Easy multi-tenant Kubernetes RWX storage with Cloud Provider OpenStack and Manila CSI

Tom Barron
tbarron@redhat.com

Victoria Martinez de la Cruz
victoria@redhat.com

Red Hat

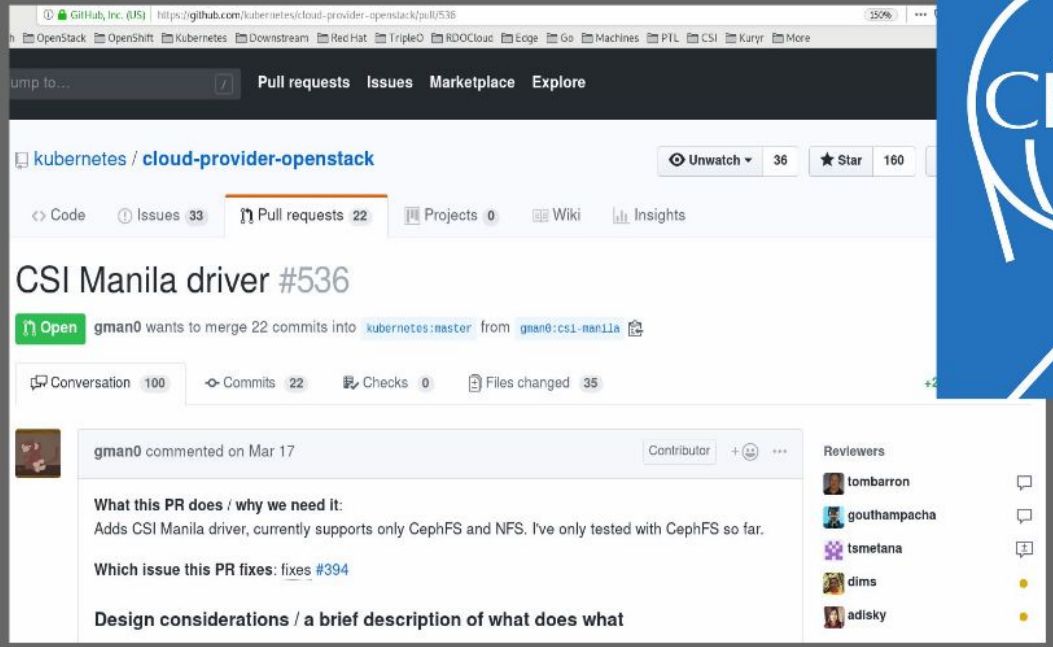
Game plan

- What is Manila CSI?
- Why RWX storage for Kubernetes with Manila CSI
- How to *deploy* Manila CSI
  - One time task for Kubernetes operators (or for Operators) (demo!)
- How to *use* Manila CSI
  - Day to day PVC and pod deployment by application developers (demo!)
- Summary and resources

What is the Manila CSI plugin?

- External, dynamic provisioner plugin for persistent Kubernetes volumes served up via OpenStack Manila
- Conforms to the new Container Storage Interface standard
- Code lives in the Kubernetes Cloud Provider Openstack repository

Manila-CSI



https://github.com/kubernetes/cloud-provider-openstack/pull/536 author: Robert Vasek

The author, Robert Vašek, initial work at CERN

He recently completed a GSOC project under Red Hat sponsorship to add snapshot capabilities to Manila CSI.

RWX Storage for Container Orchestrators with CephFS and Manila - slide 49

Why use a ***Cloud Provider*** *OpenStack* plugin?

- Why Cloud Provider Openstack rather than vendor-specific or backend-specific plugins?
- No lock in -- abstraction layer over multiple back ends
  - Manila supports ~35 storage back ends
- Keystone-based *hard* multi-tenant separation for multiple K8s clusters with independent ownership
  - Enables dynamic, elastic sharing of enterprise or public-cloud scale storage resources by multiple K8s clusters
  - OpenStack is IAAS, multiple CAAS clusters are IAAS customers
  - CAAS customers (applications developers/devops) don’t need to know anything about OpenStack

Why use the ***Manila*** plugin?

- There's is a perfectly good Cinder-CSI plugin.
- But the Cinder plugin offers only RWO file mode access, not RWX.
- Kubernetes makes it easy to scale out containerized compute via *pods* but provisioning consistent persistent storage for replicated pods is tricky.*
- RWX PVCs pointing to Storage Classes from Manila CSI can enable safe multi-writer pod deployments with familiar, straightforward application design.

