Administrative Tasks

- Nodes and Nodegroups
- Cluster Autoscaling
- Backup
- Upgrades
- One Click Deployments
- StorageClasses

Nodes and Nodegroups

Resizing Nodegroups

Nodes are organized in nodegroups. If you want to create new nodes or delete existing ones you have to resize the nodegroup. The nodegroup defines which flavor will be used for new nodes.

To resize a nodegroup select the "Nodegroups" tab and click on the "Resize" button in the context menu. Note that it is not possible to resize the master nodegroup. Select the desired node count and finally click on "Resize Nodegroup".

Create Nodegroups

A cluster has two nodegroups by default. The master nodegroup and the default nodegroup. You can't delete these nodegroups. In the case of the default nodegroup it is possible to scale the node count to zero, though.

To create a nodegroup click on the "Create Nodegroup" button and customize the new nodegroup to your wishes.

Cluster Autoscaling

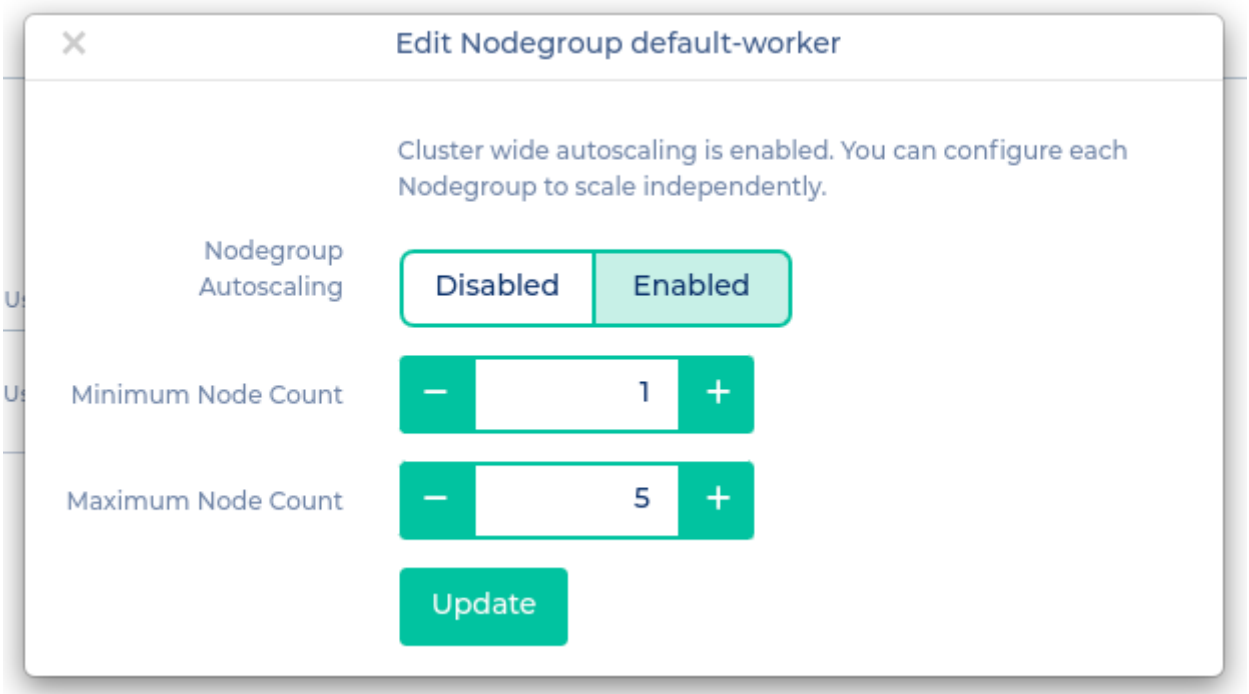
Here is how to enable Cluster Autoscaling:

First click "Enable Autoscaling" in the cluster's context menu. This will create the cluster-autoscaler deployment in your cluster. To ensure that everything went fine, run the following command:

```
$ kubectl get deployment -n kube-system cluster-autoscaler

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
cluster-autoscaler  1/1     1            1           12m
```

After successfully deploying the cluster-autoscaler you need to configure each nodegroup that should be autoscaled. Switch to the nodegroups tab and click on the "Edit" button of the nodegroup's context menu. Choose "Enabled" and select the desired mininum and maximum node count.



