-runtime direct-volume stats
send gPRC volumeStatsRequest
collect and return volume stats

Direct Assigned PV: Resize Volume



CSI driver plugin

Kata Runtime

Hypervisor

kata-agent

kata-runtime direct-volume resize

resize block device

send gPRC resizeVolumeRequest

refresh the filesystem meta data

Implementing Direct Assigned Storage

Container Runtime specific delegation logic in CSI Plugin

Demo

Implementing Direct Assigned Storage

Current: Container Runtime specific delegation logic in CSI Plugin

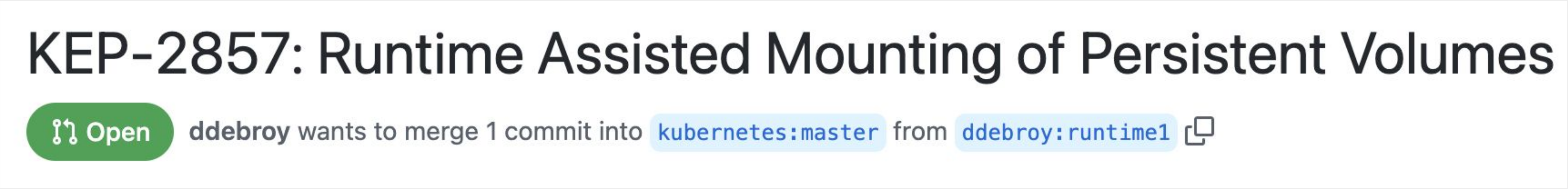
- No changes in Kubelet, CRI, OCI and CSI specs
- Most post-mount configurations work

- CSI Plugin needs to lookup runtimeClass of pod
- Subpath support not possible

Implementing Direct Assigned Storage

Future: Container Runtime agnostic delegation logic in CSI Plugin

- Enhancements in Kubelet, RuntimeClass and CSI spec
- Kubelet will match capabilities of CSI plugin and runtime
- CSI plugins can use a common proxy to delegate operations
- All post-mount configurations will be supported



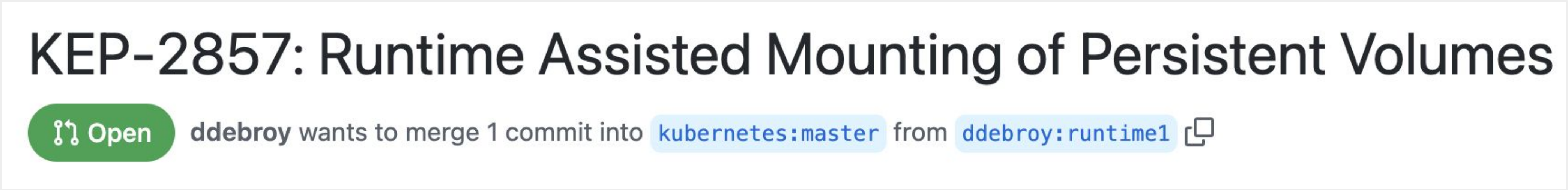
KEP-2857: Runtime Assisted Mounting of Persistent Volumes

Open **ddebroy** wants to merge 1 commit into `kubernetes:master` from `ddebroy:runtime1`

Implementing Direct Assigned Storage

Future: Container Runtime agnostic delegation logic in CSI Plugin

- Scoped to ReadWriteOncePod access modes for safety
- Should not be used for mount scenarios that require a secret



KEP-2857: Runtime Assisted Mounting of Persistent Volumes

ddebroy wants to merge 1 commit into `kubernetes:master` from `ddebroy:runtime1`

Takeaways and Next Steps

- Explored alternatives to the standard mount flow for persistent storage
  - from a microVM perspective

- Ways to delegate mount and post mount configuration to container runtime
  - for block based PVs
  - avoid bind mounts and file-system projections

- Please get involved!
  - KEP-2857 (sig-storage) [#sig-storage in kubernetes.slack.com]
  - Kata community [katacontainers.slack.com]

Questions

Please scan the QR Code above to leave feedback on this session