Volsync Dokuwiki MicroShift Demo

Infrastructure:
source, destination GCP instances

- e2-standard-2 (v2CPU, 8GB)
- CentOS8 Stream

local machine (fedora35)

Step 0: Deploy MicroShift on source and destination machines

```
# install and start CRI-O

sudo -i
curl -L -o /etc/yum.repos.d/devel:kubic:libcontainers:stable.repo
https://download.opensuse.org/repositories/devel:/kubic:/libcontainers:/stable/CentOS_8_Stream/devel:kubic:libcontainers:stable.repo

curl -L -o
/etc/yum.repos.d/devel:kubic:libcontainers:stable:cri-o:1.21.repo
https://download.opensuse.org/repositories/devel:kubic:libcontainers:stable:cri-o:1.21/CentOS_8_Stream/devel:kubic:libcontainers:stable:cri-o:1.21.repo

sudo dnf install -y cri-o cri-tools
sudo systemctl enable crio --now

# deploy MicroShift containerized

sudo dnf install -y podman
sudo curl -o /etc/systemd/system/microshift.service
https://raw.githubusercontent.com/redhat-et/microshift/main/packaging/systemd/microshift-containerized.service
sudo systemctl enable microshift --now

# cp kubeconfig to home directory for scp access from local machine

mkdir ~/.kube && sudo podman cp
microshift:/var/lib/microshift/resources/kubeadmin/kubeconfig
~/.kube/config

sudo chown -R $(whoami):$(whoami)~/.kube
```

Set up access to both clusters from local machine

```
# install oc, kubectl (either or both) on local machine if necessary

curl -O
https://mirror.openshift.com/pub/openshift-v4/x86_64/clients/ocp/stable/openshift-client-linux.tar.gz

sudo tar -xvf openshift-client-linux.tar.gz -C /usr/local/bin oc kubectl

# access :6443 of source machine from local machine
gcloud beta compute ssh --zone "zone" "source-machine-name" -- -L
8888:127.0.0.1:6443

# switch to local/new terminal
gcloud beta compute scp --zone "zone" --project “project”
gcpusername@source-machine-name:~/.kube/config ~/kubeconfig-source

# to locally access cluster at 127.0.0.1:8888
# (it’s really 127.0.0.1:6443 in gcp instance)
sed -i 's/6443/8888/g' ~/kubeconfig-source
# confirm you can access
oc --kubeconfig ~/kubeconfig-source get pods -A
oc --kubeconfig ~/kubeconfig-source new-project dokuwiki

# access :6443 of destination machine from local machine
gcloud beta compute ssh --zone "zone" "dest-machine-name" -- -L
9999:127.0.0.1:6443

# switch to local terminal
gcloud beta compute scp --zone "zone" --project “project”
gcpusername@dest-machine-name:~/.kube/config ~/kubeconfig-dest

# to locally access cluster at 127.0.0.1:9999
# (it’s really 127.0.0.1:6443 in gcp instance)
sed -i 's/6443/9999/g' ~/kubeconfig-dest
# confirm you can access
oc --kubeconfig ~/kubeconfig-dest get pods -A
oc --kubeconfig ~/kubeconfig-dest new-project dokuwiki

# Merge kubeconfigs and rename contexts (for convenience)
export KUBECONFIG=~/kubeconfig-source:~/kubeconfig-dest
oc config rename-context dokuwiki/127-0-0-1:9999/system:masters destination
oc config rename-context dokuwiki/127-0-0-1:8888/system:masters source
```

```
oc config get-contexts
  * should have destination & source and can now easily switch between
```

You can now work from your local machine (port-forwarding from GCP)

Configure Storage on source and destination machines

(https://github.com/kubernetes-csi/external-snapshotter)

```
SNAPSHOTTER_VERSION=v4.2.1
```

Apply VolumeSnapshot CRDs

```
# oc config use-context source, execute red, then oc config use-context
destination, repeat red

oc config use-context <source, destination>

oc apply -f
https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNA
PSHOTTER_VERSION}/client/config/crd/snapshot.storage.k8s.io_volumesnapshotc
lasses.yaml

oc apply -f
https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNA
PSHOTTER_VERSION}/client/config/crd/snapshot.storage.k8s.io_volumesnapshotc
ontents.yaml

oc apply -f
https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNA
PSHOTTER_VERSION}/client/config/crd/snapshot.storage.k8s.io_volumesnapshots
.yaml
```

Create snapshot controller

```
oc apply -f
https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNA
PSHOTTER_VERSION}/deploy/kubernetes/snapshot-controller/rbac-snapshot-contr
oller.yaml

oc apply -f
https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/${SNA
PSHOTTER_VERSION}/deploy/kubernetes/snapshot-controller/setup-snapshot-cont
roller.yaml
```

Deploy hostpath driver, configure storageclass

```
git clone https://github.com/kubernetes-csi/csi-driver-host-path.git &&
cd csi-driver-host-path

# oc config use-context source, execute red, then oc config use-context
destination, repeat red

oc config use-context <source, destination>

./deploy/kubernetes-latest/deploy.sh

for i in ./examples/csi-storageclass.yaml ./examples/csi-pvc.yaml
./examples/csi-app.yaml; do oc apply -f $i; done

oc annotate sc/kubevirt-hostpath-provisioner
storageclass.kubernetes.io/is-default-class="false" --overwrite

oc annotate sc/csi-hostpath-sc
storageclass.kubernetes.io/is-default-class="true" --overwrite

oc annotate volumesnapshotclass/csi-hostpath-snapclass
snapshot.storage.kubernetes.io/is-default-class="true"
```

Example from scribe-poc (volsync=scribe) repository

https://github.com/sallyom/scribe-poc/tree/microshift-dokuwiki (microshift-dokuwiki branch!)

Deploy Dokuwiki to source and destination

```
oc --context source apply -f
https://raw.githubusercontent.com/sallyom/scribe-poc/microshift-dokuwiki/mi
croshift-dokuwiki/dokuwiki-crb-dep-sa.yaml -n dokuwiki

oc --context destination apply -f
https://raw.githubusercontent.com/sallyom/scribe-poc/microshift-dokuwiki/mi
croshift-dokuwiki/dokuwiki-crb-dep-sa.yaml -n dokuwiki
```

Deploy Volsync Operator to both machines

```
# see volsync documentation
https://volsync.readthedocs.io/en/stable/installation/index.html

# Not sure how helm uses config/contexts, so I export KUBECONFIG for each
```

```
# machine, then after, merge the kubeconfigs again
# (do for source, destination machines)

export KUBECONFIG=~/kubeconfig-source
helm repo add backube https://backube.github.io/helm-charts/
helm install --create-namespace -n volsync-system volsync backube/volsync
oc -n volsync-system get pods
# then repeat with KUBECONFIG=~/kubeconfig-dest
export KUBECONFIG=~/kubeconfig-source:~/kubeconfig-dest
```

Replication Source

```
# see rclone.conf example here and create a file ~/rclone.conf:
https://volsync.readthedocs.io/en/stable/usage/rclone/rclone-secret.html#what-is-rclone-secret

oc --context source create secret generic rclone-secret
--from-file=rclone.conf=/path/to/rclone.conf

oc –context source apply -f
https://raw.githubusercontent.com/sallyom/scribe-poc/microshift-dokuwiki/microshift-dokuwiki/rclone-deployment/source-rclone/replicationsource.yaml -n
dokuwiki
```

Replication destination

```
oc config use-context destination

oc create secret generic rclone-secret -n dokuwiki
--from-file=rclone.conf=/path/to/rclone.conf -n dokuwiki

oc apply -f
https://raw.githubusercontent.com/sallyom/scribe-poc/microshift-dokuwiki/microshift-dokuwiki/rclone-deployment/destination-rclone/replicationdestination.yaml -n dokuwiki
```

Update wiki and sync the source and destination

```
# Edit source dokuwiki at http://external-ip-of-source-machine:30007
```

```
# Scale destination dokuwiki to 0 replicas

oc config use-context destination

oc scale deployment/dokuwiki-deployment --replicas=0 -n dokuwiki

IMAGE=$(oc get replicationdestination dokuwiki-destination -n dokuwiki
--template={{.status.latestImage.name}})

oc delete pvc dokuwiki-pvc

curl -o sync-pvc.yaml
https://raw.githubusercontent.com/sallyom/scribe-poc/microshift-dokuwiki/microshift-dokuwiki/rclone-deployment/destination-rclone/sync-pvc.yaml

cp sync-pvc.yaml sync-pvc-new.yaml && sed -i
's/snapshotToReplace/'"${IMAGE}"'/g' sync-pvc-new.yaml

oc apply -f sync-pvc-new.yaml

oc scale deployment/dokuwiki-deployment --replicas=1 -n dokuwiki
```