NVMesh Kubernetes Operator Guide

2.5 — Last update: 22 February 2022

Excelero, Ltd.

Table of Contents

1. Copyright and Trademark Information

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

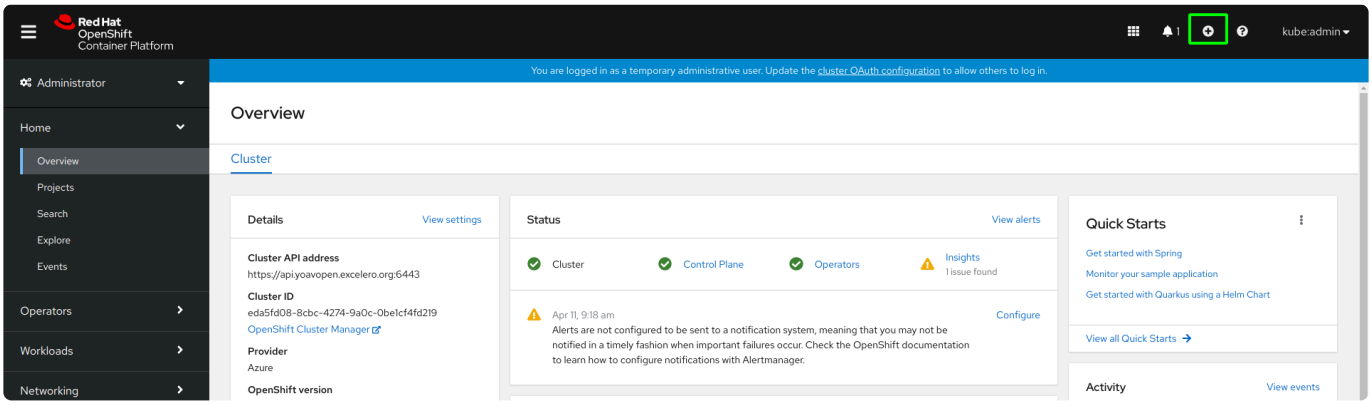
In this part we will describe the full procedure for installing NVMesh operator and running NVMesh on the OpenShift cluster

3.1. YAML Installation

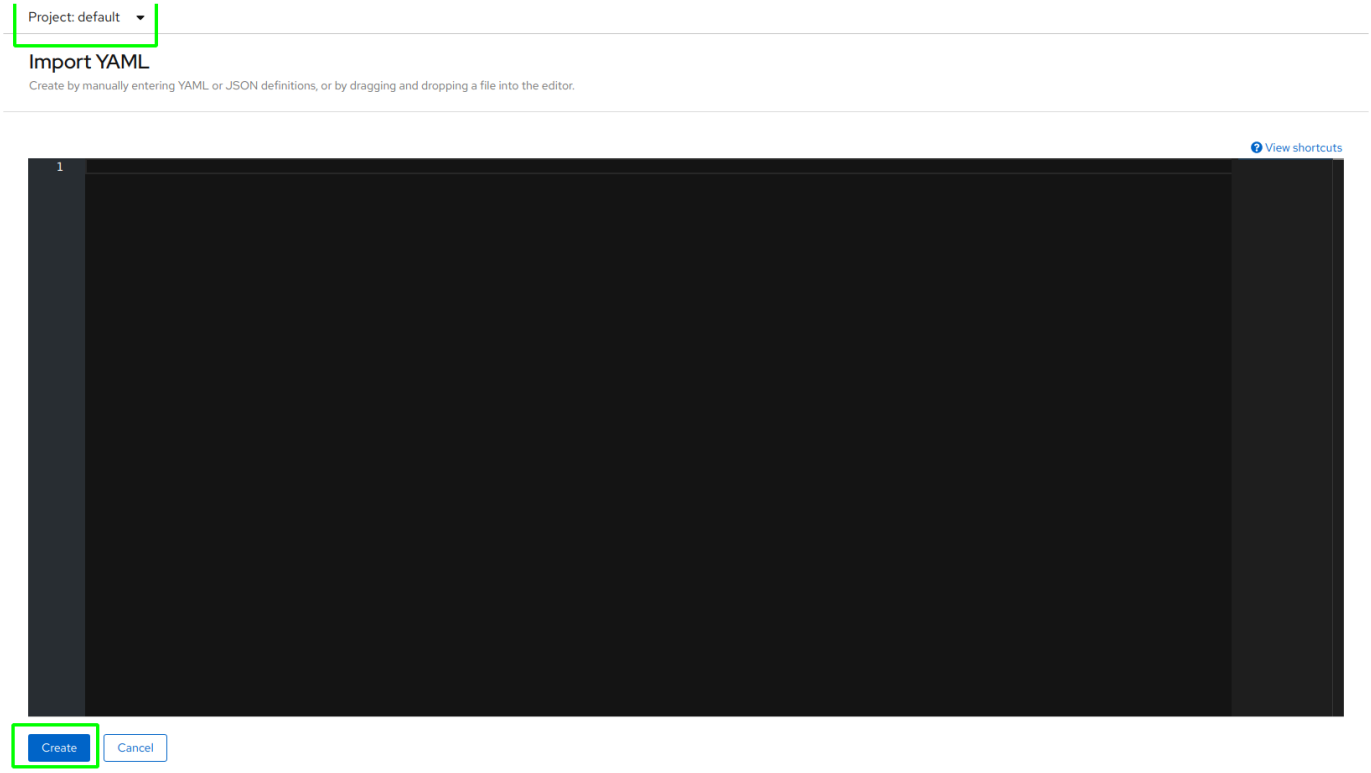
Installing YAMLs is a common procedure when working with OpenShift.

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.

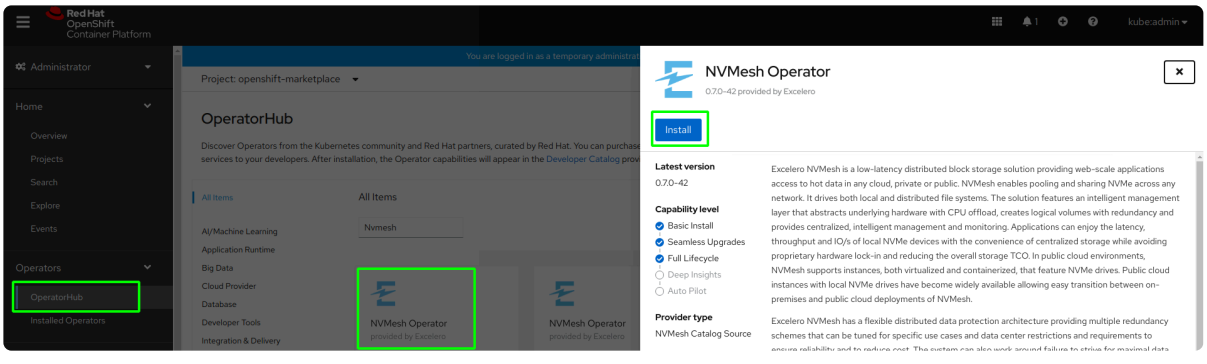
1. Obtain YAMLs containing the required secrets for repository access from sales@excelero.com.
2. Import the secrets YAMLs, see Installing YAMLs.

There are two secret YAMLs.

