K8s lab part 2– Setting up the Unity CSI driver v1.3

In this part we will add a CSI driver to our K8s cluster. We assume you already have a Unity VSA running. This guide is based on the Unity CSI driver install guide found at <https://github.com/dell/csi-unity> . You can also read through the official install guide called *CSI Driver for Dell EMC Unity Product Guide.pdf* in the GitHub repository.

Steps:

1. Install the iSCSI initiator and native multipath driver:

yum install -y iscsi-initiator-utils

yum install -y device-mapper-multipath

systemctl enable iscsi

1. Connect to the Unity VSA’s ISCSI target. Check from Unisphere under Block…iSCSI interfaces that an interface is present. Create one if not! Next use that IP address here:

iscsiadm -m discovery -t st -p 192.168.192.211

1. For NFS to work we need to get and start the rpcbind service:

yum install -y rpcbind

systemctl start rpcbind

systemctl enable rpcbind

1. Create the unity namespace on the controller node where the CSI container can live:

kubectl create namespace unity

1. Install Helm3 on the controller node:

curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash

1. Get the certificate from the Unity VSA:

openssl s\_client -showcerts -connect 192.168.192.210:443 </dev/null 2>/dev/null | openssl x509 -outform PEM > ca\_cert\_0.pem

kubectl create secret generic unity-certs-0 --from-file=cert-0=ca\_cert\_0.pem -n unity

1. Download the CSI driver:

sudo yum install -y git

cd ~

git clone <https://github.com/dell/csi-unity>

1. Configure the values for your specific Unity-VSA:

cp ~/csi-unity/helm/csi-unity/values.yaml ~/csi-unity/dell-csi-helm-installer/myvalues.yaml

cd ~/csi-unity/dell-csi-helm-installer

nano myvalues.yaml

* Scroll down and uncheck the storagaArrayList and fill in the name of your Unity VSA (VIRTxxxx)
* Set “isDefaultArray” to “true”
* NOTE: The pool ID and the NAS server need the “CLI ID”. Make sure you use those (probably “pool\_1” and “nas\_1”)

nano secret.json

* Fill while changing ip, credentials and arrayID:

{

"storageArrayList": [

{

"username": "admin",

"password": "Password",

"restGateway": "https://192.168.x.x",

"arrayId": "VIRTxxxxxxxxxx",

"insecure": true,

"isDefaultArray": true

}

]

}

kubectl create secret generic unity-creds -n unity --from-file=config=secret.json

In case you need to recreate, REPLACE the secret with this command:

kubectl create secret generic unity-creds -n unity --from-file=config=secret.json -o yaml --dry-run=client | kubectl replace -f -

1. Install the Snapshot provider (note: it is expected to get warnings as you install these)

kubectl apply -f <https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-2.0/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml>

kubectl apply -f <https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-2.0/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml>

kubectl apply -f <https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-2.0/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml>

kubectl apply -f <https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/master/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml>

kubectl apply -f <https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/master/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml>

* Check that you have a pod “snapshot-controller-0” running

kubectl get pods

1. Install the CSI driver!

./csi-install.sh --namespace unity --values ./myvalues.yaml

1. Check that the CSI driver is up and running

kubectl -n unity get pods

(You should get a pod unity-controller-0 with 5 containers running and one unity-node-xxxxx node running 2 containers each)

If for some reason the deployment was unsuccessful, start by looking here:

kubectl describe pod unity-controller-0

kubectl -n unity logs unity-controller-0 driver

**SIDENOTES**

* In case a PVC gets “stuck” and won’t deprovision in K8s use this to “shoehorn” it to destruction:

kubectl patch pvc <PVC\_NAME> -p '{"metadata":{"finalizers": []}}' --type=merge

* Note2.