NetApp Kompakt LiveLab

Datenmanagement für Kubernetes

Johannes Wagner & Hendrik Land
Solution Architect EMEA
06.04.2023

Agenda

- Vorstellungsrunde
- Theorie Storage in Kubernetes
- Hands-on Storage in Kubernetes
- Theorie Snapshots/Clone & Kontrolle
- Hands-on Snapshots/Clone & Kontrolle
- Theorie Astra Control
- Hands-on Astra Control

Storage in Kubernetes - Volume Types

Volume: Directory accessible to the containers in a pod

Resources exposed as a volume

- ConfigMap
- Secret
- …

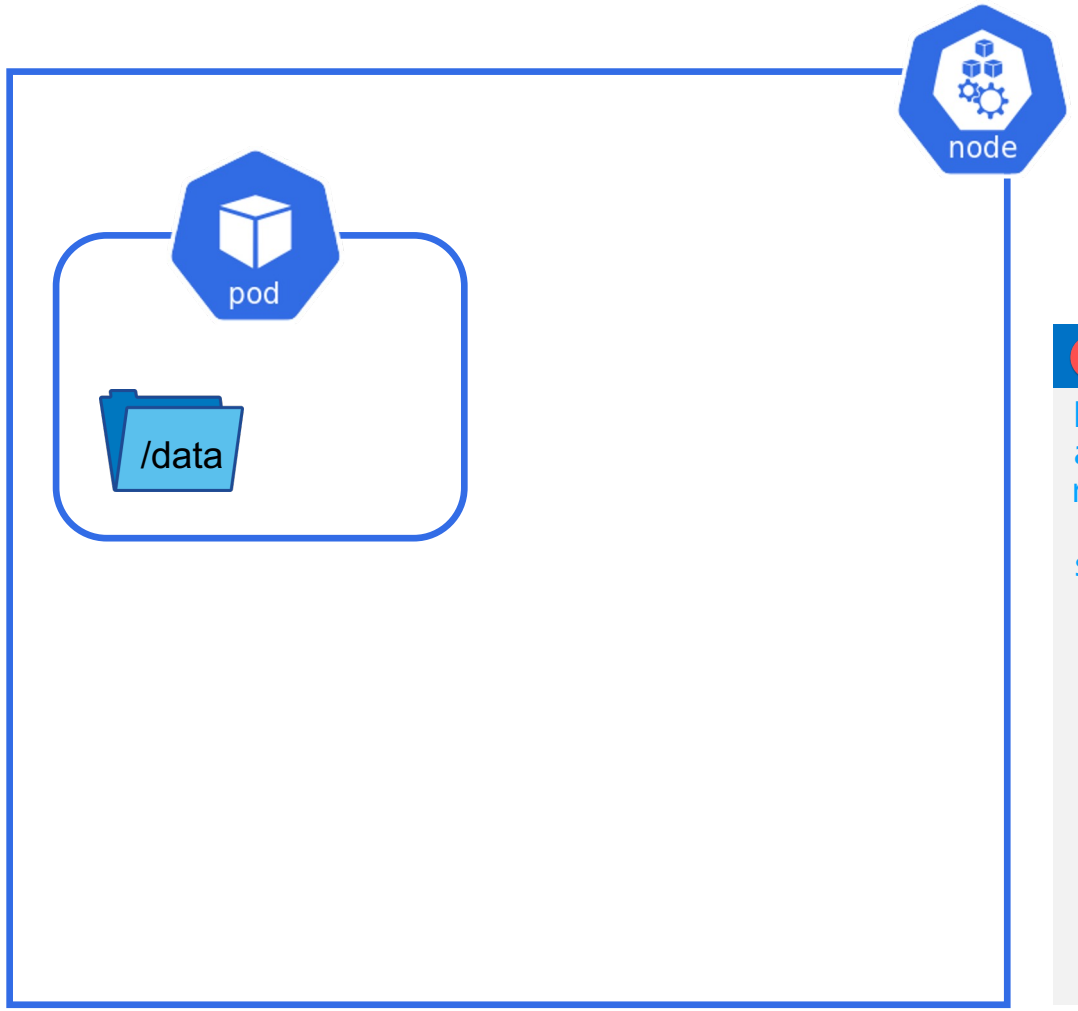
Storage – Place to store data

- 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)

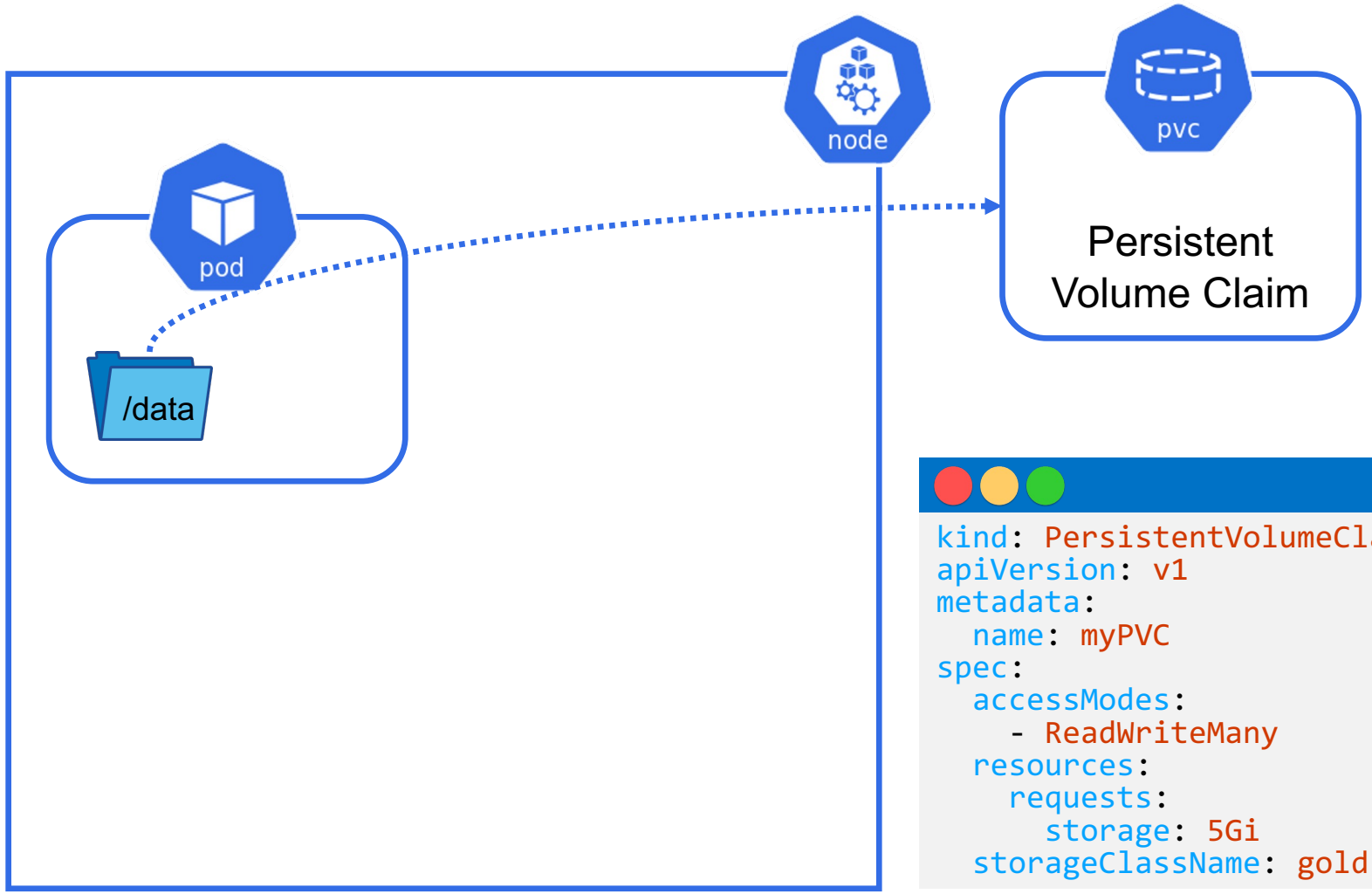
Dynamic storage provisioning in K8s



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
```

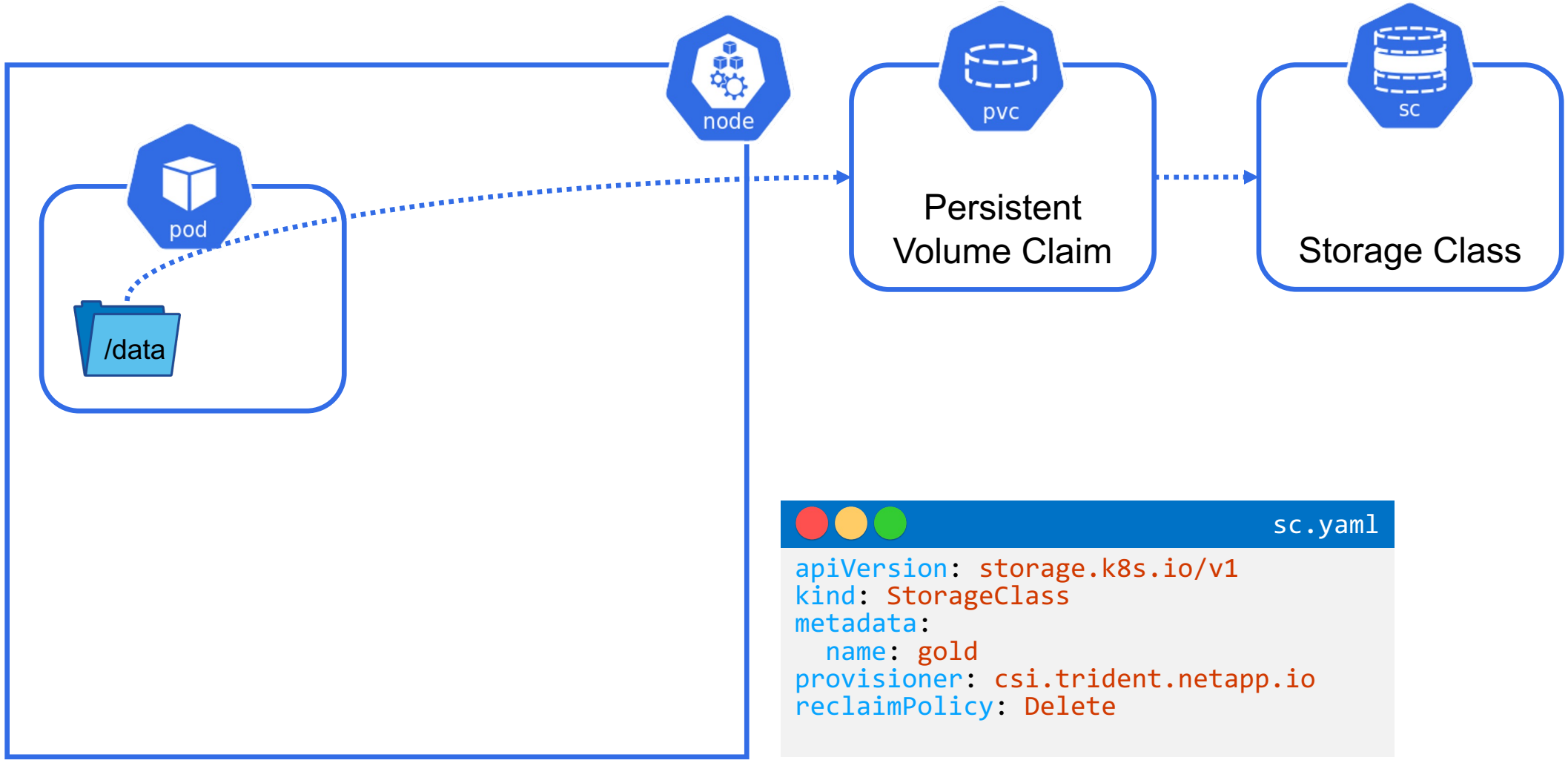
Dynamic storage provisioning in K8s



pvc.yaml

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

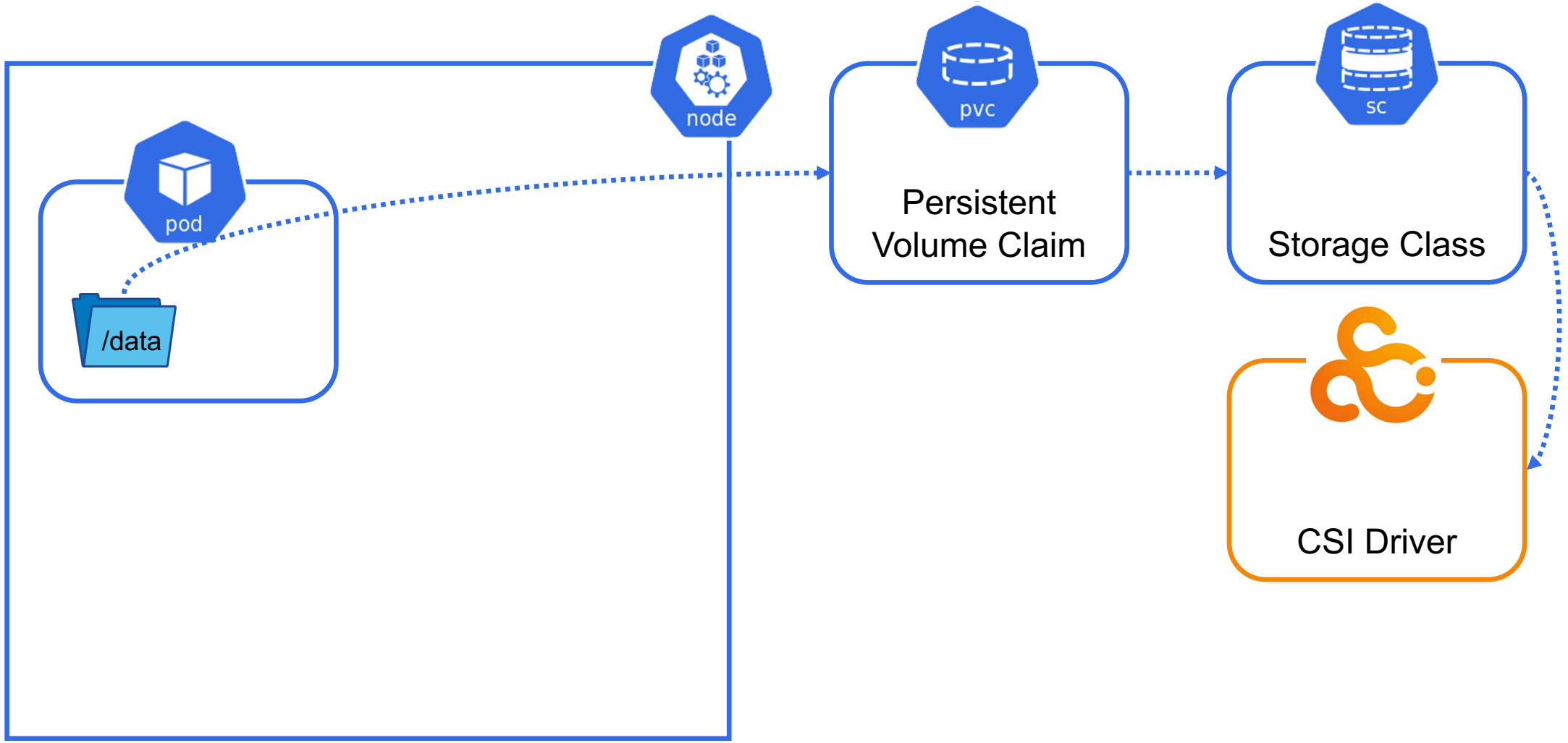
Dynamic storage provisioning in K8s



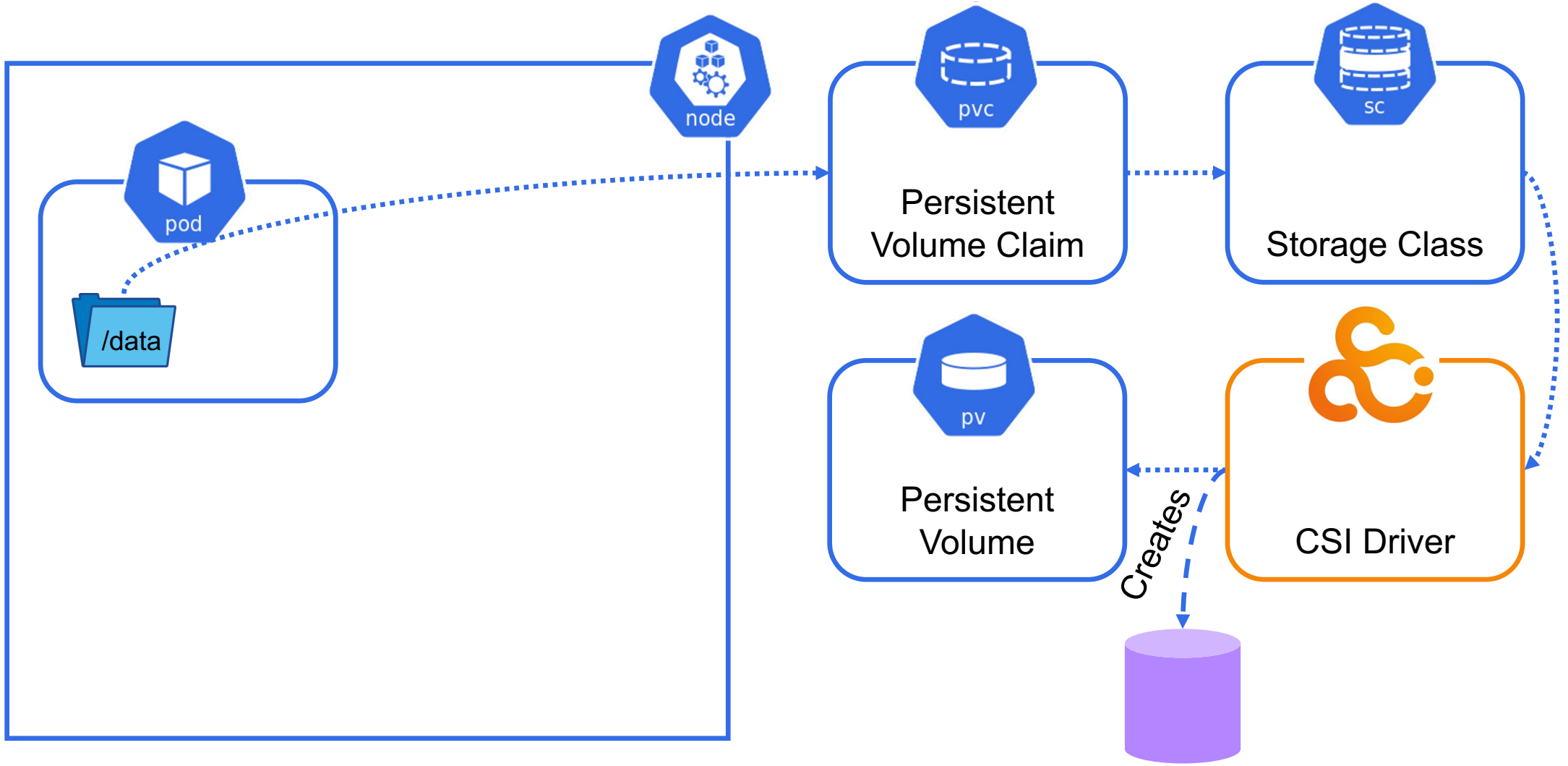
sc.yaml

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

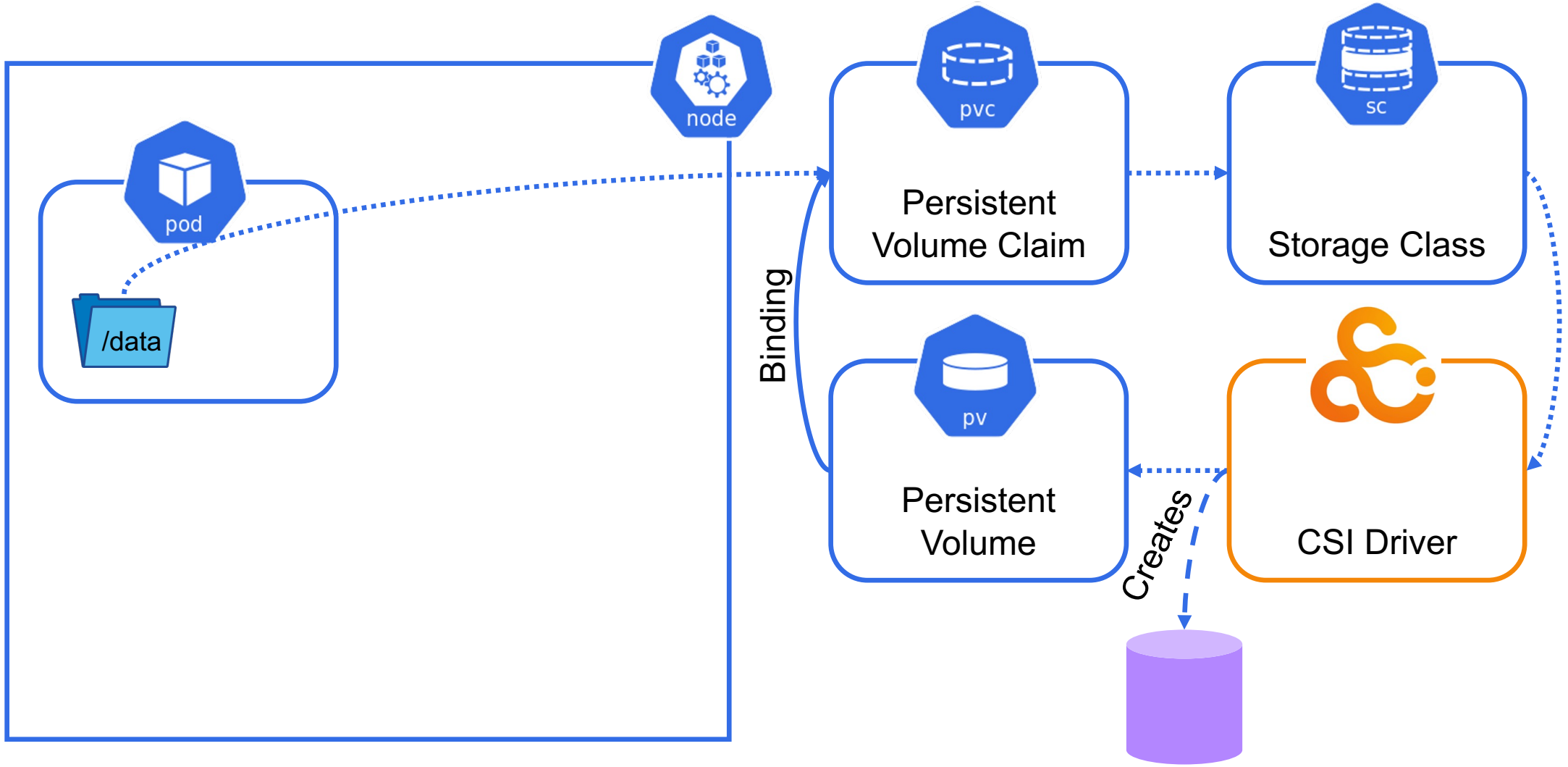
Dynamic storage provisioning in K8s



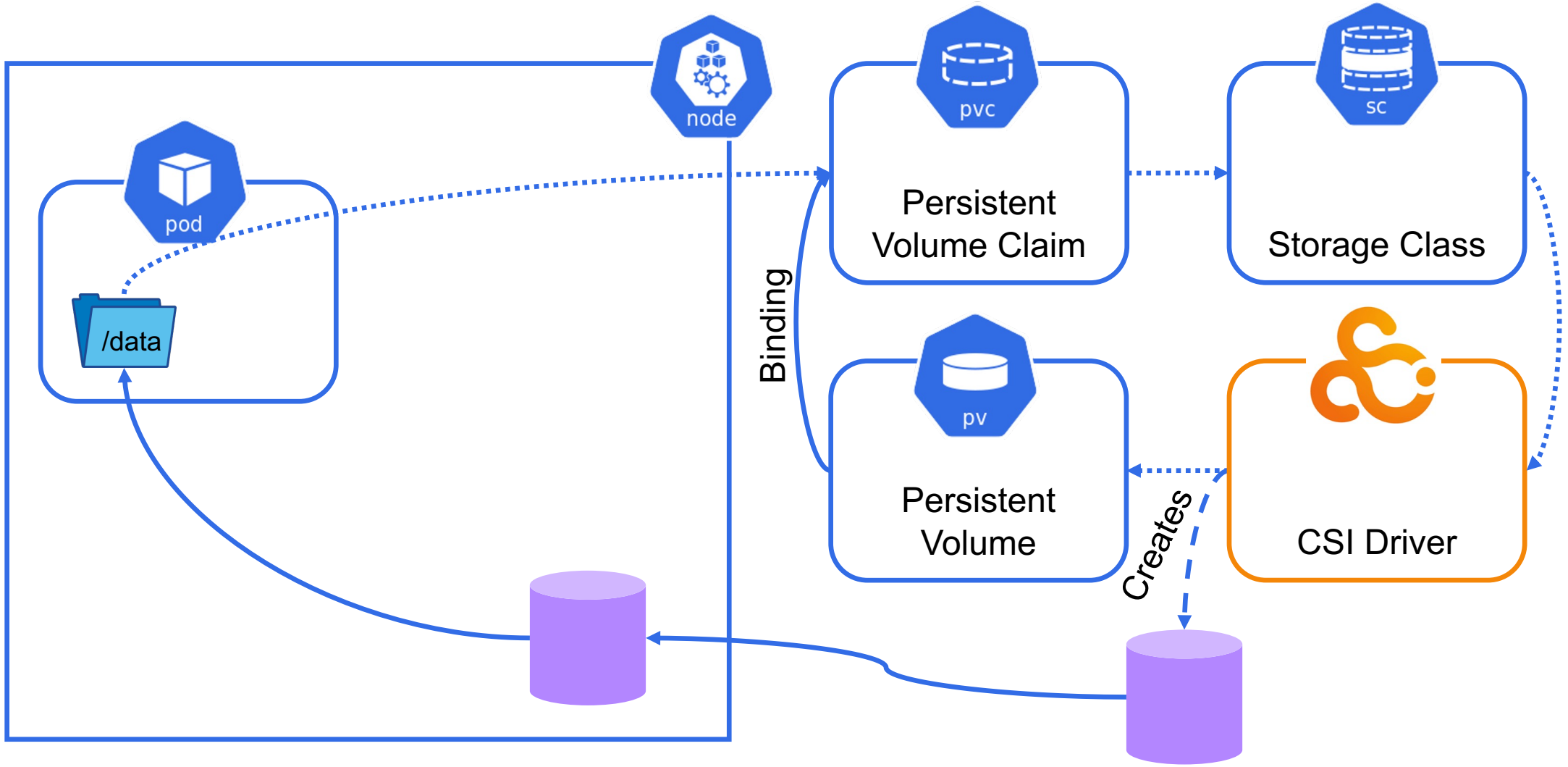
Dynamic storage provisioning in K8s



Dynamic storage provisioning in K8s



Dynamic storage provisioning in K8s



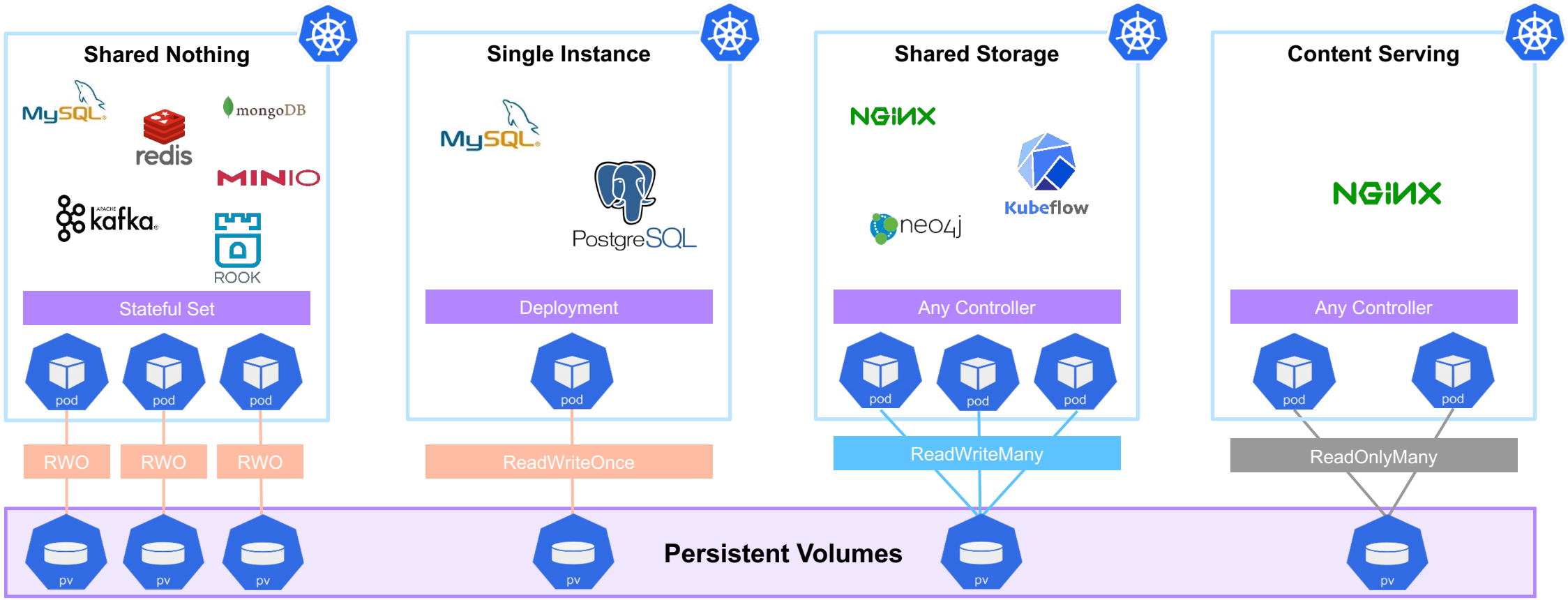
Persistent Volume Claim Access Modes

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



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

Object

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

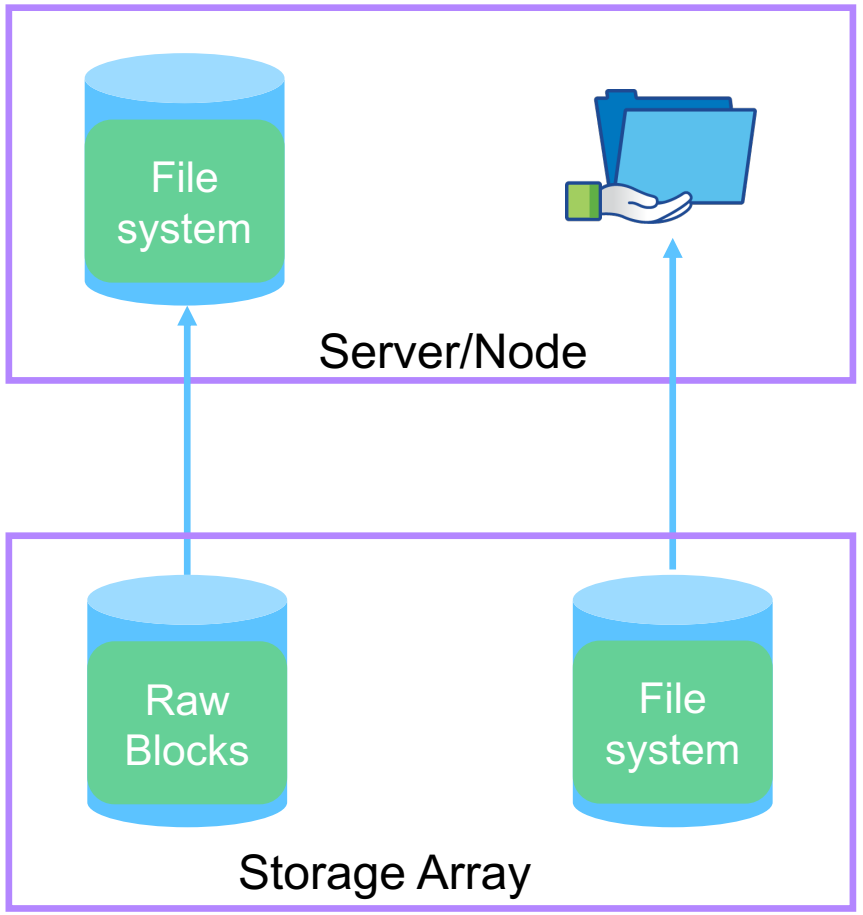
Container Storage Interface (CSI)

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

File/Block storage

Make the right choice per application

- Performance
  - Files and Block usually deliver similar performance