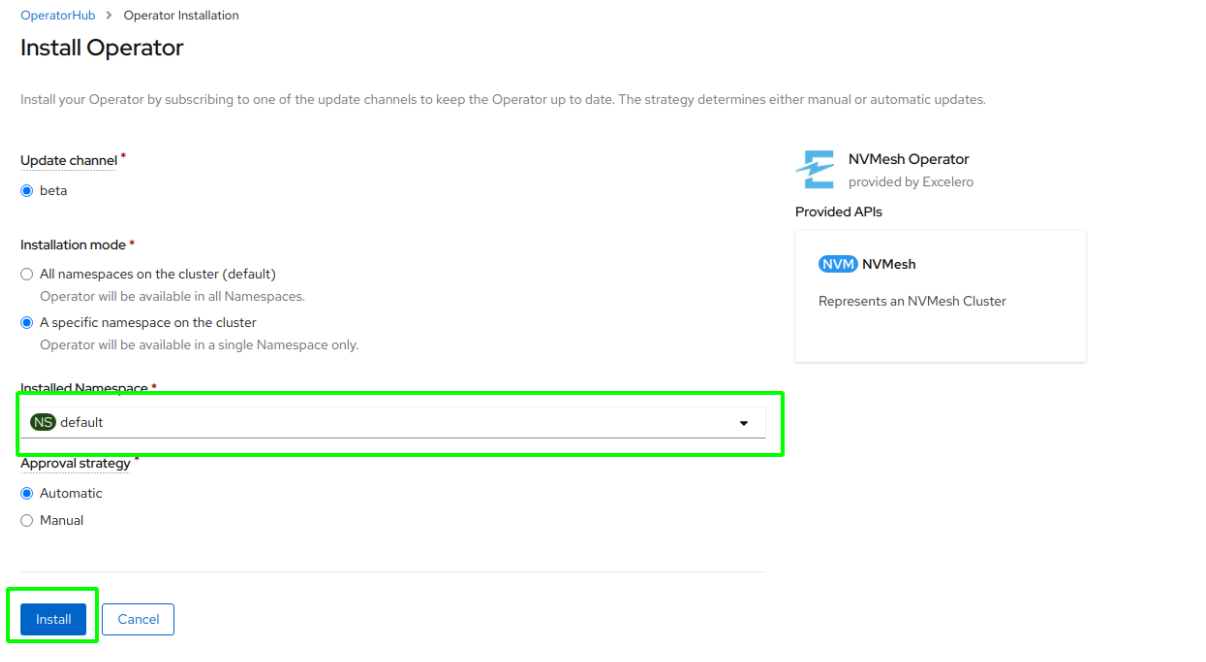
- The first provides access to Excelero's docker registry
- The second provides access to Excelero's RPM repositories

3.3. Operator Installation

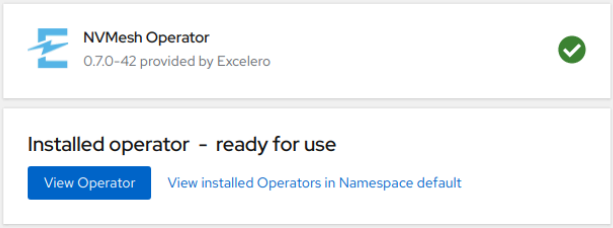
Go to the OperatorHub page and search for ***NVMesh***.



Install the operator in the default namespace.

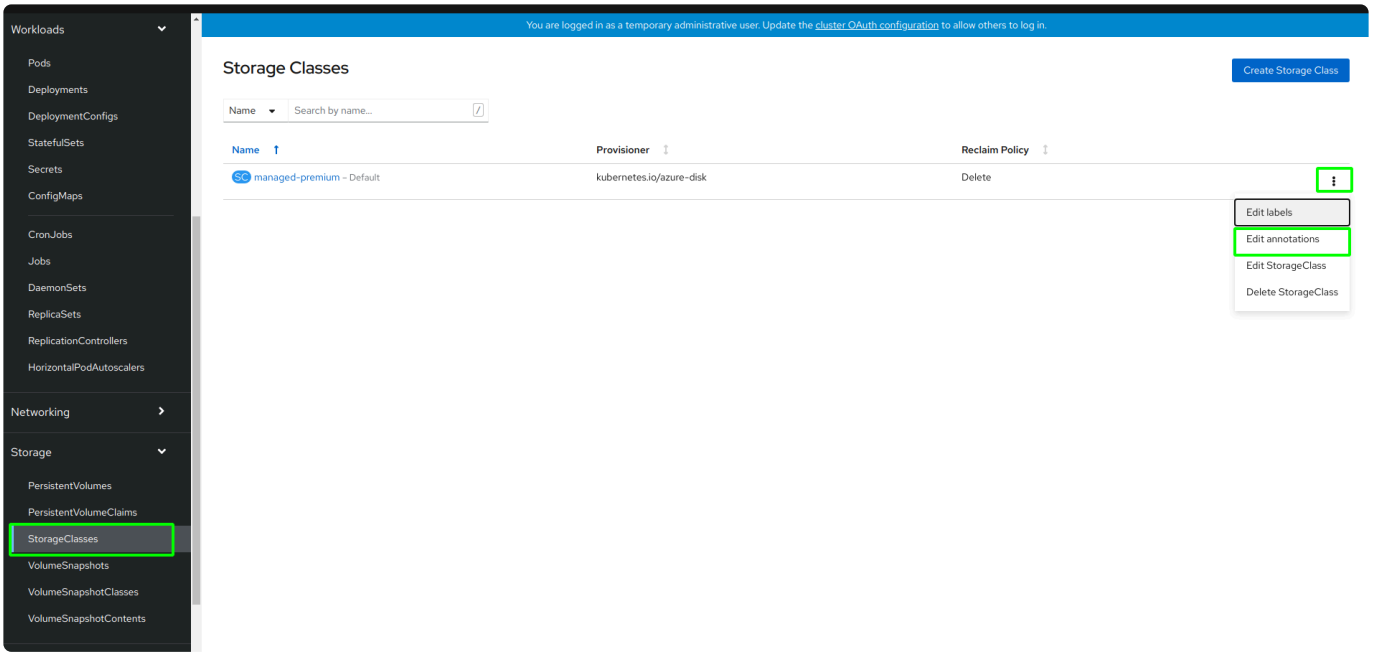


Upon installation completion, the following page should appear.

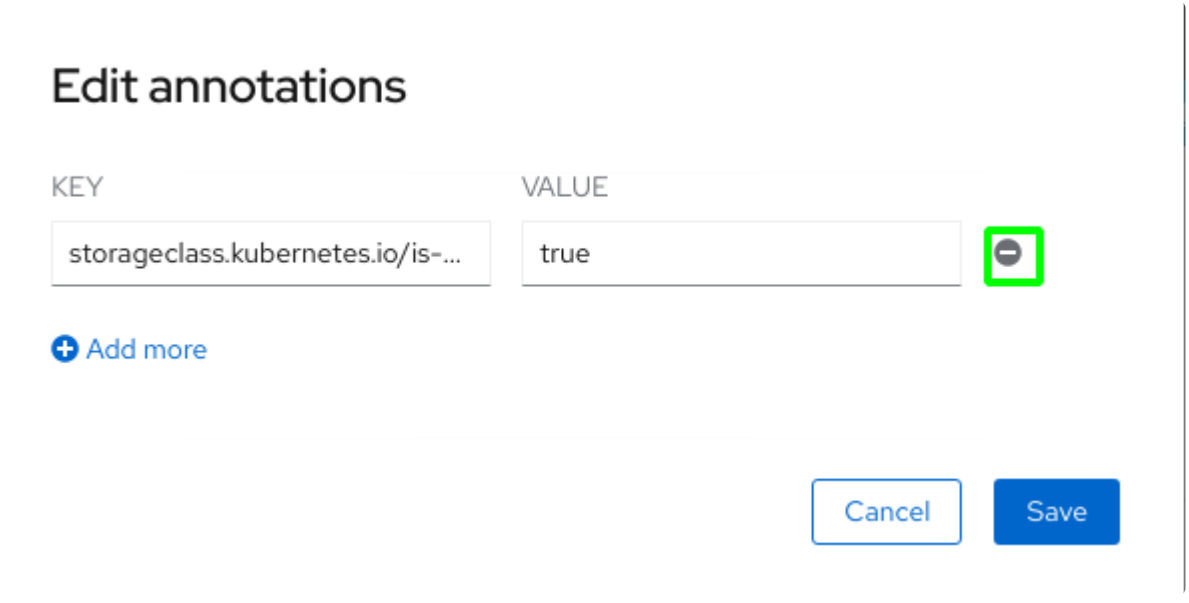


3.4. Remove the Default Storage Class

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.



