OpenSDS mini summit COPENHAGEN

Managing and Protecting Persistent Volumes for Kubernetes

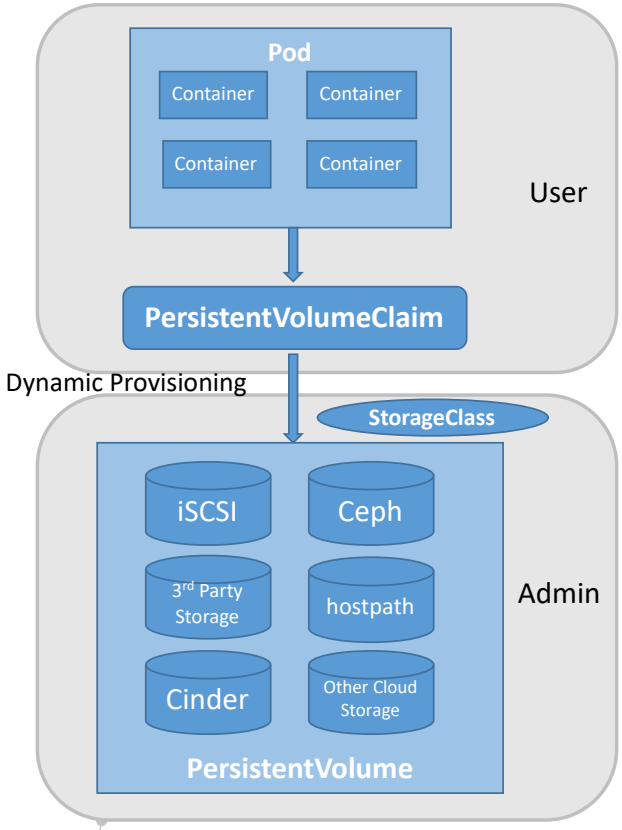
Xing Yang, Principal Architect, Huawei

Agenda

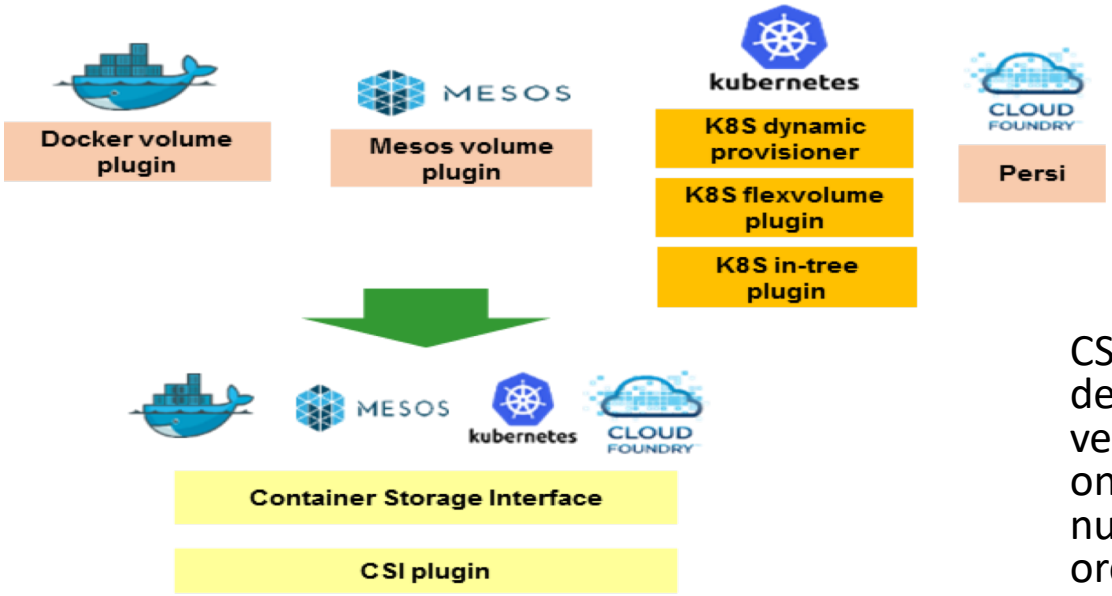
- Kubernetes Persistent Volumes
- Container Storage Interface (CSI)
- Provision and Manage Persistent Volumes using OpenSDS
  - OpenSDS Architecture
  - Mapping OpenSDS Profile to K8S StorageClass
  - Policy Driven SPDM
- Data Protection for Persistent Volumes
- Disaster Recovery for Persistent Volumes
  - Array-based Replication
  - Host-based Replication
- Demo

Kubernetes Persistent Volumes

- A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator.
- A PersistentVolumeClaim (PVC) is a request for storage by a user through a StorageClass.
- A StorageClass provides a way for administrators to describe the “classes” of storage they offer. Different classes might map to different quality-of-service levels (or ”profiles”) in other storage systems.
- A StorageClass needs to specify a provisioner for dynamic provisioning.

