Tutorial: Introduction to Using the Container Storage Interface (CSI) Primitives

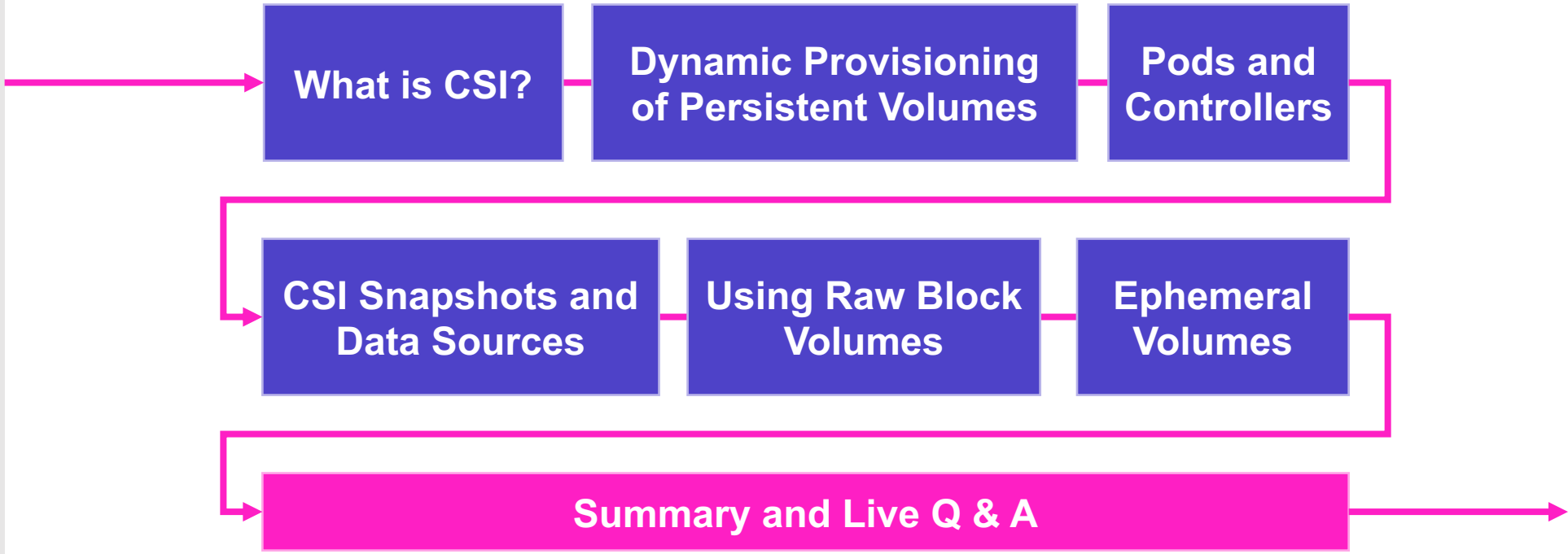
Michael Mattsson, Hewlett Packard Enterprise
@datamattsson

Tutorial: Introduction to Using the Container Storage Interface (CSI) Primitives

Michael Mattsson, Hewlett Packard Enterprise
@datamattsson

Tutorial Overview

CSI stands for Container Storage Interface



Shell

```
$ git clone https://github.com/datamattsson/kcna2020
$ cd kcna2020
```

What is CSI?

