

Kubernetes

Erstellt von:

Erstellt am:

Version:

Inhaltsverzeichnis

Kubernetes	3
What does Managed mean?	4
Getting Started	5
Create NWS Managed Kubernetes	6
Install kubectl and kubelogin	9
kubectl	9
kubelogin (kubectl oidc-login)	9
Deciding on a CNI	10
Flannel	10
Cilium	10
Connect to the created cluster	11
Starting further clusters in the same K8s project	12
Caution!	13
Administrative Tasks	14
Nodes and Nodegroups	15
Resizing Nodegroups	15
Create Nodegroups	15
Cluster Autoscaling	16
Backup	17
ETCD Backups for the state of your k8s cluster	17
PVC Backups for your workload	17
Upgrades	19
Kubernetes Upgrades	19
What is the recommended way to upgrade NWS Managed Kubernetes	19
How do I start a Kubernetes version upgrade on the masters?	19
How do I upgrade extra worker nodes?	19
What will be upgraded?	20
How is the Upgrade performed?	20
How long will the upgrade take?	20
Where can I get help for Kubernetes upgrades?	20
Operating System Updates	20
One Click Deployments	21
Prometheus monitoring	21
Loki Logging	21
StorageClasses	22
standard (Default)	22
nws-storage	22
high-iops	23
encrypted(-high-iops)	23
Custom	23
ReadWriteMany	24
Advanced Topics	25
CoreDNS	26
Static Hostnames	26
High-IOPS PersistentVolumes	27
Node Taints	28
NetworkPolicy	29
Use cases	29
Types	29
Observing Cluster-Traffic - Cilium	31
Webui	31
CLI	31
Installation	31
Remote exec	32
Observing	32
Related Links	35
Monitoring	36

Kubernetes

- [What does Managed mean?](#)
- [Getting Started](#)
 - [Create NWS Managed Kubernetes](#)
 - [Install kubectl and kubelogin](#)
 - [Deciding on a CNI](#)
 - [Connect to the created cluster](#)
 - [Starting further clusters in the same K8s project](#)
 - [Caution!](#)
- [Administrative Tasks](#)
 - [Nodes and Nodegroups](#)
 - [Cluster Autoscaling](#)
 - [Backup](#)
 - [Upgrades](#)
 - [One Click Deployments](#)
 - [StorageClasses](#)
- [Advanced Topics](#)
 - [CoreDNS](#)
 - [High-IOPS PersistentVolumes](#)
 - [Node Taints](#)
 - [NetworkPolicy](#)
 - [Observing Cluster-Traffic - Cilium](#)
- [Related Links](#)
- [Monitoring](#)

What does Managed mean?

We offer a managed Kubernetes service.

What does Managed mean at NMS:

1. We install the cluster

- [here](#) you can find more info about starting your cluster

2. We operate and monitor your cluster

- If a master or worker changes to "not ready", we will be alerted and fix the problem
- We monitor the K8s system namespace in Prometheus and make sure that e.g. the load balancers are working or the PVCs are mapped correctly
- More information about monitoring can be found [here](#)

3. We offer updates

- The updates are divided into operating system updates and K8s version updates
- For the K8s version updates we test the new version in advance and make sure that the normal standard installation still works with the new version
- You can find more information about the updates [here](#)

4. We offer Backups

- Etcd Backup
- PVC Backup
- [here](#) you will find more information about backups

Getting Started

Create NWS Managed Kubernetes

To start our Managed Kubernetes service, you must first create an account on our [NWS Customer Interface](#) and provide a valid [payment method](#).


Create Kubernetes

Stefan Schneider
Stefans Spielwiese

Kubernetes

Your Happy-Go-Lucky Container Orchestration Service Hosted in Germany.

[Learn more](#)



Cluster in Minutes

Define your cluster status, set the number of nodes, download your kubeconfig and start using kubectl. In just a few steps, your Kubernetes Cluster is ready. We take care of its high availability, updates, backups and a stable underlying cloud infrastructure.

Managed Control Plane

The Control Plane is the brain of the cluster that takes the action and enforces your desired state. You can decide between a standalone or a high available control plane. We monitor and operate it 24 hours a day and 7 days a week.

Persistent Volume Claims

Managing data within containers can be tricky. In theory there is only stateless workload for your pods, for everything else we support Persistent Volume Claims. Store your data, e.g. for the operation of databases, beyond the life cycle of containers.

Horizontal Scaling and Autoscaling

Scale your application with a simple command or use the Kubernetes web interface. Depending on CPU usage or other metrics, new containers can be started automatically. Autoscaling detects when you actually run out of space in your cluster and starts new VMs, which collapse again when you no longer need them. Saves

0.0 €/Month [See pricing table](#)

[Create](#)

NETWAYS
WEB SERVICES

- Overview
- Kubernetes
- MyEngineer
- New Contract
- CLOUD
 - Virtual Private Cloud
 - Server
- KUBERNETES
 - Kubernetes
 - Cluster
- DATABASE
 - Vitess Cluster
 - Database
- APPS
 - Icinga 2 Master
 - Icinga 2 Satellite
 - Rocket.Chat
 - Nextcloud
 - SuiteCRM
 - Gitlab CE
 - Request Tracker
 - Gitlab EE
 - Cachet
 - Jitsu

Kubernetes 7080-kubernetes-ec03b PREMIUM

Get started

- Clusters
- Nodegroups
- Servers
- Networks
- PVCs
- Backups
- Snapshots
- Cost Explorer
- Object Storage
- Load Balancers
- Manage Contract

More Information

First Steps

- Create a Kubernetes Cluster first
- Install kubectl on your client
- Download your config and save it as `phone/kube/config`
- Check out your Cluster: `kubectl get nodes`

External Resources

Learn more about your product

- [Documentation](#)
- [Faq](#)
- [Blog](#)
- [Tutorials](#)

Kubernetes Dashboard

Deploy and manage your applications in a web-based dashboard.

Autoscaling

Activating Autoscaling will install additional services in your cluster which will monitor the load of your nodes and spin up new nodes on demand. Once the...

Monitoring with Prometheus and Grafana

If you would like to get an insight into various kinds of metrics, view the current usage of your nodes or setup alerts for some services, Prometheus and Graf... You can [enable this feature](#) individually for your Kubernetes clusters!

Logging with Loki and Grafana

By gathering logs from all running pods and services in your Kubernetes Cluster, Loki and Grafana provide a great and easy to use solution to query and vie... You can [enable this feature](#) individually for your Kubernetes clusters!

Loki LogCLI

Analyze all the logs in Loki right from your local CLI. Installation options:

- ✓ User is ready
- ✓ Project is ready

The first step to the cluster would now be to start the first cluster in the Clusters submenu. In the concrete example with the smallest requirements and in version 1.25.2

NETWAYS WEB SERVICES

Overview

Kubernetes

MyEngineer Get Support Now

New Contract

CLOUD

Virtual Private Cloud

Server

KUBERNETES

Kubernetes

Cluster

DATABASE

Vitess Cluster

Database

APPS

Icinga 2 Master

Icinga 2 Satellite

Rocket.Chat

Nextcloud

SuiteCRM

Gitlab CE

Request Tracker

Gitlab EE

Cachet

Itisi

Kubernetes 7080-kubernetes-ec03b PREMIUM

Get started

Clusters

Nodegroups

Servers

Networks

PVCs

Backups

Snapshots

Cost Explorer

Object Storage

Load Balancers

Manage Contract

0 Clusters

+ Create Cluster

Create new cluster

Choose a title for your cluster. Leave blank to use the generated name.

Stefans Testcluster

Kubernetes Project

Select one of your existing projects or create a new one.

7080-kubernetes-ec03b

+ New Project

Kubernetes Version

Choose your desired Kubernetes version.

1.25.2



Control Plane Nodes

Decide between a setup with a single master node or a high available setup with three master nodes with load balancing.

Single master setup

High availability setup

Accessibility

Choose "Private" if you don't want your Kubernetes-API to be accessible over a public floating IP. A private cluster will only be accessible over VPN. To get VPN access, please send your request to nws@netways.de or create a ticket for the MyEngineer.

Public

Private

Custom Subnet

Choose a private subnet the nodes will be launched in. Use CIDR notation. Defaults to 10.0.0.0/24.

10.0.0.0/24

Control Plane Flavor

Choose a flavor. The chosen flavors are not editable after creation.

S1

S2

4 vCPU 4 GB RAM 50 GB Storage	6 vCPU 8 GB RAM 75 GB Storage	8 vCPU 16 GB RAM 100 GB Storage	12 vCPU 32 GB RAM 150 GB Storage	16 vCPU 64 GB RAM 200 GB Storage
s1.medium	s1.large	s1.xlarge	s1.xxlarge	s1.xxxlarge

Worker Nodes

Select the initial number of worker nodes. You can change the number of Nodes again later.

-

2

+

Worker Node Flavor

Choose the flavor of your worker nodes. Later it is possible to add additional Nodegroups to your running cluster with other types of flavors.

S1

S2

4 vCPU 4 GB RAM 50 GB Storage	6 vCPU 8 GB RAM 75 GB Storage	8 vCPU 16 GB RAM 100 GB Storage	12 vCPU 32 GB RAM 150 GB Storage	16 vCPU 64 GB RAM 200 GB Storage
s1.medium	s1.large	s1.xlarge	s1.xxlarge	s1.xxxlarge

In the background, NWS automations start, create an OpenStack project, create the machines, and configure the cluster with all the necessary components. After 5-10 minutes, the cluster is ready for use.

Getting Started

Install kubectl and kubelogin

kubectl

kubectl is the command-line tool to manage your Kubernetes clusters and is available for Linux, Windows and MacOS. For an easy installation follow the official instructions on kubernetes.io.

kubelogin (kubectl oidc-login)

kubelogin is a plugin that extends kubectl with OpenID Connect. This is mandatory to use NWS-ID with your Kubernetes cluster. Follow the [official instructions](#) for easy installation.

Getting Started

Deciding on a CNI

We support two different CNIs, that being Flannel and Cilium. Flannel is known for it's simplicity and Cilium for it's advanced even service mesh like features.

Flannel



Flannel focuses on the integral part that is the network connection itself. It does not provide any NetworkPolicies or traffic encryption, but it is rock solid when it comes to inter pod communication. That makes it a good choice if you want to chain CNIs and/or add a Service-Mesh on top of it.

Cilium



If you are interested in more advanced CNI features like NetworkPolicies, traffic encryption, mutual TLS and network Observability, Cilium is the right choice for you. It can provide many features that would otherwise necessitate a full blown ServiceMesh like [Istio](https://istio.io). You can find out more on their website: <https://cilium.io>.

Connect to the created cluster

Now that the cluster is built and kubectl is already installed, it is time to connect to the cluster. This is done by clicking on "Download-Config" in the context menu next to the cluster in the NWS backend.

The top screenshot shows the Netways Web Services interface for 'Stefans erstes K8s Projekt'. The 'First Steps' section is visible, with step 3 'Download your config and save it as \$HOME/.kube/config' highlighted by a red circle. The bottom screenshot shows the '1 Clusters' section with a context menu open next to the 'Test' cluster, highlighting the 'Download Config' option.

The just downloaded Config must now be moved to the correct place.

To do this, we create a directory in the user home that is still required (if it does not exist) and copy the file into it. At the end we adjust the rights. All work is done as a local user:

```
mkdir ~/.kube
mv ~/Downloads/config ~/.kube/
chmod 0600 ~/.kube/config
```

Kubectl should now automatically use the new config. To see if works we can try to list all cluster nodes as done below. If you use NWS-ID, your browser will open for authentication. After that, just switch back to the terminal.

```
$ kubectl get nodes
NAME                                STATUS  ROLES  AGE  VERSION
workshop-virgin2-ytwudzfjco6-master-0  Ready  master  17h  v1.23.1
workshop-virgin2-ytwudzfjco6-node-0    Ready  <none>  17h  v1.23.1
```

Getting Started

Starting further clusters in the same K8s project

If you need a staging cluster and a production cluster, this can easily be done in the same Kubernetes app - but you'll need to launch separate clusters for each.



The screenshot shows the NETWAYS Kubernetes dashboard. On the left sidebar, the 'Cluster' option under the 'KUBERNETES' section is highlighted with a red circle and a red arrow. In the main content area, the '1 Clusters' section shows a cluster named 'Test' with details like API URL, version, and nodes. A red arrow points to the '+ Create Cluster' button in the top right corner of the main content area.

Multiple clusters can be started in the same product, even with the same subnet.

The respective systems of the cluster are also provided normally in the same subnet, but then receive different addresses from the DHCP in the same subnet.

The communication with each cluster is done via its own Config and thus each cluster can also work only with its workers.

Getting Started

Caution!

With NWS Managed Kubernetes you have full control over all resources in your cluster. Please adhere to these rules:

- do not schedule your own pods on master nodes
- keep out of the kube-system namespace *

* unless you want to do stuff that is well documented on this site. For example [setting static hostnames in CoreDNS](#).

Scheduling your own workloads on master nodes could cause API downtimes due to OOM events. Editing or creating resources in the kube-system namespace could break critical cluster services.

Administrative Tasks

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 minimum and maximum node count.

×

Edit Nodegroup default-worker

Cluster wide autoscaling is enabled. You can configure each Nodegroup to scale independently.

Nodegroup Autoscaling

Disabled

Enabled

Minimum Node Count

–

1

+

Maximum Node Count

–

5

+

Update

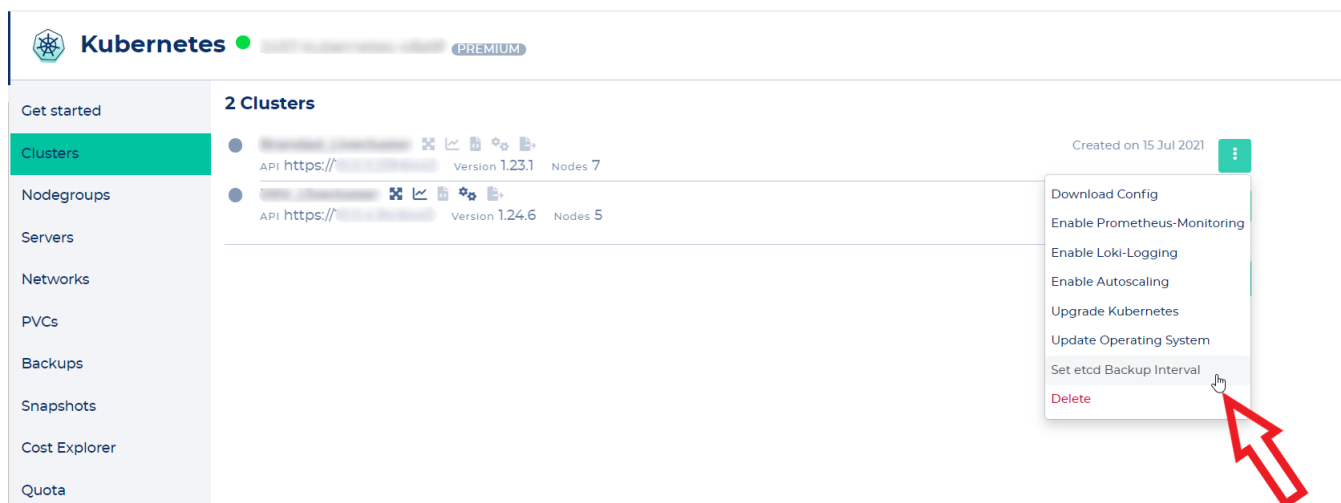
Backup

ETCD Backups for the state of your k8s cluster

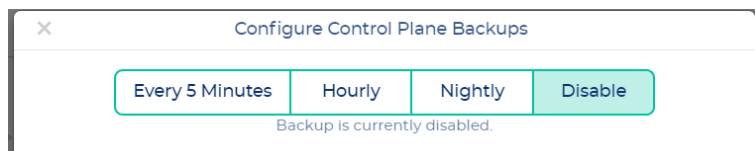
If a K8s cluster is damaged, the state of the K8s cluster can be restored with an etcd backup. Here is stored for example which containers and which volumes should be running.

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.

Kubernetes
PREMIUM

Get started

Clusters

Nodegroups

Servers

Networks

PVCs

pvc- 5 GB 0 0
attached to -

pvc- 10 GB 0 0
attached to -

pvc- 25 GB 0 0
attached to network-debug

pvc- 10 GB 0 0
attached to -

Created on 17 Sep 2021

Created on 23 Nov 2021

Configure Backup

Configure Snapshot

Show PVC configuration

Destroy

Now choose your desired backup interval.

Set backup configuration for pvc-

For a backup of your server you can set the desired retention period here. We take care of everything else and make a daily backup of your server.

Days

Disable backup

Cancel Save

Upgrades

Kubernetes Upgrades

What is the recommended way to upgrade NWS Managed Kubernetes

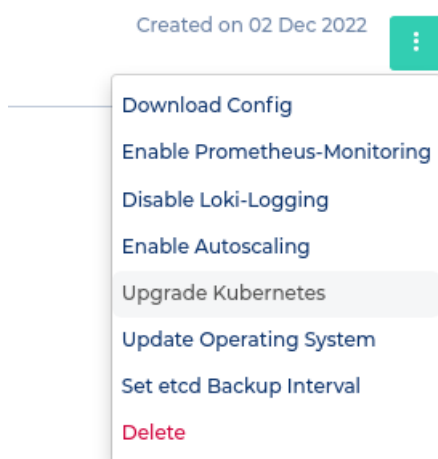
We recommend to upgrade the masters by two minor versions. Afterwards you can upgrade extra nodegroups to the version of the masters. This way you can skip a minor version on the nodes.

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.

The screenshot shows the Netways Kubernetes management console. On the left is a sidebar with navigation links: Get started, Clusters, Nodegroups (selected), Servers, Networks, PVCs, Backups, and Snapshots. The main area displays the 'Nodegroups' section for the cluster '4-kubernetes-36cee'. It lists three node groups: 'default-master' (Role: master, Version: 1.21.4, Nodes: 1, Flavor: s1.medium, Autoscaling: Off, CPUs: 4, RAM: 4, Disk: 50), 'default-worker' (Role: Worker, Version: 1.21.4, Nodes: 1, Flavor: s1.medium, Autoscaling: Off, CPUs: 4, RAM: 4, Disk: 50), and 'upg-ng-1' (Role: Worker, Version: 1.20.15, Nodes: 2, Flavor: s1.medium, Autoscaling: Off, CPUs: 4, RAM: 4, Disk: 50). A context menu is open over the 'upg-ng-1' node group, showing options: Edit, Resize, Upgrade, and Destroy.

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 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.

Advanced Topics

CoreDNS

Static Hostnames

Static hostnames can be set in an extra ConfigMap called "coredns-extra-hosts" inside the kube-system namespace.

```
apiVersion: v1
data:
  hosts.list: |
    192.168.100.100 example.com
kind: ConfigMap
metadata:
  name: coredns-extra-hosts
  namespace: kube-system
```

High-IOPS PersistentVolumes

To use High-IOPS PersistentVolumes in your Managed Kubernetes Cluster you need to manually create a High-IOPS StorageClass and reference it in the PVC resources.

```
kind: StorageClass
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "false"
  name: high-iops
parameters:
  type: Ceph-High-IOPS
provisioner: cinder.csi.openstack.org
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Node Taints

Do not hesitate to taint worker nodes.

Our DaemonSets (e.g. csi-cinder-nodeplugin) will tolerate all of them and still be scheduled on the worker nodes.

NetworkPolicy

To be able to use NetworkPolicies, you'll need to run Cilium as your CNI provider. Flannel does not support it.

Use cases

If you want to secure access between pods and only allow specific traffic, NetworkPolicy are the tool of choice. They basically work like a firewall and only allow the ports/traffic you specifically specified. A basic NetworkPolicy might look like this:

```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: egress-namespaces
spec:
  podSelector:
    matchLabels:
      app: myapp
  policyTypes:
  - Egress
  egress:
  - to:
    - namespaceSelector:
        matchExpressions:
        - key: namespace
          operator: In
          values: ["frontend", "backend"]
```

This will allow the myapp pods to communicate with the pods found in the namespaces `frontend` and `backend`.

Types

The afore mentioned Policy is based on the vanilla `NetworkPolicy` resource that comes with Kubernetes. Cilium on the other hand as extended this resource to form a `CiliumNetworkPolicy`. It has advanced features like L7 and DNS traffic inspection.

```
---
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "fqdn"
spec:
  endpointSelector:
    matchLabels:
      org: empire
      class: mediabot
  egress:
  - toFQDNs:
    - matchName: "api.github.com"
  - toEndpoints:
```

```
- matchLabels:
  "k8s:io.kubernetes.pod.namespace": kube-system
  "k8s:k8s-app": kube-dns
toPorts:
- ports:
  - port: "53"
    protocol: ANY
rules:
  dns:
  - matchPattern: "*"
- toEndpoints:
  - matchLabels:
    app: nginx
toPorts:
- ports:
  - port: "80"
    protocol: TCP
rules:
  http:
  - method: "GET"
    path: "/public/*"
  - method: "GET"
    path: "/secret/index.html"
  headers:
  - 'X-My-Header: true'
```

This policy for example allows any traffic to `api.github.com` and http traffic to a nginx deployment. Some routes also need to be accessed with special headers.

Another custom resource by cilium is the `CiliumClusterwideNetworkPolicy` that as the name suggests isn't namespace scoped and will be applicable for the whole cluster.

Cilium has an extension called Hubble. Similar to the well known telescope it specializes in observability. With its hubble-ui component you are able to graphically visualize all traffic flowing in your cluster. But you are not constrained to the web-ui and can also use the hubble cli, which is even more powerful and can help you debug issues faster.

To gain access to the webui, you'll need to port-forward traffic to the web-ui service into the cluster. It works as follows:

```
$ kubectl get svc -n kube-system hubble-ui
$ kubectl port-forward -n kube-system svc/hubble-ui 8080:80
```

CLI

To gain access to the hubble cli, you'll either need to install the hubble locally or execute everything in the cilium container.

Installation

```
HUBBLE_VERSION=$(curl -s https://raw.githubusercontent.com/cilium/hubble/master/stable.txt)
HUBBLE_ARCH=amd64
if [ "$(uname -m)" = "aarch64" ]; then HUBBLE_ARCH=arm64; fi
curl -L --fail --remote-name-all https://github.com/cilium/hubble/releases/download/$HUBBLE_VERSION/hubble-linux-
${HUBBLE_ARCH}.tar.gz{..sha256sum}
```

```
sha256sum --check hubble-linux-${HUBBLE_ARCH}.tar.gz.sha256sum
sudo tar xzvfC hubble-linux-${HUBBLE_ARCH}.tar.gz /usr/local/bin
rm hubble-linux-${HUBBLE_ARCH}.tar.gz{,}.sha256sum}
```

For hubble to work locally it needs access to the API as well.

```
$ kubectl port-forward -n kube-system svc/hubble-relay 4245:80 &
Forwarding from 0.0.0.0:4245 -> 4245
Forwarding from [::]:4245 -> 4245
```

Remote exec

```
$ alias hubble='kubectl exec -in kube-system ds/cilium -c cilium-agent -- hubble'
$ hubble status
Healthcheck (via unix:///var/run/cilium/hubble.sock): Ok
Current/Max Flows: 4,095/4,095 (100.00%)
Flows/s: 4.07
```

Observing

To observe any traffic (much like tcpdump) you can just run observe in --follow mode. It will show any traffic that runs through your cluster.

```
$ hubble observe --follow
Sep  4 07:28:18.255: 10.100.2.60:48428 (host) -> 10.100.2.77:4240 (health) to-endpoint FORWARDED (TCP Flags: ACK, PSH)
Sep  4 07:28:18.256: 10.100.2.60:48428 (host) <- 10.100.2.77:4240 (health) to-stack FORWARDED (TCP Flags: ACK, PSH)
Sep  4 07:28:23.290: 10.100.5.13:55176 (remote-node) <- 10.100.2.77:4240 (health) to-overlay FORWARDED (TCP Flags: ACK)
Sep  4 07:28:23.292: 10.100.5.13:55176 (remote-node) -> 10.100.2.77:4240 (health) to-endpoint FORWARDED (TCP Flags: ACK)
Sep  4 07:28:23.613: 10.100.2.60:37112 (host) -> kube-system/coredns-69bc699795-trnrxn:8181 (ID:17050) to-endpoint FORWARDED (TCP Flags: SYN)
Sep  4 07:28:23.613: 10.100.2.60:37112 (host) <- kube-system/coredns-69bc699795-trnrxn:8181 (ID:17050) to-stack FORWARDED (TCP Flags: SYN, ACK)
Sep  4 07:28:23.613: 10.100.2.60:37112 (host) -> kube-system/coredns-69bc699795-trnrxn:8181 (ID:17050) to-endpoint FORWARDED (TCP Flags: ACK)
Sep  4 07:28:23.613: 10.100.2.60:34644 (host) -> kube-system/coredns-69bc699795-trnrxn:8080 (ID:17050) to-endpoint FORWARDED (TCP Flags: SYN)
```

Hubble can also filter based on many different identities, like pod labels, namespaces and dns lookups.

```
$ hubble observe --follow \
  --pod default/nginx-5f8f49fff4-m8m9h \
  --not --label k8s-app=kube-dns
Sep  4 08:23:29.510: default/nginx-5f8f49fff4-m8m9h:53700 (ID:37906) -> 142.250.184.238:80 (world) to-stack FORWARDED (TCP Flags: SYN)
Sep  4 08:23:29.519: default/nginx-5f8f49fff4-m8m9h:53700 (ID:37906) -> 142.250.184.238:80 (world) to-stack FORWARDED (TCP Flags: ACK)
Sep  4 08:23:29.519: default/nginx-5f8f49fff4-m8m9h:53700 (ID:37906) -> 142.250.184.238:80 (world) to-stack FORWARDED (TCP Flags: ACK, PSH)
Sep  4 08:23:29.542: default/nginx-5f8f49fff4-m8m9h:53700 (ID:37906) -> 142.250.184.238:80 (world) to-stack
```

FORWARDED (TCP Flags: ACK, FIN)

Sep 4 08:23:29.548: default/nginx-5f8f49fff4-m8m9h:53700 (ID:37906) -> 142.250.184.238:80 (world) to-stack

FORWARDED (TCP Flags: ACK)

That for example will monitor all traffic of the nginx pod found in the `default` namespace except DNS lookups.

Another common use case would be to filter based on destination port. This can be done with the `--to-port` Flag. If you need more info, the output can be formatted as json:

```
$ hubble observe --follow --pod nginx-5f8f49fff4-m8m9h --to-port 80 -o json | jq
{
  "flow": {
    "time": "2023-09-04T08:25:35.610232081Z",
    "uuid": "c488a8f9-1301-4490-84f1-7ed96afd36f3",
    "verdict": "FORWARDED",
    "ethernet": {
      "source": "d6:5b:64:ee:1c:86",
      "destination": "e2:1e:63:4a:0b:cf"
    },
    "IP": {
      "source": "10.100.3.241",
      "destination": "142.250.184.238",
      "ipVersion": "IPv4"
    },
    "I4": {
      "TCP": {
        "source_port": 33610,
        "destination_port": 80,
        "flags": {
          "SYN": true
        }
      }
    },
    "source": {
      "ID": 740,
      "identity": 37906,
      "namespace": "default",
      "labels": [
        "k8s:app=nginx",
        "k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=default",
        "k8s:io.cilium.k8s.policy.cluster=default",
        "k8s:io.cilium.k8s.policy.serviceaccount=default",
        "k8s:io.kubernetes.pod.namespace=default"
      ],
      "pod_name": "nginx-5f8f49fff4-m8m9h",
      "workloads": [
        {
          "name": "nginx",
          "kind": "Deployment"
        }
      ]
    },
    "destination": {
      "identity": 2,
      "labels": [
        "reserved:world"
      ]
    }
  }
}
```

```
]
},
"Type": "L3_L4",
"node_name": "cl-cilium-15-jibbo4pnpgn7-node-1",
"event_type": {
  "type": 4,
  "sub_type": 3
},
"traffic_direction": "EGRESS",
"trace_observation_point": "TO_STACK",
"is_reply": false,
"Summary": "TCP Flags: SYN"
},
"node_name": "cl-cilium-15-jibbo4pnpgn7-node-1",
"time": "2023-09-04T08:25:35.610232081Z"
}
```

Related Links

On our [tutorials page](#) you will find numerous articles about Kubernetes.

Monitoring

- We monitor the health of your K8s cluster 24/7
- If a master or worker changes to an unhealthy state, we are alerted and fix the problem
- In addition, we deploy the **nws-agent-stack** in each cluster
 - It consists of a **prometheus-agent** pod and a **kube-state-metrics** pod
 - The purpose of the nws-agent-stack is to monitor the system services (everything within the kube-system namespace) in your cluster. This is used, for example, to ensure that PVCs are mapped correctly
 - **Do not change its configuration!**
- We do **not** monitor the containers located in your namespaces by default
- If you want to monitor your workloads, you need to install your own monitoring solution in your cluster
- Check out the [NWS-Prometheus-Stack](#) for a customizable solution.
Our MyEngineers are also happy to help with setup here.