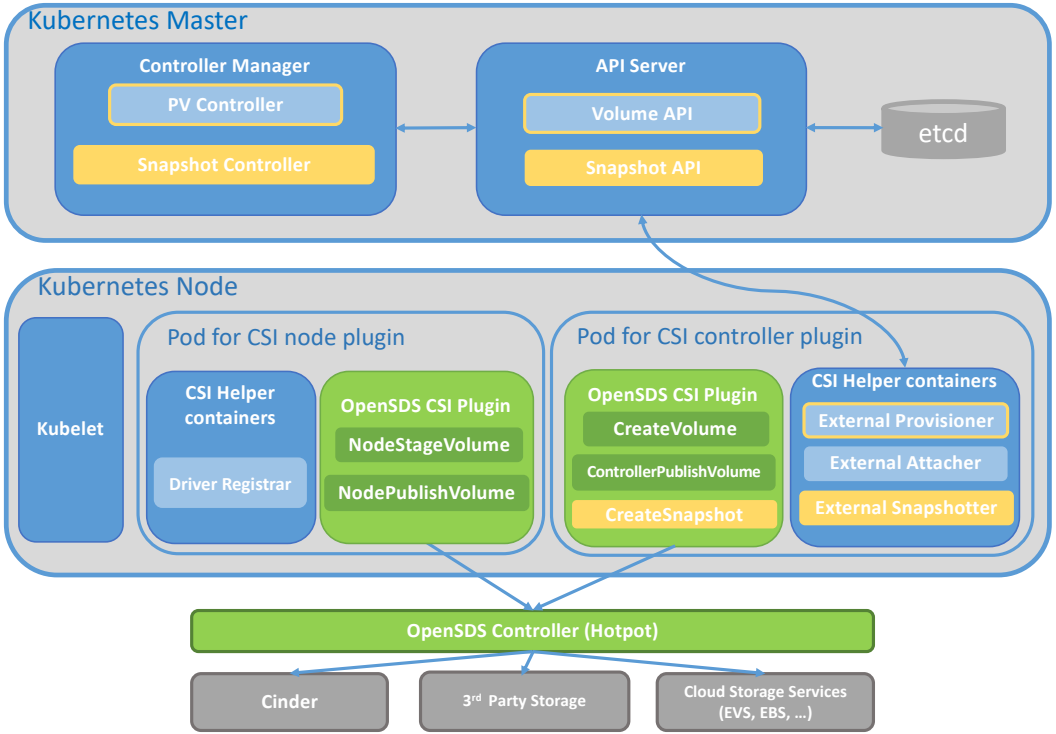


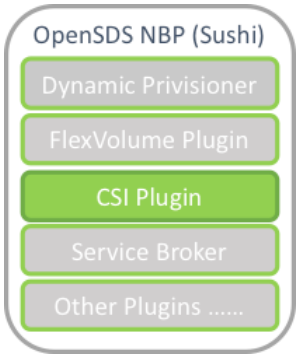
Container Storage Interface (CSI)



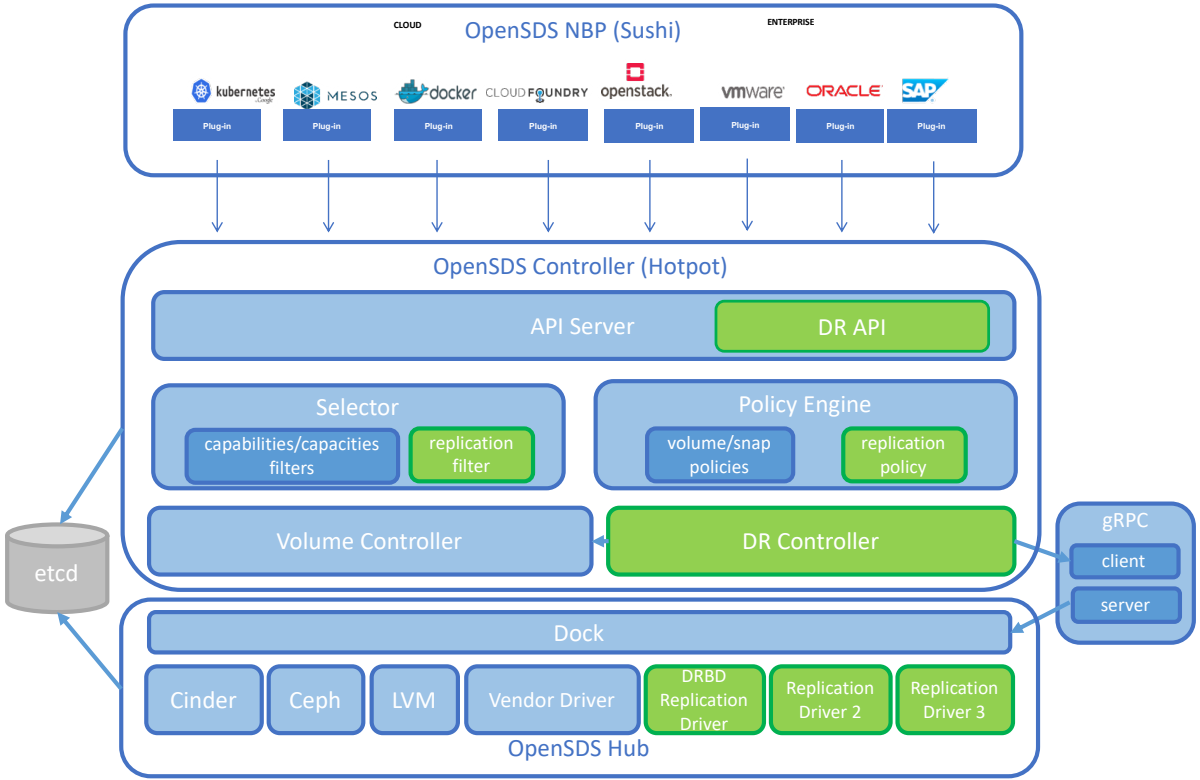
CSI is an industry standard defined to enable storage vendors to develop a plugin once and have it work across a number of container orchestration systems.