Container Storage Interface

SIG stands for Special Interest Group

CONTAINER STORAGE INTERFACE

CSI Specification

- https://github.com/container-storage-interface/spec

Kubernetes SIG Storage

- https://github.com/kubernetes/community/tree/master/sig-storage

CSI Documentation

- https://kubernetes-csi.github.io/

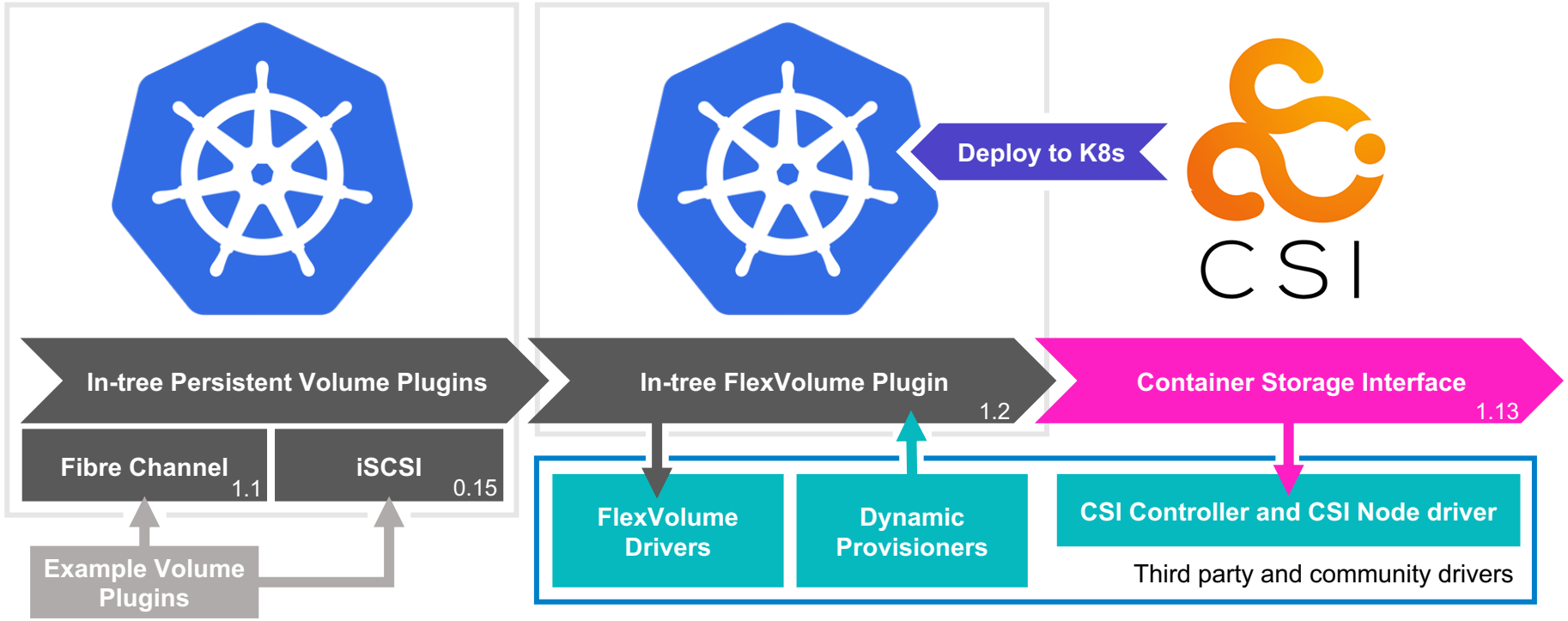
Container Orchestrators (CO)

