# Everything you always wanted to know about storage in Kubernetes

(But were too afraid to ask)

Johannes Wagner, Hendrik Land

2022-11-22

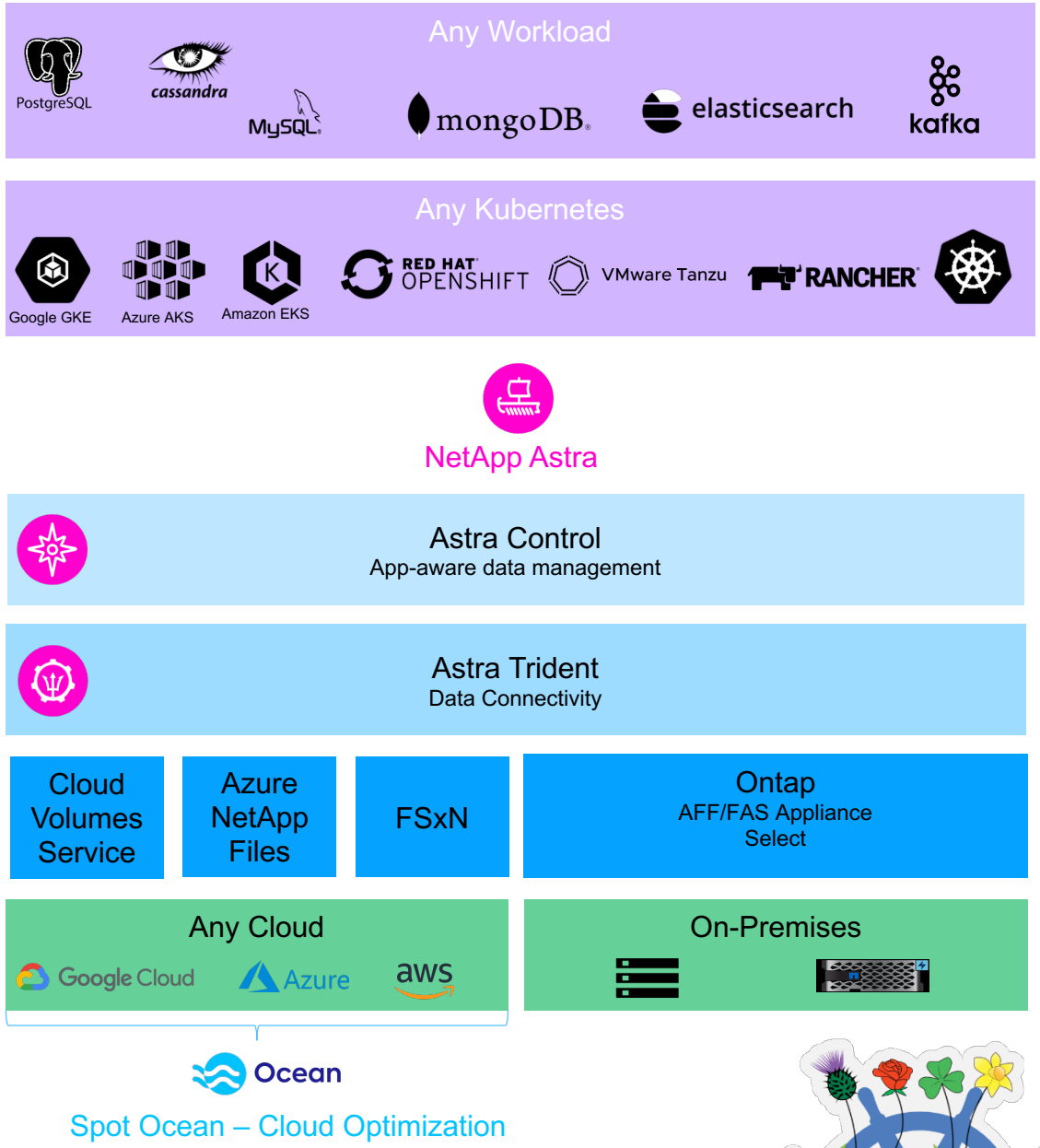


NetApp

# NetApp – Who are we?

- **Astra – Storage & Data Management for K8s**
  - On-Premises & Any-Cloud
  - Cloud-Native 1st party storage services
  - Any K8s

- **Spot – Continuous Cloud Optimization**
  - Ensure availability and optimize cost
  - Container-driven autoscaling
  - Container right-sizing