Provision and Manage Persistent Volumes using OpenSDS

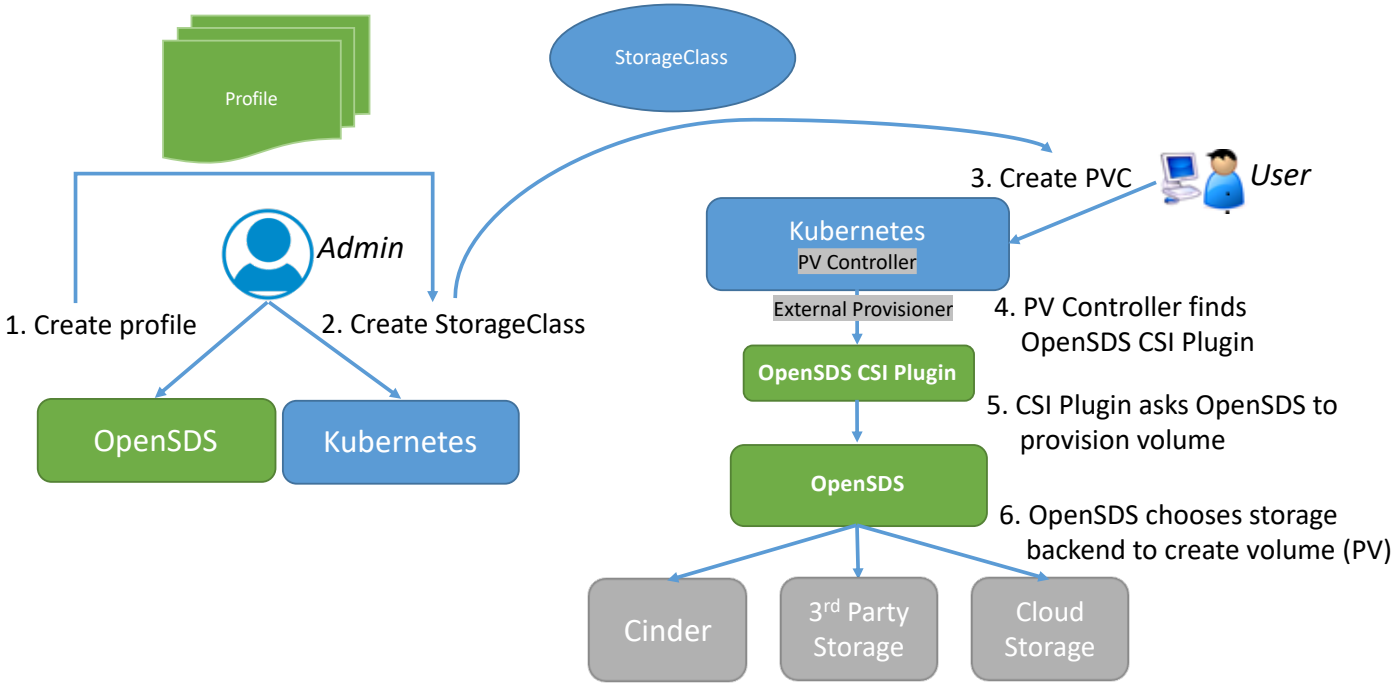




OpenSDS Architecture

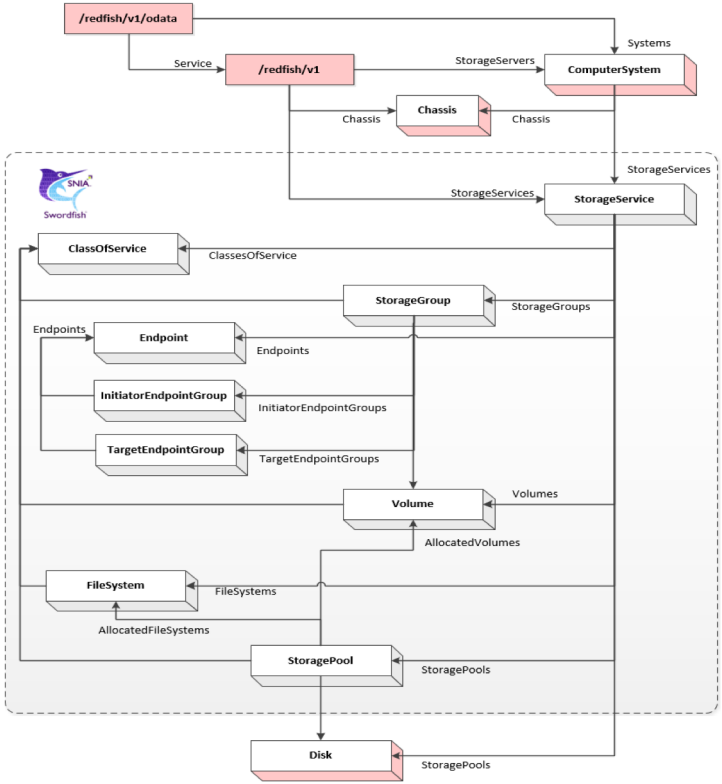


Mapping OpenSDS Profile to K8S StorageClass



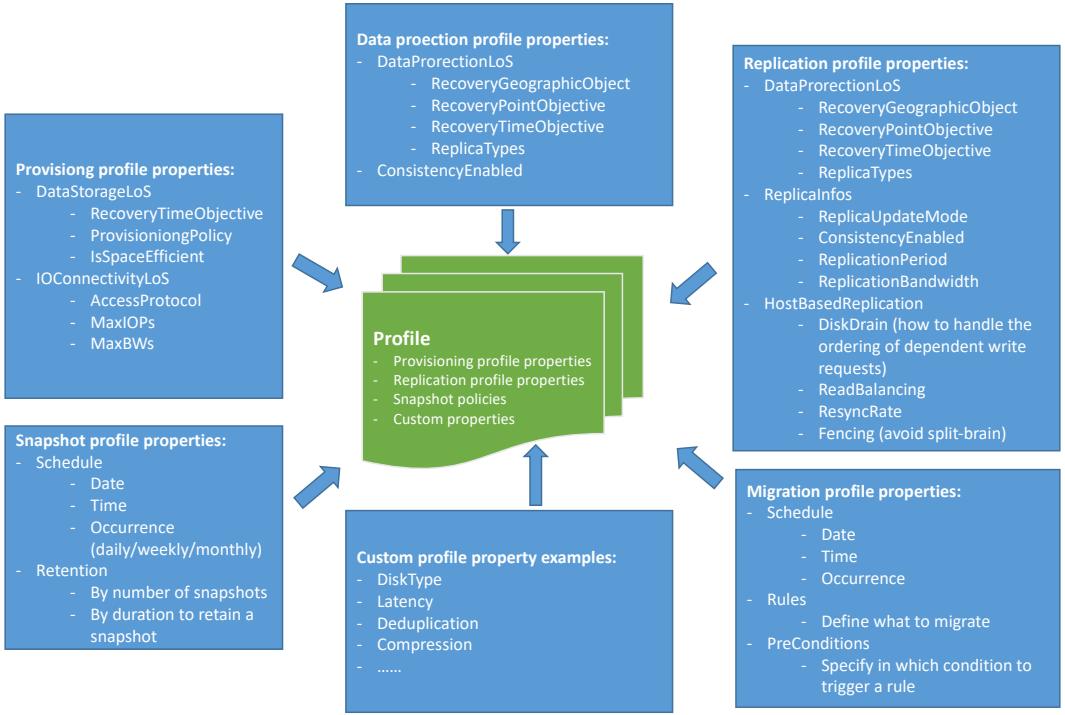
Policy Driven SPDM

- OpenSDS profile is based on Swordfish specification.
- The SNIA Swordfish™ specification helps to provide a unified approach for the management of storage and servers in hyperscale and cloud infrastructure environments, supported by multiple storage vendors.
- An extension of the DMTF (Distributed Management Task Force) Redfish specification.
