CSI-based storage Orchestration System

Dennis.Chen@arm.com
Staff Software Engineer

March 20, 2018
Linaro Connect HKG18
Agenda

• Background
• What’s CSI
• FlexVolume vs CSI
• How CSI works
• CSI Usage Workflow
• CSI Sample Driver
Background

1. In-tree Volume Plugins
   - Those are linked, compiled, built and shipped with the core k8s binaries
   - Development is tightly coupled and dependent on k8s releases
   - Bugs in volume plugin can crash critical k8s components, instead of just the plugin
   - Will not be accepted in the future

2. Out-of-Tree Volume Plugins (customized plugins by storage vendors)
   - FlexVolume driver
   - CSI driver (*)
What’s CSI

• Container Storage Interface (CSI) is a standardized mechanism for Container Orchestration Systems (COs), including Kubernetes, to expose arbitrary storage systems to containerized workloads. Storage Provider (SP) develops once and this works across a number of COs.

• The goal of CSI is to become the primary volume plugin system for k8s in the future.

• k8s 1.9 release has already included the alpha feature of CSI implementation.

• The CSI spec can be found at:

  https://github.com/container-storage-interface/spec/blob/master/spec.md
CSI vs FlexVolume

Two mainstream Volume Plugin mechanisms in K8s – FlexVolume and CSI

1. FlexVolume plugin framework:
   - Makes the 3rd party storage providers’ plugin as “out-of-tree” (same as CSI does)
   - exec based API for external volume plugins
   - Needs to access the root filesystem of node and master machines when deploying
   - Doesn’t address the pain point of dependencies.

2. CSI overcomes the limitations of FlexVolume listed above. CSI is the preferred solution, for now CSI and FlexVolume can co-exist.
How CSI works

- A new in-tree CSI Volume plugin (K8s) + out-of-tree CSI Volume driver (3\textsuperscript{rd} party)
- Communication channel via a Unix Domain Socket (UDS) created by 3\textsuperscript{rd} Volume Driver

The socket file also called a ‘EndPoint’ in form of like: /var/lib/kubelet/plugins/rook-ceph/csi.sock
How CSI works

Recommended Mechanism for Deploying CSI Drivers on k8s
How CSI works

1. Communication between k8s and the CSI driver – The ‘proxy’ containers by k8s team.

   Proxy/Sidecar container that watches k8s VolumeAttachment objects and triggers ControllerPublish/UnpublishVolume against a CSI endpoint to call CSI driver

   • external-provisioner: Sidecar container that monitors PersistentVolumeClaim objects and triggers CreateVolume/DeleteVolume against the 3rd CSI volume driver.

   • external-attacher: Sidecar container that monitors VolumeAttachment objects and triggers ControllerPublish/Unpublish against the 3rd CSI volume driver.

   • driver-registrar: The container that 1) registers the CSI driver with kubelet and 2) adds the drivers custom NodeId to a label on the k8s Node API object.

UDS is used for communication between Sidecar Container and 3rd party CSI Driver.
How CSI works

2. Codes repositories as the specific examples:

   • external-provisioner
     The codes hosted by [https://github.com/kubernetes-csi/external-provisioner](https://github.com/kubernetes-csi/external-provisioner)

   • external-attacher
     The codes hosted by [https://github.com/kubernetes-csi/external-attacher](https://github.com/kubernetes-csi/external-attacher)

   • driver-registrar
     The codes hosted by [https://github.com/kubernetes-csi/driver-registrar](https://github.com/kubernetes-csi/driver-registrar)
CSI Usage Workflow

1. Provisioning & Deleting

A cluster admin creates a StorageClass with the name of the external-provisioner (1:1).

An end user creates a PVC object referencing this StorageClass, triggering the provisioner calls CreateVolume() of CSI driver.

---

**Storage Class Setup (By Administrator)**

- kind: StorageClass
- apiVersion: storage.k8s.io/v1beta1
- metadata:
  - name: ext-storage
- provisioner: com.example.team/csi-provisioner
- parameters:
  - type: pd-ssd

**PersistentVolumeClaim Creation (By User)**

- apiVersion: v1
- kind: PersistentVolumeClaim
- metadata:
  - name: request-for-storage
- spec:
  - resources:
    - requests:
      - storage: 5Gi
  storageClassName: ext-storage
CSI Usage Workflow

2. Attaching & Detaching

*VolumeAttachment* captures the intent to attach or detach the specified volume to/from the specified node.

An external “attacher” watch the k8s API on behalf of the external CSI volume driver to handle attach/detach requests:

- A new k8s *VolumeAttachment* API object is created by k8s attach/detach controller
- external-attacher watches and calles *ControllerPublishVolume* against the CSI volume driver to attach the volume to the specified node.
CSI Sample Driver

- [https://github.com/kubernetes-csi/drivers](https://github.com/kubernetes-csi/drivers)
  The drivers here are provided purely for illustrative purpose, and should not be used for production workloads.

- The real CSI drivers are to be housed on their own repo, like:
  [https://github.com/ceph/ceph-csi](https://github.com/ceph/ceph-csi)

Source: Alvaro Miranda
Thank You!
Danke!
Merci!
谢谢!
ありがとうございます!
Gracias!
Kiitos!