# Storage in Kubernetes - Volume Types

Volume: Directory accessible to the containers in a pod

**Resources exposed as a volume**

- ConfigMap
- Secret
- …

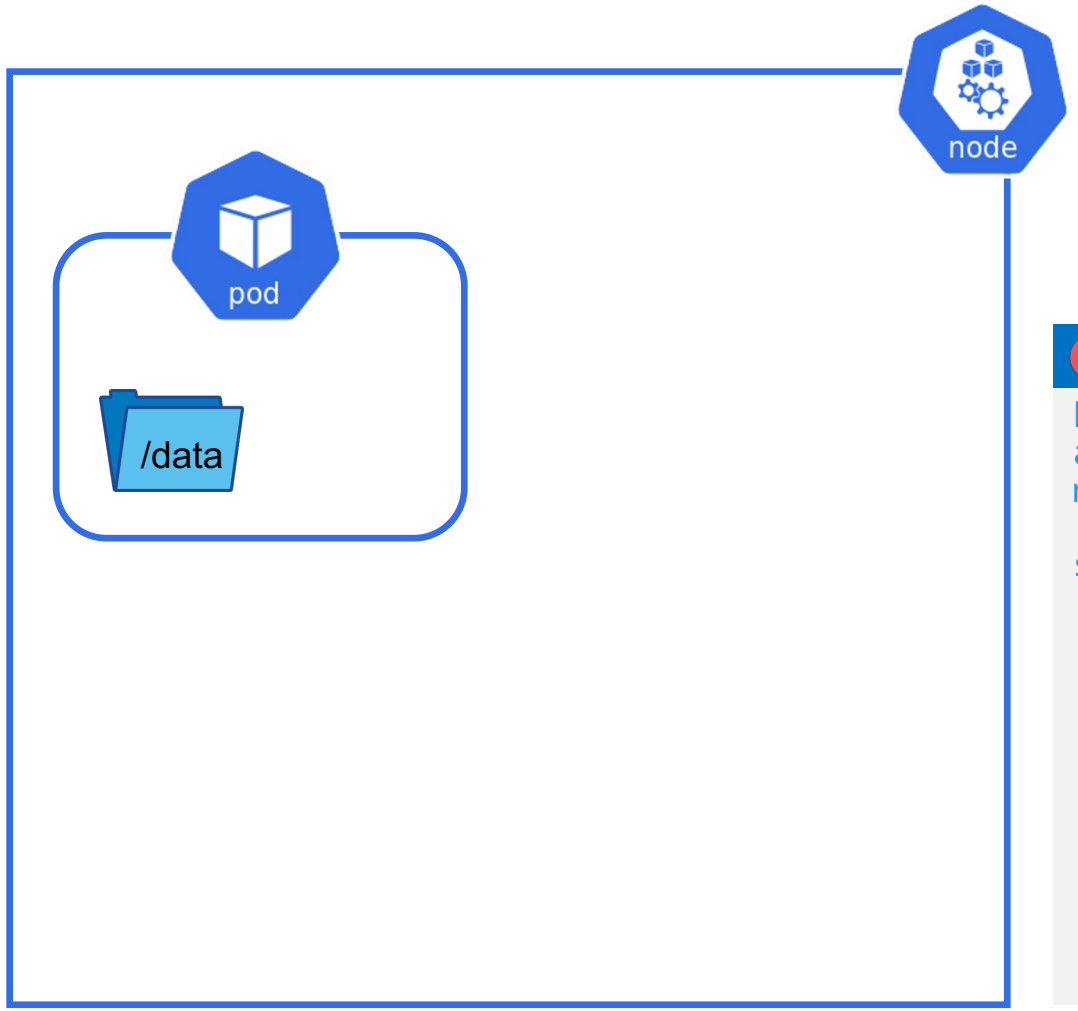
**Storage – Place to store data**

- In-tree & flexVolume storage drivers – deprecated!
- emptyDir
- Local
- PersistentVolumeClaim & CSI Drivers

# CSI – Container Storage Interface

- Open Standard for attaching storage to containers based on file and block storage
- Provides dynamic provisoning of storage resources
- Provides storage as a file system to the container
- CSI alpha in K8s 1.9, beta in 1.11, GA in 1.13
- CNCF standard (though mostly used with K8s)

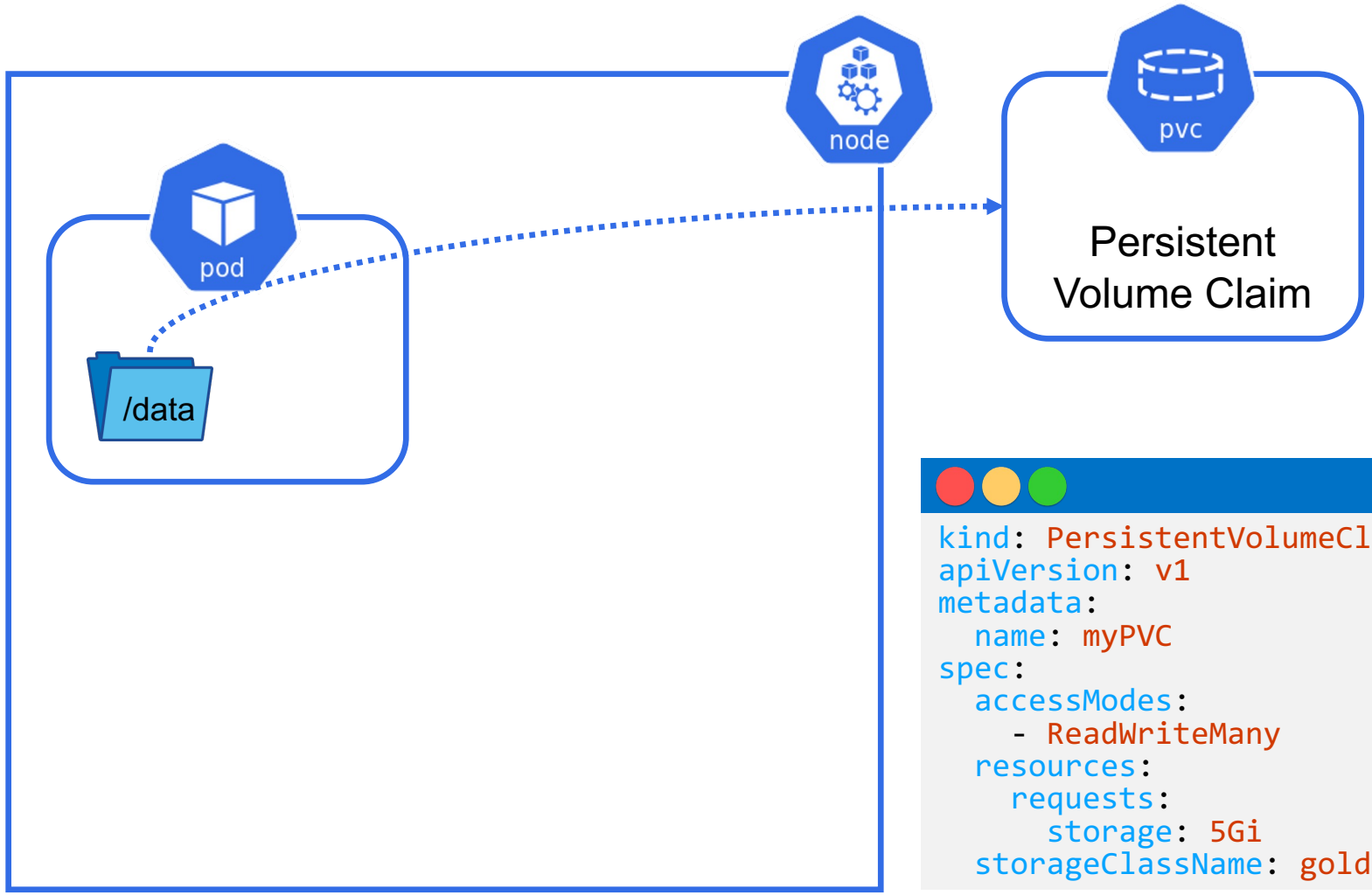
# Dynamic storage provisioning in K8s




pod.yaml

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: myLittlePod
spec:
  volumes:
    - name: myVol
      persistentVolumeClaim:
       claimName: myPVC
  containers:
    - name: ubuntu-test
      image: ubuntu
      command: ["/<do something>"]
      volumeMounts:
        - mountPath: "/data"
          name: myVol
```

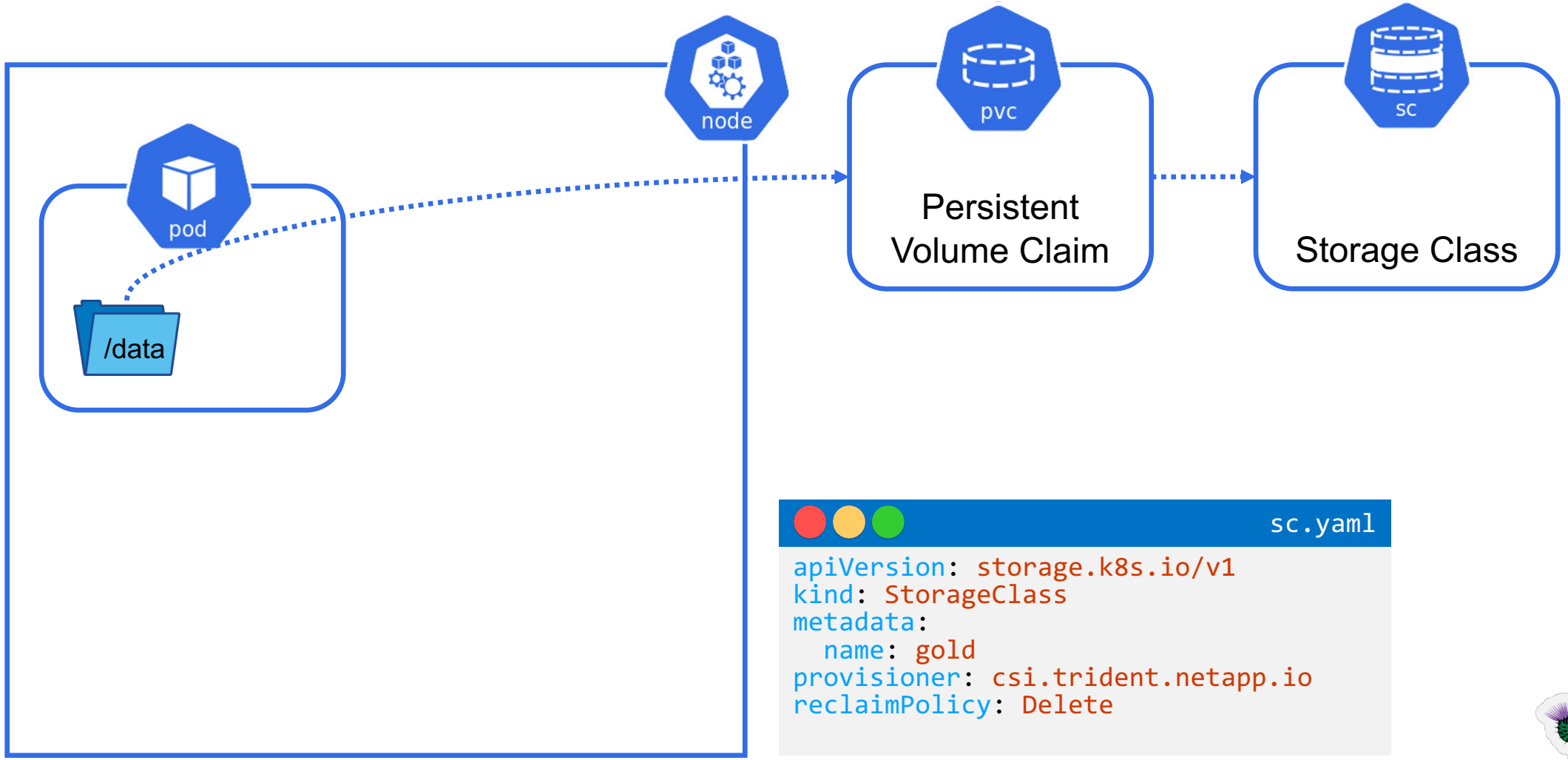
# Dynamic storage provisioning in K8s




pvc.yaml

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myPVC
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 5Gi
  storageClassName: gold
```