  - Redfish is designed by the DMTF's Scalable Platforms Management Forum (SPMF) to create and publish an open industry standard specification and schema for management of scalable platform hardware. It is a RESTful interface over HTTPS in JSON format based on OData v4.

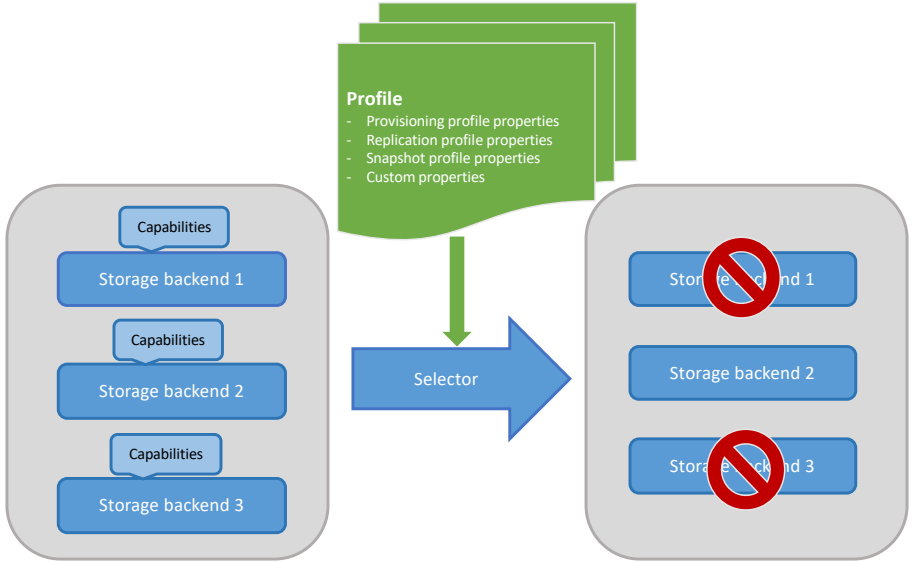


Source: Swordfish_v1.0.5_Specification

Profile Definitions



Mapping Profiles to Capabilities



Profile Example

DATA STORAGE

- **DataStorageLoS**
  - RecoveryTimeObjective (Immediate, Nearline, Offline, Online)
  - ProvisioningPolicy (thin, thick)
  - IsSpaceEfficient (true, false)