* See Kubernetes Storage 101, David Zhu and Jan Šafránek, especially slides 45ff.

Why use a **CSI** plugin?

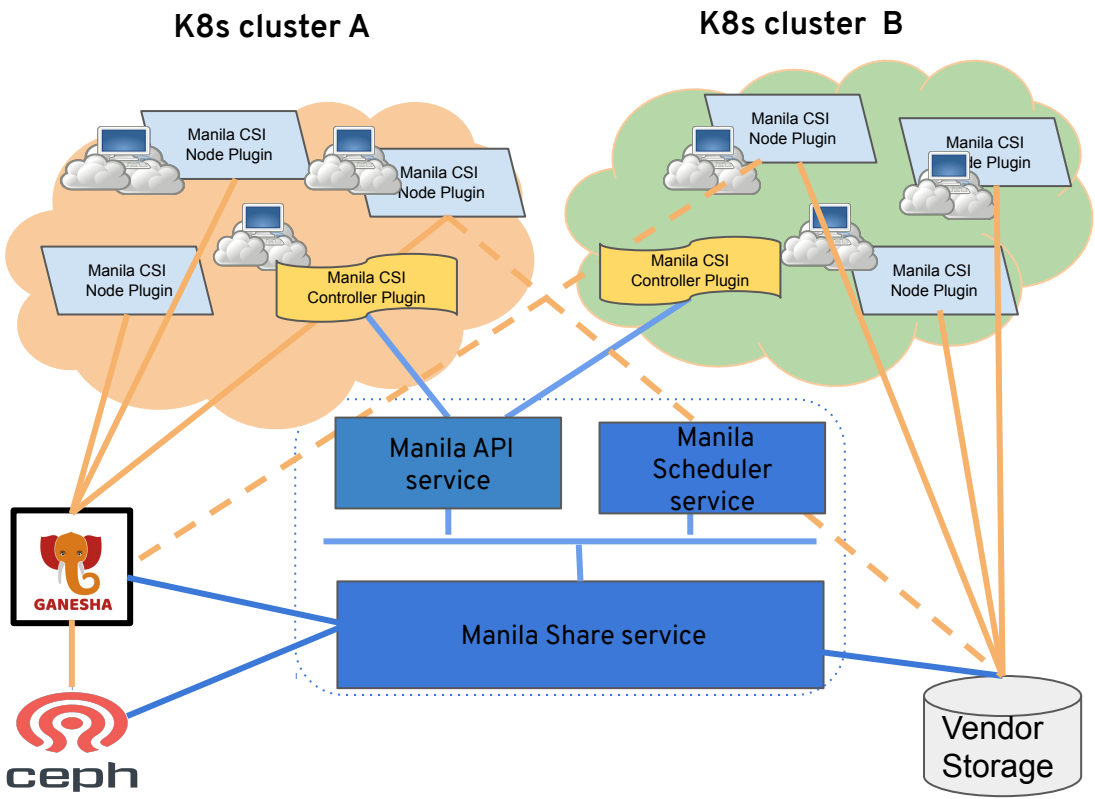
- There's a nice Manila provisioner already in cloud provider openstack repository
  - It's already external to the K8s codebase so can be changed on its own life cycle, doesn't impact K8s core security, etc. (faster bug fixes and features)
  - It already can support both static and dynamic provisioning
- CSI is a standard interface for K8s, docker, Mesos, and other COs
  - But maybe you just care about K8s :)
- Bottom line: this is where the new development is happening
  - New features and developer/testing attention are focused on the CSI plugins rather than the non-CSI external provisioner plugins.

OpenStack Manila CSI for Kubernetes

- K8s nodes are VMs or Bare Metal
- OpenStack Admin is the Storage Admin's customer (can be same individuals of course)
- K8s Admins are separate OpenStack customers (separate tenants – each with their own OpenStack user privileges)
- K8s users are customers of the K8s Admin. Users don't need to know anything about Manila or OpenStack

Control Path (PVCs and Manila CRUD)

Data Path (mount PVs)