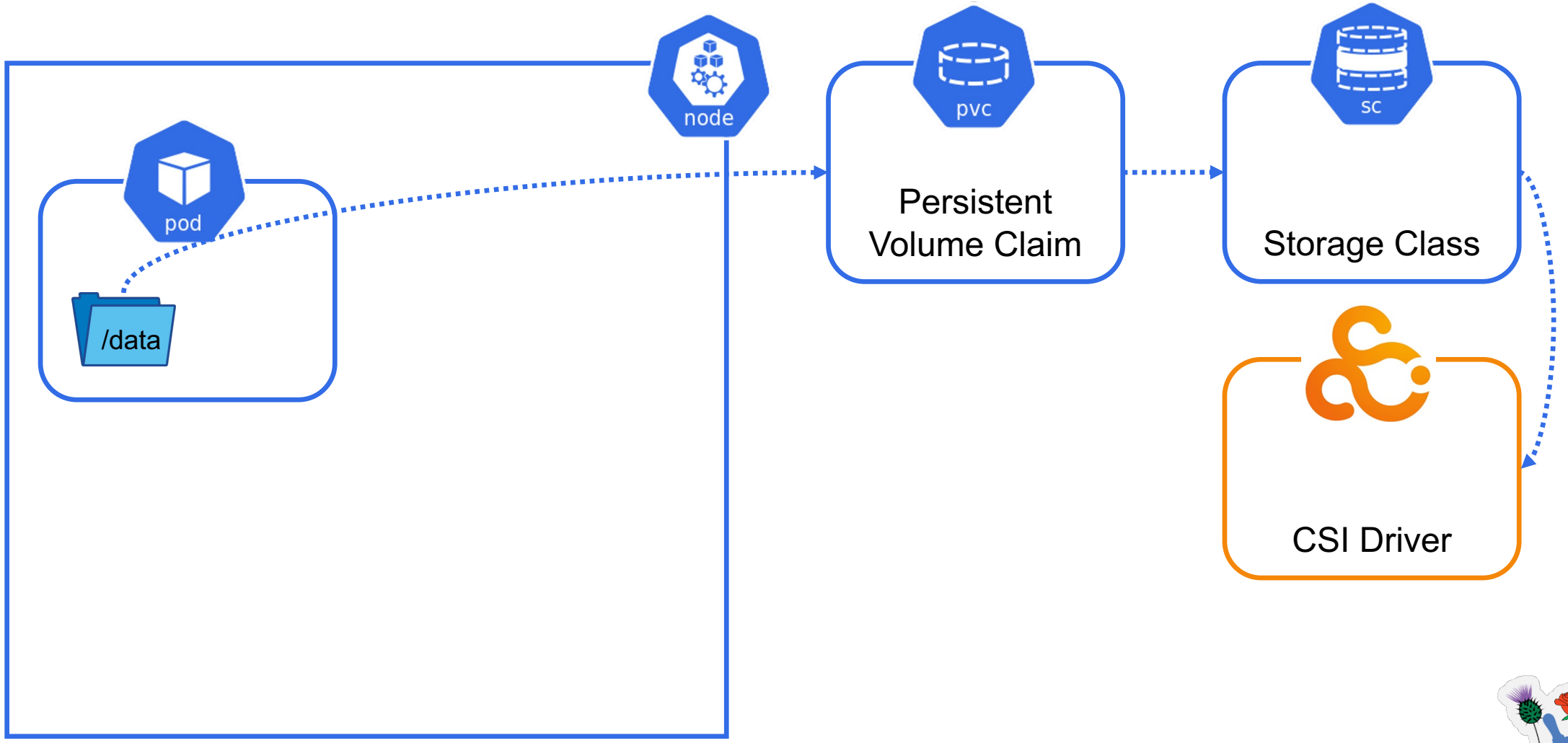
# Dynamic storage provisioning in K8s



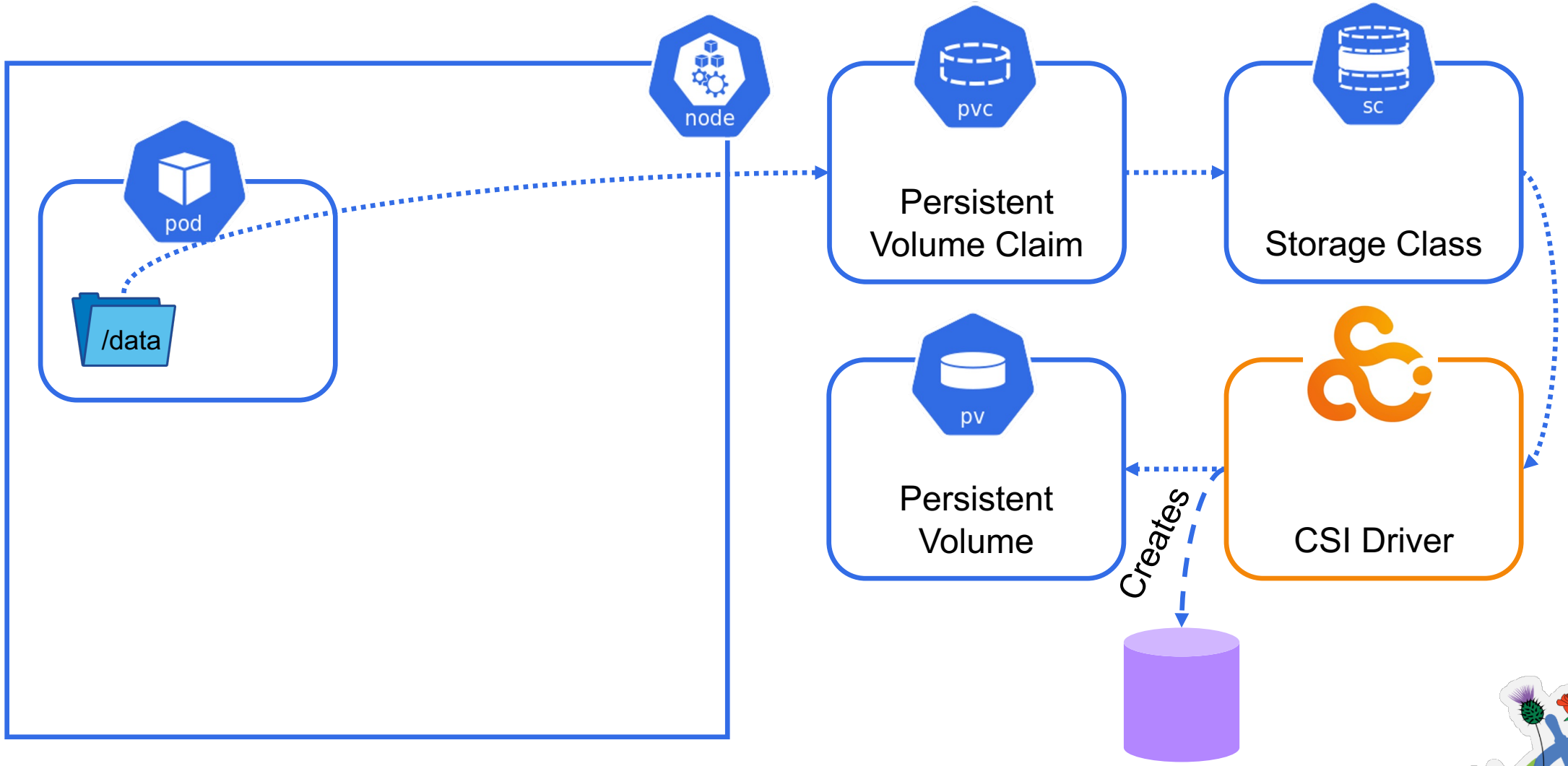

sc.yaml

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
```

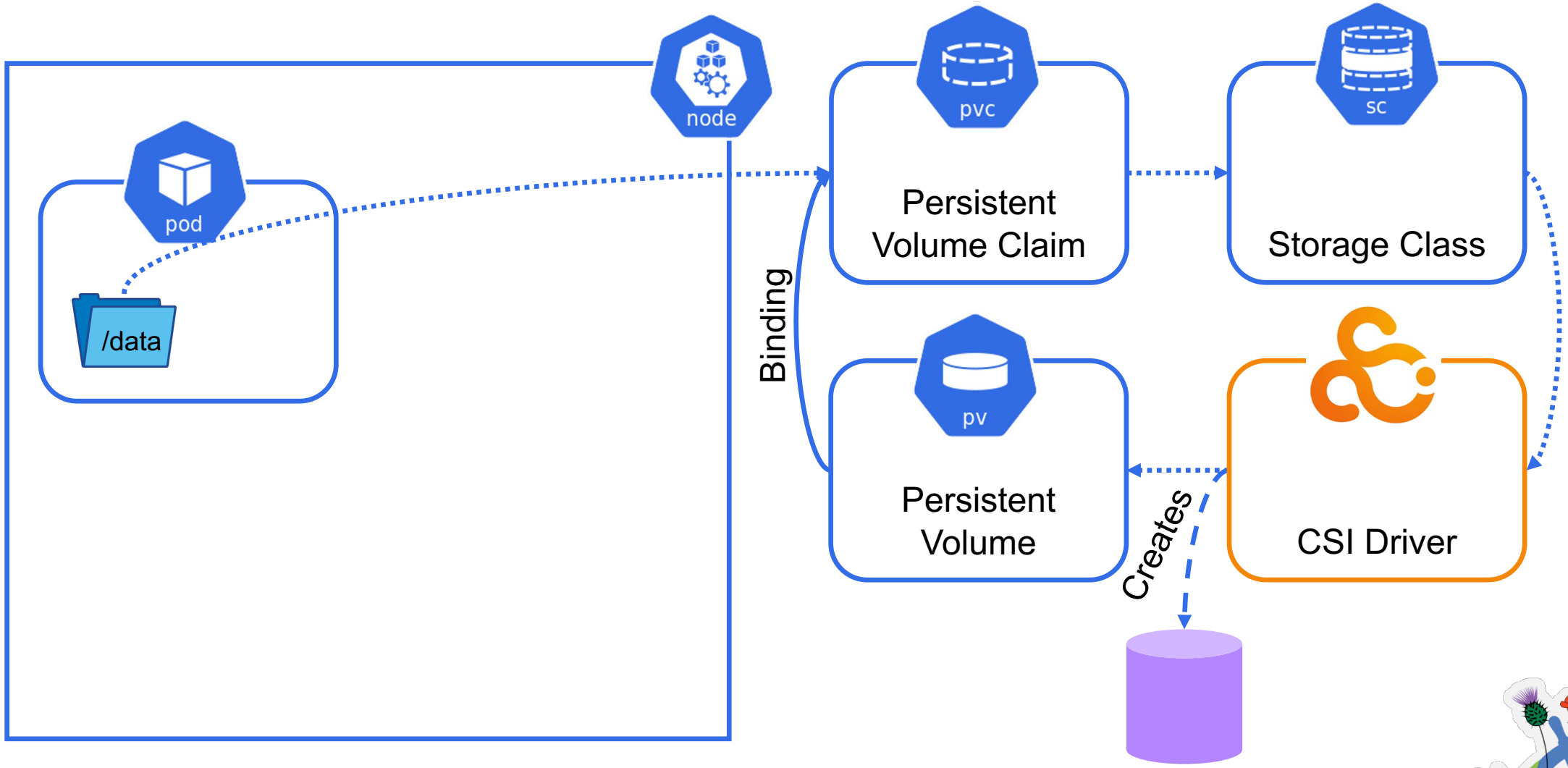
# Dynamic storage provisioning in K8s



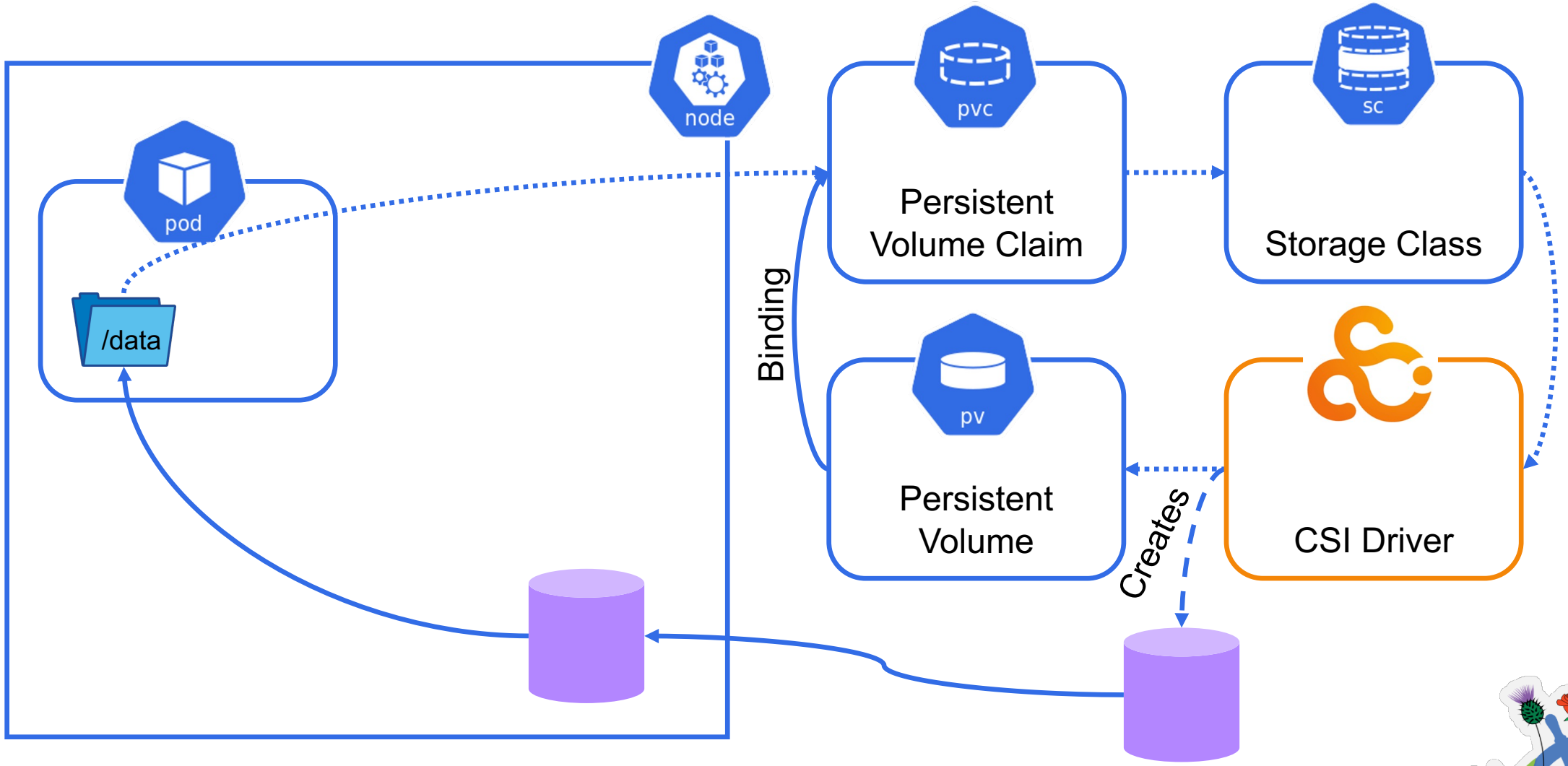
# Dynamic storage provisioning in K8s



# Dynamic storage provisioning in K8s



# Dynamic storage provisioning in K8s



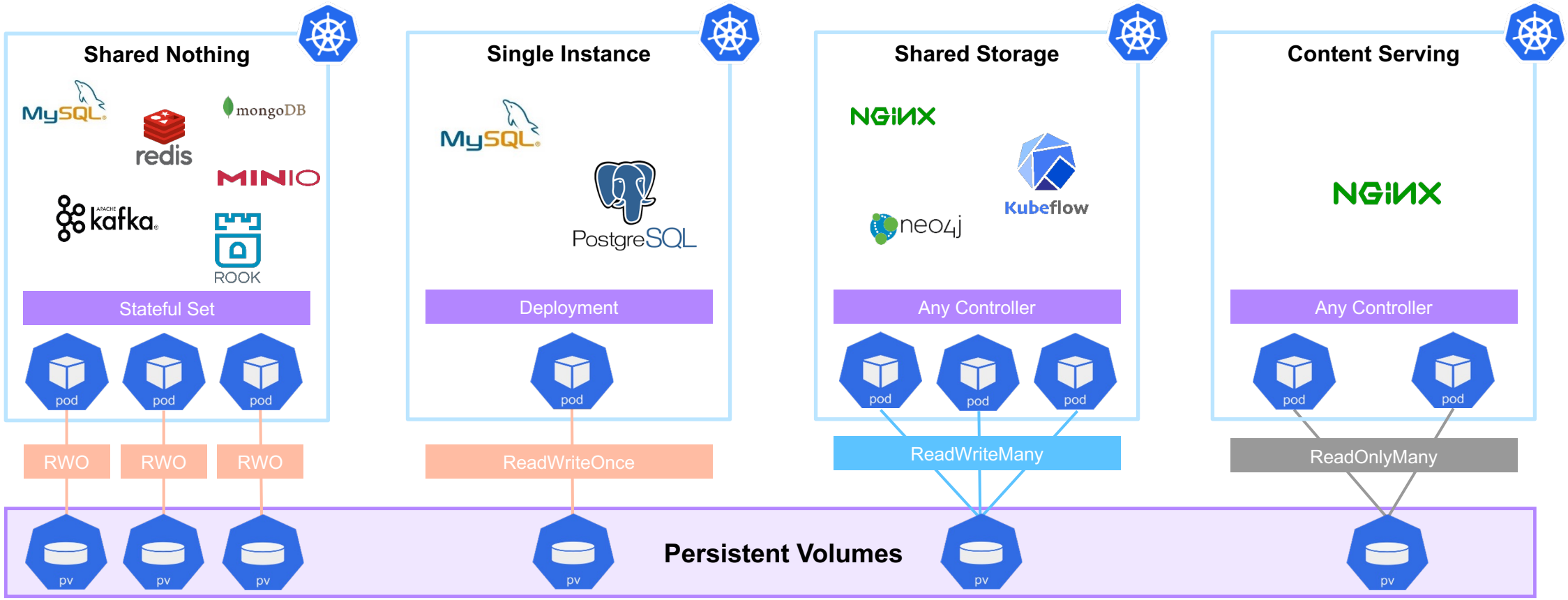
# Persistent Volume Claim Access Modes

Different applications have different needs