DATA PERFORMANCE

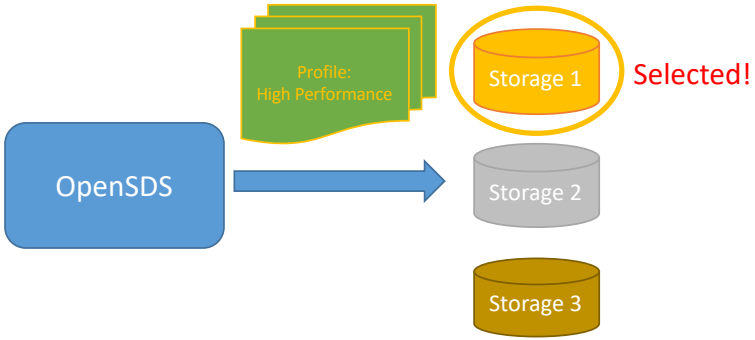
- **IOConnectivitLoS**
  - AccessProtocol (iSCSI, FC, RBD …)
  - MaxIOPS
  - MaxBWS

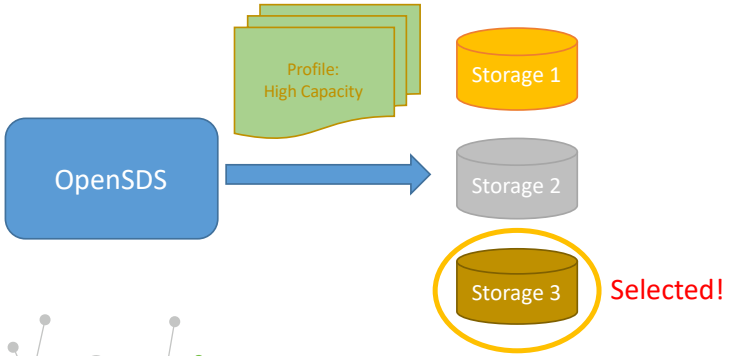
REPLICATION

- **DataProtectionLoS**
  - RecoveryGeographicObjective
  - RecoveryTimeObjective
  - RecoveryPointObjective
  - ReplicaTypes
- **ReplicaInfos**
  - ReplicaitonUpdateMode
  - ConsistencyEnabled
  - ReplicationPeriod
  - ReplicationBandwidth

SNAPSHOT

- **Schedule**
  - Date
  - Time
  - Occurrence (daily/weekly/monthly)
- **Retention**
  - By number of snapshots
  - By duration to retain a snapshot





StorageClass with Profile Parameter

HighPerformanceSC.yaml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: opensds-csi-high-performance-sc
provisioner: csi-opensdsplugin
parameters:
   profile: High-Performance
```

HighPerformancePVC.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
   name: opensds-csi-high-performance-pvc
spec:
   accessModes:
   - ReadWriteOnce
   resources:
      requests:
         storage: 10Gi
   storageClassName: opensds-csi-high-performance-sc
```

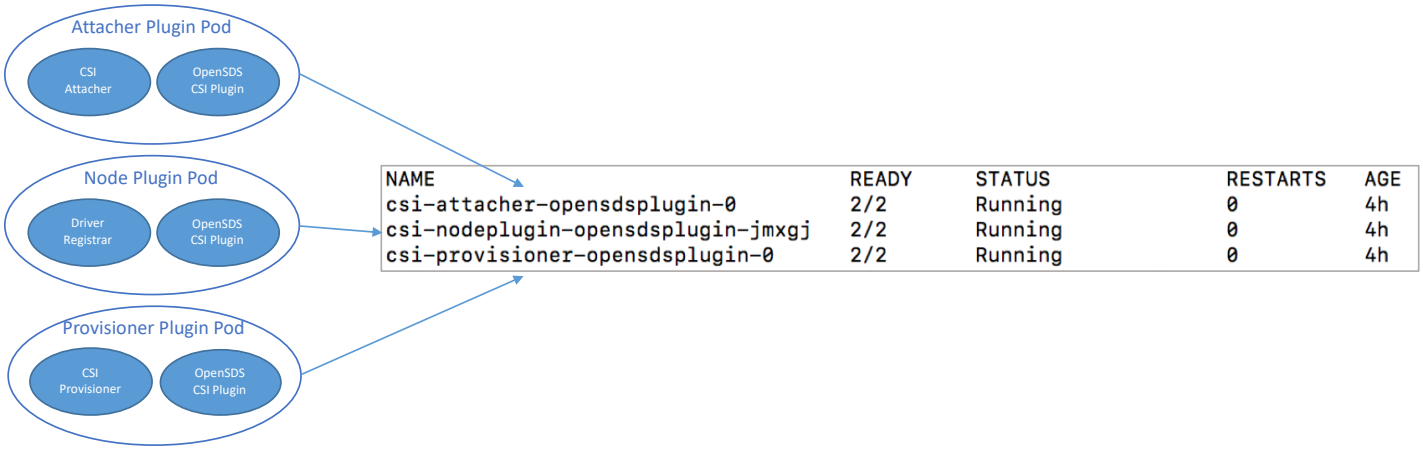
Note: profile parameter can be profile id or name

Running OpenSDS CSI Plugin

- Create OpenSDS CSI plugin pods:

  `kubectl create -f csi/server/deploy/kubernetes`

- Three pods can be found by `kubectl get pod`:



Using OpenSDS Volume

- Create nginx application

  kubectl create -f

  csi/server/examples/kubernetes/nginx.yaml

- An OpenSDS volume is mounted at */var/lib/www/html*.