  - Small file workload (millions of small files) is usually faster on block storage
  - (Re-) mount is usually faster with file storage
- POSIX behaviour
  - NFS Shared filesystem (necessarily) has to behave different in specific areas
  - “Silly rename” – When deleting a file that you continue to use
- Access mode
  - File storage supports ReadWriteMany (RWX)
- Ease of use
  - NFS is often perceived easier to setup/operate
  - PVC Resize is immediate with NFS
- In general, follow the application vendor recommendations



Trident Driver for Ontap

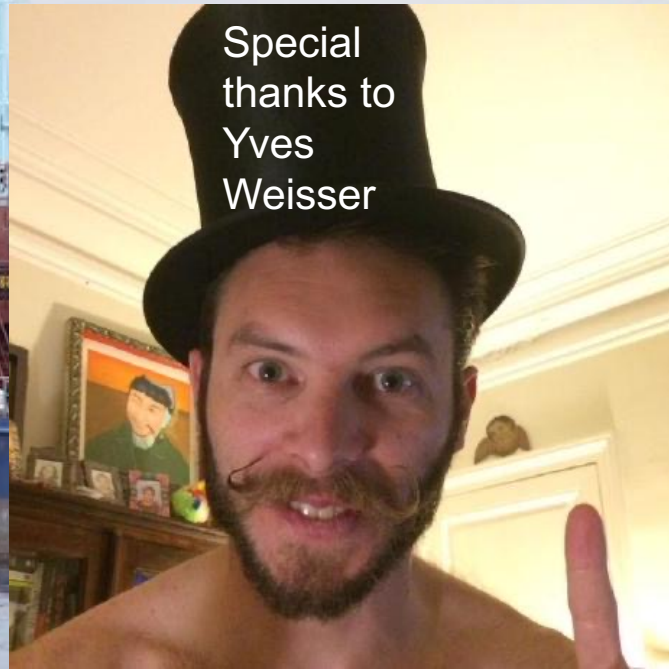
SAN / Block / iSCSI

- ontap-san
  - PVC = LUN in dedicated Ontap volume
- ontap-san-economy
  - PVC = LUN in shared Ontap volume
  - Reduces number of Ontap volumes required

NAS / File / NFS

- ontap-nas
  - PVC = Ontap Volume
- ontap-nas-economy
  - PVC = Ontap qtree
  - multiple PVC share Ontap Volume
  - Reduces number of Ontap volumes required
  - No PVC-granular Snapshot/Cloning support
  - Only use if you do **not** need data management
- ontap-nas-flexgroup
  - PVC = Ontap FlexGroup
  - For Volumes >100TB
  - No cloning support

Hands-on Part 1



Before we start