- Kubernetes
- Nomad
- Cloud Foundry
- Mesos

Container Storage Interface

In-tree storage plugins are being deprecated

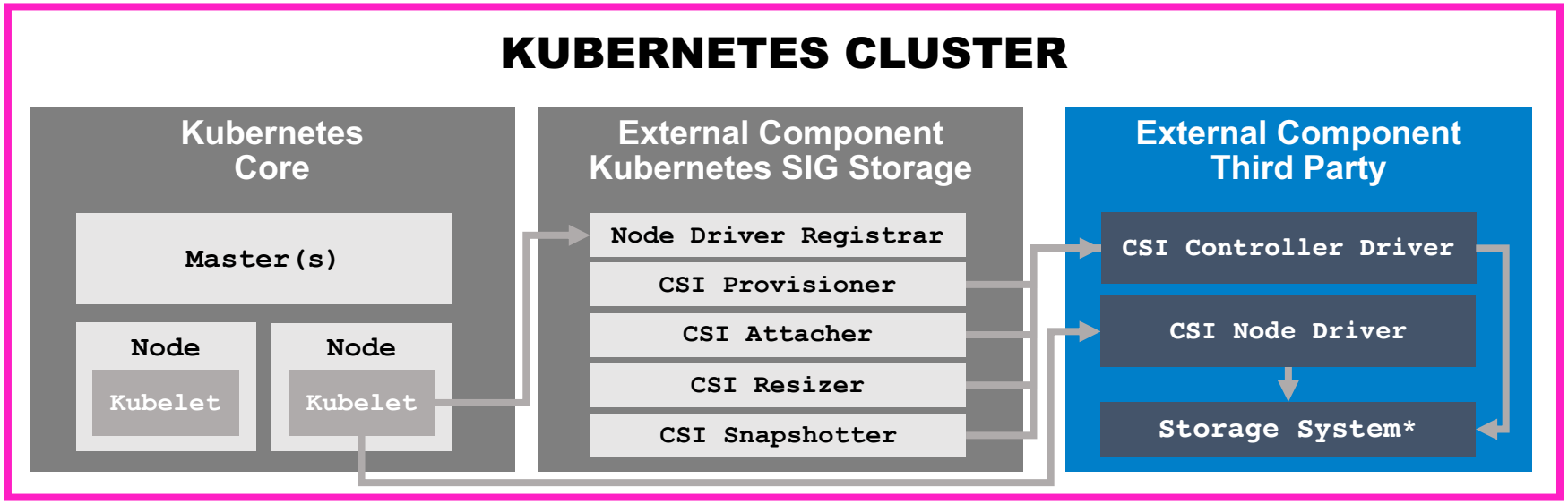
Kubernetes Storage Plugins History



Container Storage Interface

CSI drivers may provide either file or block storage

CSI Architecture (simplified)



* May run on Kubernetes or be completely external

Container Storage Interface

`csi.hpe.com` will be used for the demos

CSI Drivers

diskplugin.csi.alibabacloud.com csi-vxflexos.dellemc.com infinibox-csi-driver quobyte-csi
nasplugin.csi.alibabacloud.com csi-xtremio.dellemc.com csi-instorage robin
ossplugin.csi.alibabacloud.com org.democratic-csi.[X] pmem-csi.intel.com csi-sandstone-plugin
arstor.csi.huayun.io dcx.csi.diamanti.com csi.juicefs.com eds.csi.sangfor.com
ebs.csi.aws.com dobs.csi.digitalocean.com org.kadalu.gluster seaweedfs-csi-driver
efs.csi.aws.com csi.drivescale.com linodebs.csi.linode.com secrets-store.csi.k8s.io
fsx.csi.aws.com [x].ember-csi.io io.drbd.linstor-csi csi-smtx-plugin
disk.csi.azure.com nvmesh-csi.excelero.com driver.longhorn.io csi.spdk.io
file.csi.azure.com pd.csi.storage.gke.io csi-macrosan storageos
csi.block.bigtera.com com.google.csi.filestore manila.csi.openstack.org csi.cio.storidge.com
csi.fs.bigtera.com gcs.csi.ofek.dev com.mapr.csi-kdf csi-driver.storpool.com
cephfs.csi.ceph.com org.gluster.glusterfs com.tuxera.csi.moosefs com.tencent.cloud.csi.cbs
rbd.csi.ceph.com org.gluster.glustervirtblock csi.trident.netapp.io com.tencent.cloud.csi.cfs
csi.chubaofs.com com.hammerspace.csi nexentastor-csi-driver.nexenta.com com.tencent.cloud.csi.cosfs
cinder.csi.openstack.org io.hedvig.csi nexentastor-block-csi-driver.nexenta.com topolvm.cybozu.com
csi.cloudscale.ch csi.hetzner.cloud com.nutanix.csi csi.vastdata.com
csi-infiblock-plugin hspc.csi.hitachi.com cstor.csi.openebs.io csi.block.xsky.com
csi-infifs-plugin csi.hpe.com csi-opensdsplugin csi.fs.xsky.com
dsp.csi.daterainc.io csi.huawei.com com.open-e.joviandss.csi secrets.csi.kubevault.com
csi-isilon.dellemc.com eu.zetanova.csi.hyperv pxd.openstorage.org csi.vsphere.vmware.com
csi-powermax.dellemc.com block.csi.ibm.com pure-csi csi.weka.io
csi-powerstore.dellemc.com spectrumscale.csi.ibm.com disk.csi.qingcloud.com yandex.csi.flant.com
csi-unity.dellemc.com vpc.block.csi.ibm.io csi-neonsan csi.zadara.com