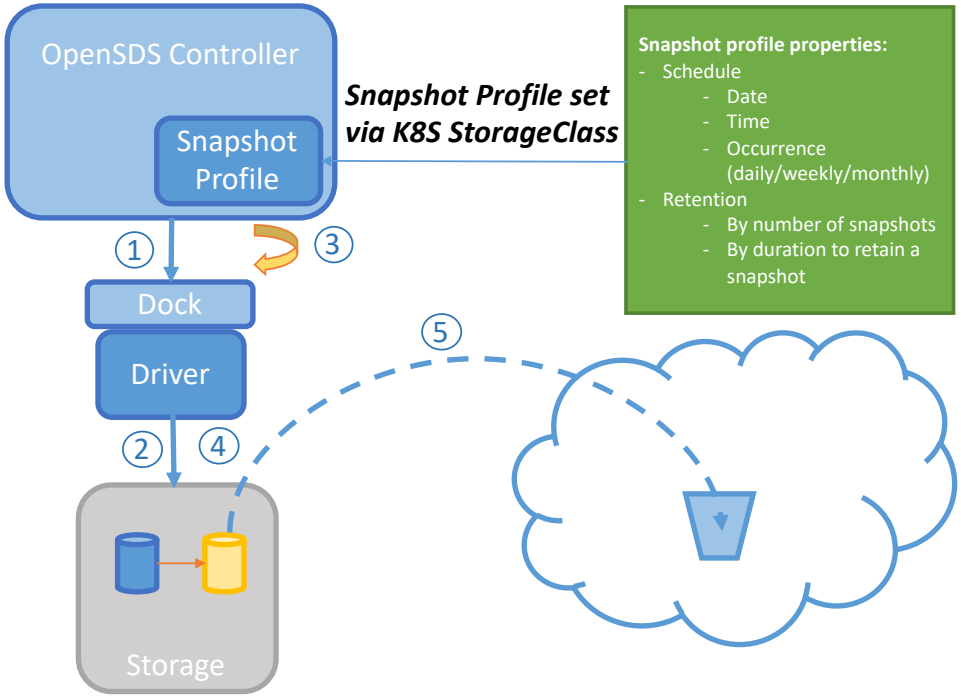
  docker exec -it <nginx container id> /bin/bash

```
root@nginx:/# mount | grep html
/dev/sda on /var/lib/www/html type ext4 (rw,relatime,data=ordered)
```