- Please raise your hand, write something in the chat or unmute yourself and ask us if you have questions or problems
- Choose only your username from the spreadsheet
- Don’t use the normal lab guide, use what is in the git repo
- Pre-setup is important
- We will stop after Scenario02 and Scenario04 for further theory sessions

Usernames for Lab on Demand

Access at https://lod-bootcamp.netapp.com
Password for all users: **8IwcKrbuQ&ze**
https://github.com/ntap-johanneswagner/kompaktlivelab23

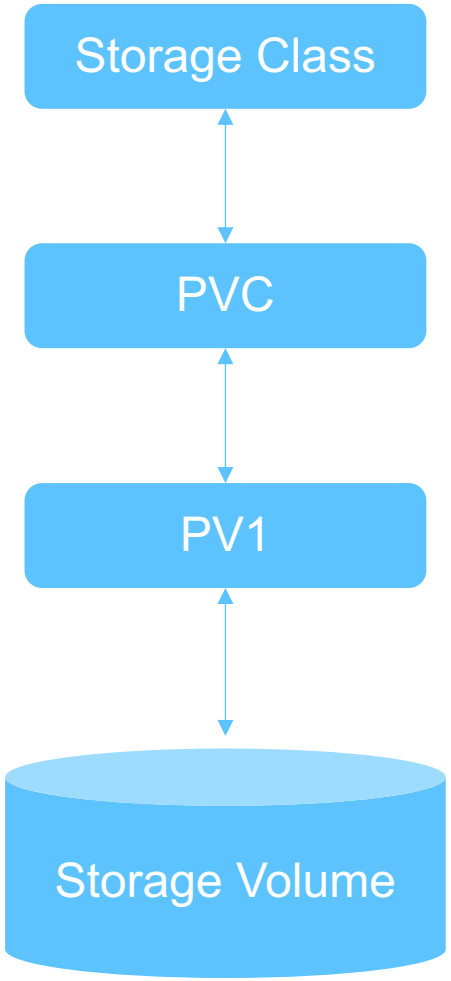
Teilnehmer	Username
Andreas Tellenbach	b11267u1
Falko Schmidt	b11267u2
Florian Lympius	b11267u3
Gabi Schmidt	b11267u4
Gerald Schneider	b11267u5
Maximilian Voit	b11267u6
Nico Bopp	b11267u7
Patrick Hilke	b11267u8
Stefan Berkel	b11267u9
Stephan Bergfeld	b11267u10
Sven Prause	b11267u11

https://github.com/ntap-johanneswagner/kompaktlivelab23

Hands-on Part 1

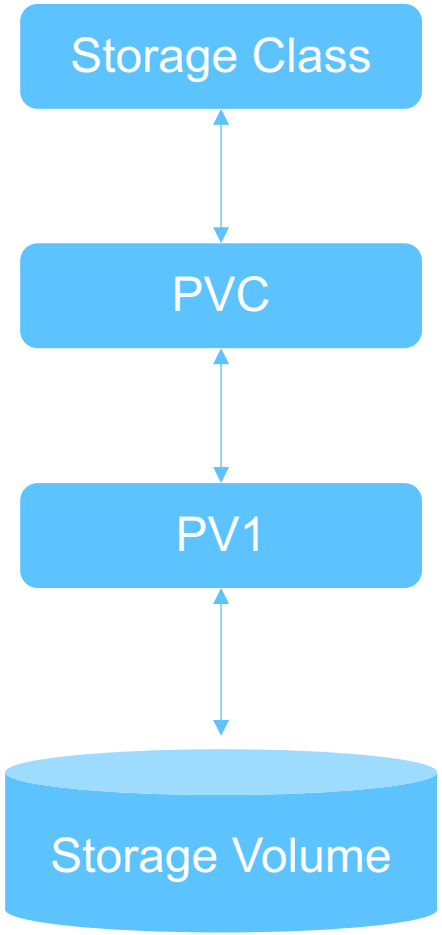
CSI Snapshots workflow

Initial configuration



CSI Snapshots workflow

Volume Snapshot Class creation



Volume Snapshot Class

volumesnapshotclass.yaml

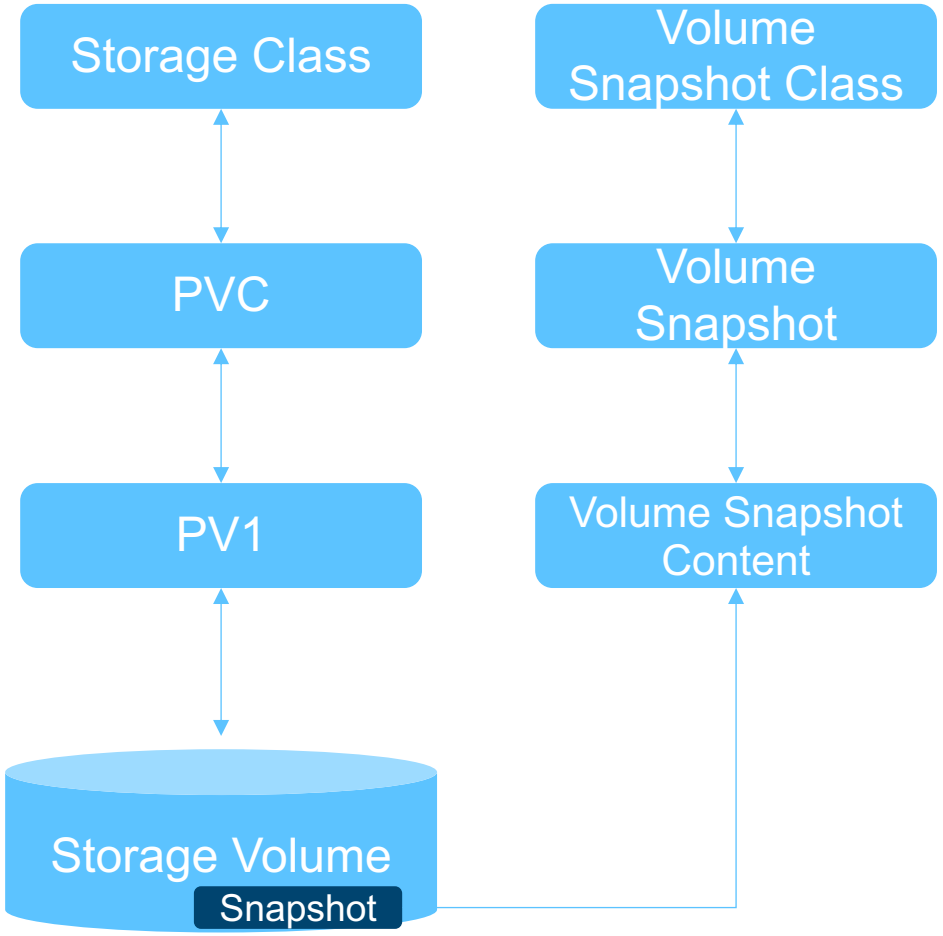
```
apiVersion: snapshot.storage.k8s.io/v1beta1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