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

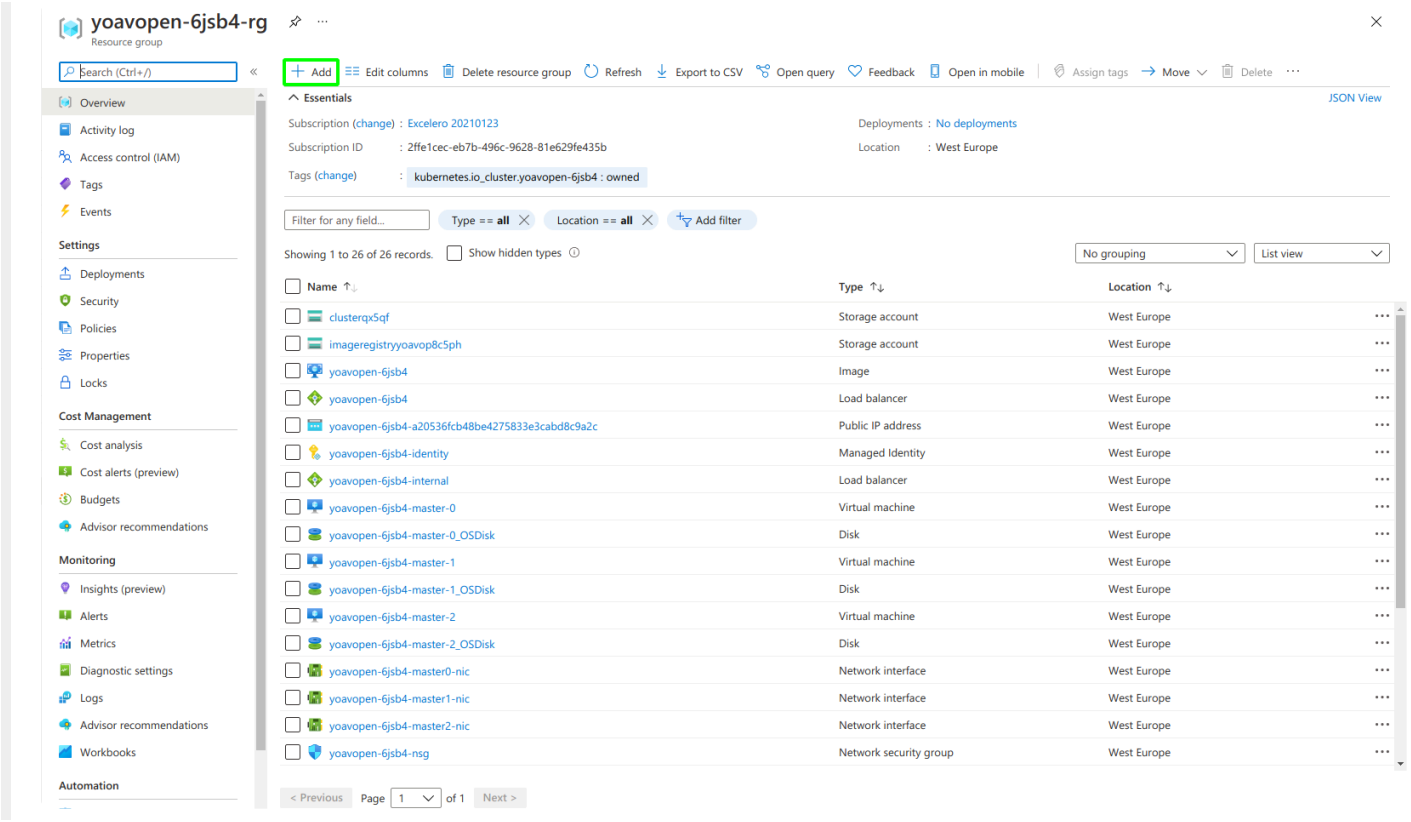


3.5. Create an NFS Server Instance

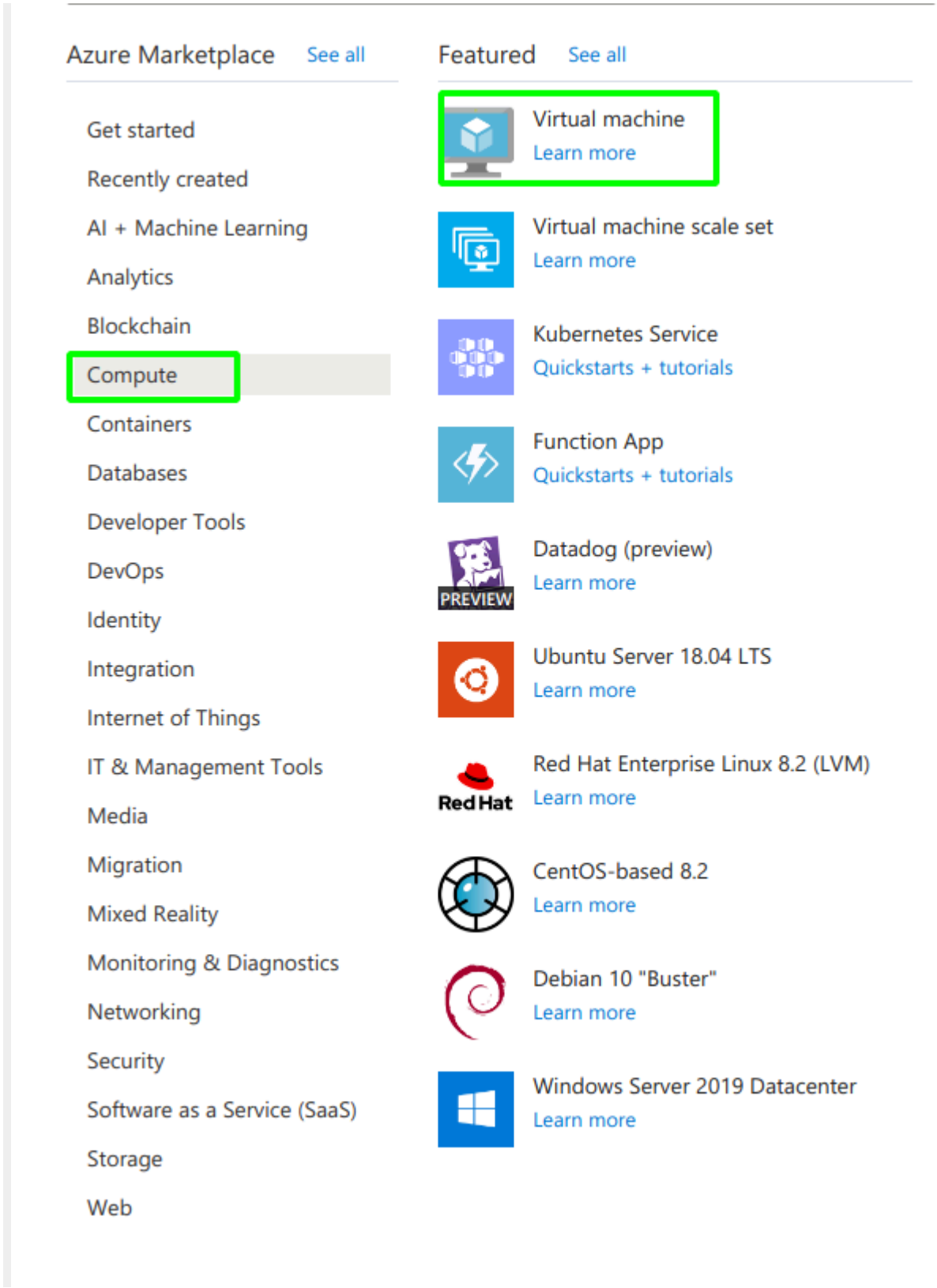
All ***NVMesh Management Servers*** should have access to two shared volumes: one for Mongo and one for backups. This can be done by Azure files, NFS server, or any other persistent volume method.
The following uses the NFS method. To use another method or if an NFS server is already set, skip to Add 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*.