https://kubernetes-csi.github.io/docs/drivers.html

Drivers supports various aspects of the CSI spec

CSI Driver Example Capabilities

	CephFS	Ceph RDB
Provisioner	cephfs.csi.ceph.com	rbd.csi.ceph.com
Specification compliance	v0.3, v1.0.0, v1.1.0, v1.2.0	v0.3, v1.0.0, v1.1.0, v1.2.0
Modes*	Persistent	Persistent
Access mode	Read/Write Multiple Pods	Read/Write Single Pod
Features	Expansion, Snapshot, Clone	Raw Block, Snapshot, Expansion, Topology, Cloning

* = May be Persistent or Ephemeral

New features are introduced through Kubernetes Enhancement Proposals (KEP)

CSI Feature Maturity

Feature	K8s maturity	Since K8s version
Dynamic Provisioning Parameter Overloading	GA	1.13
Topology	GA	1.17
Volume Limits	GA	1.17
Raw Block Volume	GA	1.18
PVC Data Source Cloning	GA	1.18
Volume Expansion	Beta	1.16
Ephemeral Local Volumes	Beta	1.16
Volume Snapshots Volume Snapshot Clone and Restore	Beta	1.17
Generic Ephemeral Volumes	Alpha	1.19

CSI drivers are usually already installed and setup when using managed Kubernetes

Installing and inspecting CSI drivers

Shell

```
$ kubectl get csidrivers
$ kubectl get csinodes
```

HANDS-ON #1

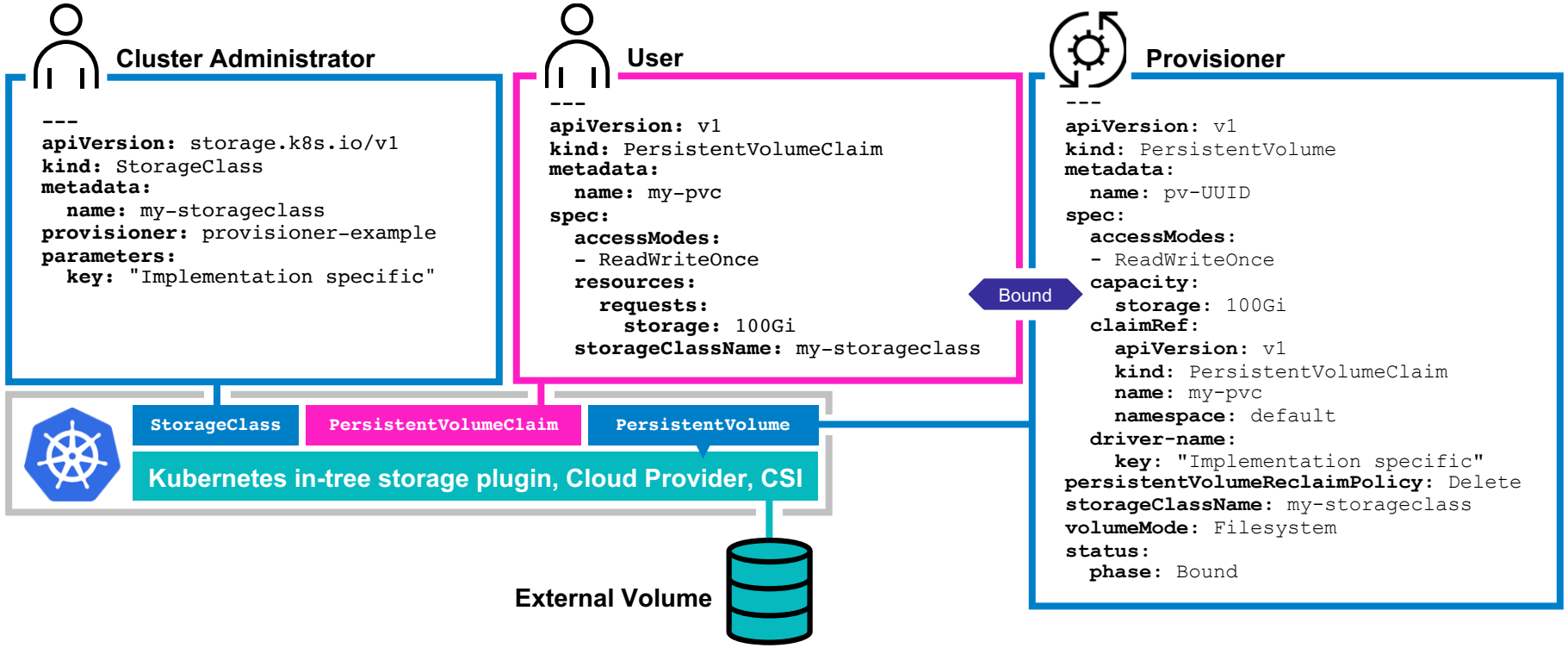
Install a CSI driver.