nginx.yaml

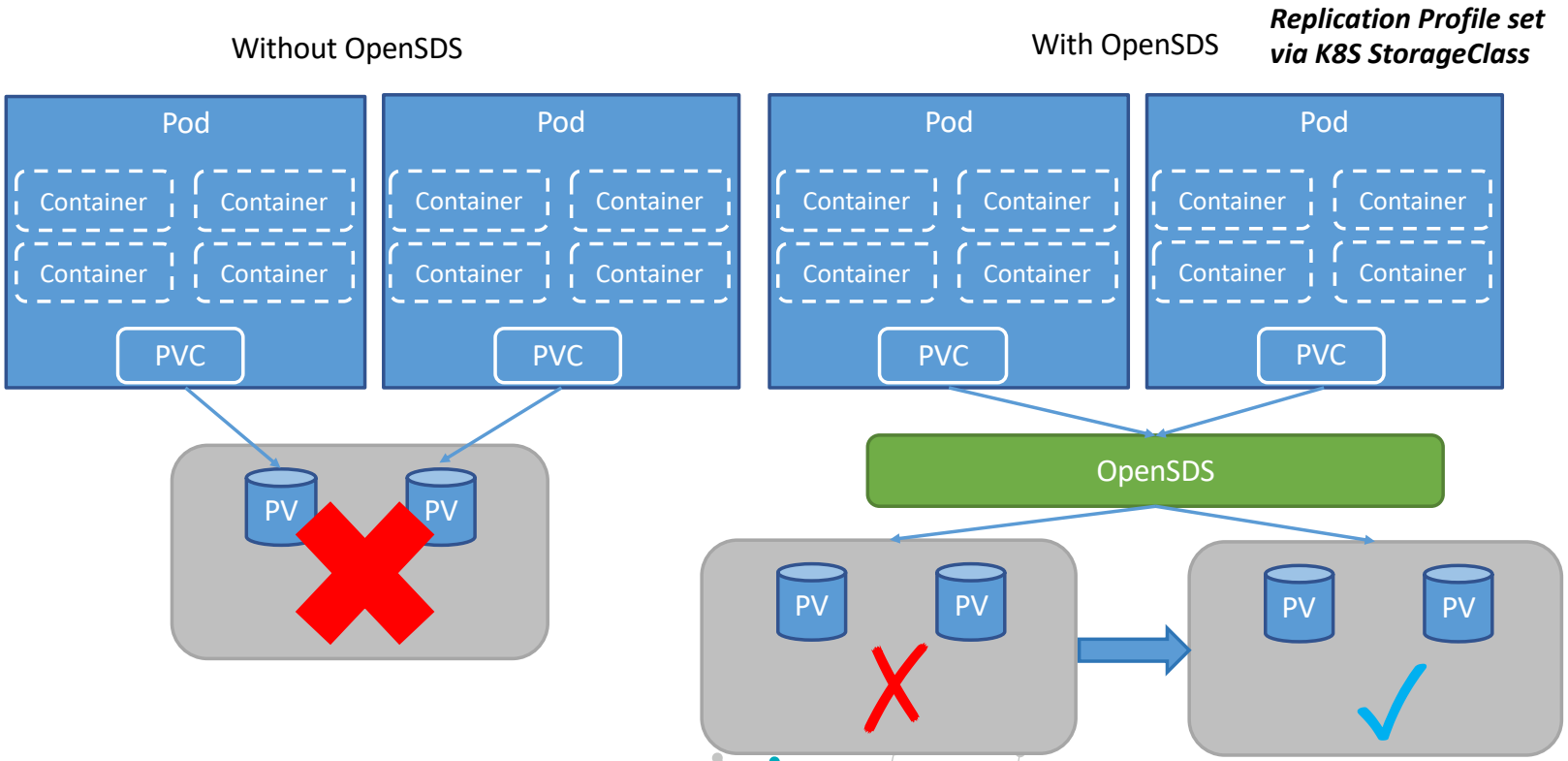
```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    imagePullPolicy: IfNotPresent
    name: nginx
    ports:
    - containerPort: 80
      protocol: TCP
    volumeMounts:
      - mountPath: /var/lib/www/html
        name: csi-data-opensdsplugin
  volumes:
  - name: csi-data-opensdsplugin
    persistentVolumeClaim:
      claimName: opensds-csi-high-performance-pvc
      readOnly: false
```

Data Protection for Persistent Volumes

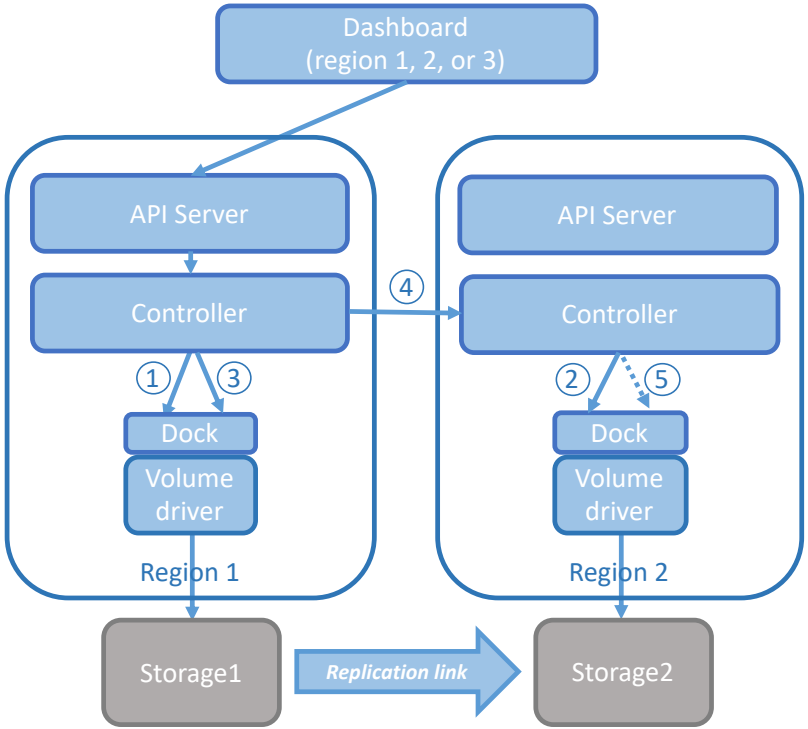


1. Controller asks driver to create a volume.
2. Driver creates a volume on the storage backend.
3. Controller periodically asks driver to create a snapshot based on policies defined in the Snapshot Profile.
4. Driver creates a snapshot on the storage backend.
5. Driver uploads the snapshot to an object store on premise or in the cloud based on the snapshot profile.