NOTE

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

CSI Snapshots workflow

Volume Snapshot creation

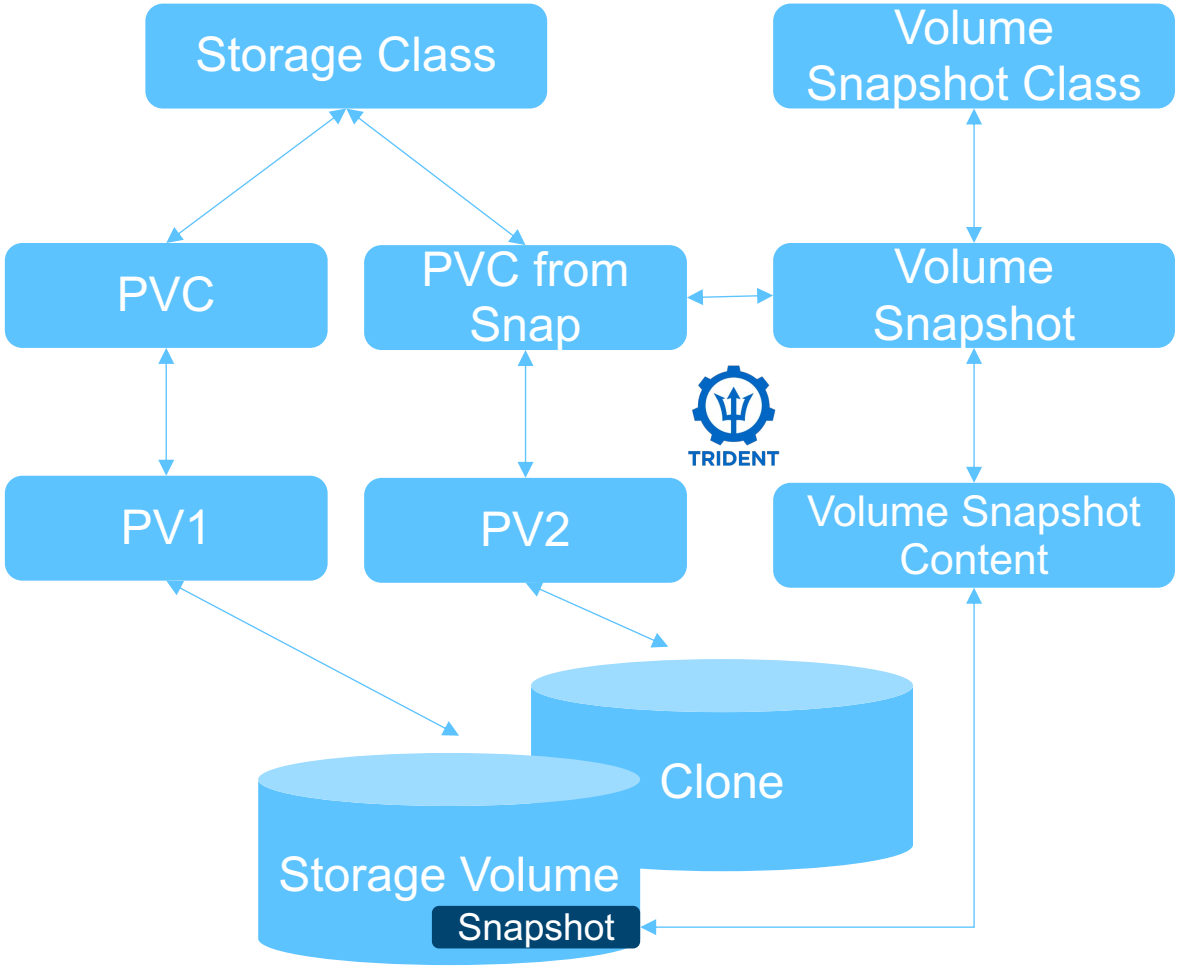


volume-snapshot.yaml

```
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

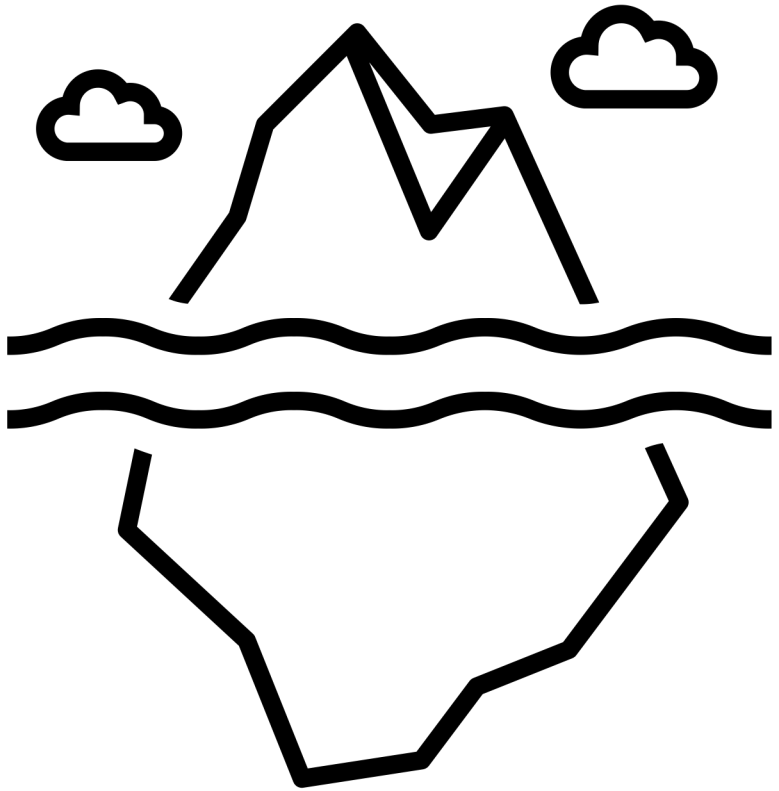
- Standard K8s mechanisms such as ResourceQuota and LimitRange apply to storage as well
- Restrict capacity and number of PVCs per namespace with a ResourceQuota
  - Total capacity
  - Capacity per StorageClass
  - Total number of PVCs
  - Number of PVCs per StorageClass
- Remember: StorageClass is a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

https://github.com/ntap-johanneswagner/kompaktlivelab23

Hands-on Part 2

EVERYTHING you always wanted to know about storage in Kubernetes?

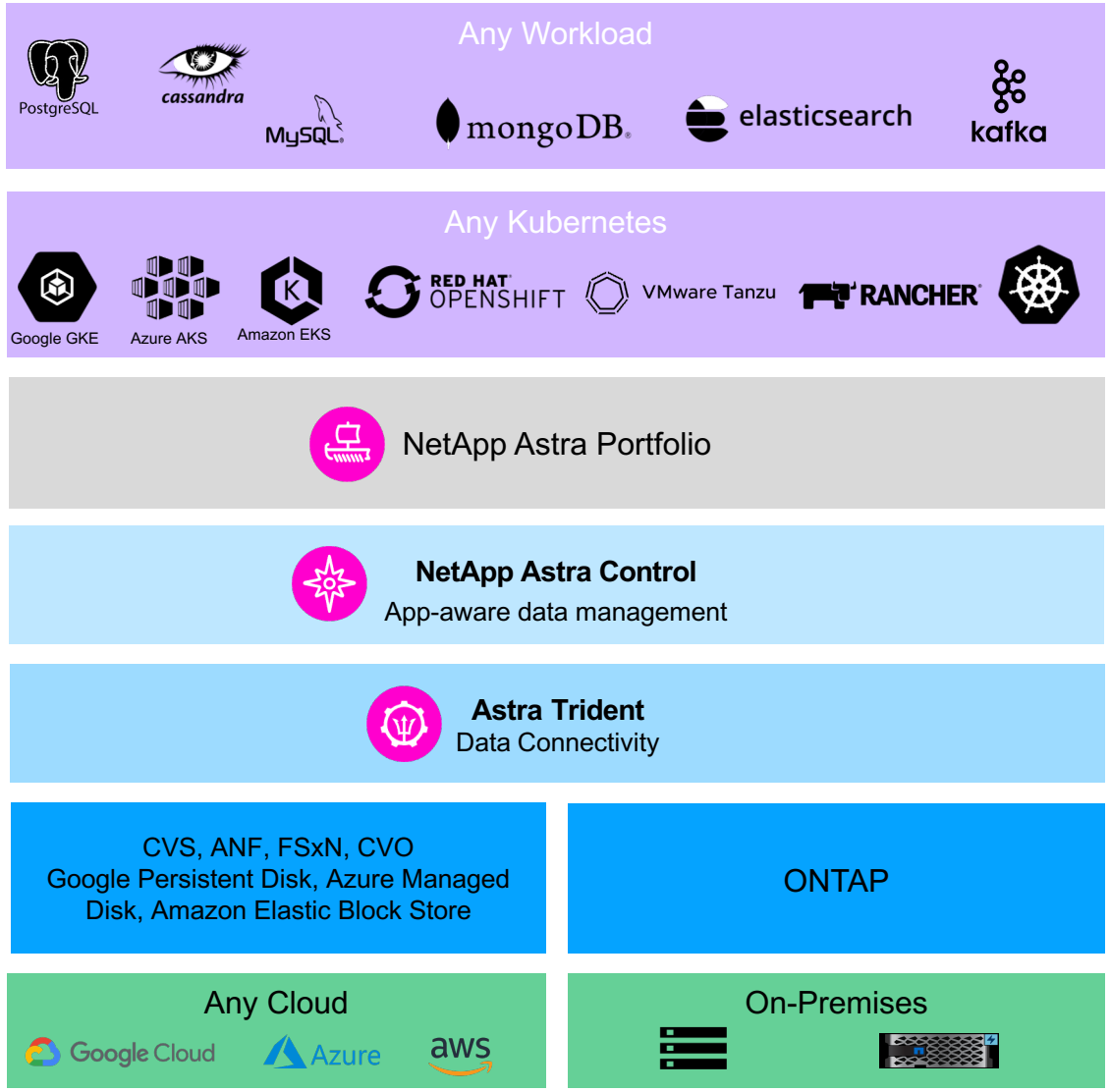
OK, there’s more…



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

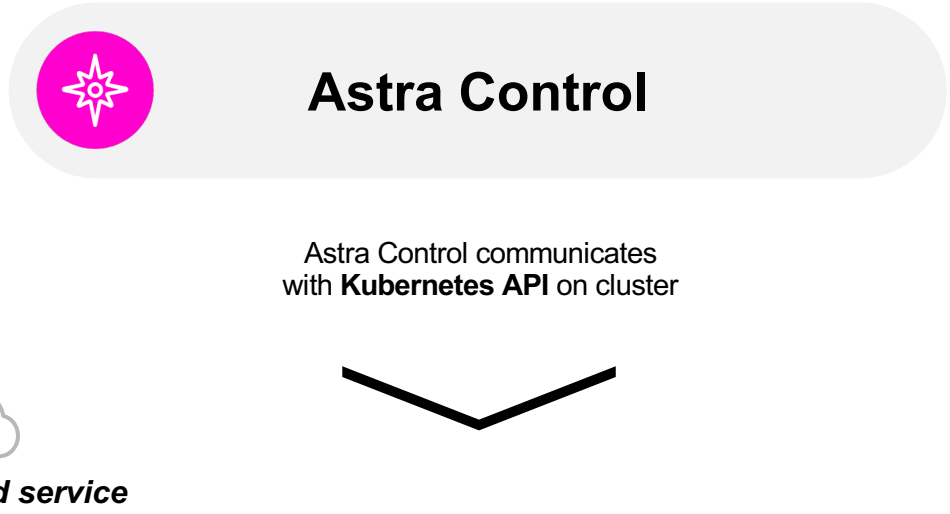
NetApp Astra

- Cloud-Native
- Application-aware
- On-premises & Any Cloud
- Any Kubernetes
- Storage & Data Management done right



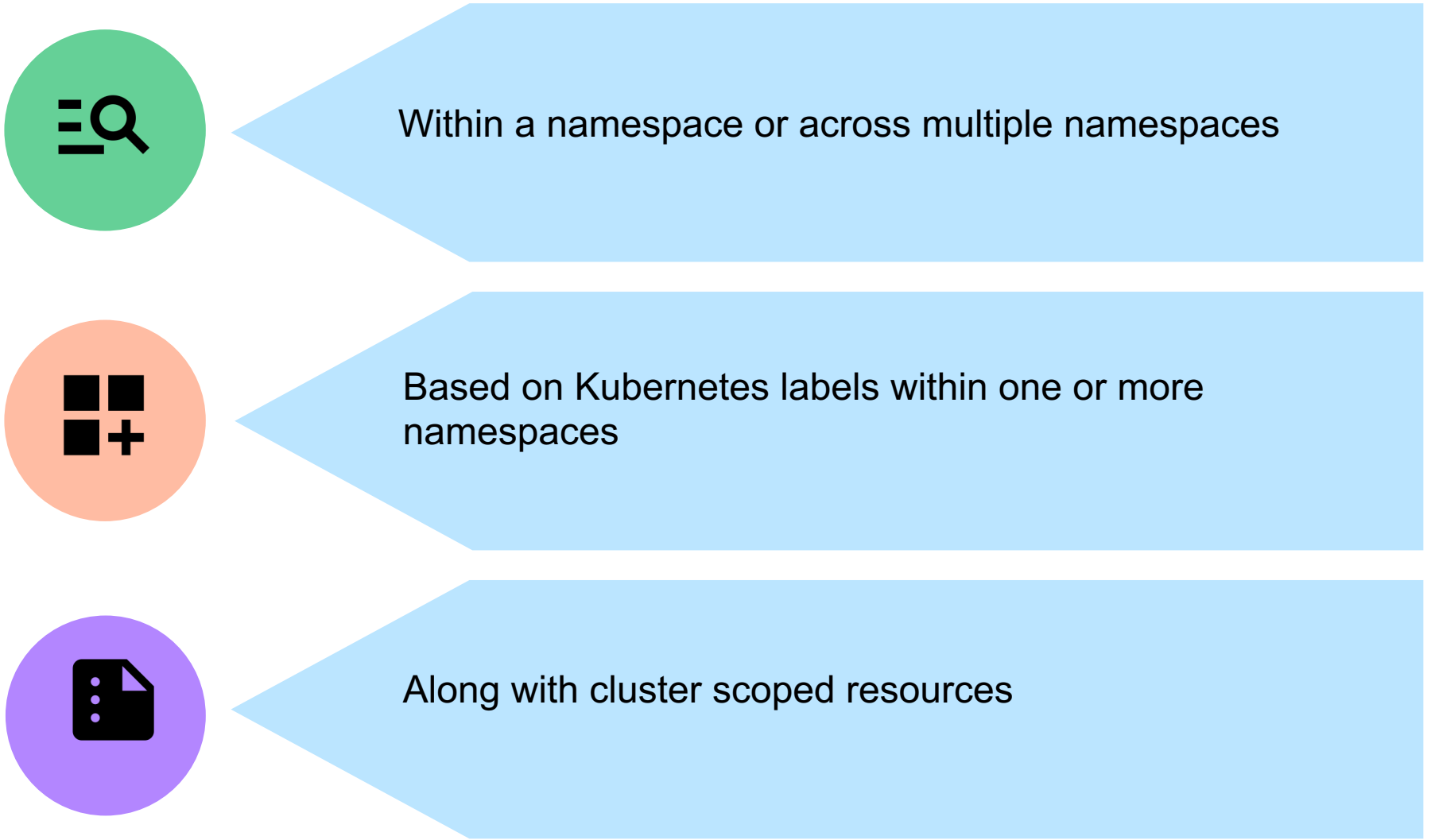
Astra Control

Multi-cloud, end-to-end application data lifecycle management



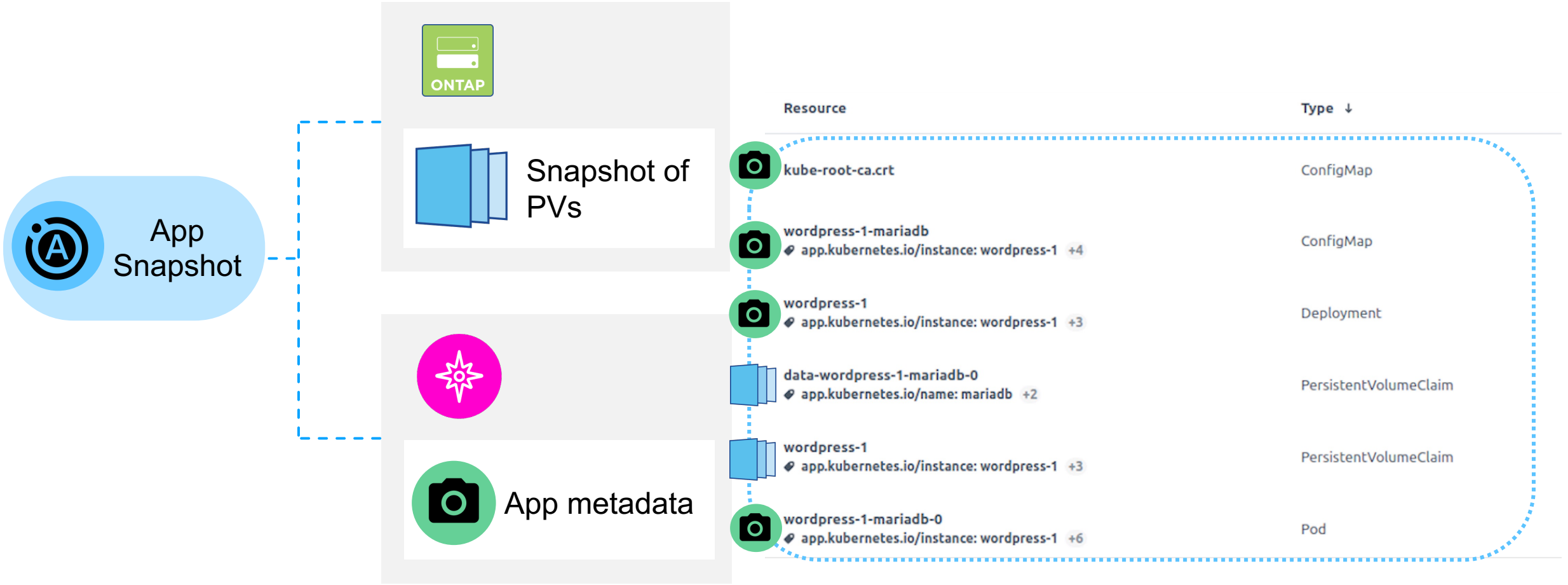