Dynamic Provisioning of Persistent Volumes

Dynamic Provisioning

Dynamic provisioning is not exclusive to CSI drivers

Dynamic Provisioning in Kubernetes



Dynamic Provisioning

`StorageClasses` are non-namespaced API objects

StorageClass Overview

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "false"
  name: my-storageclass
provisioner: csi.vendor.io
parameters:
  csi.storage.k8s.io/fstype: xfs
  csi.storage.k8s.io/controller-publish-secret-name: csi
  csi.storage.k8s.io/controller-publish-secret-namespace: vendor
  ...
  csi.storage.k8s.io/controller-expand-secret-name: csi
  csi.storage.k8s.io/controller-expand-secret-namespace: vendor
reclaimPolicy: Delete
volumeBindingMode: Immediate
allowVolumeExpansion: "false"
```

● Optional, with their respective defaults
... Each CSI Sidecar needs to reference a secret

Dynamic Provisioning

A `PVC` is confined to a `Namespace`

Persistent Volume Claim Overview

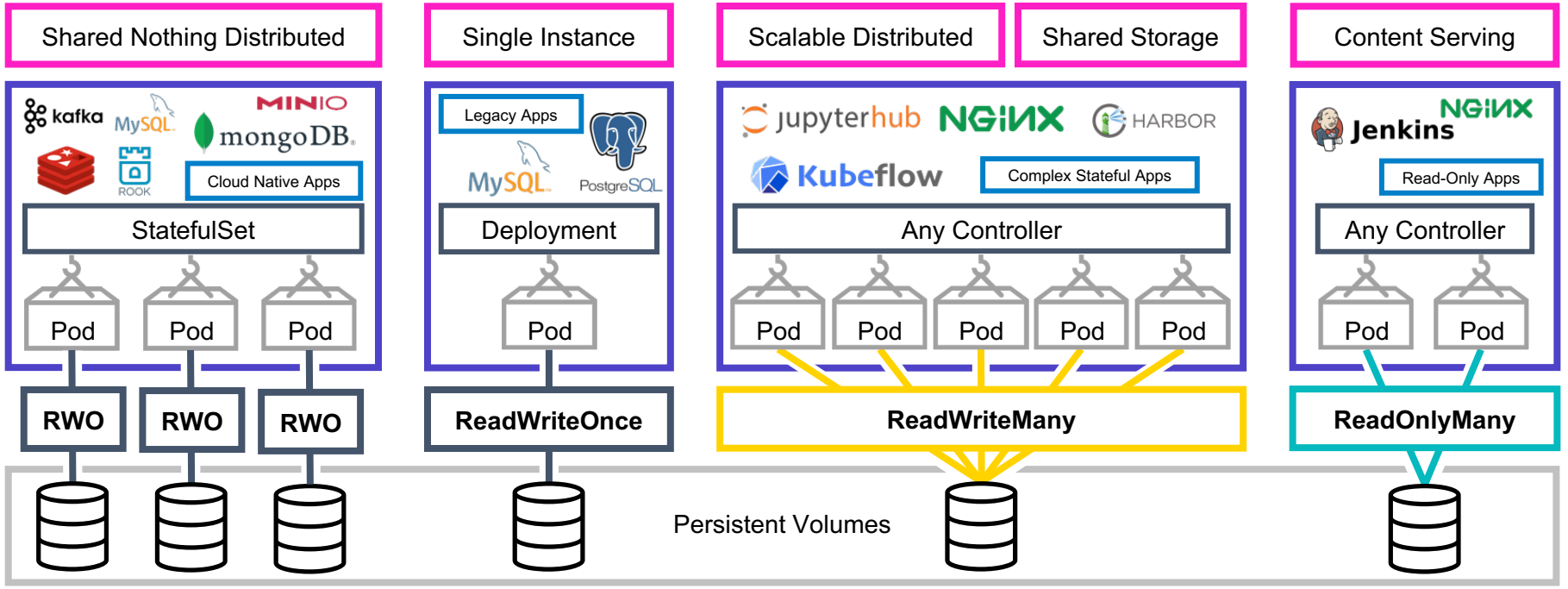
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Ti
  volumeMode: Filesystem
  storageClassName: my-storageclass
```

- Optional, with their respective defaults
- Uses the default **`StorageClass`** if omitted

Dynamic Provisioning

A `PVC` is confined to a `Namespace`

PVC Access Modes



Dynamic Provisioning

The `PV` describes the storage resource in a way the CSI driver understand

Persistent Volume Overview

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  annotations:
    pv.kubernetes.io/provisioned-by: csi.vendor.io
finalizers:
  - kubernetes.io/pv-protection
  - external-attacher/csi-vendor-io
  name: pvc-f03444b1-2aca-4006-a75b-d126d98c0eaf
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 2Ti
  csi:
    controllerPublishSecretRef:
      name: csi
      namespace: vendor
    ...
    controllerExpandSecretRef:
      name: csi
      namespace: vendor
    driver: csi.vendor.io
    fsType: xfs
    volumeAttributes:
      pool: default
      volumeAccessMode: mount
    volumeHandle: 0617faf9c71562b78a0000000000000000005e20
  persistentVolumeReclaimPolicy: Delete
  storageClassName: my-storageclass
  volumeMode: Filesystem
status: {}
```

`...` Each CSI operation needs to reference a secret

Note: `claimRef` omitted

HANDS-ON #2

Create a StorageClass and PVC. Attach a workload and expand volume.

Dynamic Provisioning

All workload controllers except the `StatefulSet` works very similar

Workload Controllers

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: my-mount
  volumes:
    - name: my-mount
      persistentVolumeClaim:
        claimName: my-pvc
```

