

NVMesh Kubernetes Operator Guide

2.4.1 — Last update: 16 December 2021

Excelero, Ltd.

Table of Contents

1. Copyright and Trademark Information	2
2. Introduction	3
3. NVMesh OpenShift Operator Installation	4
3.1. YAML Installation	5
3.2. Setting up Repository Access	6
3.3. Operator Installation	7
3.4. Remove the Default Storage Class	9
3.5. Create an NFS Server Instance	10
3.6. Add a PV to the Cluster	19
3.7. Install NVMesh Pods	21
3.8. Expose Management Routes	25
4. Creating PVCs with NVMesh Storage	26
4.1. Format Drives	27
4.2. Create PersistentVolumeClaims	28
5. NVMesh OpenShift Operator on Azure	29
5.1. Prerequisites	30
5.1.1. Resource Limits	31
5.1.2. Roles and Permissions	37
5.1.3. Public DNS Zone	38
5.1.4. Azure CLI	39
5.2. OpenShift Cluster on Azure	40
5.2.1. Deploying the Cluster	41
5.2.2. Accessing the Cluster	47
5.2.3. Infiniband Only	48
6. NVMesh on Azure Kubernetes Service	56
6.1. Quick Summary	57
6.2. Deploying the AKS Cluster	59
6.2.1. Login via the Azure CLI	61
6.2.2. Verify Azure Subscription Access	62
6.2.3. Create a Resource Group	63
6.2.4. Create the Cluster	64
6.3. Deploying NVMesh on AKS	65
6.3.1. Set up Repository Access	66
6.3.2. Deploy Node Pool for Targets	67
6.3.3. Get the NVMesh Operator	69
6.3.4. Deploy the NVMesh Operator	70
6.3.5. Deploy NVMesh Pods	71

6.3.6. Deploy Node Pool for Compute / Clients	73
6.3.7. Create PVCs (Persistent Volume Claims)	74
6.4. Performance Tuning.....	76

1. Copyright and Trademark Information

© 2015-2021 Excelero, Inc. All rights reserved. Specifications are subject to change without notice.

Excelero, the Excelero logo, MeshProtect, and Remote-Direct-Drive-Access (RDDA) are trademarks of Excelero, Inc. in the United States and/or other countries. NVMesh® is a registered trademark of Excelero, Inc. in the United States.

Mellanox and ConnectX are registered trademarks of NVIDIA.

Intel is a registered trademark of Intel Corporation. Xeon and Core are trademarks of Intel Corporation. All other brands or products are trademarks or registered trademarks of their respective holders and should be treated as such.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo and OpenShift are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

Ubuntu® is a trademark of Canonical or its subsidiaries in the United States and/or other countries.

Node.js® is an official trademark of Joyent. NVMesh is not formally related to or endorsed by the official Joyent Node.js open source or commercial project

OpenStack® is an official trademark of the OpenStack Foundation.

MONGO and MONGODB are trademarks of MongoDB Inc., registered in the United States and other countries.

Microsoft® and Azure® are trademarks of Microsoft, Inc., registered in the United States and other countries.

2. Introduction

The purpose of this guide is to provide simple instructions for installing and using the **NVMesh** operator in an OpenShift environment.

Basic knowledge of OpenShift administration is recommended.

3. NVMesh OpenShift Operator Installation

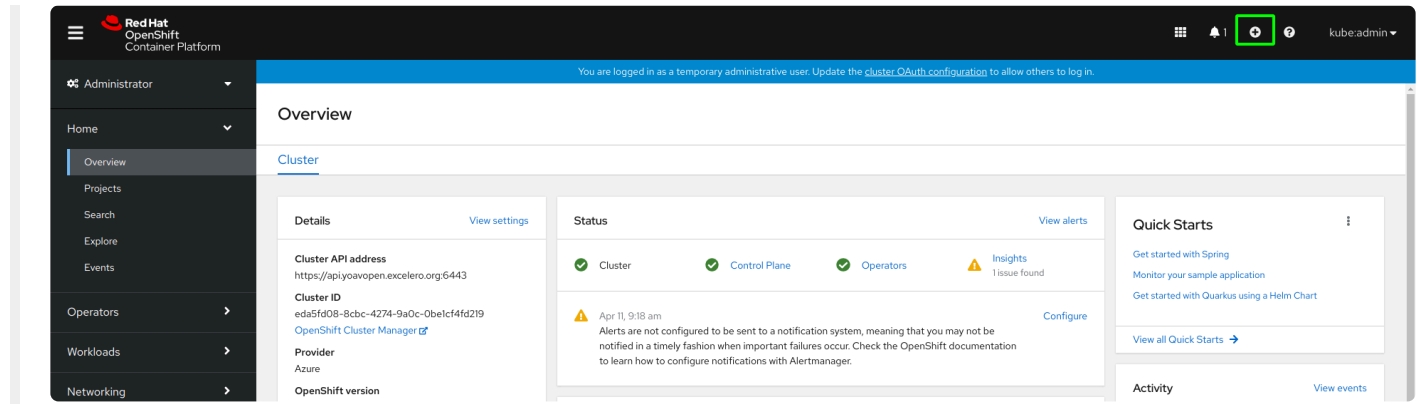
In this part we will describe the full procedure for installing NVMesh operator and running NVMesh on the OpenShift cluster

3.1. YAML Installation

Installing YAMLs is a common procedure when working with OpenShift.

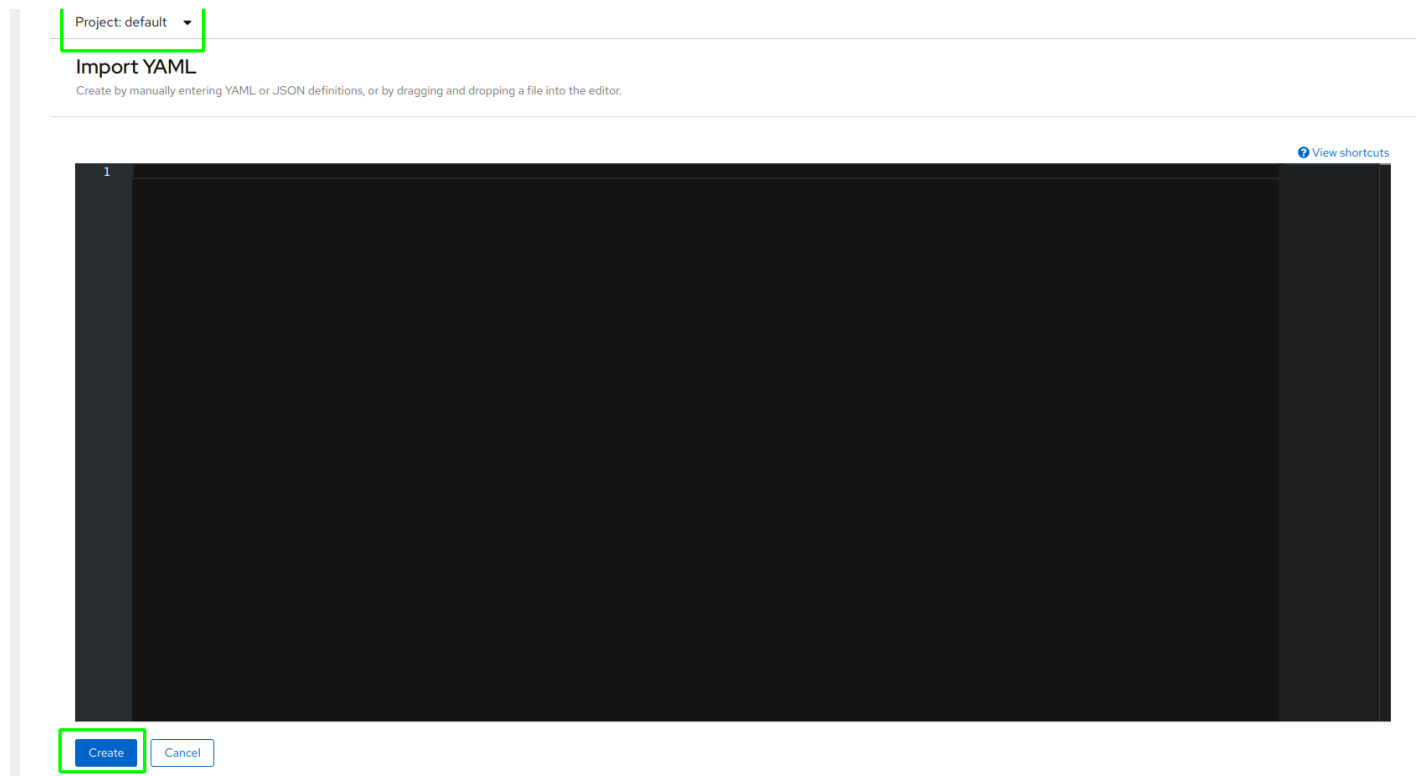
To install a component to an OpenShift cluster, use the following procedure.

First, click + at the top right of the UI



On the Import page, select project “default” from the dropdown at the top.

Paste the YAML in the text box and click “Create”



3.2. Setting up Repository Access

To access the **NVMesh** repository, import access credentials.

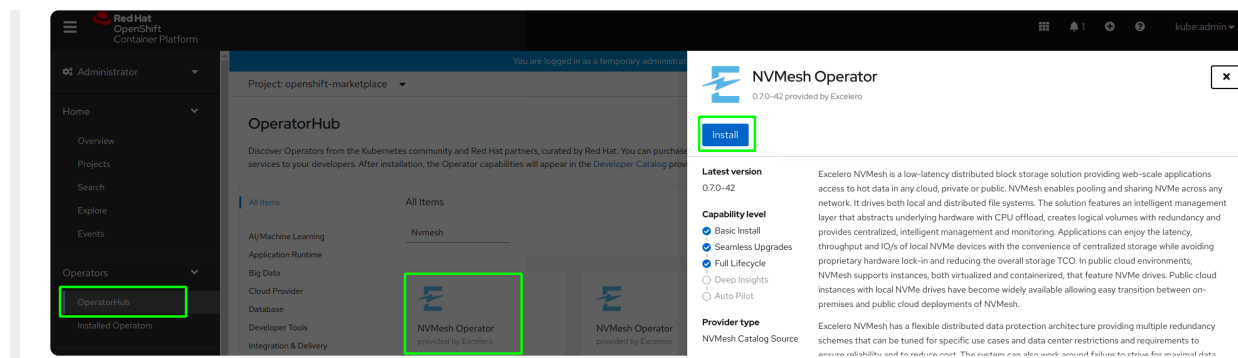
1. Obtain YAMLS containing the required secrets for repository access from sales@excelero.com.
2. Import the secrets YAMLS, see [Installing YAMLS](#).