- ReadWriteOnce (RWO)
  - Can be mounted read-write by a single node
- ReadWriteMany (RWX)
  - Can be mounted read-write by several nodes
- ReadOnlyMany
  - Read-Only access by several nodes
- ReadWriteOncePod
  - Read-Write access for a single pod only
  - New in K8s 1.22

# Persistent Volume Claim Access Modes

Different applications have different needs






Kubernetes Community Days UK 2022

# Storage Protocol Choices

## File

- Shared filesystem (RWX)
- Good fit for Pod lifecycle
- Open Standard: NFS

## Block

- Required by some applications (Prometheus, Kafka,…)
- Open Standards: iSCSI, NVMe

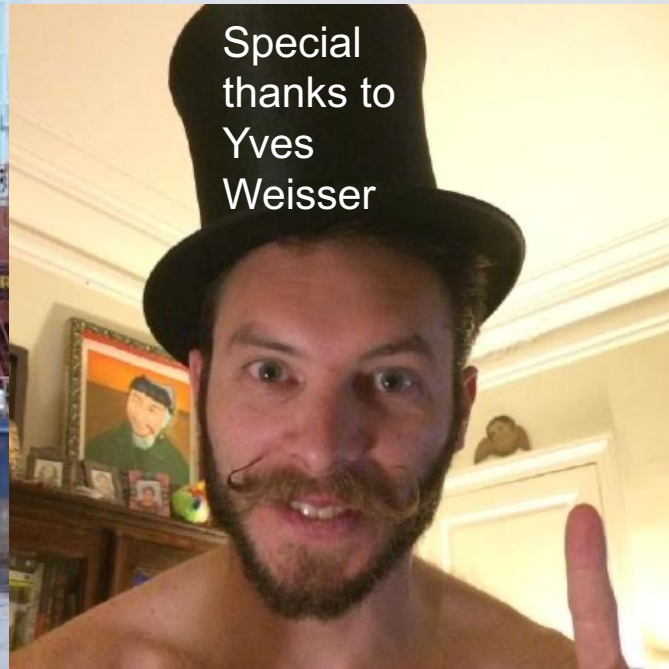
Container Storage Interface (CSI)

## Object

- Data Service, not a file system inside the container
- “Standard“: S3

Container Object Storage Interface (COSI) – Alpha in K8s 1.25

# Hands-on Part 1



# Before we start

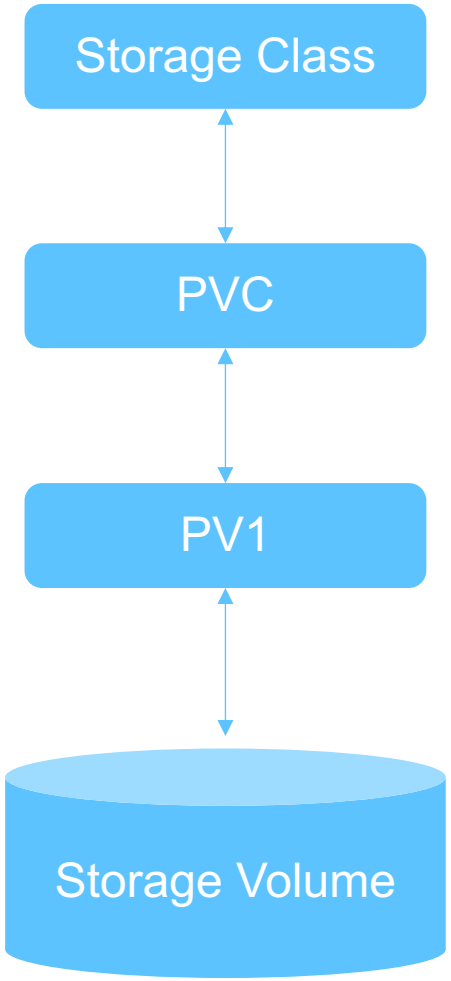
- Please raise your hand or ask us if you have questions or problems
- Choose your username from the spreadsheet, enter your name to avoid double usage
- Pre-setup is important
- Scenario 01 contains basic setup steps and is a must, following scenarios depend on it
- Before Scenario 03 we want to do a short theory session again

# Hands-on
# Part 1

https://github.com/kcdstoragews/lab

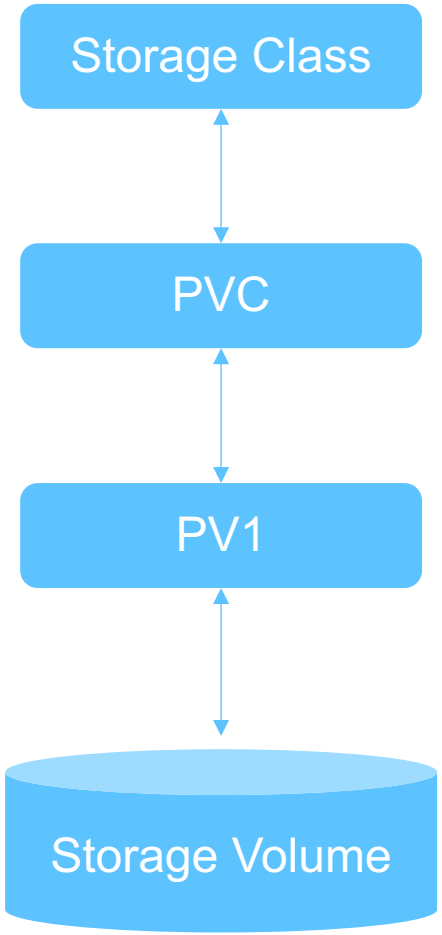
# CSI Snapshots workflow

Initial configuration



# CSI Snapshots workflow

## Volume Snapshot Class creation




Volume Snapshot Class

volumesnapshotclass.yaml

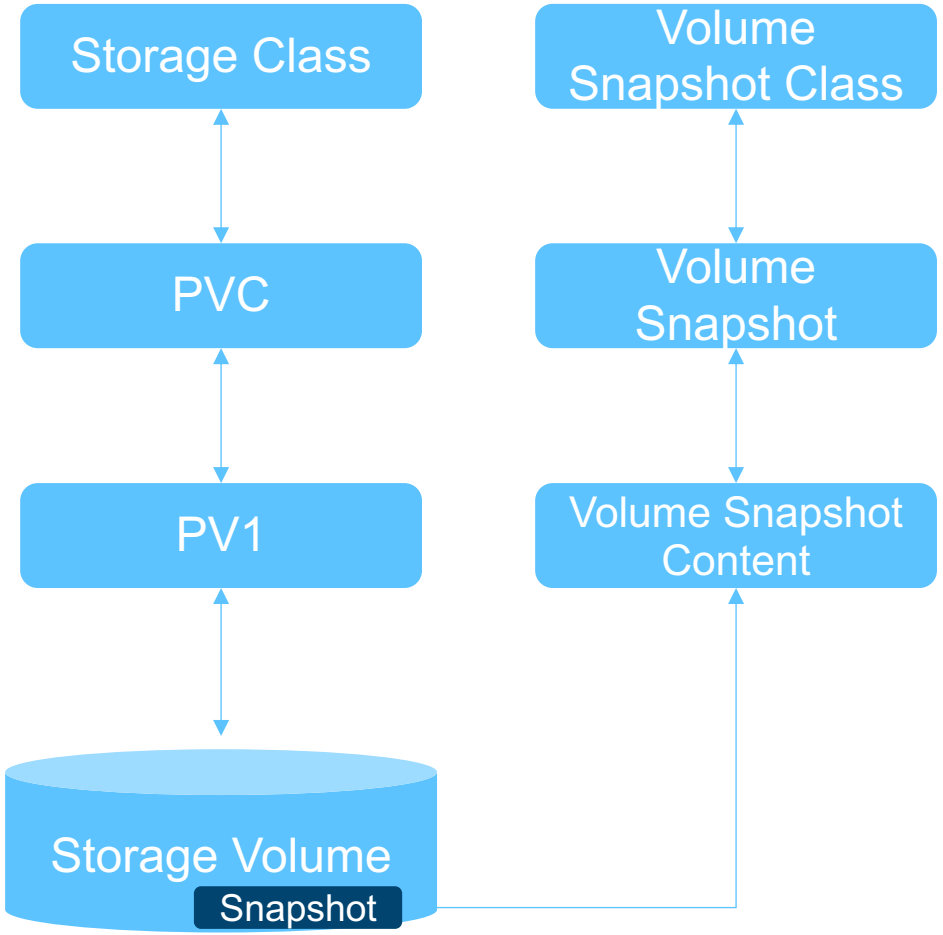
```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