Disaster Recovery for Persistent Volumes

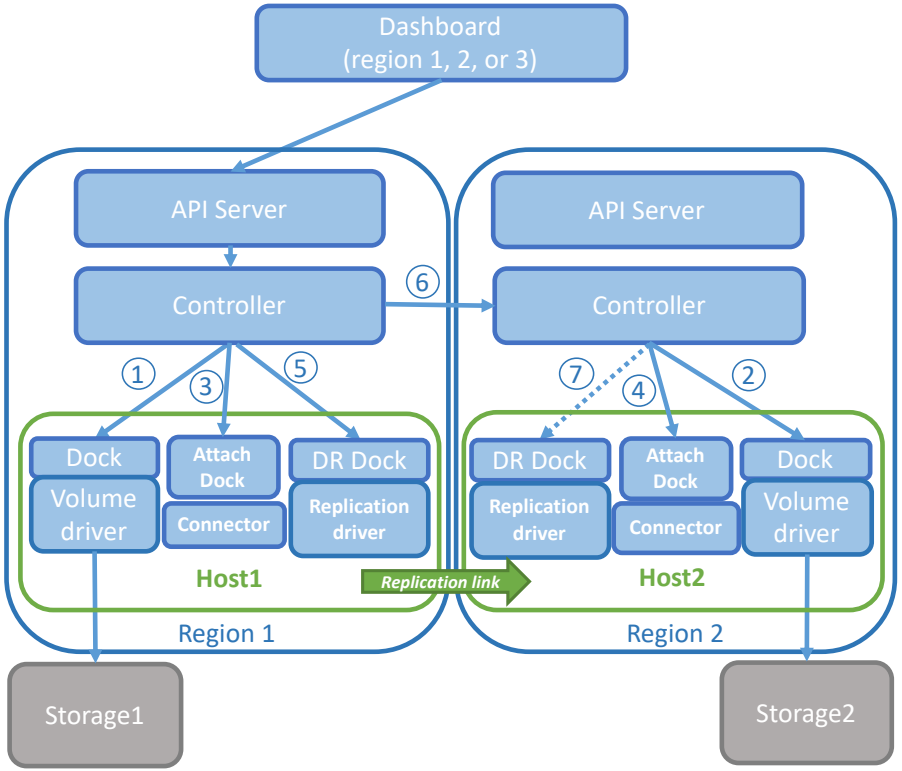


Array-based Replication



1. Creates source volume
   - Creates entry in db
   - Creates volume on Storage1.
2. Creates target volume
   - Creates entry in db
   - Creates volume on Storage2
3. Creates source replication
   - Creates entry in db
   - Creates replication relationship on Storage1 and Storage2
4. Controller 1 communicates with controller 2 to create target replication
5. Controller 2 creates entry in db

Host-based Replication



1. Creates source volume
   - Creates entry in db
   - Creates volume on Storage1
2. Creates target volume
   - Creates entry in db
   - Creates volume on Storage2
3. Attach source volume to Host1
   - Update volume entry in db with host info
4. Attach target volume to Host2
   - Update volume entry in db with host info
5. Controller 1 Creates source replication
   - Creates entry in db
   - Creates replication relationship on Host1 and Host2 (Host1 is primary)
6. Controller 1 communicates with controller 2 to create target replication
7. Controller 2 creates entry for target replication in db

Replication Functionalities

Create Replication:
osdsctl replication create <primary volume id> <secondary volume id> [flags]

Flags:

-d, --description string	the description of created replication
-h, --help	help for create
-n, --name string	the name of created replication
-p, --primary_driver_data string	the primary replication driver data of created replication
-m, --replication_mode string	the replication mode of created replication, value can be sync/async
-t, --replication_period int	the replication period of created replication, the value must be greater than 0
-s, --secondary_driver_data string	the secondary replication driver data of created replication

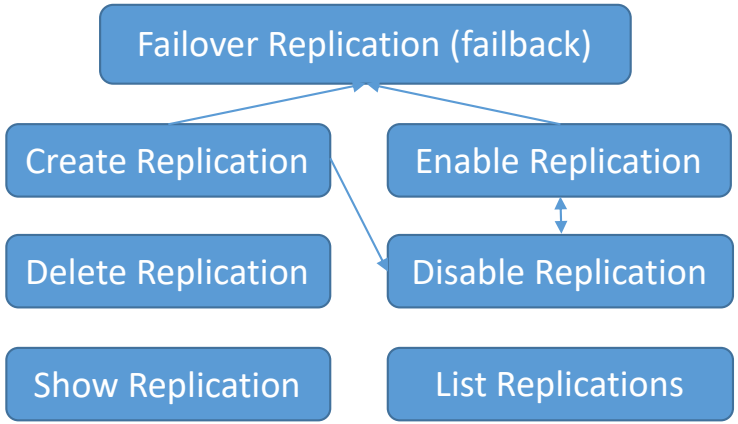
Enable Replication:
osdsctl replication enable <replication id>

Disable Replication:
osdsctl replication disable <replication id>

Failover Replication:
osdsctl replication failover <replication id> [flags]

Flags:

-a, --allow_attached_volume	whether allow attached volume when failing over replication
-h, --help	help for failover
-s, --secondary_backend_id string	the secondary backend id of failover replication



Demo

- Provision storage using OpenSDS CSI plugin based on profile definition
- Create tenant, create profile, create volume and snapshot via UI
- Array-based replication using Dorado
- Note: Host-based replication using DRBD is WIP

Future Development

- Array-based and Host-based Migration
- Multi-Cloud Storage
- Multi-OpenStack

Join Us

- Repos: https://github.com/opensds
- Slack: opensds.slack.com
- Mailing list: opensds-tech-discuss@lists.opensds.io

Thank You

@opensds_io

OpenSDS mini summit COPENHAGEN