There are two secret YAMLS.

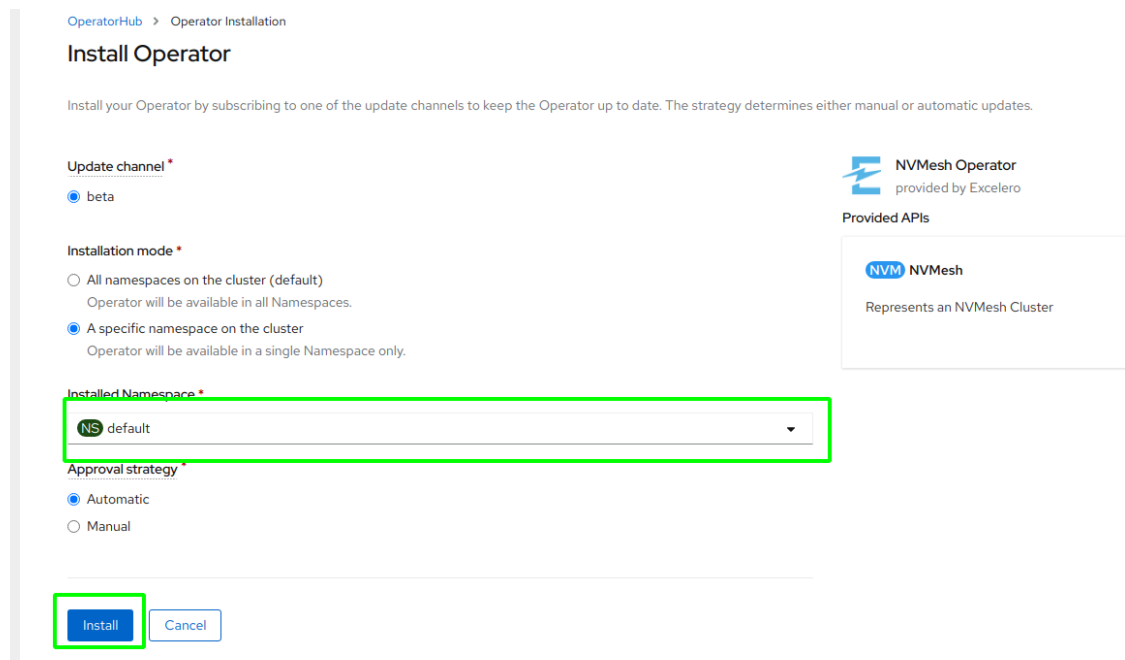
- The first provides access to Excelero's docker registry
- The second provides access to Excelero's RPM repositories

3.3. Operator Installation

Go to the OperatorHub page and search for **NVMesh**.



Install the operator in the default namespace.



Upon installation completion, the following page should appear.

**NVMesh Operator**

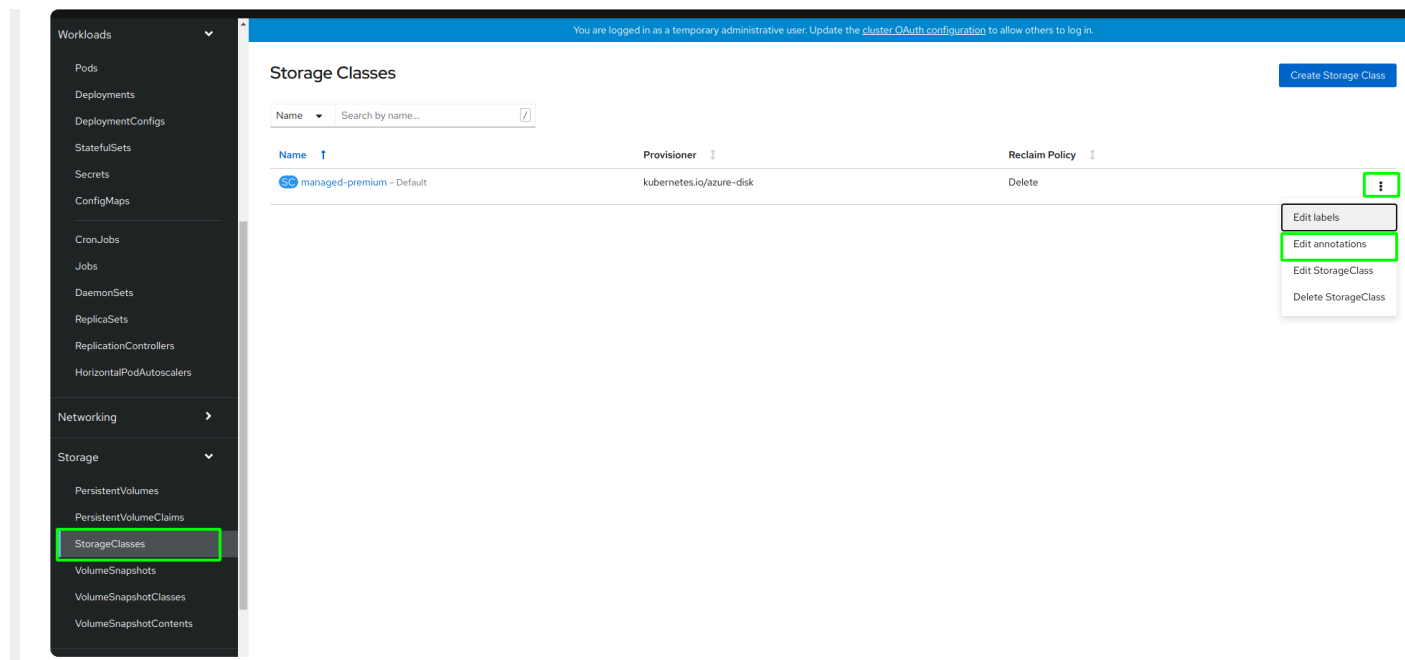
0.7.0-42 provided by Excelero

**Installed operator - ready for use**[View Operator](#)[View installed Operators in Namespace default](#)

3.4. Remove the Default Storage Class

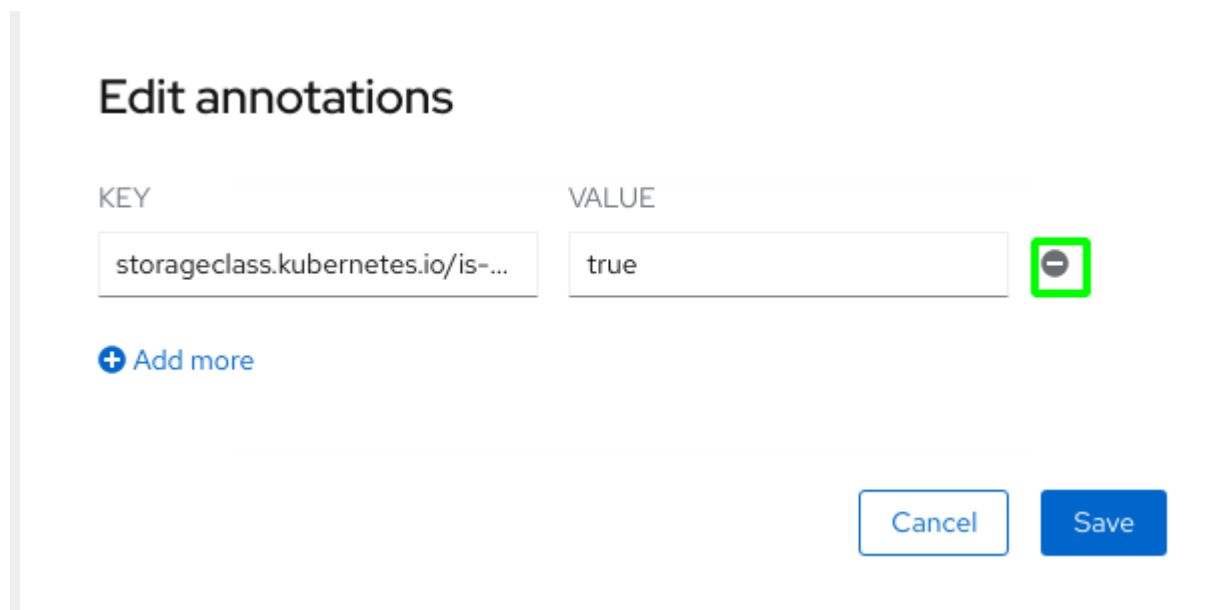
By default, Mongo PVCs will be created with a default storage class, which means they will be bound only to PVs created from this storage class. In order to bypass this, we will unset the default storage class.

Go to *Storage* and then *StorageClasses* using the left menu, click options for *managed-premium* storage class and click *Edit annotations* from the drop-down menu at the right.



The screenshot shows the Kubernetes dashboard interface. On the left sidebar, the 'Storage' section is expanded, and 'StorageClasses' is highlighted. The main content area displays the 'Storage Classes' page. At the top, there is a search bar and a 'Create Storage Class' button. Below this is a table with columns: Name, Provisioner, and Reclaim Policy. The table contains one entry: 'managed-premium - Default' with provisioner 'kubernetes.io/azure-disk' and reclaim policy 'Delete'. A dropdown menu is open for the first entry, showing options: 'Edit labels', 'Edit annotations' (highlighted), 'Edit StorageClass', and 'Delete StorageClass'.