Deploying Manila CSI

One time task for Kubernetes Administrators

Manifests

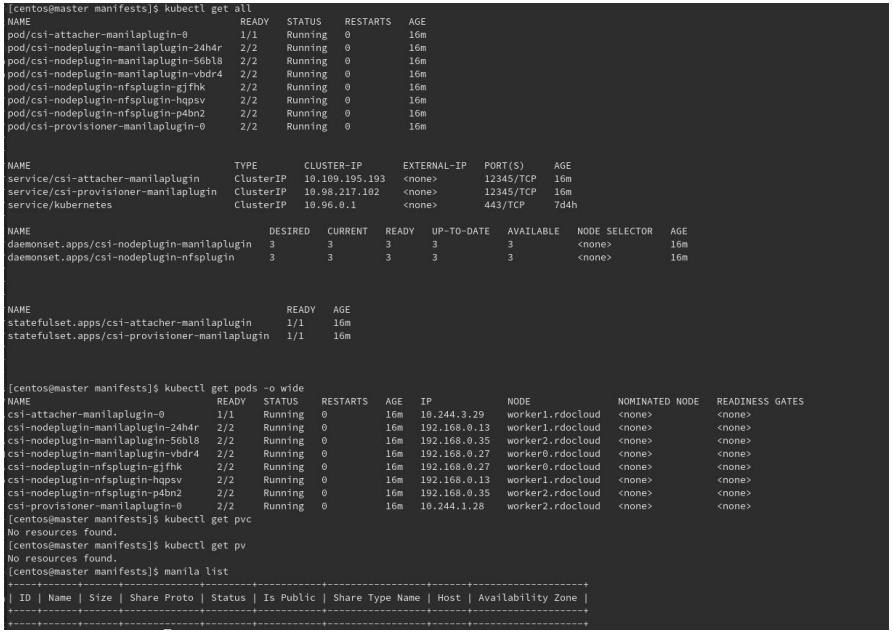
```
$ tree admin-manifests
admin-manifests
├── 00-nfscsi-nodeplugin                ← protocol partner node plugin
│   ├── 00-rbac.yaml
│   └── 11-daemonset.yaml
├── 11-manilacsi-nodeplugin             ← defines forwarding to partner node plugin
│   ├── 00-rbac.yaml
│   └── 11-daemonset.yaml
├── 22-manilacsi-attacher               ← essentially a no-op for manila-csi
│   ├── 00-rbac.yaml
│   └── 11-stateful-set.yaml
├── 33-manilacsi-provisioner            ← fulfills PVCs via Manila API
│   ├── 00-rbac.yaml
│   └── 11-stateful-set.yaml
├── 44-secrets                          ← OpenStack  user credentials
│   └── 00-secrets.yaml                    for the K8s admin
└── 55-storage-class                    ← Used by PVCs to select the
    └── 00-storage-class.yaml              dynamic external provisioner
```

Admin Manila CSI Deployment

Setting up Manila CSI in the K8s cluster
(follow link for demo)

The manifests used in the demo are available here.

- One time setup by K8s administrator
- Can use the helm chart now provided in the cloud provider openstack repo instead
- In our downstream OCP product we'll make an Operator to do this as well as manage day2, etc.
- So this will be even easier than what we are demoing here



Plugins running post CSI deployment, no storage provisioned

Red Hat

Using Manila CSI

Using Manila CSI

Application developers can dynamically provision RWX storage and deploy pods with applications that safely consume it using yaml manifests that are themselves completely decoupled from Manila and from its CSI plugin.

- Use the same pod and pvc definitions on premises that you use with OpenShift on AWS, GCP, Azure, etc *except for the storage class reference in the PVC*

Simple Multi-Writer scenario

```
$ cat 00-writer-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: writer-one
spec:
  restartPolicy: Never
  containers:
   - image: gcr.io/google_containers/busybox
        command:
        - "/bin/sh"
        - "-c"
        - "while true; do echo $(date) >> /mnt/test/$(hostname);
sleep 10; done"
        name: busybox
        volumeMounts:
          - name: mypvc
            mountPath: /mnt/test
  Volumes:
   - name: mypvc
     persistentVolumeClaim:
       claimName: myclaim
       readOnly: false
```

```
$ diff 00-writer-pod.yaml 11-writer-pod.yaml
4c4
<   name: writer-one
---
>   name: writer-two
```

- 00-writer and 11-writer differ only in their names
- They mount the same volume via mypvc at /mnt/test
- They write to different files at /mnt/test/$hostname
- The name of the PVC used

PVC definition

```
$ cat rwx-persistent-volume-claim.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
        - ReadWriteMany
  resources:
        requests:
        storage: 10Gi
  storageClassName: csi-manila-nfs
```

- K8s administrator created this storage class - csi-manila-nfs
  - End user doesn't need to know anything about Manila CSI, just needs to refer to this Storage class
- Pod definitions refer to this name to use this PVC
- Use RWX so that the PV that fulfills this PVC will can be mounted to multiple pods on multiple nodes in the cluster

End user deploys multi-writer application with RWX storage

```
[centos@master user-manifests]$ kubectl exec writer-one mount | grep '/mnt/test'
192.168.0.27:/volumes/_nogroup/a2fcd804-8535-4e64-89c4-be72d6c7f87e on /mnt/test type nfs4 (rw,relatim
one,addr=192.168.0.27)
[centos@master user-manifests]$ kubectl exec writer-one -- ls -R /mnt
/mnt:
test

/mnt/test:
writer-one
writer-two
[centos@master user-manifests]$ kubectl exec writer-one -- tail /mnt/test/writer-two
Wed Jul 3 21:39:10 UTC 2019
Wed Jul 3 21:39:20 UTC 2019
Wed Jul 3 21:39:30 UTC 2019
Wed Jul 3 21:39:40 UTC 2019
Wed Jul 3 21:39:50 UTC 2019
Wed Jul 3 21:40:00 UTC 2019
Wed Jul 3 21:40:10 UTC 2019
Wed Jul 3 21:40:20 UTC 2019
Wed Jul 3 21:40:30 UTC 2019
Wed Jul 3 21:40:40 UTC 2019
[centos@master user-manifests]$ kubectl exec writer-two mount | grep '/mnt/test'
192.168.0.27:/volumes/_nogroup/a2fcd804-8535-4e64-89c4-be72d6c7f87e on /mnt/test type nfs4 (rw,relatim
one,addr=192.168.0.27)
[centos@master user-manifests]$ kubectl exec writer-two -- tail /mnt/test/writer-one
Wed Jul 3 21:40:01 UTC 2019
Wed Jul 3 21:40:11 UTC 2019
Wed Jul 3 21:40:21 UTC 2019
Wed Jul 3 21:40:31 UTC 2019
Wed Jul 3 21:40:41 UTC 2019
Wed Jul 3 21:40:51 UTC 2019
Wed Jul 3 21:41:01 UTC 2019
Wed Jul 3 21:41:11 UTC 2019
Wed Jul 3 21:41:21 UTC 2019
Wed Jul 3 21:41:31 UTC 2019
[centos@master user-manifests]$
```

Writer-one sees what writer two is writing and vice versa.

Easy end-user multi-writer deployment to RWX volume (follow link for demo)

The manifests used in the demo are available here.

Manila CSI supports RWO mode too

```
$ cat rwx-persistent-volume-claim.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
        - ReadWriteMany
  resources:
        requests:
        storage: 10Gi
  storageClassName: csi-manila-nfs
```

```
$ cat rwo-persistent-volume-claim.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
        - ReadWriteOnce
  resources:
        requests:
        storage: 10Gi
  storageClassName: csi-manila-nfs
```

Just change the *accessMode* in the PVC manifest

Same applications with RWO PVC

multi-writer deployment with RWO PVC (follow link for demo)

The manifests used in the demo are available here.

```
Events:
  Type     Reason              Age   From                     Message
  ----     ------              ----  ----                     -------
  Normal   Scheduled           23s   default-scheduler        Successfully assigned default/writer-two to worker0.rdocloud
  Warning  FailedAttachVolume  23s   attachdetach-controller  Multi-Attach error for volume "pvc-dc9b62cd-c316-4557-8d08-8e88cf82e96b" Volume is already use
d by pod(s) writer-one
[centos@master user-manifests]$ ## writer two cannot mount!
```

Second pod gets stuck and cannot come up -- as it should since RWO mode is being enforced.

Features and Futures

- Share Expand and Shrink
- HA improvements (daemon set for controller with leader election)
- Create volume from snapshot compatibility layer
  - When Manila back ends can’t do this themselves
- Complete OpenLab CI
- Improve concurrency for long-running tasks (like CephFS create from volume)
- Integrated handler for multiple share protocols?
- Topology awareness (AZs)

Summary, Resources and Q&A

- Cloud provider openstack code repository (includes manila-csi plugin)
- Kubernetes Storage 101, David Zhu and Jan Šafránek, Kubecon Barcelona 2019.
- Manila-kube repository for deploying Kubernetes cluster on OpenStack with manila-csi
- RWX storage for container orchestrators with CephFS and Manila
- Manila CSI Manifests used in the demo
- GSOC snapshots project

Thank you!

Reach us out for Q&A:

tbarron@redhat.com

vkmc@redhat.com

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat