Container Storage Interface Addons: Extending CSI specification to provide advanced storage operations

Rakshith R
Software Engineer @ IBM
Maintainer @ CephCSI & CSI-Addons
Core Contributor @ Rook

https://github.com/Rakshith-R
https://in.linkedin.com/in/rakshith-r
Agenda

- Container & Container Orchestration
- In-Tree Storage Drivers
- CSI
- CSI Deployment
- CSI-Addons
- CSI-Addons Deployment
- CSI-Addons: Reclaim Space Operation
- CSI-Addons: Network Fence Operation
- CSI-Addons: Volume Replication Operation
- Future Roadmap
- References
Container & Container Orchestration

**Container:**
- A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.
- They are portable, lightweight, secure and widely used.

**Container Orchestration:**
- Container orchestration automates the deployment, management, scaling, and networking of containers.
- Popular Container Orchestration Platforms:
  - Kubernetes
  - Docker Swarm
  - Apache Mesos
  - Nomad
In-Tree Storage Drivers

- Container Orchestrators realised the need for persistent storage for stateful applications.
- **Storage Drivers** were used to provide access to persistent storage for containers.
- These **Storage Drivers** were “in-tree”, they were part of each CO’s codebase and shipped with the core CO’s binaries.
- This had a lot of disadvantages:
  - **Storage Vendors (SV)** had to write different volume plugins for each CO.
  - **SVs** were forced to align with the CO’s release process even for bug fixes.
  - Third-party storage code caused reliability and security issues in core CO’s binaries.
  - It was often difficult (and in some cases impossible) for CO’s maintainers to test and maintain plugin code.
**CSI**

- **Container Storage Interface** (CSI) was proposed as a solution to problems faced by in-tree volume plugins.
- **CSI Specification** defines APIs (RPCs) to enable:
  - Dynamic provisioning and deprovisioning of a volume.
  - Mounting/unmounting a volume from a node.
  - Creating and deleting a snapshot.
  - Provisioning a new volume from a snapshot.
- **SVs** now had to develop only a single **CSI Driver** and it would work across a number of container orchestration (CO) systems.
CSI Deployment

A **CSI Driver** Consists of :

- **Provisioner Deployment** :
  - For Volume/Snapshot Creation, Expansion and Deletion.
  - It contains CSI Driver, external provisioner, external snapshotter, external attacher and external resizer containers.
  - Usually deployed with count two and leader election enabled for HA.

- **Nodeplugin Daemonset** :
  - For Volume Mounting and Unmounting.
  - It contains CSI Driver and node driver registrar containers.
  - Deployed one per node.
CSI Deployment

Node(s)
- Pod
- CSI driver Nodeplugin (DaemonSet)
  - driver-registrar
  - csi-plugin
- Kubelet

Node
- Pod
- CSI driver Provisioner (Deployment)
  - csi-provisioner
  - csi-resizer
  - csi-attacher
  - csi-snapshotter
- csi-plugin
- Kubelet

Kubernetes Services
- API Server
- Controller Manager
- Snapshot Controller

Ceph
CSI-Addons hosts the extensions to the CSI specification that provides advanced storage operations.

Various components involved:

- **CSI-Addons Specification**
  - Defines APIs (RPCs) to provide:
    - Identity service
    - Reclaim Space service
    - Network Fence service
    - Volume Replication service

- **CSI-Addons Controller**
  - Watches and responds to Custom Resources.
  - Connects to Sidecar and sends operation requests.

- **CSI-Addons Sidecar**
  - Advertises its presence to the controller.
  - Relays requests from Controller to the CSI Driver.
CSI-Addons: Reclaim Space Operation

- **Reclaim Space** operation executes
  - **ONLINE operation**: This operation is run from **CSI Driver** on the node where the volume is mounted. For example: `\textsf{fstrim}` on filesystem mode volumes.
  - **OFFLINE operation**: This operation is forwarded to the leader CSI Driver which supports it and can be executed regardless of volume is mounted or not. For example: `\textsf{rbd sparsify}` run on a rbd volume.
- This enables storage admins to have accurate view of storage consumption in a cluster.
CSI-Addons: Network Fence Operations

- **Network Fence** operation provides an API for blocking a list of given CIDR IP ranges.
- This plays a critical role in **Metro Disaster Recovery** and **Node-Loss** scenarios.
CSI-Addons: Volume Replication Operation

Volume Replication Operation

- **Volume Replication** operation provides common and reusable APIs for storage disaster recovery.
- It allows enabling/disabling mirroring and changing state (primary/secondary) of rbd mirrored images.
- The volume replication operation automates rbd-mirroring, allowing **promote**, **demote**, **resync** and get volume replication information operations on rbd images.
- This plays a critical role in **Regional Disaster Recovery**.
CSI-Addons: Future Roadmap

- Rotation of Key Encryption Keys (KEKs) for encrypted volumes.
- Volume Group Replication.
- Repairing corrupted Filesystem.
References

- Container Storage Interface (CSI) Specification.
- CSI-Addons · GitHub
- GitHub - csi-addons/spec: Storage Provider extensions to the CSI Specification
- CSI-Addons implementation and APIs for Kubernetes
- CSI driver for Ceph
- GitHub - rook/rook: Storage Orchestration for Kubernetes