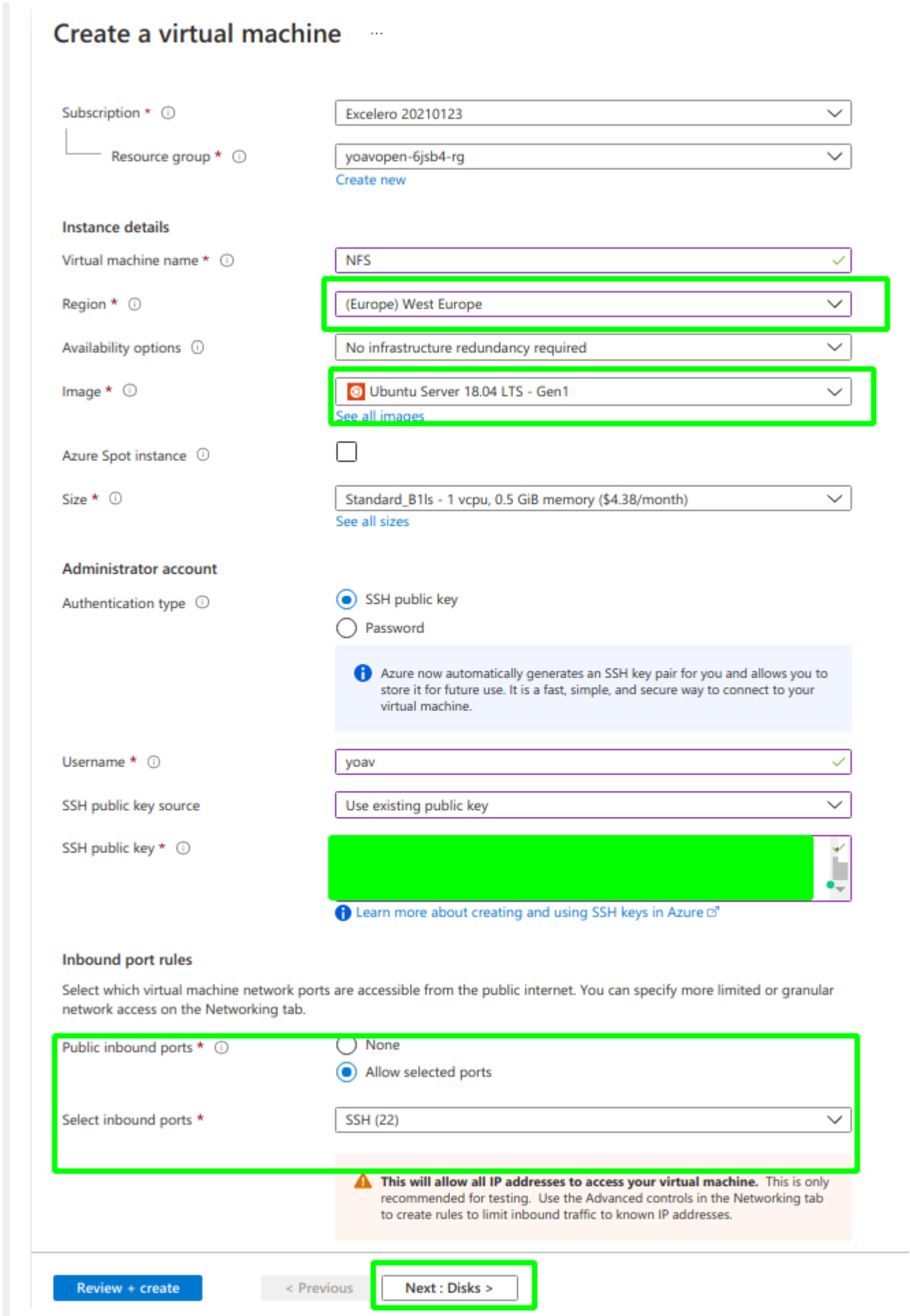
On the left, click *Compute* and then choose *Virtual Machine*.



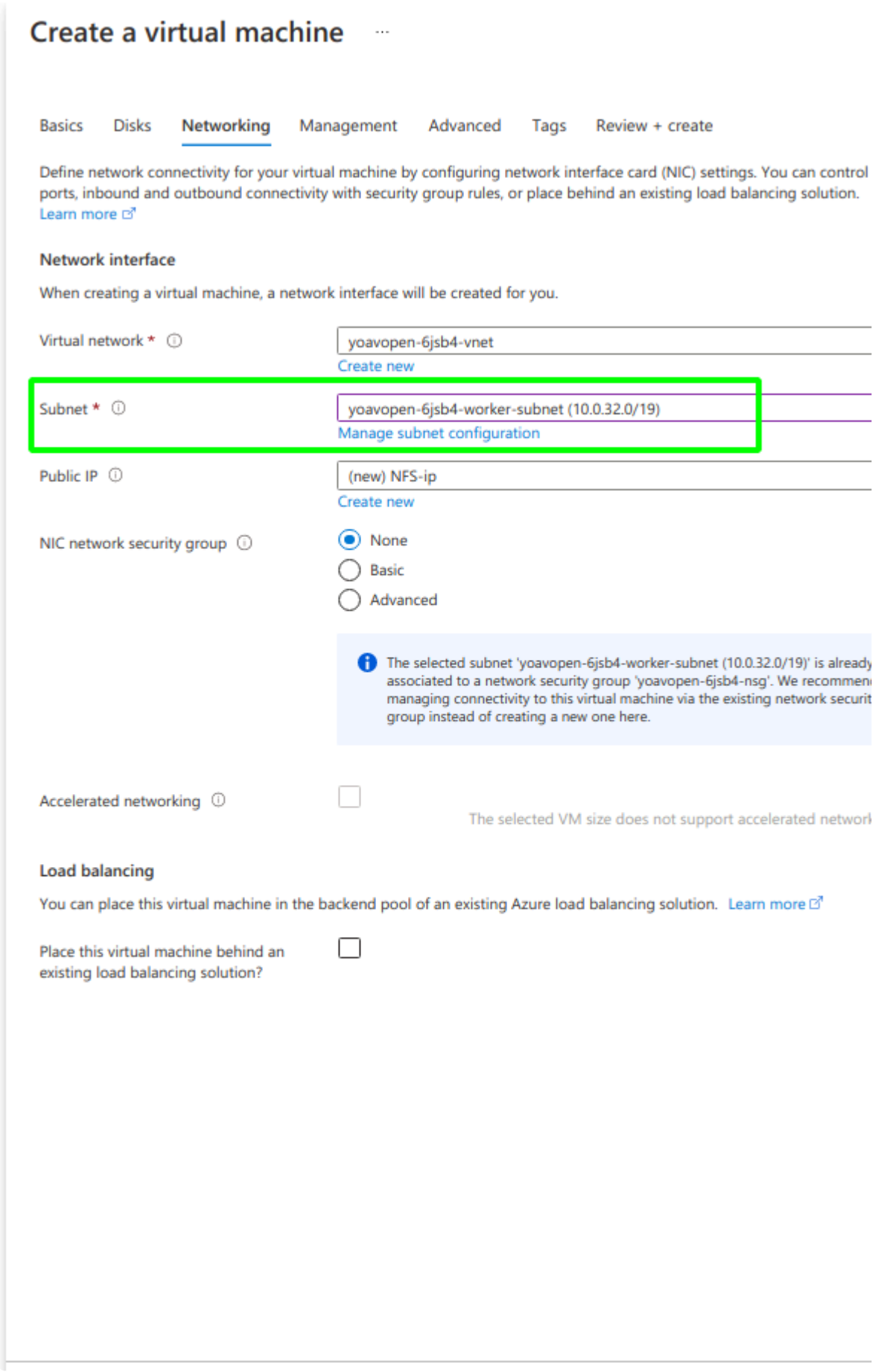
Edit the virtual machine settings as follows:

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

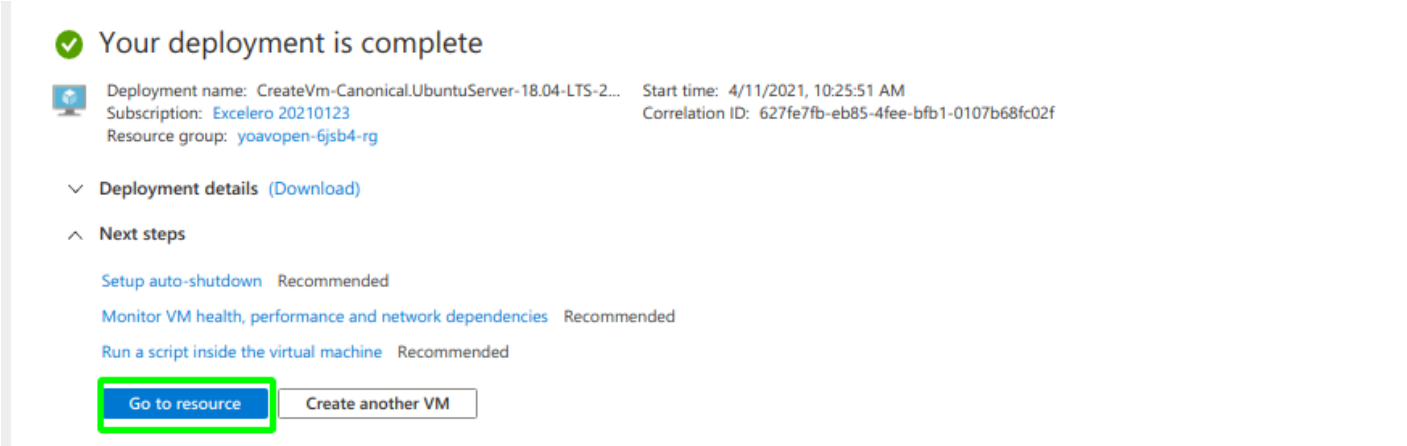
- Size: a minimal machine should be sufficient, for instance “Standard B1ls (1 vcpu, 0.5 GiB memory)”
- Administrator account: it is easiest to use SSH and paste an existing public key



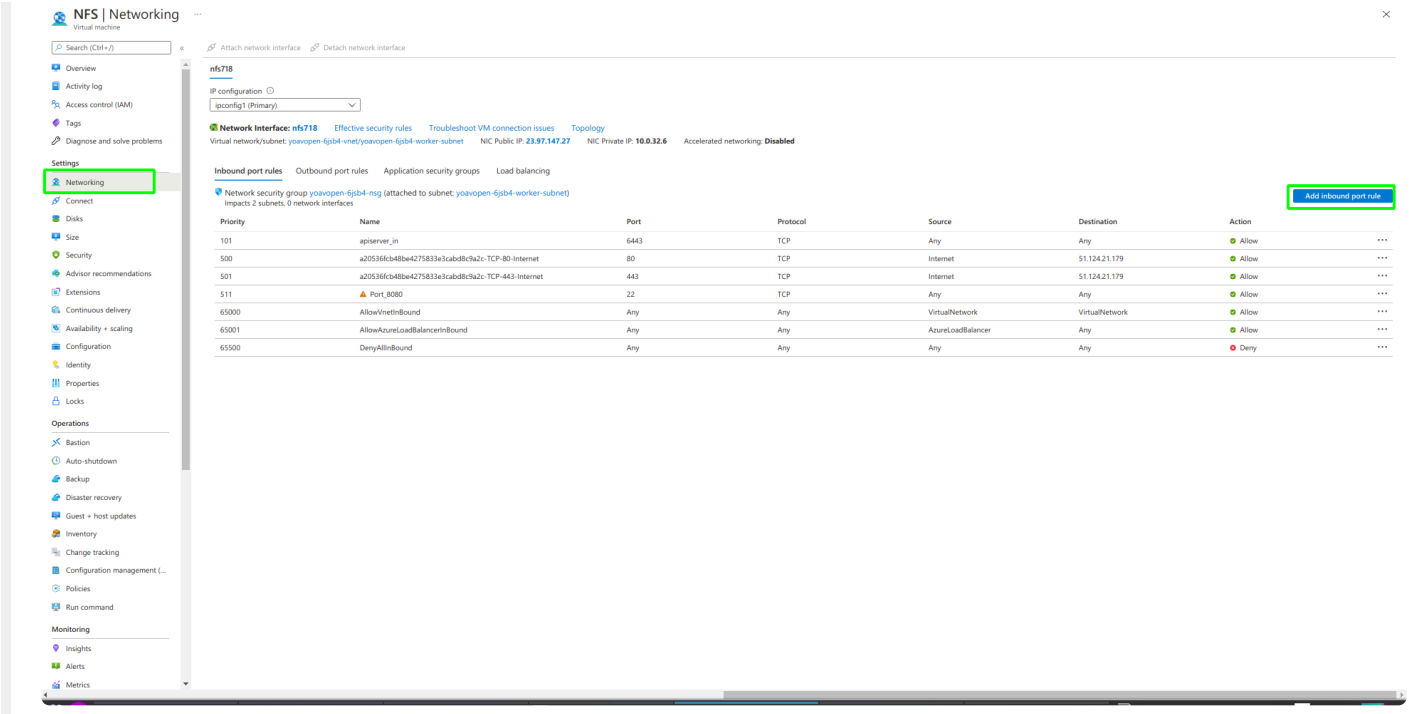
Click *Next* and then *disks*.
Click *Next* and then *networking*.
Choose the worker subnet as the subnet on this machine will run the managements pods, and use a public IP (don’t touch the field).

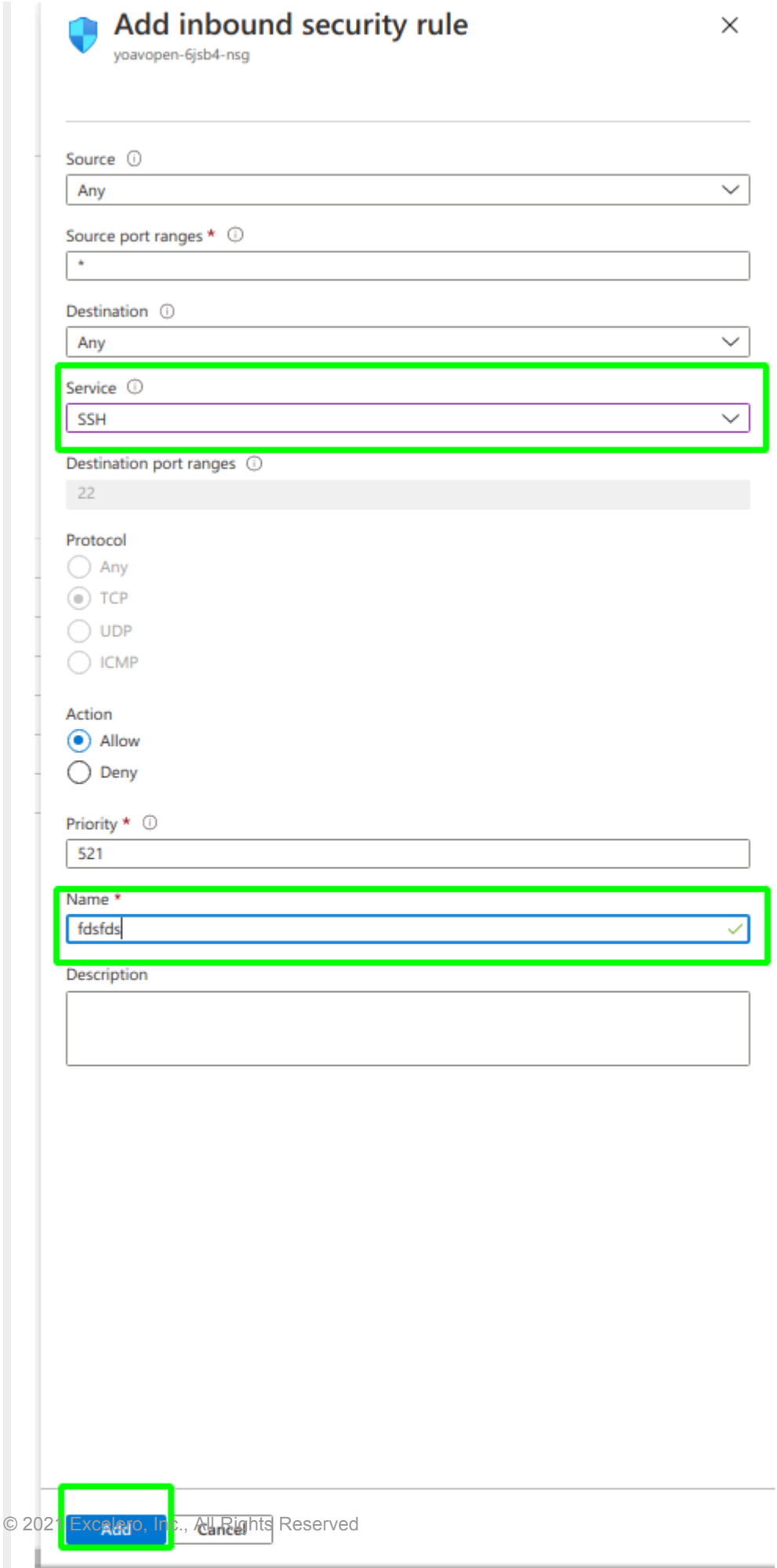


Click *review* and *create* and then *create* again and wait for the VM to start. When the machine will be created the following screen will appear. Click *Go-to resource*.



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





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

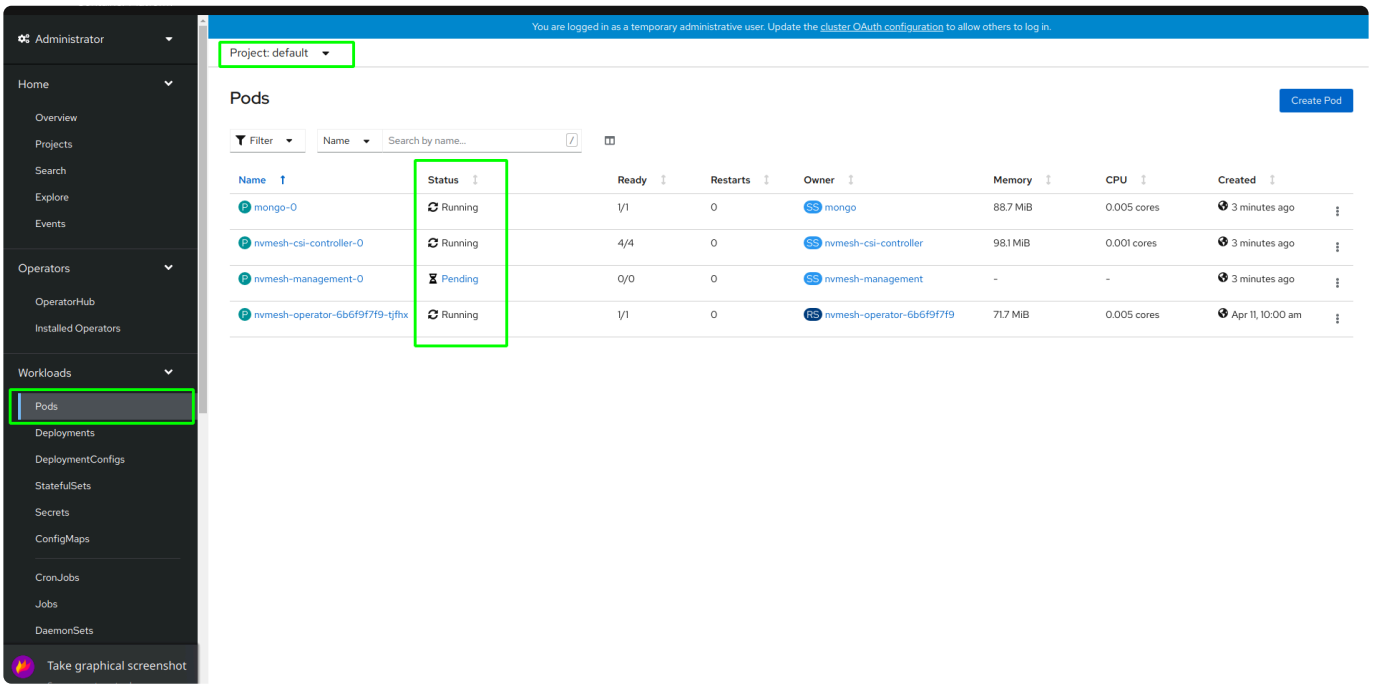
```
apiVersion: nvmesh.excelero.com/v1
kind: NVMesh
metadata:
  name: cluster1
spec:
  core:
    version: 2.2.0-490
    tcpOnly: true
    configuredNICs: eth0
    azureOptimized: true
  csi:
    controllerReplicas: 1
    version: v1.1.4-7
  management:
    imageRegistry: registry.excelero.com
    mongoDB: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

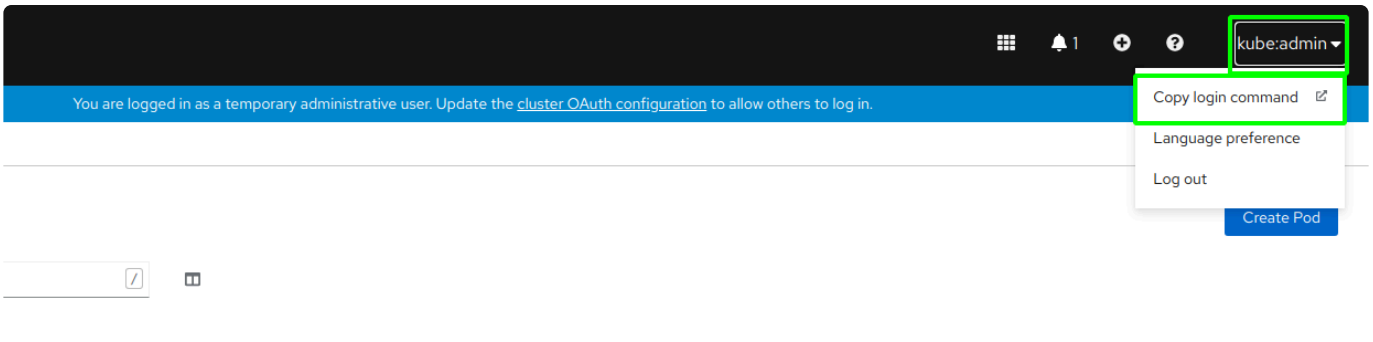
```
    controllerReplicas: 1
    version: v1.1.4-7
  management:
    imageRegistry: registry.excelero.com
    mongoDB:
      replicas: 1
    replicas: 1
    version: 2.2.0
```

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.



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

- 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	SS mongo	-	-	3 minutes ago
nvmesh-client-driver-container-6kmqc	Running	1/1	0	DS nvmesh-client-driver-container	-	-	a minute ago
nvmesh-client-driver-container-7bqhd	Running	1/1	0	DS nvmesh-client-driver-container	-	-	2 minutes ago
nvmesh-client-driver-container-thxkr	Running	1/1	0	DS nvmesh-client-driver-container	-	-	a minute ago
nvmesh-csi-controller-0	Running	4/4	0	SS nvmesh-csi-controller	-	-	Apr 28, 3:19 pm
nvmesh-csi-node-driver-g2nrr	Running	2/2	0	DS nvmesh-csi-node-driver	-	-	a minute ago
nvmesh-csi-node-driver-pdlkc	Running	2/2	0	DS nvmesh-csi-node-driver	-	-	2 minutes ago
nvmesh-csi-node-driver-x69lk	Running	2/2	0	DS nvmesh-csi-node-driver	-	-	a minute ago
nvmesh-management-0	Running	1/1	0	SS nvmesh-management	-	-	3 minutes ago
nvmesh-mcs-agent-bqx58	Running	2/2	1	DS nvmesh-mcs-agent	-	-	a minute ago
nvmesh-mcs-agent-g428r	Running	2/2	1	DS nvmesh-mcs-agent	-	-	2 minutes ago
nvmesh-mcs-agent-qmh2t	Running	2/2	1	DS nvmesh-mcs-agent	-	-	a minute ago
nvmesh-operator-c4fbcdd94-zg6rv	Running	1/1	0	RS nvmesh-operator-c4fbcdd94	-	-	Apr 28, 3:18 pm
nvmesh-target-driver-container-5cgvj	Running	3/3	0	DS nvmesh-target-driver-container	-	-	a minute ago
nvmesh-target-driver-container-gwwrb	Running	3/3	0	DS nvmesh-target-driver-container	-	-	a minute ago
nvmesh-target-driver-container-tc22s	Running	3/3	0	DS 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.

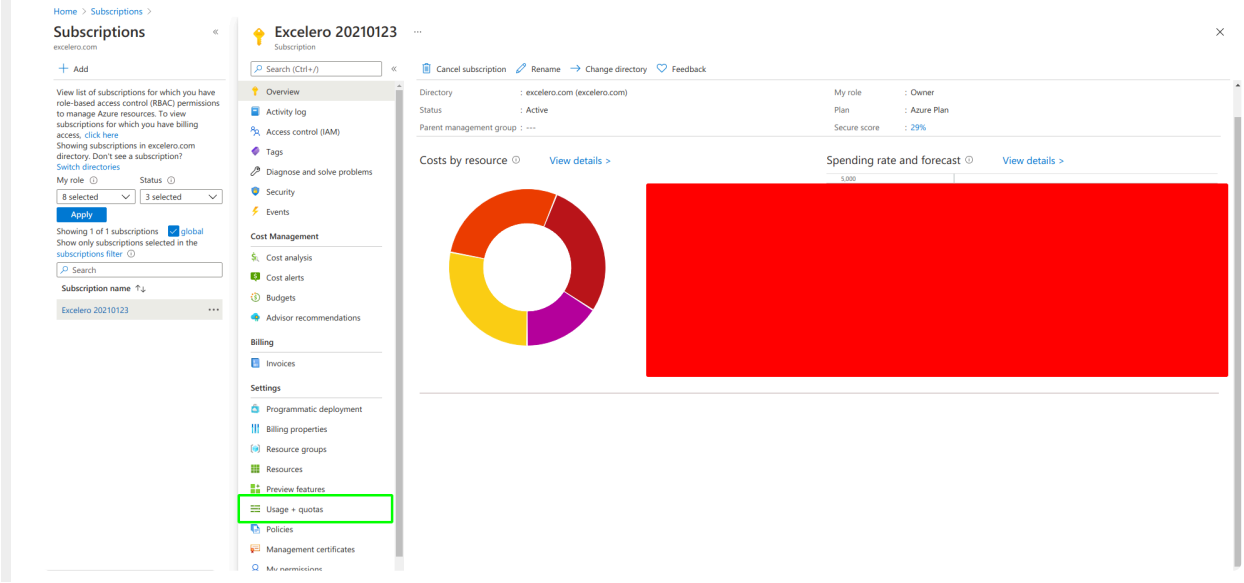
5.1. Prerequisites

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

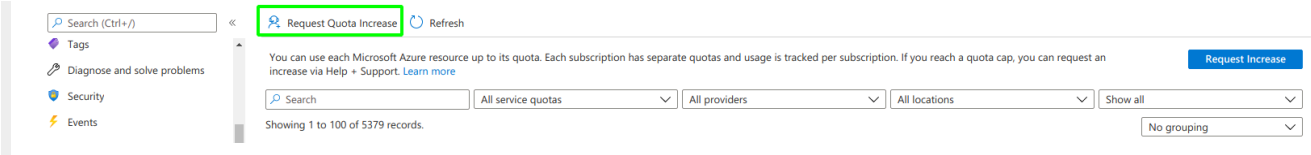
5.1.1. Resource Limits

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	1. DSv3 – 24. 8 each for 3 masters 2. **Infiniband**: HBv3 – 360. 120 each for 3 workers ***or*** **TCP**: LSv2 – 64. 32 each for 2 workers 3. DSv4 – 4 for the bootstrap machine	20 per region	A default cluster requires 40 vCPUs, so you must increase the account limit. By default, each cluster creates the following instances: 1. 1 bootstrap

			machine, which is removed after installation. 2. 3 control plane machines. 3. 3 compute machines. 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. The bootstrap node VM, which uses 4 vCPUs, is used only during installation. To deploy more worker nodes, enable autoscaling, deploy large workloads or use a different instance type,

			increase the vCPU limit to ensure that the cluster can deploy the machines required. 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.
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	Each cluster creates network security groups for each subnet in the VNet. The default cluster creates network security groups for the control plane and for the compute node subnets: ***controlplane***: Allows the control plane machines to be reached on port 6443 from anywhere. ***node***: Allows worker nodes to be reached from the Internet on ports 80 and 443.
Network Load Balancers	3	1000 per region	Each cluster creates the following load

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

			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.

```
[
  {
      "cloudName": "AzureCloud",
      "homeTenantId": "xxxxx-x-x-x--x-x-x-x-",
      "id": "xxxxxxxxxxxxxxxxxxxx",
      "isDefault": true,
      "managedByTenants": [],
      "name": "Just a name",
      "state": "Enabled",
      "tenantId": "xxxxx-xxxxxxxx-xxxx-xxxx-xxx",
      "user": {
        "name": "xxxx@excelero.com",
        "type": "user"
      }
    }
  ]
```

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-xxxxx-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 "azurerm_role_assignment" "main":

> ERROR 161: resource “azurerm_role_assignment” “main” …. this can randomly 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: "xxxx", and password: "gdfklgdflgdfgfd"
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', variab
les('virtualNetworkName'))]",
    "nodeSubnetName" : "[concat(parameters('baseName'), '-worker-subnet')]",
    "nodeSubnetRef" : "[concat(variables('virtualNetworkID'), '/subnets/', variab
les('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('locatio
n'), '-', 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('ava
ilabilitySetName')))]"
      ],
      "properties" : {
        "mode" : "Incremental",
        "template" : {
          "$schema" : "http://schema.management.azure.com/schemas/2015-01-01/depl
oymentTemplate.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",
```

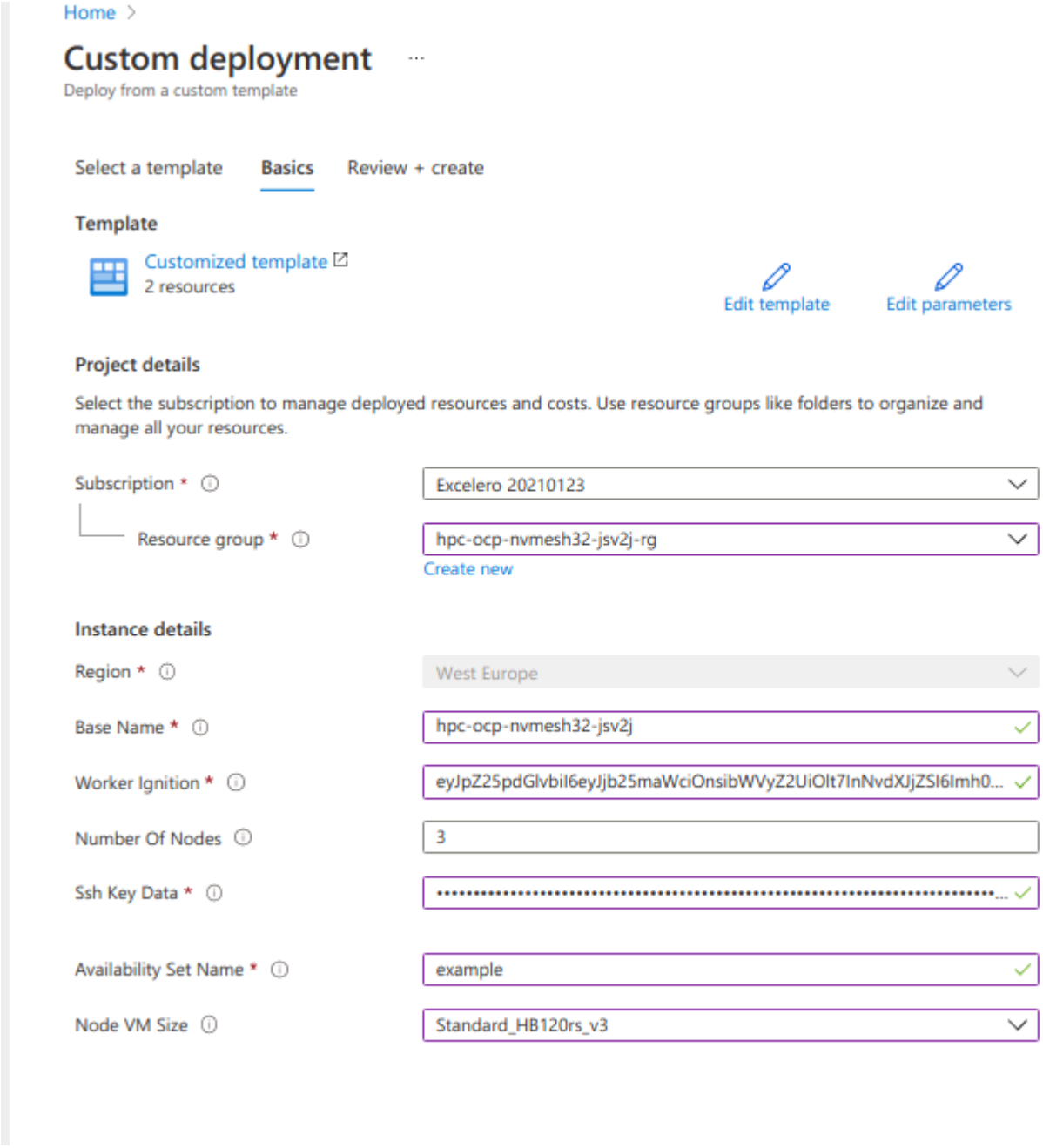
```
                    "properties" : {
                      "privateIPAllocationMethod" : "Dynamic",
                      "subnet" : {
                        "id" : "[variables('nodeSubnetRef')]"
                      },
                      "loadBalancerBackendAddressPools" : [
                        {
                          "id" : "[resourceId('Microsoft.Network/loadBalancers/ba
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/userAssignedIdentitie
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('ima
geName'))]"
                    },
                    "osDisk" : {
                      "name": "[concat(variables('vmNames')[copyIndex()],'_OSDis
k')]",
                      "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', parame
ters('availabilitySetName'))]"
              }
              }
```

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

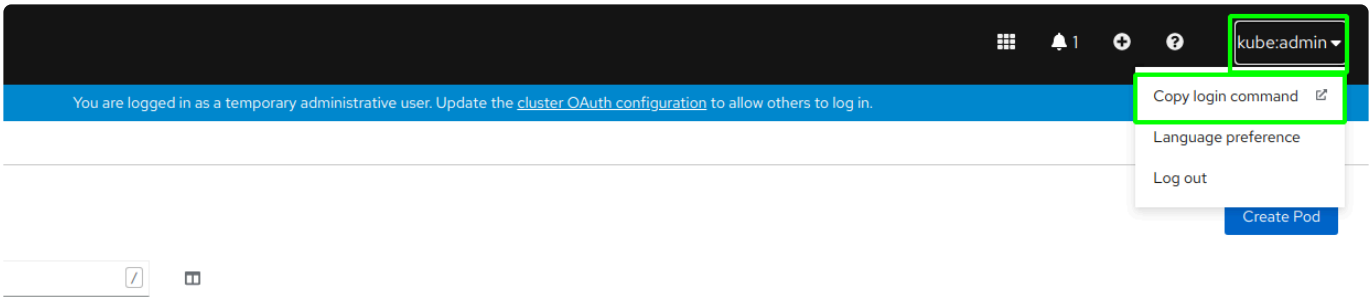
Now fill in the following elements.

- **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



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.

!(zoom)

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 * 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*** OpensShift Operator on Azure Prerequisites. All sections apply, except that the Azure CLI should be built-in already.

6.1. Quick Summary

Action	Command
Login	`az login`
Verify login	`az account list --refresh`
Optional: choose a different account	`az account set -s <id>`
Create a resource group	`az group create --name <resourceGroupName> --location  <region>`
Optional: install kubectl	`sudo az aks install-cli`
Create the AKS cluster	`az aks create --resource-group <resourceGroupName> --name <clusterName> --node-count <node-count>  --generate-ssh-keys`
Update local credentials for the newly created cluster	`az aks get-credentials --resource-group <resourceGroupName> --name <clusterName>`
Verify AKS cluster creation	`kubectl get nodes`
Create a proximity placement group	`az ppg create -n <ppgName> -g <resourceGroupName> -l <region> -t standard`
Check node availability by zone	`az vm list-skus -l eastus2 --zone --size "Standard_L48s_v2"`
Deploy a node pool for ***Targets***	`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>`

Import YAMLs with ***NVMesh 2.5*** operator secrets	
Install and access the ***NVMesh 2.5*** operator	`git clone git@gitlab.excelero.com:excelero/openshift-operator.git` `cd openshift-operator`
Deploy the ***NVMesh 2.5*** operator	`kubectl apply -f deploy/`
Verify the operator deployment	`kubectl get pods`
Deploy ***NVMesh 2.5*** pods	`kubectl apply -f deploy/samples/nvmesh/nvmesh_v1_AKS_tcp.yaml`
Verify the pods deployment	`kubectl get pods --watch` or `watch -d kubectl get pods -o wide`
Generate an ***NVMesh 2.5*** volume	`kubectl apply -f <volume.yaml>`
List PVCs	`kubectl get pvcs`
List PVs	`kubectl get pvs`
Performance Tuning	`kubectl edit configmap nvmesh-core-config` – edit the configuration `kubectl delete ds nvmesh-client nvmesh-target nvmesh-mcs-agent` – apply the configuration

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.

```
[
  {
      "cloudName": "AzureCloud",
      "homeTenantId": "xxxxx-x-x-x--x-x-x-x-",
      "id": "xxxxxxxxxxxxxxxxxxxx",
      "isDefault": true,
      "managedByTenants": [],
      "name": "Just a name",
      "state": "Enabled",
      "tenantId": "xxxxx-xxxxxxxx-xxxx-xxxx-xxx",
      "user": {
        "name": "xxxx@excelero.com",
        "type": "user"
      }
    }
  ]
```

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:

```
{
  "id": "/subscriptions/cf41518e-baf3-4748-9ebd-7b2c35f34207/resourceGroups/myResourceGroup",
  "location": "eastus",
  "managedBy": null,
  "name": "myResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```


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-co
unt 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.

```
[
  {
      "cloudName": "AzureCloud",
      "homeTenantId": "xxxxx-x-x-x--x-x-x-x-",
      "id": "xxxxxxxxxxxxxxxxxxxx",
      "isDefault": true,
      "managedByTenants": [],
      "name": "Just a name",
      "state": "Enabled",
      "tenantId": "xxxxx-xxxxxxxx-xxxx-xxxx-xxx",
      "user": {
        "name": "xxxx@excelero.com",
        "type": "user"
      }
    }
  ]
```

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:


```
{
  "id": "/subscriptions/cf41518e-baf3-4748-9ebd-7b2c35f34207/resourceGroups/myResourceGroup",
  "location": "eastus",
  "managedBy": null,
  "name": "myResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null,
  "type": "Microsoft.Resources/resourceGroups"
}
```

6.2.4. Create the Cluster

```
az aks create --resource-group <resourceGroupName> --name <clusterName> --node-co
unt <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-ffbls              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 2.5 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 2.5*** and the storage in the ***Targets*** node pool, run:

```
az aks nodepool add --resource-group <resourceGroupName> --cluster-name <clusterN
ame> --name d32spool --node-vm-size Standard_D32s_v3 --node-count 1 [ --ppg <ppgN
ame> ] --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    VOLUM
E                                    CAPACITY   ACCESS MODES   STORAGECLAS
S      AGE
backups-nvmesh-management-0   Bound     nvmesh-backu
p-0                            5Gi        RWO            default           43h
block-pvc                     Bound     pvc-632172ae-5ac9-49f3-8cba-e63edc283c2
a   10Gi       RWX            nvmesh-raid1      40h
block-pvc-r0                  Bound     pvc-7fc781e7-e744-4f0e-bac2-ec272c96462
2   10Gi       RWX            nvmesh-raid0      23h
block-pvc-r10                 Bound     pvc-017fa85e-ce4c-4ff5-96a4-a02d0b1c2cc
5   10Gi       RWX            nvmesh-raid10     23h
data-volume-mongo-0           Bound     data-volume-mongod
b                       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:

```
kubectl 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`.