Managed Applications in Astra Control

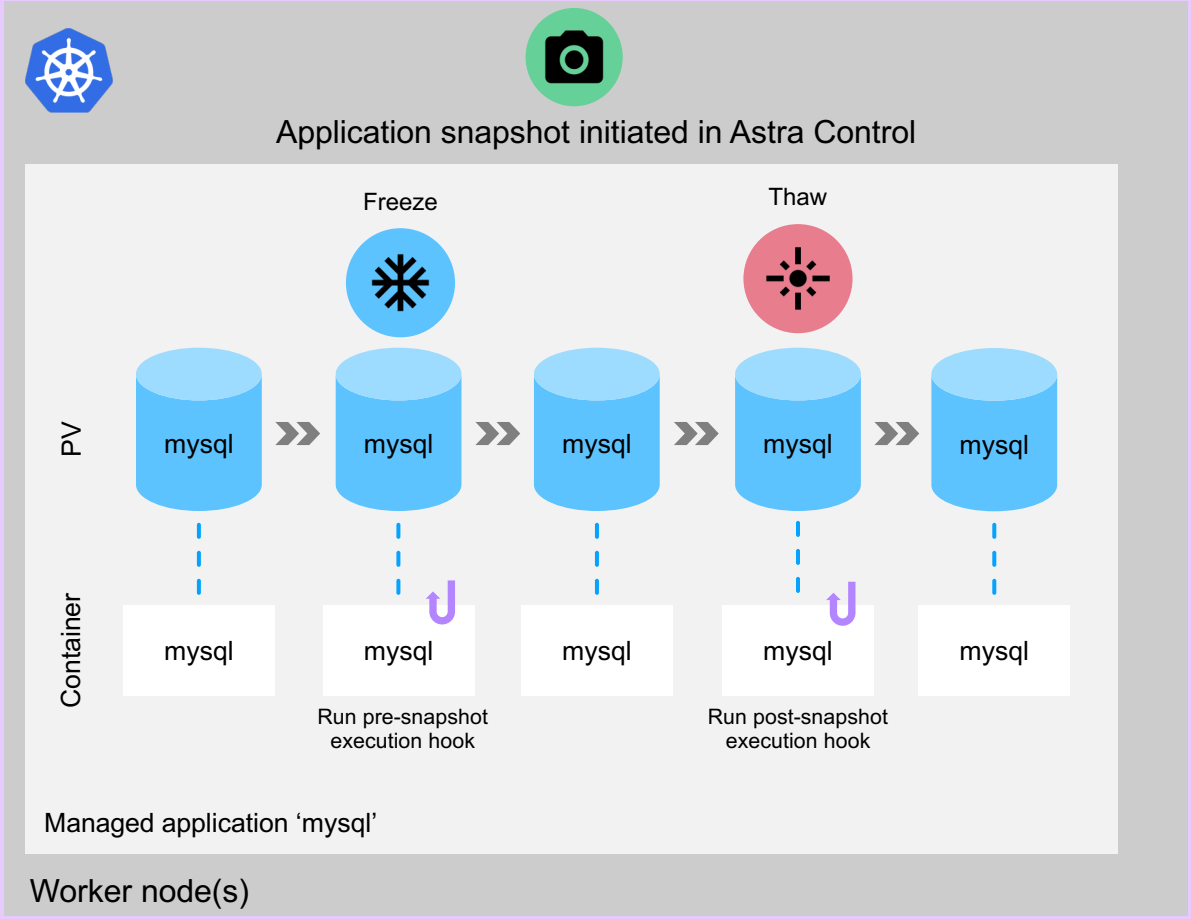
Multiple options to protect applications and data



Data Protection On-Demand or Scheduled

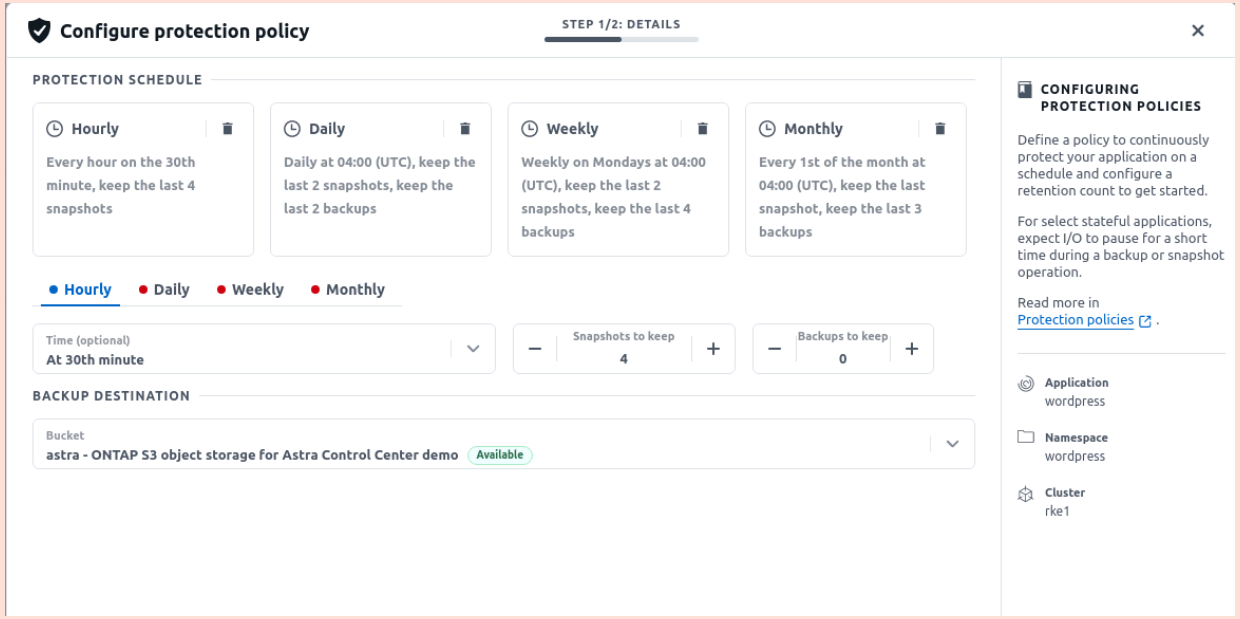
Protect your Application, Metadata and Persistent Volumes

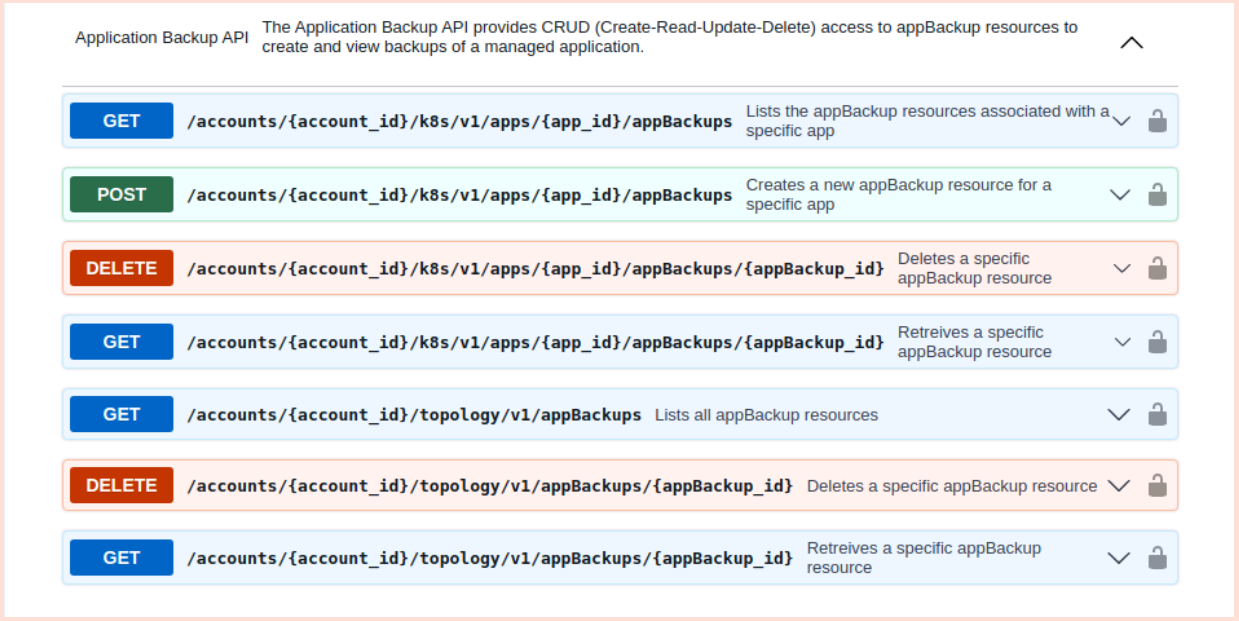




A good backup needs consistence

Without execution hooks you don’t know the state of your application





Stop manual work

Use policies and API

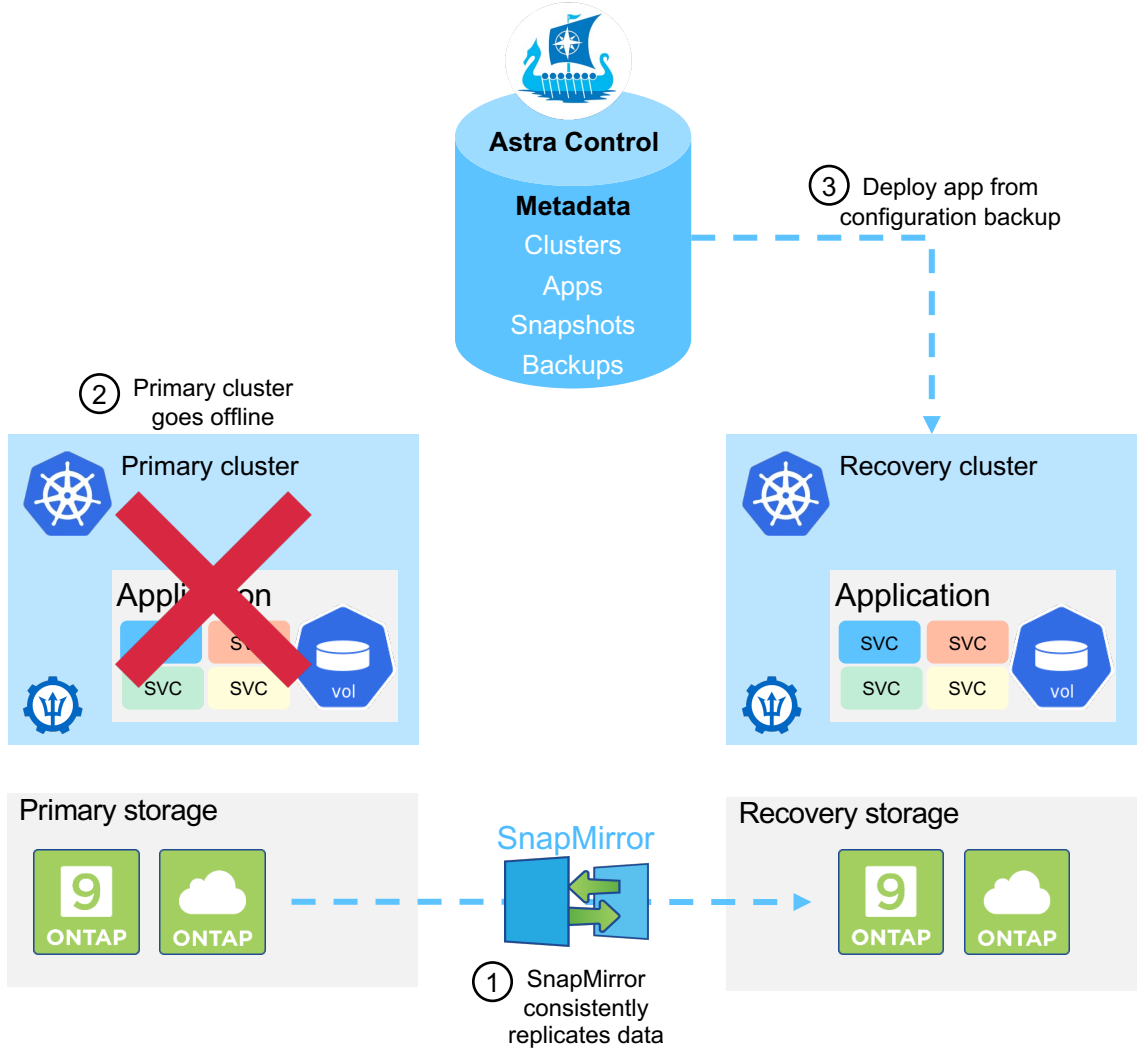
Business continuity

Problem

Enterprise applications require business continuity to meet RPO/RTO objectives

Solution

Using Astra Control, you can quickly protect applications to a remote cluster in preparation for disaster recovery using SnapMirror



*** Includes support for multi-namespace applications**

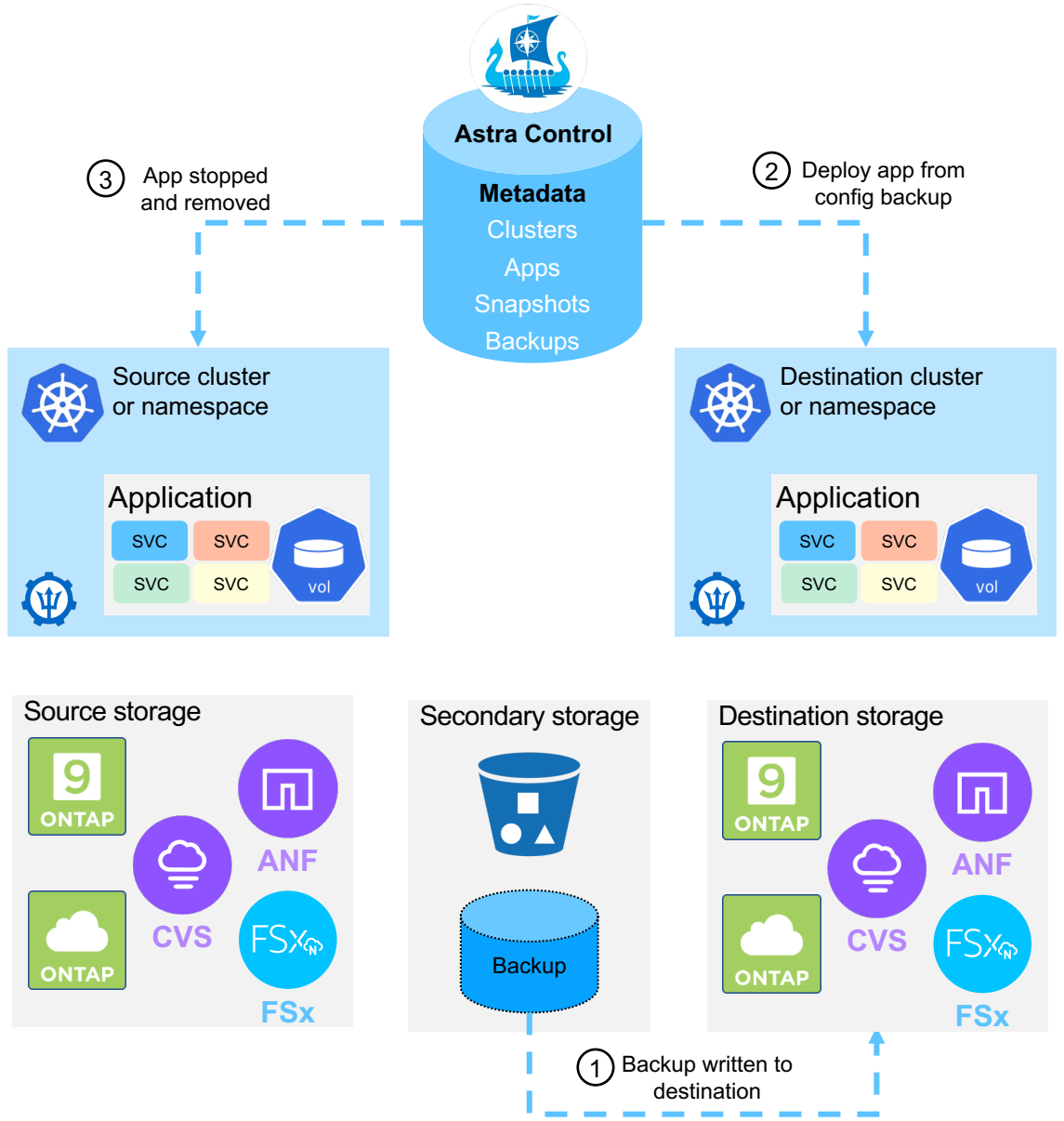
Application mobility

Problem

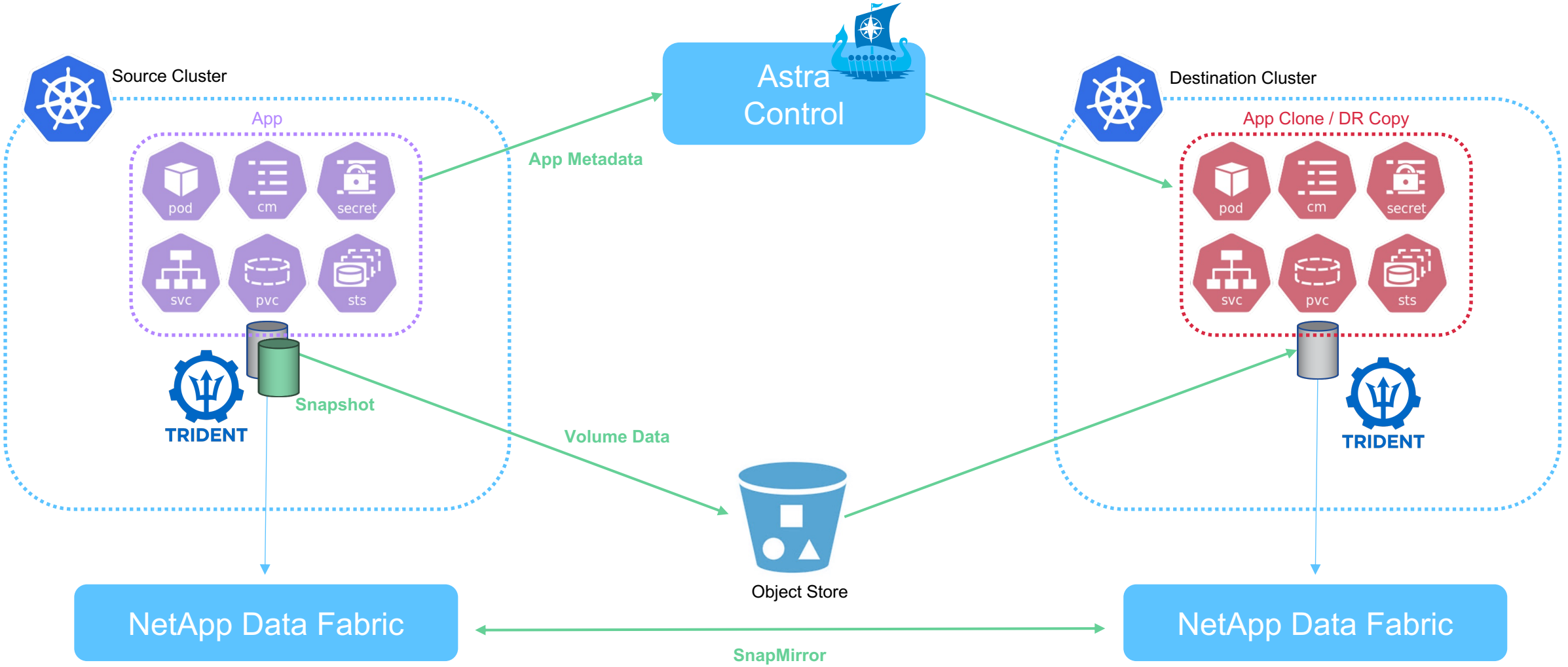
Requirement to move data due to data residency, compliance, or regulatory reasons

Solution

Astra can clone and move application data freeing the app to move between clusters either in the cloud or on-premises



DataFlow



https://github.com/ntap-johanneswagner/kompaktlivelab23

Hands-on Part 3