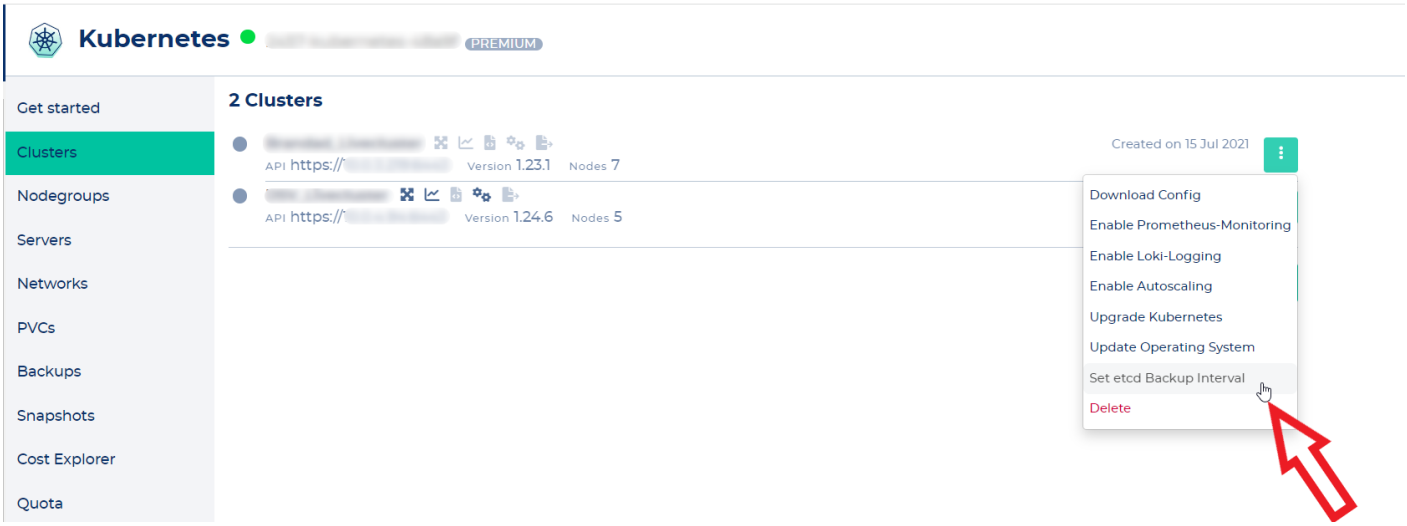
Backup

ETCD Backups for the state of your k8s cluster

If a K8s cluster is damaged, the state of the K8s cluster can be restored with an etcd backup. Here is stored for example which containers and which volumes should be running.

The state of your cluster is stored in the etcd key-value-store. You can configure periodic backups for the etcd store.

To enable etcd backups you need to click "Set etcd Backup interval" in the cluster's context menu.



Now choose your desired backup interval.

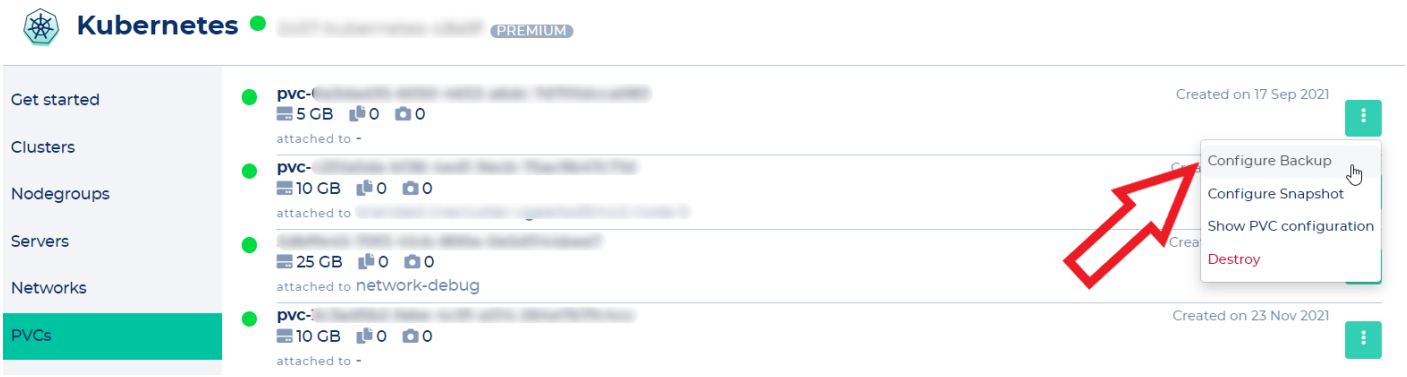
The etcd backups are stored in the S3-Object-Storage. You can find the credentials in the "Object Storage" section of the Kubernetes interface.

PVC Backups for your workload

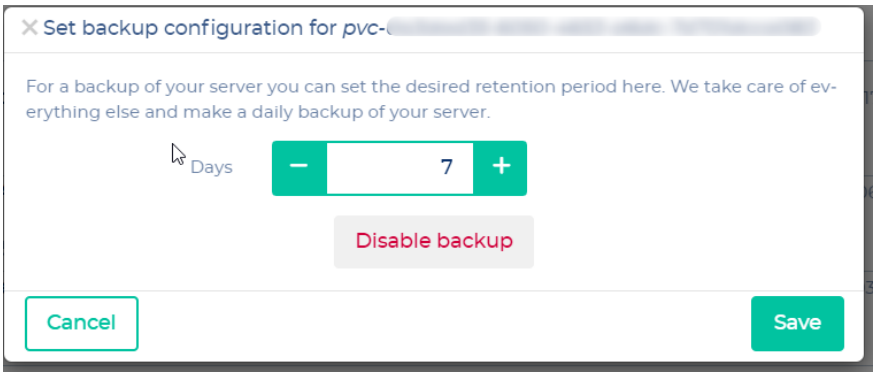
etcd Backups just save the state of your K8s cluster, but not your workload.

The actual application and its data must be backed up separately via the PVC backups.

To enable backups of your workload, you need to click "Configure Backup" in the PVC's context menu.



Now choose your desired backup interval.



Upgrades

Kubernetes Upgrades

What is the recommended way to upgrade NWS Managed Kubernetes

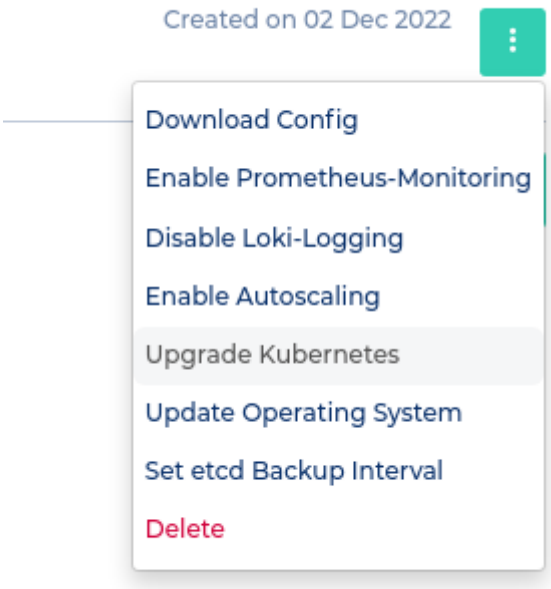
We recommend to upgrade the masters by two minor versions. Afterwards you can upgrade extra nodegroups to the version of the masters. This way you can skip a minor version on the nodes.

Tip: Replacing extra nodegroups is in most cases faster than upgrading existing ones. Just start new nodegroups (the nodes will spawn with the current Kubernetes version of the masters) and delete the old ones. Only do this if you are sure that you don't have persistent data stored on the nodes' filesystems.

How do I start a Kubernetes version upgrade on the masters?

Note that the steps below will only upgrade the master nodegroup and the default-worker nodegroup. See the next Question to find out how to upgrade extra nodegroups.

To upgrade to a more recent Kubernetes version you have to press the "Upgrade Kubernetes" button in the cluster's context menu. Afterwards choose the Kubernetes version you want to upgrade to and press "Upgrade" in the modal to start the upgrade.

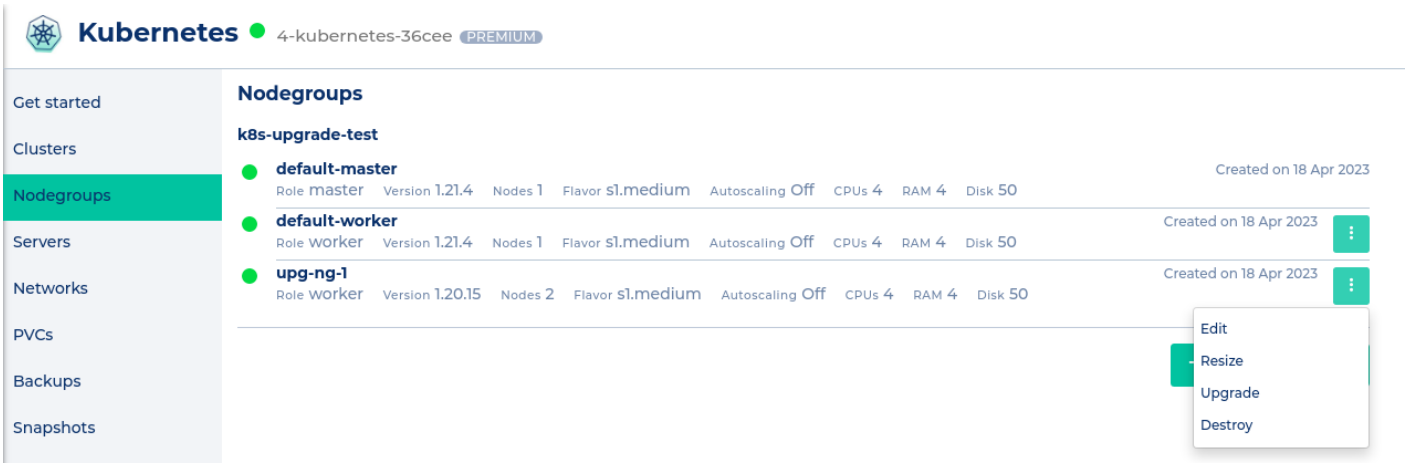


- If you do not see an upgrade button you have to disable OS Upgrades first
- Do not enable OS Upgrades while running a Kubernetes Upgrade
- Please make sure that the cluster health status is "healthy" before upgrading

How do I upgrade extra worker nodes?

To upgrade extra worker nodes you have to switch to the nodegroup menu. In the context menu of the nodegroups you can select "Upgrade". Click the "Upgrade!" button in the modal to start the upgrade.

Note that you can not choose the Kubernetes version of extra node groups. It is only possible to upgrade to the current master version.



What will be upgraded?

On the master node(s)

- etcd
- kube-apiserver
- kube-controller-manager
- kube-scheduler

On master and worker node(s)

- kube-proxy
- kubelet

Also the cluster services in kube-system namespace will be upgraded.

How is the Upgrade performed?

The nodes are upgraded one by one.

Important: Each node will be drained during the upgrade, which means that all the pods on a node are evicted and rescheduled. Make sure to have enough resources left in your cluster so that pods can be rescheduled quickly on other nodes.

How long will the upgrade take?

The Upgrade takes 5 to 10 minutes per node.

Where can I get help for Kubernetes upgrades?

You should always consider to get a MyEngineer involved if you upgrade one of your NWS Kubernetes clusters, especially on production clusters. Our support can help you to detect breaking changes you will run into when upgrading. With the help of MyEngineer you can keep upgrade related downtimes as low as possible.

Operating System Updates

To configure automatic OS Upgrades for your Kubernetes nodes, you have to click on "Update Operating System" in the cluster's context menu.

The upgrades are orchestrated by zincati. You get to choose between immediate, periodic and lock-based upgrades. Keep in mind that your nodes will be rebooted if an upgrade takes place. Take a look at the zincati documentation for further explanations.

We maintain our own FCOS updates graph to be able to test the official releases before making them available to you.

One Click Deployments

We prepared some software stacks that you can deploy to your cluster with just one click.

Prometheus monitoring

Click the "Enable Prometheus-Monitoring" button in the cluster's context menu to install the kube-prometheus-stack helm chart to your cluster. The helm release name is nws-prometheus-stack and it will be installed in the kube-system namespace.

Check out the instructions in the "Get started" tab to find out how to access the components of the prometheus stack.

Loki Logging

Click the "Enable Loki-Logging" button in the cluster's context menu to install the loki and grafana helm chart to your cluster. The helm release names are nws-loki and nws-loki-grafana, both residing in the kube-system namespace.

Check out the instructions in the "Get started" tab to find out how to access loki and grafana.

StorageClasses

Our kubernetes cluster comes 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 into the Cinder-CSI Driver documentation or feel free to contact our support team to get 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` as being specified by it's annotation is the default class that is used when no SC gets specified. It will provision an ext4 formatted Volume in OpenStack immediately and delete it if the PVC will be 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 PVCs with a lot of small files like Databases or Logging systems for example as xfs is able to scale inodes dynamically. This stands in contrast to EXT4, which will create 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` SC 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 our Volume-Type that transparently enable encryption for the volume. The two classes differ they way `standard` and `high-iops` do. One uses the normal IOPS and the other the high IOPS profile.

Custom

You can of course also create your own storageclass with the any of the paramters and options available to the Cinder CSI Driver and Kubernetes. The following custom storageclass for example would use the `Ceph-Encrypted` Volume type with the XFS format while retaining the created 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 are unable to provide Volumes with the `RWX` access type. This is on our roadmap, but we cannot commit to any timeframe at this point in time. For now you'll need to build your own solution based on NFS or Rook Ceph for example. We would be happy to assist you with that.