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: my-web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        volumeMounts:
        - name: my-mount
          mountPath: "/usr/share/nginx/html"
  volumeClaimTemplates:
  - metadata:
      name: my-mount
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: my-storageclass
      resources:
        requests:
          storage: 2Ti
```

HANDS-ON #3

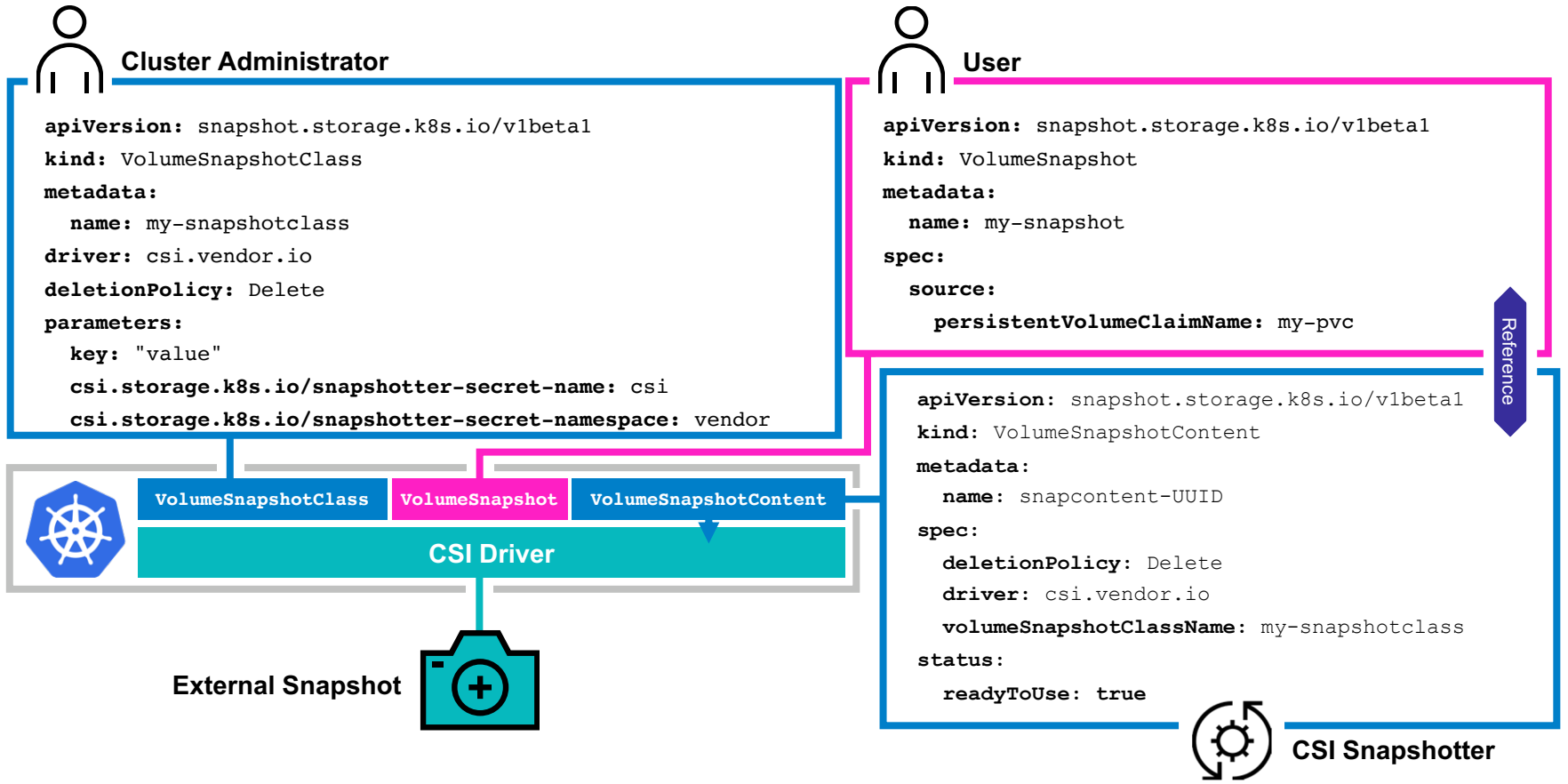
Deploy an application utilizing a StatefulSet.

CSI Snapshots and PVC Data Sources

CSI Snapshot and dataSource

Snapshots uses storage system snapshots abstracted to API objects

CSI Snapshots



HANDS-ON #4

Deploy CSI snapshotter, create a VolumeSnapshotClass and VolumeSnapshots.

CSI Snapshot and dataSource

New `PVCs` may be created from `VolumeSnapshots`

PVC from VolumeSnapshot

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-new-pvc
spec:
  dataSource:
    name: my-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Ti
```

HANDS-ON #5

Create a new PVC from a VolumeSnapshot and attach an application.

CSI Snapshot and dataSource

A PVC may be created from an existing PVC

PVC Cloning

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-clone-pvc
spec:
  dataSource:
    name: my-pvc
    kind: PersistentVolumeClaim
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Ti
```

HANDS-ON #6

Create a new PVC from an existing PVC and attach an application.

CSI Snapshot and dataSource

Restoring an application from a `VolumeSnapshot` is a cloning operation

Restore an application

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  dataSource:
    name: my-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Ti
```

HANDS-ON #7

Restore an application from a VolumeSnapshot.

Using Raw Block Volumes

Using Raw Block Volumes

The default `volumeMode` is filesystem

Raw Block Volume PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-block-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Ti
  volumeMode: Block
  storageClassName: my-storageclass
```

- Uses the default **`StorageClass`** if omitted

Using Raw Block Volumes

Devices are referenced and enumerated slightly different from filesystems

Raw Block Volume Pod specification

```
apiVersion: v1
kind: Pod
metadata:
  name: ioping
spec:
  containers:
  - name: ioping
    image: datamattsson/ioping:edge
    args: [ "/dev/xvda" ]
    volumeDevices:
    - name: volume
      devicePath: /dev/xvda
  volumes:
  - name: volume
    persistentVolumeClaim:
      claimName: my-block-pvc
```

HANDS-ON #8

Create a raw block device and attach a workload.

Using Raw Block Volumes

Rook is a CNCF incubating project

Use Case for Raw Block Volumes

Rook

- Open-Source, Cloud-Native Storage for Kubernetes
- Provides File, Block and Object to Kubernetes
- Uses CEPH
- Manage distributed storage on Kubernetes effortless

Deployment pattern

- Installs as an Operator
- Create `CephCluster` CRD
- May leverage `volumeMode: Block` PVC

HANDS-ON #9

Deploy Rook.

Ephemeral Volumes

Ephemeral Volumes

Ephemeral Volumes exist only for the duration of the **`Pod`** and inherently **`ReadWriteOnce`** per **`Pod`**

Ephemeral Local Volumes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: my-mount
  volumes:
  - name: my-mount
    csi:
      driver: csi.vendor.io
      volumeAttributes:
        csi.storage.k8s.io/ephemeral: "true"
        inline-volume-secret-name: csi
        inline-volume-secret-namespace: vendor
        size: 64Gi
```

Ephemeral Volumes

Ephemeral Volumes exist only for the duration of the **Pod**

Ephemeral Local Volumes (alternative secretRef)

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: my-mount
  volumes:
  - name: my-mount
    csi:
      driver: csi.vendor.io
      volumeAttributes:
        csi.storage.k8s.io/ephemeral: "true"
        nodePublishSecretRef:
          name: csi
        size: 64Gi
```

HANDS-ON #10

Using Ephemeral Local Volumes.

Ephemeral Volumes

Generic Ephemeral Volumes is an Alpha feature introduced in 1.19

Generic Ephemeral Volumes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/usr/share/nginx/html"
        name: my-mount
  volumes:
  - name: my-mount
    ephemeral:
      volumeClaimTemplate:
        spec:
          accessModes: [ "ReadWriteOnce" ]
          resources:
            requests:
              storage: 64Gi
          storageClassName: my-storageclass
```

- Uses the default **`StorageClass`** if omitted

HANDS-ON #11

Using Generic Ephemeral Volumes.

Summary

KubeCon CloudNativeCon North America 2020 Virtual

✅ We are done!

Thank you for participating!

CONTAINER STORAGE INTERFACE

What we covered

- Introduction to CSI drivers
- Dynamic provisioning in Kubernetes
- CSI snapshots, restore and PVC cloning
- Accessing raw block storage
- Ephemeral volumes

Source files (YAML, .pptx and asciinema cast files)

- https://github.com/datamattsson/kcna2020

CSI Specification

- https://github.com/container-storage-interface/spec

Kubernetes SIG Storage

- https://github.com/kubernetes/community/tree/master/sig-storage

CSI Documentation

- https://kubernetes-csi.github.io/


KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual