NETWAYS WEB SERVICES

StorageClasses

Erstellt von: Justin Lamp

Erstellt am: 19 April 2024 15:23:58

Version: 7

Inhaltsverzeichnis

StorageClasses

Our Managed Kubernetes clusters come with a couple of predefined storage classes. If there is no storage class that meets your needs, custom ones can be created. For more information take a look at the Cinder-CSI Driver documentation or feel free to contact our support team for assistance.

```
$ kubectl get sc
NAME                 PROVISIONER                RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION
encrypted            cinder.csi.openstack.org   Delete          Immediate              true
encrypted-high-iops  cinder.csi.openstack.org   Delete          Immediate              true
high-iops            cinder.csi.openstack.org   Delete          Immediate              true
nws-storage          cinder.csi.openstack.org   Delete          Immediate              true
standard (default)   cinder.csi.openstack.org   Delete          Immediate              true
```

standard (Default)

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: standard
allowVolumeExpansion: true
parameters:
  csi.storage.k8s.io/fstype: ext4
provisioner: cinder.csi.openstack.org
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

The StorageClass `standard` is the default class that is used when no StorageClass is specified.

It provisions an **ext4 formatted** Volume in OpenStack and deletes it if the referencing PersistentVolumeClaim (**PVC**) gets deleted. **VolumeExpansion** is also supported. It enables the user to update the size of a PVC and let Kubernetes handle the resize.

The **IOPS limit** is set to 1000 IOPS, but enables a boost of up to 2000 IOPS for 60s.

nws-storage

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nws-storage
allowVolumeExpansion: true
parameters:
  csi.storage.k8s.io/fstype: xfs
provisioner: cinder.csi.openstack.org
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

The `nws-storage` class is similar to `standard`, but uses **xfs** as it's filesystem.

This is useful for PersistentVolumeClaims with many small files, e.g. databases or logging systems, as xfs can scale inodes dynamically. This stands in contrast to ext4, which creates a fixed size of inodes at creation time.

high-iops

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: high-iops
parameters:
  csi.storage.k8s.io/fstype: ext4
  type: Ceph-High-IOPS
allowVolumeExpansion: true
provisioner: cinder.csi.openstack.org
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

The `high-iops` StorageClass uses a different volume type in OpenStack called `Ceph-High-IOPS`, which allows the system **up to 2000 IOPS** for sustained loads and 4000 IOPS for bursts (60s).

encrypted(-high-iops)

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: encrypted
parameters:
  csi.storage.k8s.io/fstype: ext4
  type: Ceph-Encrypted(-High-IOPS)
allowVolumeExpansion: true
provisioner: cinder.csi.openstack.org
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

The StorageClasses starting with `encrypted` use OpenStacks volume type that transparently enables encryption for the volume. The two classes differ in the same way `standard` and `high-iops` do.

Custom

You can of course also create your own StorageClass with any of the parameters and options available to the Cinder CSI Driver and Kubernetes.

For reference, the following custom StorageClass would use the `Ceph-Encrypted` Volume type with the XFS format while retaining the backing PersistentVolume after deleting the PersistentVolumeClaim:

```
$ kubectl apply -f - <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```
metadata:
  name: encrypted-xfs-retain
parameters:
  csi.storage.k8s.io/fstype: xfs
  type: Ceph-Encrypted
allowVolumeExpansion: true
provisioner: cinder.csi.openstack.org
reclaimPolicy: Retain
volumeBindingMode: Immediate
EOF
```

ReadWriteMany

Unfortunately, we currently don't provide Volumes with the `RWX` access type. This is on our roadmap, but we cannot commit to any timeframe at this point. For now, you will need to build your own solution, e.g. based on NFS or Rook Ceph. We would be happy to assist you with that.