Remove the only key defined by using the minus button and click *Save*.



The screenshot shows the 'Edit annotations' dialog. It has a title 'Edit annotations'. Below the title is a table with two columns: 'KEY' and 'VALUE'. The first row has KEY 'storageclass.kubernetes.io/is-...' and VALUE 'true'. A minus button is highlighted next to the VALUE field. Below the table is a '+ Add more' button. At the bottom right are 'Cancel' and 'Save' buttons.

3.5. Create an NFS Server Instance

All **NVMesh Management Servers** should have access to two shared volumes: one for Mongo and one for backups. This can be done by Azure files, NFS server, or any other persistent volume method.

The following uses the NFS method. To use another method or if an NFS server is already set, skip to [Add a Persistent Volume to the Cluster](#) stage.

Create an NFS Server Instance

Go to [Microsoft Azure – Resource Groups](#) and choose the resource group associated with the cluster (it should have a name starting with the cluster name). Click *Add*.

The screenshot shows the Azure portal interface for the resource group 'yoavopen-6jsb4-rg'. The 'Add' button is highlighted in the top navigation bar. The left sidebar shows the 'Compute' section selected. The main area displays a list of resources in the resource group, including storage accounts, images, load balancers, virtual machines, disks, and network interfaces, all located in West Europe.

Name	Type	Location
clusterqx5qf	Storage account	West Europe
imageregistryyoavop8c5ph	Storage account	West Europe
yoavopen-6jsb4	Image	West Europe
yoavopen-6jsb4	Load balancer	West Europe
yoavopen-6jsb4-a20536fcb48be4275833e3cabd8c9a2c	Public IP address	West Europe
yoavopen-6jsb4-identity	Managed identity	West Europe
yoavopen-6jsb4-internal	Load balancer	West Europe
yoavopen-6jsb4-master-0	Virtual machine	West Europe
yoavopen-6jsb4-master-0_OSDisk	Disk	West Europe
yoavopen-6jsb4-master-1	Virtual machine	West Europe
yoavopen-6jsb4-master-1_OSDisk	Disk	West Europe
yoavopen-6jsb4-master-2	Virtual machine	West Europe
yoavopen-6jsb4-master-2_OSDisk	Disk	West Europe
yoavopen-6jsb4-master0-nic	Network interface	West Europe
yoavopen-6jsb4-master1-nic	Network interface	West Europe
yoavopen-6jsb4-master2-nic	Network interface	West Europe
yoavopen-6jsb4-nsg	Network security group	West Europe

On the left, click *Compute* and then choose *Virtual Machine*.

Azure Marketplace [See all](#)

Get started

Recently created

AI + Machine Learning

Analytics

Blockchain

Compute

Containers

Databases

Developer Tools

DevOps

Identity

Integration

Internet of Things

IT & Management Tools

Media

Migration

Mixed Reality

Monitoring & Diagnostics

Networking

Security

Software as a Service (SaaS)

Storage

Web

Featured [See all](#)

Virtual machine

[Learn more](#)

Virtual machine scale set

[Learn more](#)

Kubernetes Service

[Quickstarts + tutorials](#)

Function App

[Quickstarts + tutorials](#)

Datadog (preview)

[Learn more](#)

Ubuntu Server 18.04 LTS

[Learn more](#)

Red Hat Enterprise Linux 8.2 (LVM)

[Learn more](#)

CentOS-based 8.2

[Learn more](#)

Debian 10 "Buster"

[Learn more](#)

Windows Server 2019 Datacenter

[Learn more](#)

Edit the virtual machine settings as follows:

- Virtual Machine name: NFS
- Region: same as the cluster
- Image: Ubuntu Server 18.04

- Size: a minimal machine should be sufficient, for instance “Standard B1ls (1 vcpu, 0.5 GiB memory)”
- Administrator account: it is easiest to use SSH and paste an existing public key

Create a virtual machine ...

Subscription * ⓘ

Resource group * ⓘ

[Create new](#)

Instance details

Virtual machine name * ⓘ

Region * ⓘ

Availability options ⓘ

Image * ⓘ

[See all images](#)

Azure Spot instance ⓘ ☐

Size * ⓘ

[See all sizes](#)

Administrator account

Authentication type ⓘ ☒ SSH public key

☐ Password

i Azure now automatically generates an SSH key pair for you and allows you to store it for future use. It is a fast, simple, and secure way to connect to your virtual machine.

Username * ⓘ

SSH public key source

SSH public key * ⓘ

i [Learn more about creating and using SSH keys in Azure](#)

Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports * ⓘ ☐ None

☒ Allow selected ports

Select inbound ports *

⚠ This will allow all IP addresses to access your virtual machine. This is only recommended for testing. Use the Advanced controls in the Networking tab to create rules to limit inbound traffic to known IP addresses.

[Review + create](#)

[< Previous](#)

[Next : Disks >](#)

Click *Next* and then *disks*.


Click *Next* and then *networking*.

Choose the worker subnet as the subnet on this machine will run the managements pods, and use a public IP (don't touch the field).

Create a virtual machine ...


Basics Disks **Networking** Management Advanced Tags Review + create

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution.

[Learn more](#) 

Network interface

When creating a virtual machine, a network interface will be created for you.

Virtual network * 


yoavopen-6jsb4-vnet

[Create new](#)

Subnet * 


yoavopen-6jsb4-worker-subnet (10.0.32.0/19)

[Manage subnet configuration](#)

Public IP 

(new) NFS-ip


[Create new](#)

NIC network security group 

- ☒ None
☐ Basic
☐ Advanced




The selected subnet 'yoavopen-6jsb4-worker-subnet (10.0.32.0/19)' is already associated to a network security group 'yoavopen-6jsb4-nsg'. We recommend managing connectivity to this virtual machine via the existing network security group instead of creating a new one here.

Accelerated networking 

☐

The selected VM size does not support accelerated network

Load balancing

You can place this virtual machine in the backend pool of an existing Azure load balancing solution. [Learn more](#) 

Place this virtual machine behind an existing load balancing solution?

☐

Click *review* and *create* and then *create* again and wait for the VM to start. When the machine will be created the following screen will appear. Click *Go-to resource*.

✓ Your deployment is complete

Deployment name: CreateVm-Canonical.UbuntuServer-18.04-LTS-2... Start time: 4/11/2021, 10:25:51 AM
 Subscription: Excelero 20210123 Correlation ID: 627fe7fb-eb85-4fee-bfb1-0107b68fc02f
 Resource group: yoavopen-6jsb4-rg

Deployment details (Download)

Next steps

Setup auto-shutdown Recommended

Monitor VM health, performance and network dependencies Recommended

Run a script inside the virtual machine Recommended

Go to resource Create another VM

If ssh times out, allow inbound port 22 to the VM. On the VM resource page, click *Networking* and add the rule.

NFS | Networking

Virtual machine

Search (Ctrl+J)

Attach network interface Detach network interface

nfs718

IP configuration (0)
 (ipconfig1 (Primary))

Network interface: **nfs718** Effective security rules Troubleshoot VM connection issues Topology
 Virtual network/subnet: yoavopen-6jsb4-vnet/yoavopen-6jsb4-worker-subnet NIC Public IP: **23.97.147.27** NIC Private IP: **10.0.32.6** Accelerated networking: **Disabled**

Inbound port rules Outbound port rules Application security groups Load balancing

Network security group yoavopen-6jsb4-nsg (attached to subnet: yoavopen-6jsb4-worker-subnet)
 Impacts 2 subnets, 0 network interfaces

Priority	Name	Port	Protocol	Source	Destination	Action
101	api-server_in	6443	TCP	Any	Any	Allow
500	a205368c48b0e4275833e3cabd8c9a2c-TCP-80-Internet	80	TCP	Internet	51.124.21.179	Allow
501	a205368c48b0e4275833e3cabd8c9a2c-TCP-443-Internet	443	TCP	Internet	51.124.21.179	Allow
511	Port_8080	22	TCP	Any	Any	Allow
65000	AllowInetInbound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancerInbound	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInbound	Any	Any	Any	Any	Deny

Add inbound port rule



Add inbound security rule



yoavopen-6jsb4-nsg

Source ⓘ

Any

Source port ranges * ⓘ

*

Destination ⓘ

Any

Service ⓘ

SSH

Destination port ranges ⓘ

22

Protocol

☐ Any☒ TCP☐ UDP☐ ICMP

Action

☒ Allow☐ Deny

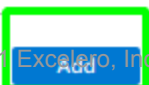
Priority * ⓘ

521

Name *

fdsfds ✓

Description



Cancel

On the machine run:

```
sudo apt install -y nfs-kernel-server
sudo mkdir -p /opt/nvmesh/backups
sudo mkdir -p /opt/nvmesh/mongo
sudo chown -R nobody:nogroup /opt/nvmesh/backups/
sudo chown -R nobody:nogroup /opt/nvmesh/mongo/
sudo chmod 777 /opt/nvmesh/backups/
sudo chmod 777 /opt/nvmesh/mongo/
```

Edit `/etc/exports` and add (with root permissions):

```
/opt/nvmesh/backups 10.0.32.0/24(rw,sync,no_subtree_check)
/opt/nvmesh/mongo 10.0.32.0/24(rw,sync,no_subtree_check)
```

Finally, run the following:

```
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
```

Run `ifconfig` to take the internal IP address of the machine and record it for future use.

3.6. Add a PV to the Cluster

Add a Shared Persistent Volume to the Cluster

Use the following YAML based on the NFS server created at the previous step. Change 10.0.32.6 to the internal IP recorded at the previous stage. If method other than NFS was used, then use the appropriate YAML for that method.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: data-volume-mongodb
  labels:
    role: mongo-for-nvmesh
spec:
  capacity:
    storage: 20Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: default
  nfs:
    server: 10.240.0.9
    path: /opt/nvmesh/mongo/
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nvmesh-backup-0
  labels:
    role: nvmesh-backups
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: default
  nfs:
    server: 10.240.0.9
```

```
path: /opt/nvmesh/backups
```

3.7. Install NVMesh Pods

The next step is to load the **NVMesh** objects to the cluster using the following YAML.

✿ Use the default project.

TCP Version

```
apiVersion: nvmesh.excelero.com/v1
kind: NVMesh
metadata:
  name: cluster1
spec:
  core:
    version: 2.2.0-490
    tcpOnly: true
    configuredNICs: eth0
    azureOptimized: true
  csi:
    controllerReplicas: 1
    version: v1.1.4-7
  management:
    imageRegistry: registry.excelero.com
  mongoDB:
    replicas: 1
    replicas: 1
    version: 2.2.0
```

Infiniband Version

```
apiVersion: nvmesh.excelero.com/v1
kind: NVMesh
metadata:
  name: cluster1
spec:
  core:
    version: 2.2.0-423-ib2
    tcpOnly: false
    configuredNICs: ib0
    azureOptimized: true
  csi:
```

```

controllerReplicas: 1
version: v1.1.4-7
management:
  imageRegistry: registry.excelero.com
mongoDB:
  replicas: 1
  replicas: 1
  version: 2.2.0

```

To validate, go to the *Workloads / Pods* page using the left menu. Check that the following pods are running or pending like in the following image.

Name	Status	Ready	Restarts	Owner	Memory	CPU	Created
mongo-0	Running	1/1	0	mongo	88.7 MiB	0.005 cores	3 minutes ago
nvmesh-csi-controller-0	Running	4/4	0	nvmesh-csi-controller	98.1 MiB	0.001 cores	3 minutes ago
nvmesh-management-0	Pending	0/0	0	nvmesh-management	-	-	3 minutes ago
nvmesh-operator-6b6f9f7f9-tjfhx	Running	1/1	0	nvmesh-operator-6b6f9f7f9	71.7 MiB	0.005 cores	Apr 11, 10:00 am

To start **Client**, **Target**, and **Management** pods, label the OpenShift workers accordingly using the OpenShift CLI, as follows:

- Create an OC token: click kube:admin and then Copy login command

- You may need to re-login with your cluster login/password, for instance if it timed out

- Click *Display Token* and copy the login command

Your API token is

sha256~wP3TJpTLllVe7QzwhY3soZFMmDNuvRwKkYYisGhhD8

Log in with this token

```
oc login --token=[REDACTED] --server=https://api.yoavopen.excelero.org:6443
```

Use this token directly against the API

```
curl -H "Authorization: Bearer sha256~wP3TJpTLllVe7QzwhY3soZFMmDNuvRwKkYYisGhhD8" "https://api.yoavopen.excelero.org:6443/apis/user.openshift.io/v1/users/~"
```

















































































[Request another token](#)

[Logout](#)

- On the local machine, run the login command

✿ **Note:** “oc” must be the path to the command downloaded during cluster install. Also, answer “y” to the insecure prompt if requested.

- Run `oc project default`
- Run `oc get nodes`
- There should now be 3 workers and 3 masters
- Tag **one** of the **workers** as **Management** by running: `oc label node <worker_name> nvmesh.excelero.com/nvmesh-management=""`. Change to the name from the previous stage).
- Now tag all **workers** as **Client** and **Target** using tags commands: `oc label node <worker_name> nvmesh.excelero.com/nvmesh-client="" && oc label node <worker_name> nvmesh.excelero.com/nvmesh-target=""`
- Go to the *Pods* page and validate that the **Client**, **Target**, and **Management** pods are up.

Name ↑	Status ↑	Ready ↑	Restarts ↑	Owner ↑	Memory ↑	CPU ↑	Created ↑	
 mongo-0	 Running	1/1	0	 mongo	-	-	 3 minutes ago	
 nvmesh-client-driver-container-6kmgc	 Running	1/1	0	 nvmesh-client-driver-container	-	-	 a minute ago	
 nvmesh-client-driver-container-7bqhd	 Running	1/1	0	 nvmesh-client-driver-container	-	-	 2 minutes ago	
 nvmesh-client-driver-container-thxkr	 Running	1/1	0	 nvmesh-client-driver-container	-	-	 a minute ago	
 nvmesh-csi-controller-0	 Running	4/4	0	 nvmesh-csi-controller	-	-	 Apr 28, 3:19 pm	
 nvmesh-csi-node-driver-g2nrr	 Running	2/2	0	 nvmesh-csi-node-driver	-	-	 a minute ago	
 nvmesh-csi-node-driver-pdlkc	 Running	2/2	0	 nvmesh-csi-node-driver	-	-	 2 minutes ago	
 nvmesh-csi-node-driver-x69lk	 Running	2/2	0	 nvmesh-csi-node-driver	-	-	 a minute ago	
 nvmesh-management-0	 Running	1/1	0	 nvmesh-management	-	-	 3 minutes ago	
 nvmesh-mcs-agent-bqx58	 Running	2/2	1	 nvmesh-mcs-agent	-	-	 a minute ago	
 nvmesh-mcs-agent-g428r	 Running	2/2	1	 nvmesh-mcs-agent	-	-	 2 minutes ago	
 nvmesh-mcs-agent-qmh2t	 Running	2/2	1	 nvmesh-mcs-agent	-	-	 a minute ago	
 nvmesh-operator-c4fbccdd94-zg0rv	 Running	1/1	0	 nvmesh-operator-c4fbccdd94	-	-	 Apr 28, 3:18 pm	
 nvmesh-target-driver-container-5cgvj	 Running	3/3	0	 nvmesh-target-driver-container	-	-	 a minute ago	
 nvmesh-target-driver-container-gwwrb	 Running	3/3	0	 nvmesh-target-driver-container	-	-	 a minute ago	
 nvmesh-target-driver-container-tc22s	 Running	3/3	0	 nvmesh-target-driver-container	-	-	 a minute ago	

3.8. Expose Management Routes

Add a *Route* to expose the **Management** pod UI into a public DNS using the following YAML. Replace `your-cluster` with `your-domain`.

```
kind: Route
apiVersion: route.openshift.io/v1
metadata:
  name: mgmt-gui
spec:
  host: ui-mgmt.apps.YOUR-CLUSTER.YOUR-DOMAIN
  to:
    kind: Service
    name: nvmesh-management-gui
    weight: 100
  port:
    targetPort: gui
  tls:
    termination: passthrough
    insecureEdgeTerminationPolicy: Redirect
  wildcardPolicy: None
```

The URL <https://ui-mgmt.apps.YOUR-CLUSTER.YOUR-DOMAIN> can be used login to the cluster.

If the OpenShift cluster is deployed on Azure, use the name of the Private DNS zone of your resource group.

4. Creating PVCs with NVMesh Storage

The following sections describe how to generate Persistent Volume Claims that will be stored on **NVMesh** volumes.

4.1. Format Drives

By default, all new drives should be automatically formatted once the NVMesh cluster is deployed.

Drives that were already formatted by a previous cluster deployment will not be formatted and will be automatically evicted. To erase any previous data and re-use the drives they will need to be manually formatted.

Manually Formatting Drives: Use the official guide and format drives, typically all, in the cluster, see [Format Drives](#).

Disable Automatic Formatting of Drives: Automatic formatting of drives can be disabled by adding the following flag to the NVMesh resource spec:

```
spec:
  management:
    disableAutoFormatDrives: true
```

4.2. Create PersistentVolumeClaims

Following is an example YAML for creating a *RAID-1* volume named *nvmesh-fast-storage*.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nvmesh-fast-storage
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 10Gi
  storageClassName: nvmesh-raid1
```

For full documentation on creating PVCs and StorageClasses see [NVMesh CSI Driver Guide – Usage](#).

5. NVMesh OpenShift Operator on Azure

This section provides instructions for deploying **NVMesh** on Microsoft Azure.

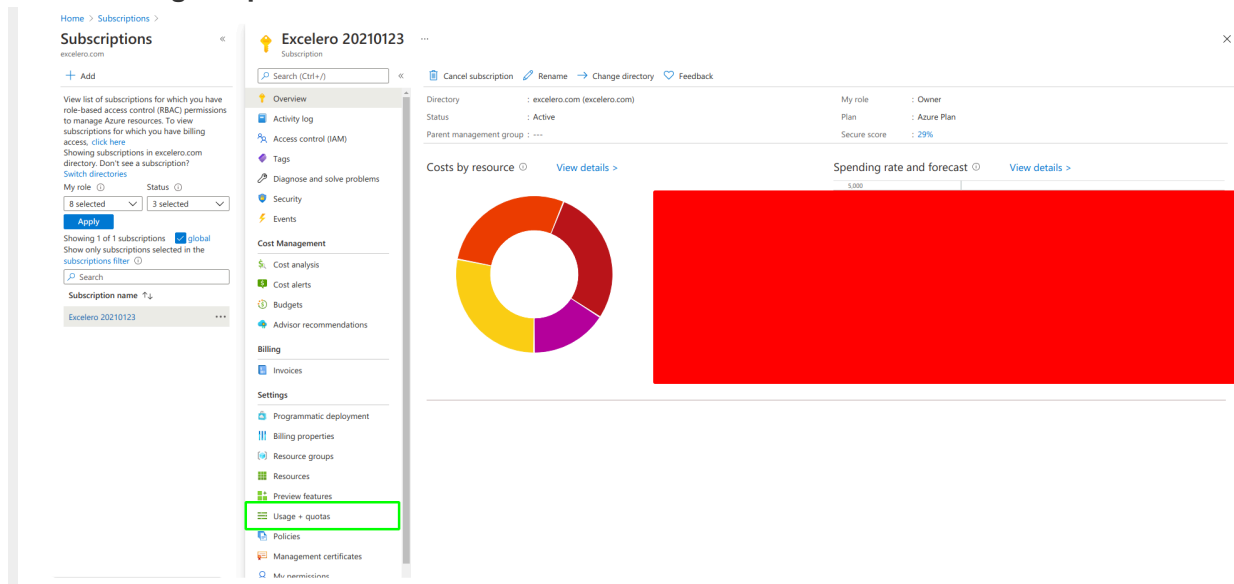
5.1. Prerequisites

This section describes the prerequisites for deploying **NVMesh** on Microsoft Azure.

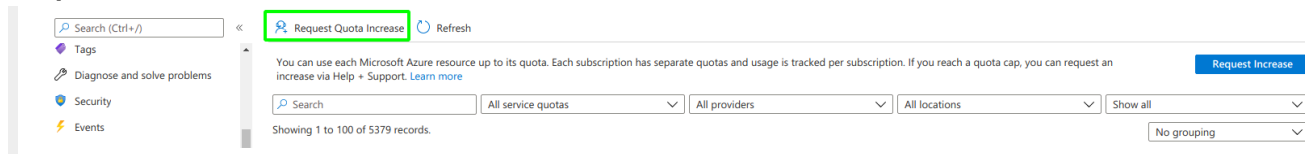
5.1.1. Resource Limits

Make sure your Azure subscription has at least the following resource limits:

1. Go to [Microsoft Azure Subscriptions](#) and choose your subscription
2. Choose **Usage + quotas** from the left sidebar



3. If any of the resource limits below are insufficient, see the table as reference, use the link at the top to **Request Quota Increase**



Component	Number of Components Required by Default	Default Azure Limit	Description
vCPUs	<ol style="list-style-type: none"> 1. D5v3 – 24. 8 each for 3 masters 2. Infiniband: HBv3 – 360. 120 each for 3 workers <p>or</p> <ol style="list-style-type: none"> TCP: L5v2 – 64. 32 each for 2 workers 3. D5v4 – 4 for the bootstrap machine 	20 per region	<p>A default cluster requires 40 vCPUs, so you must increase the account limit.</p> <p>By default, each cluster creates the following instances:</p> <ol style="list-style-type: none"> 1. 1 bootstrap

		<p>machine, which is removed after installation.</p> <p>2. 3 control plane machines.</p> <p>3. 3 compute machines.</p> <p>As the bootstrap machine uses D4s_v3 machines with 4 vCPUs, the control plane machines use D8s_v3 virtual machines with 8 vCPUs and the worker machines use D4s_v3 machines with 4 vCPUs, a default cluster requires 40 vCPUs.</p> <p>The bootstrap node VM, which uses 4 vCPUs, is used only during installation.</p> <p>To deploy more worker nodes, enable autoscaling, deploy large workloads or use a different instance type,</p>
--	--	---

			<p>increase the vCPU limit to ensure that the cluster can deploy the machines required.</p> <p>By default, the installation program distributes control plane and compute machines across all availability zones within a region. To ensure high availability for the cluster, select a region with at least 3 availability zones. If the region contains fewer than 3 availability zones, the installation program places more than one control plane machine in the available zones.</p>
Virtual Networks	1	1000 per region	Each default cluster requires a Virtual Network (VNet), which contains 2 subnets
Network	6	65,536 per region	Each default

Interfaces			cluster requires 6 network interfaces. If additional machines are created or workloads deployed create load balancers, the cluster uses more network interfaces.
Network Security Groups	2	5000	<p>Each cluster creates network security groups for each subnet in the VNet.</p> <p>The default cluster creates network security groups for the control plane and for the compute node subnets:</p> <p>controlplane: Allows the control plane machines to be reached on port 6443 from anywhere.</p> <p>node: Allows worker nodes to be reached from the Internet on ports 80 and 443.</p>
Network Load Balancers	3	1000 per region	Each cluster creates the following load

			<p>balancers:</p> <p>default – Public IP address that load balances requests to ports 80 and 443 across worker machines.</p> <p>internal – Private IP address that load balances requests to ports 6443 and 22623 across control plane machines.</p> <p>external – Public IP address that load balances requests to port 6443 across control plane machines.</p> <p>If applications create additional Kubernetes LoadBalancer service objects, the cluster will use additional load balancers.</p>
Public IP Addresses	3		<p>Each of the 2 public load balancers uses a public IP address. The bootstrap machine also</p>

			uses a public IP address so that SSH can be used to troubleshoot issues during installation. The IP address for the bootstrap node is used only during installation.
Private IP Addresses	7		The internal load balancer, each of the 3 control plane machines and each of the 3 worker machines each use a private IP address.

5.1.2. Roles and Permissions

The Microsoft Azure account must have the role **User Access Administrator** for the subscription.

To validate:

1. Go to [Microsoft Azure Subscription](#) and choose the subscription
2. On the left bar click **Access control (IAM)**
3. Choose “**Role Assignments**” and check if the user is shown under **User Access Administrator**

For additional information on how to assign roles, see [Assign Azure roles using the Azure portal](#).

The user should also be an **Application Administrator**.

To validate:

1. Go to [Microsoft Azure – All users](#)
2. Click the username
3. Click **Assigned Roles** in the left sidebar

If the user is not an **Application Administrator**, contact the Azure Admin to assign the role.

5.1.3. Public DNS Zone

A Public DNS Zone in Azure is also a prerequisite. Use the following steps, originally from the OpenShift tutorial, to create one.

- Identify your domain, or subdomain, and registrar. You can transfer an existing domain and registrar or obtain a new one through Azure or another source.

For more information about purchasing domains through Azure, see [Buy a custom domain name for Azure App Service](#) in the Azure documentation.

- If you are using an existing domain and registrar, migrate its DNS to Azure. See [Migrate an active DNS name to Azure App Service](#) in the Azure documentation.
- Configure DNS for your domain. Follow the steps in the [Tutorial: Host your domain in Azure DNS](#) in the Azure documentation to create a public hosted zone for your domain or subdomain, extract the new authoritative name servers, and update the registrar records for the name servers that your domain uses.
Use an appropriate root domain, such as openshiftcorp.com, or subdomain, such as clusters.openshiftcorp.com.
- If you use a subdomain, follow your company's procedures to add its delegation records to the parent domain.

5.1.4. Azure CLI

Azure CLI is also a prerequisite. Following these steps to install it.

1. To install Azure CLI on a local machine, see [Install the Azure CLI for Linux manually](#)
2. Create a user or login to an existing one on the Red Hat portal <https://cloud.redhat.com/>
3. Follow step 1 only as described here, <https://cloud.redhat.com/openshift/install/azure/installer-provisioned>. Download the OpenShift CLI (OC) and openshift-install and record the pull secret for later. Untar OC and openshift-install using `tar xzvf <filenames>`.

5.2. OpenShift Cluster on Azure

This section describes how to deploy OpenShift cluster on Azure and how to subsequently access it.



For setups working with Infiniband mode, change the definitions manually as described in section [Infiniband Only](#).

5.2.1. Deploying the Cluster

Connect to Azure using its CLI.

1. Most often this can be done from a local computer shell.
2. Use `az login` to begin the connection.
3. Follow the shell steps to complete the login process.

Verify that the expected Azure subscriptions are accessible.

1. Use `az account list --refresh` to see all available subscriptions, for example as follows.

```
1  [
2    {
3      "cloudName": "AzureCloud",
4      "homeTenantId": "xxxxx-x-x-x--x-x-x-x-",
5      "id": "xxxxxxxxxxxxxxxxxxxxxx",
6      "isDefault": true,
7      "managedByTenants": [],
8      "name": "Just a name",
9      "state": "Enabled",
10     "tenantId": "xxxxx-xxxxxxxx-xxxx-xxxx-xxx",
11     "user": {
12       "name": "xxxx@excelero.com",
13       "type": "user"
14     }
15   }
16 ]
```

2. Use `az account set -s <id>` to choose a specific subscription.

Run `az account show`.

1. Record the values of `tenantId` and `id` for future use.

Create a service principal, which is needed for each cluster using `az ad sp create-for-rbac --role Contributor --name <service_principal_name>`.

1. Make a note of the values for `appId` and `password` from the output for future use.
2. **Note:** The error, “When using this permission, the backing application of the service principal being created must in the local tenant” seems like a transient bug. Rerun the command until it works.

Grant permissions to the created Server Principal using the `appId` recorded above.

1. `az role assignment create --role "User Access Administrator" --assignee-object-id $(az ad sp show --id <service-principal-name> -o tsv --query objectId)`
2. `az ad app permission add --id <appId> --api 00000002-0000-0000-c000-000000000000 --api-permissions 824c81eb-e3f8-4ee6-8f6d-de7f50d565b7=Role`
3. `az ad app permission grant --id <appId> --api 00000002-0000-0000-c000-000000000000`

Choose one of the following YAMLs, TCP YAML or Infiniband YAML, and save it in the same folder as openshift-installer binary and name it `install-config.yaml`.

Example YAML for TCP-based environments

```
apiVersion: v1
baseDomain: <your_base_dns>
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    azure:
      type: Standard_L32s_v2
      osDisk:
        diskSizeGB: 512
      zones:
        - "1"
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: <your_desired_cluster_name>
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
    - 172.30.0.0/16
```

```
platform:
  azure:
    baseDomainResourceGroupName: nvmeshrg
    cloudName: AzurePublicCloud
    outboundType: Loadbalancer
    region: westeurope
publish: External
sshKey: <your_public_ssh_key(not_path)>
pullSecret: '<your_pull_secret(not_path)>'
```

Example YAML for Infiniband-based environments

```
apiVersion: v1
baseDomain: <your_base_dns>
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    azure:
      type: Standard_HB120rs_v3
      osDisk:
        diskSizeGB: 512
      zones:
        - "1"
  replicas: 0
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: 3
metadata:
  creationTimestamp: null
  name: <your_desired_cluster_name>
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  machineNetwork:
    - cidr: 10.0.0.0/16
  networkType: OpenShiftSDN
  serviceNetwork:
```

```

- 172.30.0.0/16
platform:
  azure:
    baseDomainResourceGroupName: nvmeshrg
    cloudName: AzurePublicCloud
    outboundType: Loadbalancer
    region: westeurope
publish: External
sshKey: <your_public_ssh_key(not_path)>
pullSecret: '<your_pull_secret(not_path)>'

```



No workers are created when working with Infiniband. This is on purpose, as openshift-install does not supported the availability set feature. Workers will be created later oo.

Edit the YAML file filling in the following.

1. your_base_dns – the public base DNS domain as configured in azure, for example excelero.org.
2. Set the number of workers or **NVMesh** nodes by changing 3 to any number bigger than 3.
3. your_desired_cluster_name.
4. your_public_ssh_key (not_path) – copy and paste a public key that will be installed on all openshift nodes.
5. your_pull_secret (not_path) – **keep the quotes** and replace the variable with copy-paste of the pull secret you download at the prerequisite stage.
6. region – can be any region from the following list that has enough limits as described in the prerequisite.
 - australiacentral (Australia Central)
 - australiaeast (Australia East)
 - australiasoutheast (Australia South East)
 - brazilsouth (Brazil South)
 - canadacentral (Canada Central)
 - canadaeast (Canada East)
 - centralindia (Central India)
 - centralus (Central US)
 - eastasia (East Asia)
 - eastus (East US)
 - eastus2 (East US 2)
 - francecentral (France Central)
 - germanywestcentral (Germany West Central)
 - japaneast (Japan East)
 - japanwest (Japan West)
 - koreacentral (Korea Central)
 - koreasouth (Korea South)

- northcentralus (North Central US)
- northeurope (North Europe)
- norwayeast (Norway East)
- southafricanorth (South Africa North)
- southcentralus (South Central US)
- southeastasia (Southeast Asia)
- southindia (South India)
- switzerlandnorth (Switzerland North)
- uaenorth (UAE North)
- uksouth (UK South)
- ukwest (UK West)
- westcentralus (West Central US)
- westeurope (West Europe)
- westindia (West India)
- westus (West US)
- westus2 (West US 2)

Run `rm -f ~/.azure/osServicePrincipal.json` to delete any previous service principal configuration on the local machine.

Run `./openshift-install create cluster --dir=./ --log-level=debug`. The process should take around 50 minutes and will provide an interactive shell.

1. Platform → choose `azure`
2. subscription id → paste the `id` recorded above
3. tenant id → paste the `tenantId` recorded above
4. service principal client id → paste `appId` recorded above
5. service principal client secret → paste `password` recorded above

Accelerate worker machine NICs once the cluster is up.

1. Go to [Microsoft Azure – Resource Groups](#)
2. Click the resource group with the cluster name defined
3. Search for **Network Interface** resources named: `-xxxx-worker-region-xxxx-nic`.
4. Click on the NIC and then click **Enabled accelerated networking** at the top.



Following are some known errors:

ERROR Error: authorization.RoleAssignmentsClient#Get: Failure responding to request: StatusCode=404 — Original Error: autorest/azure: Service returned an error. Status=404 Code="RoleAssignmentNotFound" Message="The role assignment '9f6023cc-81c9-7914-5c89-03cc7ea74ea1' is not found."

ERROR

ERROR on ../../tmp/openshift-install-935734395/main.tf line 161, in resource "azure_rm_role_assignment" "main":

ERROR 161: resource "azure_rm_role_assignment" "main" this can randomly happen after creating the cluster machines role_assignment: fix immediate read after write issue by dlamotte · Pull Request #9698 · terraform-providers/terraform-provider-azurerm due to a TF provider bug which was fixed. If happen destroy the cluster and rerun:

If there are errors, run `./openshift-install destroy cluster` and revert to the create cluster step.

5.2.2. Accessing the Cluster

When the install finishes, which typically takes around 50 minutes, output such as follows is expected.

```
DEBUG Cluster is initialized
INFO Waiting up to 10m0s for the openshift-console route to be created...
DEBUG Route found in openshift-console namespace: console
DEBUG OpenShift console route is admitted
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export KUBECONFIG=/home/YOURUSER/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-console.apps.YOURCLUSTER.YOURDOMAIN
INFO Login to the console with user: "xxx", and password: "gdfklgdfgdfgfd"
DEBUG Time elapsed per stage:
```

It should now be possible to login to the cluster using the link, user and password shown.

5.2.3. Infiniband Only

To use Infiniband mode, create the worker's VMs in the same **Availability Set** so that they have the same Infiniband pkey critical for Infiniband communications. This is not possible from openshift-installer. Instead, an ARM template is used.

1. Go to your openshift-install folder location and run `./openshift-install create ignition-configs`.
2. Run `cat ./worker.ign | base64 | tr -d '\n' > ignition_base64`.
3. Run `cat terraform.tfvars.json | grep cluster_id` and record your cluster_id/base name for future use.
4. Go to <https://portal.azure.com/#create/Microsoft.Template> and click **Build your own template in the editor**
5. Copy and paste the following YAML and click **save**.

```
{
  "$schema" : "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion" : "1.0.0.0",
  "parameters" : {
    "baseName" : {
      "type" : "string",
      "minLength" : 1,
      "metadata" : {
        "description" : "Base name to be used in resource names (usually the cluster's Infra ID)"
      }
    },
    "workerIgnition" : {
      "type" : "string",
      "metadata" : {
        "description" : "Ignition content for the worker nodes"
      }
    },
    "numberOfNodes" : {
      "type" : "int",
      "defaultValue" : 3,
      "minValue" : 1,
      "maxValue" : 30,
      "metadata" : {
        "description" : "Number of OpenShift compute nodes to deploy"
      }
    }
  }
}
```

```

"sshKeyData" : {
  "type" : "securestring",
  "metadata" : {
    "description" : "SSH RSA public key file as a string"
  }
},
"availabilitySetName": {
  "type" : "string",
  "metadata" : {
    "description" : "Availability Set Name"
  }
},
"nodeVMSize" : {
  "type" : "string",
  "defaultValue" : "Standard_HB120rs_v3",
  "allowedValues" : [
    "Standard_D2s_v3",
    "Standard_D4s_v3",
    "Standard_HB120rs_v3"
  ],
  "metadata" : {
    "description" : "The size of the each Node Virtual Machine"
  }
},
"variables" : {
  "location" : "[resourceGroup().location]",
  "virtualNetworkName" : "[concat(parameters('baseName'), '-vnet')]",
  "virtualNetworkID" : "[resourceId('Microsoft.Network/virtualNetworks', variables('virtualNetworkName'))]",
  "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
  "nodeSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/', variables('nodeSubnetName'))]",
  "infraLoadBalancerName" : "[parameters('baseName')]",
  "sshKeyPath" : "/home/capi/.ssh/authorized_keys",
  "identityName" : "[concat(parameters('baseName'), '-identity')]",
  "imageName" : "[concat(parameters('baseName'), '')]",
  "copy" : [
    {
      "name" : "vmNames",
      "count" : "[parameters('numberOfNodes')]",
      "input" : "[concat(parameters('baseName'), '-worker-', variables('location'), '-', copyIndex('vmNames', 1))]"
    }
  ]
}

```

```

    ]
  },
  "resources" : [
    {
      "type": "Microsoft.Compute/availabilitySets",
      "name": "[parameters('availabilitySetName')]",
      "apiVersion": "2019-03-01",
      "location": "[variables('location')]",
      "properties": {
        "platformFaultDomainCount": "3",
        "platformUpdateDomainCount": "5"
      },
      "sku": {
        "name": "Aligned"
      }
    },
    {
      "apiVersion" : "2019-05-01",
      "name" : "[concat('node', copyIndex())]",
      "type" : "Microsoft.Resources/deployments",
      "copy" : {
        "name" : "nodeCopy",
        "count" : "[length(variables('vmNames'))]"
      },
      "dependsOn" : [
        "[resourceId('Microsoft.Compute/availabilitySets', concat(parameters('availabilitySetName')))]"
      ],
      "properties" : {
        "mode" : "Incremental",
        "template" : {
          "$schema" : "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
          "contentVersion" : "1.0.0.0",
          "resources" : [
            {
              "apiVersion" : "2018-06-01",
              "type" : "Microsoft.Network/networkInterfaces",
              "name" : "[concat(variables('vmNames')[copyIndex()], '-nic')]",
              "location" : "[variables('location')]",
              "properties" : {
                "ipConfigurations" : [
                  {
                    "name" : "pipConfig",

```

```

        "properties" : {
            "privateIPAllocationMethod" : "Dynamic",
            "subnet" : {
                "id" : "[variables('nodeSubnetRef')]"
            },
            "loadBalancerBackendAddressPools" : [
                {
                    "id" : "[resourceId('Microsoft.Network/loadBalancers/back
ckendAddressPools',variables('infraLoadBalancerName'), variables('infraLoadBalanc
erName'))]"
                }
            ]
        }
    },
    {
        "apiVersion" : "2018-06-01",
        "type" : "Microsoft.Compute/virtualMachines",
        "name" : "[variables('vmNames')[copyIndex()]]",
        "location" : "[variables('location')]",
        "tags" : {
            "kubernetes.io-cluster-ffranzupi": "owned"
        },
        "identity" : {
            "type" : "userAssigned",
            "userAssignedIdentities" : {
                "[resourceID('Microsoft.ManagedIdentity/userAssignedIdentie
s/', variables('identityName'))]" : {}
            }
        },
        "dependsOn" : [
            "[concat('Microsoft.Network/networkInterfaces/', concat(variable
s('vmNames')[copyIndex()], '-nic'))]"
        ],
        "properties" : {
            "hardwareProfile" : {
                "vmSize" : "[parameters('nodeVMSize')]"
            },
            "osProfile" : {
                "computerName" : "[variables('vmNames')[copyIndex()]]",
                "adminUsername" : "capi",
                "customData" : "[parameters('workerIgnition')]"
            }
        }
    }
]

```

```

        "linuxConfiguration" : {
            "disablePasswordAuthentication" : true,
            "ssh" : {
                "publicKeys" : [
                    {
                        "path" : "[variables('sshKeyPath')]",
                        "keyData" : "[parameters('sshKeyData')]"
                    }
                ]
            }
        },
        "storageProfile" : {
            "imageReference": {
                "id": "[resourceId('Microsoft.Compute/images', variables('imageName'))]"
            },
            "osDisk" : {
                "name": "[concat(variables('vmNames')[copyIndex()], '_OSDisk')]",
                "osType" : "Linux",
                "createOption" : "FromImage",
                "managedDisk": {
                    "storageAccountType": "Premium_LRS"
                },
                "diskSizeGB": 512
            }
        },
        "networkProfile" : {
            "networkInterfaces" : [
                {
                    "id" : "[resourceId('Microsoft.Network/networkInterfaces', concat(variables('vmNames')[copyIndex()], '-nic'))]",
                    "properties": {
                        "primary": true
                    }
                }
            ]
        },
        "availabilitySet": {
            "id": "[resourceId('Microsoft.Compute/availabilitySets', parameters('availabilitySetName'))]"
        }
    }
}

```

```
    }  
  ]  
}  
}  
}  
]  
}
```

Now fill in the following elements.

- **Subscription**
- **Resource Group**
- **Region** – Choose the same region used for the OpenShift cluster
- **Base Name** – The cluster_id recorded above
- **Worker Ignition** – The content of the **ignition_base64** file created above
- **Number Of Nodes** – 3
- **Ssh Key Data** – the public ssh key itself
- **Availability Set Name** – choose any name
- **Node VM Size** – choose Standard_HB120rs_v3

[Home](#) >

Custom deployment

Deploy from a custom template

Select a template **Basics** Review + create

Template

Customized template 
2 resources Edit template Edit parameters

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Excelero 20210123



Resource group * ⓘ

hpc-ocp-nvmesh32-jsv2j-rg

[Create new](#)

Instance details

Region * ⓘ

West Europe

Base Name * ⓘ

hpc-ocp-nvmesh32-jsv2j ✓

Worker Ignition * ⓘ

eyJpZ25pdGlvbil6eyJjb25maWciOnsibWVyZ2UiOlt7InNvdXJjZSI6Imh0... ✓

Number Of Nodes ⓘ

3

Ssh Key Data * ⓘ

..... ✓

Availability Set Name * ⓘ

example ✓

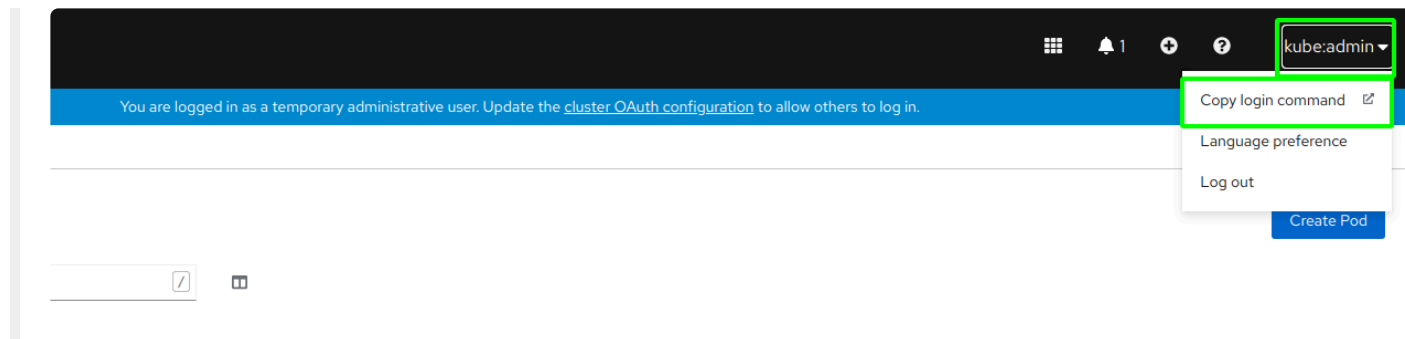
Node VM Size ⓘ

Standard_HB120rs_v3

Click **Review+Create** and then **Create**.

When the creation process ends, accept the new workers VMs using the OC CLI.

1. Create the oc token, click **kube:admin** and then copy the login command.



You may need to re-login with the cluster login/password if it timed out.
Click **Display Token** and copy the login command.



On a local machine, run the login command. Note: the “oc” command should be in the standard command PATH from the cluster install. Also, you may need to answer “y” to the insecure prompt.

1. Run `oc project default`
2. Run `oc get nodes`

Use the following link which describes how to accept new nodes to OpenShift cluster:

https://docs.openshift.com/container-platform/4.7/installing/installing_azure/installing-azure-user-infra.html#installation-approve-csrs_installing-azure-user-infra, number of new nodes should match the number of workers.

The **NVMesh** tracer requires $5 * 4 * \text{num_of_cpus}$ threads process IDs, increase the PID limit of the workers. Use the following guide to increase it to 4096 to be on the safe side, [How to change the value of pids_limit in OpenShift 4.x](#)

6. NVMesh on Azure Kubernetes Service

For an overview of Azure Kubernetes Service, hereon AKS, see this [link](#).

The following steps outline how to initialize an AKS cluster for use with **NVMesh**. Beyond that, the instructions from the previous sections apply.

Specifically, see the prerequisites for [NVMesh Openshift Operator on Azure Prerequisites](#). All sections apply, except that the Azure CLI should be built-in already.

6.1. Quick Summary

Action	Command
Login	<code>az login</code>
Verify login	<code>az account list --refresh</code>
Optional: choose a different account	<code>az account set -s <id></code>
Create a resource group	<code>az group create --name <resourceGroupName> --location <region></code>
Optional: install kubectl	<code>sudo az aks install-cli</code>
Create the AKS cluster	<code>az aks create --resource-group <resourceGroupName> --name <clusterName> --node-count <node-count> --generate-ssh-keys</code>
Update local credentials for the newly created cluster	<code>az aks get-credentials --resource-group <resourceGroupName> --name <clusterName></code>
Verify AKS cluster creation	<code>kubectl get nodes</code>
Create a proximity placement group	<code>az ppg create -n <ppgName> -g <resourceGroupName> -l <region> -t standard</code>
Check node availability by zone	<code>az vm list-skus -l eastus2 --zone --size "Standard_L48s_v2"</code>
Deploy a node pool for Targets	<code>az aks nodepool add --resource-group myResourceGroup --cluster-name myAKSCluster --name <nodepool name> --node-vm-size Standard_L48s_v2 --node-count <node-count> --ppg <myPPGResourceID> --labels nvmesh.excelero.com/nvmesh-management="" nvmesh.excelero.com/nvmesh-client="" nvmesh.excelero.com/nvmesh-target="" --zones <zone-id></code>

Import YAMLs with NVMesh 2.4.1 operator secrets	
Install and access the NVMesh 2.4.1 operator	<pre>git clone git@gitlab.excelero.com:excelero/openshift-operator.git cd openshift-operator</pre>
Deploy the NVMesh 2.4.1 operator	<pre>kubectl apply -f deploy/</pre>
Verify the operator deployment	<pre>kubectl get pods</pre>
Deploy NVMesh 2.4.1 pods	<pre>kubectl apply -f deploy/samples/nvmesh/nvmesh_v1_AKS_tcp.yaml</pre>
Verify the pods deployment	<pre>kubectl get pods --watch or watch -d kubectl get pods -o wide</pre>
Generate an NVMesh 2.4.1 volume	<pre>kubectl apply -f <volume.yaml></pre>
List PVCs	<pre>kubectl get pvcs</pre>
List PVs	<pre>kubectl get pvs</pre>
Performance Tuning	<pre>kubectl edit configmap nvmesh-core-config – edit the configuration kubectl delete ds nvmesh-client nvmesh-target nvmesh-mcs-agent – apply the configuration</pre>

6.2. Deploying the AKS Cluster

It is assumed that the user is acquainted with [Azure CLI](#).

Login via the Azure CLI

```
az login
```

Follow the instructions in the shell to complete the login.

Verifying access to the Azure subscriptions

```
az account list --refresh
```

This will show all available subscriptions, for example, as follows.

```
1  [  
2    {  
3      "cloudName": "AzureCloud",  
4      "homeTenantId": "xxxxx-x-x-x--x-x-x-x-",  
5      "id": "xxxxxxxxxxxxxxxxxxxxx",  
6      "isDefault": true,  
7      "managedByTenants": [],  
8      "name": "Just a name",  
9      "state": "Enabled",  
10     "tenantId": "xxxxx-xxxxxxxx-xxxx-xxxx-xxx",  
11     "user": {  
12       "name": "xxxx@excelero.com",  
13       "type": "user"  
14     }  
15   }  
16 ]
```

Use `az account set -s <id>` to choose a specific subscription.

Creating a Resource Group

```
az group create --name <resourceGroupName> --location <region>
```

Example output follows:

```
3  ✓ {
4    "id": "/subscriptions/cf41518e-baf3-4748-9ebd-7b2c35f34207/resourceGroups/myResourceGroup",
5    "location": "eastus",
6    "managedBy": null,
7    "name": "myResourceGroup",
8  ✓  "properties": {
9    "provisioningState": "Succeeded"
10  },
11  "tags": null,
12  "type": "Microsoft.Resources/resourceGroups"
13 }
```

Creating a Proximity Placement Group

```
az ppg create -n <ppgName> -g <resourceGroupName> -l <region> -t standard
```

Creating an AKS Cluster

```
az aks create --resource-group <resourceGroupName> --name <clusterName> --node-count 3 --generate-ssh-keys --ppg <ppgResourceID>
```

If `kubectl` is not installed, it can be easily installed using:

```
sudo az aks install-cli
```

To verify that it was created, use:

```
kubectl get nodes
```

6.2.1. Login via the Azure CLI

```
az login
```

Follow the instructions in the shell to complete the login.

6.2.2. Verify Azure Subscription Access

```
az account list --refresh
```

This will show all available subscriptions, for example, as follows.

```
1  [  
2    {  
3      "cloudName": "AzureCloud",  
4      "homeTenantId": "xxxxx-x-x-x--x-x-x-x-",  
5      "id": "xxxxxxxxxxxxxxxxxxxx",  
6      "isDefault": true,  
7      "managedByTenants": [],  
8      "name": "Just a name",  
9      "state": "Enabled",  
10     "tenantId": "xxxxx-xxxxxxxx-xxxx-xxxx-xxx",  
11     "user": {  
12       "name": "xxxx@excelero.com",  
13       "type": "user"  
14     }  
15   }  
16 ]
```

Use `az account set -s <id>` to choose a specific subscription.

6.2.3. Create a Resource Group

```
az group create --name <resourceGroupName> --location <region>
```

Example output follows:

```
3  ∨ {
4    "id": "/subscriptions/cf41518e-baf3-4748-9ebd-7b2c35f34207/resourceGroups/myResourceGroup",
5    "location": "eastus",
6    "managedBy": null,
7    "name": "myResourceGroup",
8  ∨  "properties": {
9      "provisioningState": "Succeeded"
10   },
11   "tags": null,
12   "type": "Microsoft.Resources/resourceGroups"
13 }
```

6.2.4. Create the Cluster

```
az aks create --resource-group <resourceGroupName> --name <clusterName> --node-count <node-count> --generate-ssh-keys
```

At a minimum, node-count should be 3.

Update the local credentials to match the newly created cluster using:

```
az aks get-credentials --resource-group <resourceGroupName> --name <clusterName>
```

If `kubectl` is not installed, it can be easily installed using:

```
sudo az aks install-cli
```

To verify that the cluster was created, use:

```
kubectl get nodes
```

6.3. Deploying NVMesh on AKS

6.3.1. Set up Repository Access

To access the **NVMesh** repository, import access credentials.

1. Obtain YAMLs containing the required secrets for repository access from sales@excelero.com.
2. Apply the secrets YAMLs using kubectl:

```
kubectl apply -f nvmesh_secrets.yaml
```

There are two secret YAMLs.

- The first provides access to Excelero's docker registry
- The second provides access to Excelero's RPM repositories

6.3.2. Deploy Node Pool for Targets

Targets are deployed using a node pool.

They can be either in one availability zone or across two.

A PPG (proximity placement group) is needed per AZ (availability zone) used.

To create the PPG, use:

```
az ppg create -n <ppgName> -g <resourceGroupName> -l <region> -t standard
```

Then deploy a node pool per AZ with the PPG id returned from the previous command, as follows:

```
az aks nodepool add --resource-group myResourceGroup --cluster-name myAKSCluster  
--name <nodepool name> --node-vm-size Standard_L48s_v2 --node-count <node-count>  
--ppg <myPPGResourceID> --labels nvmesh.excelero.com/nvmesh-management="" nvmes  
h.excelero.com/nvmesh-client="" nvmesh.excelero.com/nvmesh-target="" --zones <zon  
e-id>
```

The node count should be 4 for a single AZ.

For cross-AZ, the node count should be 2 for each AZ. To ensure failover, add to the node pool another node of any type that will be used as an arbiter. Contact [Excelero Technical Support](#) for more in-depth instructions for ensuring failover.

For more storage, simply add additional node pools.

To know in which zones there are nodes of this type, use:

```
az vm list-skus -l eastus2 --zone --size "Standard_L48s_v2"
```

This will provide a response, see the following example excerpt, that lists the zones where the nodes are available.

```
"locationInfo": [  
  {  
    "location": "eastus2",  
    "zoneDetails": [  
      {  
        "Name": [  
          "3",
```

```
        "2",
        "1"
    ],
    "capabilities": [
        {
            "name": "UltraSSDAvailable",
            "value": "True"
        }
    ],
    "name": null
}
],
"zones": [
    "3",
    "2",
    "1"
]
}
],
"locations": [
    "eastus2"
],
"name": "Standard_L48s_v2",
"resourceType": "virtualMachines",
"restrictions": [],
"size": "L48s_v2",
"tier": "Standard"
```

6.3.3. Get the NVMesh Operator

To obtain the **NVMesh** operator, run the following command

```
git clone git@gitlab.excelero.com:excelero/openshift-operator.git
```

Upon completion, a new directory name `openshift-operator` will contain files for operator deployment.

Alternatively, this step can be skipped by referring to operator files using the gitlab URL. Examples will be given in the next section.

6.3.4. Deploy the NVMesh Operator

To deploy the **NVMesh** operator, use the following if the git was cloned and after choosing the appropriate git branch.

```
kubectl apply -f deploy/
```

Alternatively, use a command such as the following to deploy without cloning the entire repository:

```
kubectl apply -f https://raw.githubusercontent.com/Excelero/nvmesh-k8s-operator/v0.8.0/deploy/operator.yaml
```

Note that version v0.8.0 was chosen by putting it in the appropriate URL location.

Then, verify that the operator pods have been created using:

```
kubectl get pods
```

For example:

```
[tomzan@localhost openshift-operator]$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nvmesh-operator-8548f59f4c-qcn2h	1/1	Running	0	24s

6.3.5. Deploy NVMesh Pods

Deploying the pods implementing **NVMesh** is typically done using the following command from the `nvmesh-operator` directory mentioned in the previous sections.

```
kubectl apply -f deploy/samples/nvmesh/nvmesh_v1_AKS_tcp.yaml
```

or as follows if the repository has not been cloned:

```
kubectl apply -f https://raw.githubusercontent.com/Excelero/nvmesh-k8s-operator/v0.8.0/deploy/samples/nvmesh/nvmesh_v1_AKS_tcp.yaml
```

The built-in sample deploys using PremiumSSD managed disks for storing system configuration information. Other than changing the cluster name, it is recommended to contact [Excelero Technical Support](#) for any other changes.

Verify the pods have started using the following command. It may take a few minutes.

```
kubectl get pods --watch
```

or using

```
watch -d kubectl get pods -o wide
```

Sample output:

```
server:~/AKS_example/openshift-operator$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mongo-0	1/1	Running	0	33m
nvmesh-client-dp84f	1/1	Running	0	85s
nvmesh-client-tj9pw	1/1	Running	0	98s
nvmesh-client-zvmq2	1/1	Running	0	92s
nvmesh-csi-controller-0	4/4	Running	0	33m
nvmesh-csi-node-driver-2rgtx	2/2	Running	0	85s
nvmesh-csi-node-driver-92cn6	2/2	Running	0	92s
nvmesh-csi-node-driver-98v9b	2/2	Running	0	98s
nvmesh-management-0	1/1	Running	0	33m

nvmesh-mcs-agent-ffb1s	2/2	Running	0	91s
nvmesh-mcs-agent-kdtsq	2/2	Running	0	85s
nvmesh-mcs-agent-ltx7q	2/2	Running	0	98s
nvmesh-operator-8548f59f4c-n4cmg	1/1	Running	0	33m
nvmesh-target-kk8gh	3/3	Running	0	73s
nvmesh-target-vcbd9	3/3	Running	0	61s
nvmesh-target-zpmbt	3/3	Running	0	67s

6.3.6. Deploy Node Pool for Compute / Clients

NVMesh 2.4.1 Clients are deployed on nodes or a node pool used for compute.

For example, to deploy a node pool while labeling the compute nodes for access to **NVMesh 2.4.1** and the storage in the **Targets** node pool, run:

```
az aks nodepool add --resource-group <resourceGroupName> --cluster-name <clusterName> --name d32spool --node-vm-size Standard_D32s_v3 --node-count 1 [ --ppg <ppgName> ] --labels nvmesh.excelero.com/nvmesh-client=""
```

Verify the pods have started using the following command. It may take a few minutes.

```
kubectl get pods --watch
```

or using

```
watch -d kubectl get pods -o wide
```



It is also possible to tag individual nodes instead of or in addition to node pools.

6.3.7. Create PVCs (Persistent Volume Claims)

Use the following command to create a PVC

```
kubectl apply -f <volume.yaml>
```

For example:

```
kubectl apply -f raid10-volume.yaml
```

For non-protected volumes, use a YAML such as this:

```
server:~/AKS_example$ cat raid0-volume.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: block-pvc-r0
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 10Gi
  storageClassName: nvmesh-raid0
```

The following example is for protected volumes:

```
server:~/AKS_example$ cat raid1-volume.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: block-pvc
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
```

```
resources:
  requests:
    storage: 10Gi
storageClassName: nvmesh-raid1
```

The following example is for protected volumes with higher performance by using more drives in parallel:

```
server:~/AKS_example$ cat raid10-volume.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: block-pvc-r10
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 10Gi
  storageClassName: nvmesh-raid10
```

After creating one PVC of each of the examples, verification will show the following:

```
server:~/AKS_example$ kubectl get pvc
NAME                                STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
backups-nvmesh-management-0p-0     Bound    nvmesh-backup-0                          5Gi        RWO            default        43h
block-pvc-a                         Bound    pvc-632172ae-5ac9-49f3-8cba-e63edc283c2  10Gi       RWX            nvmesh-raid1   40h
block-pvc-r02                      Bound    pvc-7fc781e7-e744-4f0e-bac2-ec272c96462  10Gi       RWX            nvmesh-raid0   23h
block-pvc-r105                    Bound    pvc-017fa85e-ce4c-4ff5-96a4-a02d0b1c2cc  10Gi       RWX            nvmesh-raid10  23h
data-volume-mongo-0b              Bound    data-volume-mongod                        20Gi       RWO            default        43h
```

6.4. Performance Tuning

Performance tuning and changing other configuration parameters is often done via `/etc/modprobe.d` files.

These parameters are typically set in the YAML used to deploy the node pool **Targets**, but they may need to be altered later on.

In the AKS environment, use the following command to edit the configuration parameters typically tuned via options in such files:

```
kubectrl edit nvmesh cluster1
```

and edit the field **spec.core.moduleParams**.

For an example for the contents, see https://github.com/Excelero/nvmesh-k8s-operator/blob/v0.8.0/deploy/samples/nvmesh/nvmesh_v1_AKS_tcp.yaml.