**NOTE**

the *deletionPolicy* parameter can be set to:
- Delete
- Retain

# CSI Snapshots workflow

Volume Snapshot creation

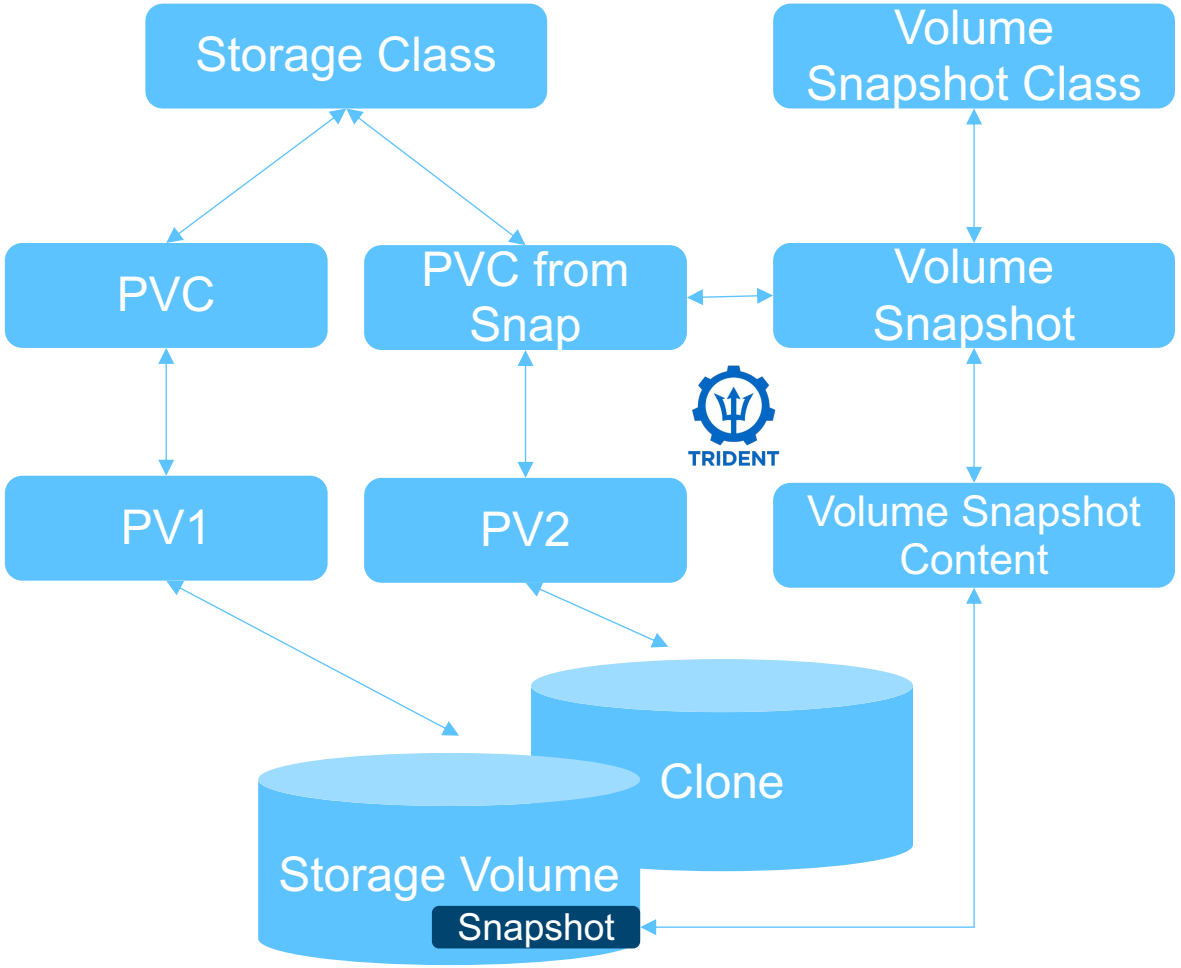



volume-snapshot.yaml

```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshot
metadata:
  name: volume-snapshot
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc
```

# CSI Snapshots workflow

## Clone creation




pvc-from-snapshot.yaml

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: sc-nas
  dataSource:
    name: volume-snapshot
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

# Consumption Control

- Standard K8s mechanisms such as ResourceQuota and LimitRange apply to storage as well
- Restrict capacity and number of PVCs per namespace with a ResourceQuota
  - Total capacity
  - Capacity per StorageClass
  - Total number of PVCs
  - Number of PVCs per StorageClass
- Remember: StorageClass is a global resource in the cluster
  - But you can assign a 0 byte quota
- Define Minimum and Maximum size of an individual PVC with LimitRange

# Generic Ephemeral Volumes

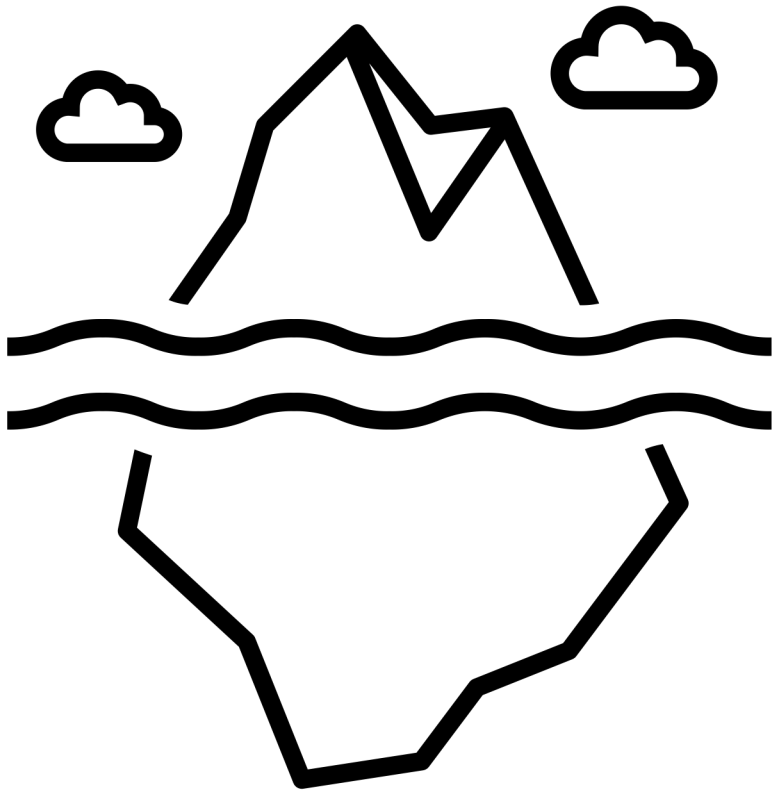
- K8s provides different types of ephemeral volumes
  - emptyDir
  - CSI ephemeral volumes
  - **generic ephemeral volumes**
- Generic ephemeral volumes provide additional features (beta in K8s 1.21, GA in 1.23)
  - External storage to not exceed disk capacity of local node
  - Snapshot & Cloning Capability
  - Resizing
  - StorageClass
- Set StorageClass binding mode to WaitForFirstConsumer so K8s scheduler is free to choose any node

# Hands-on
# Part 2

https://github.com/kcdstoragews/lab

**EVERYTHING you always wanted to know about storage in Kubernetes?**

OK, there's more…



- CSI Topology
- (Capacity) monitoring
- Non-graceful shutdown
- Security