Setup

NetApp Solutions

NetApp
February 16, 2023

Table of Contents

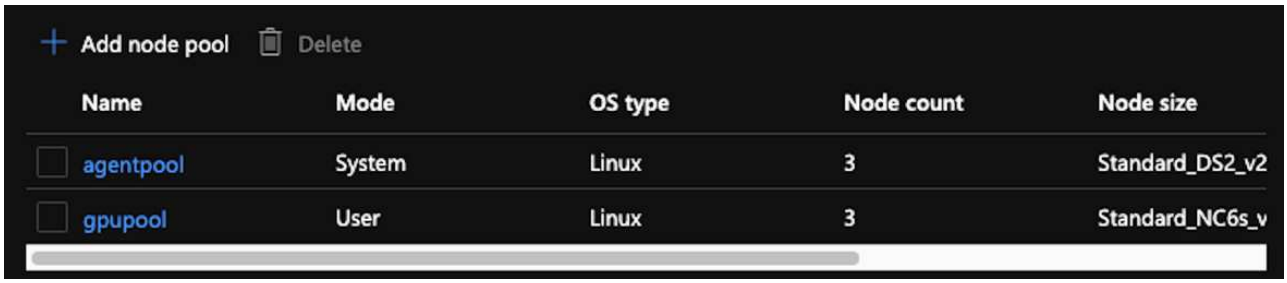
Setup

Install and set up the AKS cluster

Previous: Click-through rate prediction use case summary.

To install and set up the AKS cluster, see the webpage Create an AKS Cluster and then complete the following steps:

1. When selecting the type of node (system [CPU] or worker [GPU] nodes), select the following:
   a. Primary system nodes should be Standard DS2v2 (`agentpool` default three nodes).
   b. Then add the worker node Standard_NC6s_v3 pool (three nodes minimum) for the user group (for GPU nodes) named `gpupool`.



2. Deployment takes 5 to 10 minutes. After it is complete, click Connect to Cluster.
3. To connect to the newly created AKS cluster, install the following from your local environment (laptop/pc):
   a. The Kubernetes command-line tool using the instructions provided for your specific OS
   b. The Azure CLI as described in the document, Install the Azure CLI
4. To access the AKS cluster from the terminal, enter `az login` and enter the credentials.
5. Run the following two commands:

```
az account set --subscription xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxx
aks get-credentials --resource-group resourcegroup --name aksclustername
```

6. Enter `Azure CLI: kubectl get nodes`.
7. If all six nodes are up and running, as shown in the following example, your AKS cluster is ready and connected to your local environment

```
verronmartina@verron-mac-0 ~ % kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-agentpool-34613062-vmss000000   Ready    agent   22m   v1.18.14
aks-agentpool-34613062-vmss000001   Ready    agent   22m   v1.18.14
aks-agentpool-34613062-vmss000002   Ready    agent   22m   v1.18.14
aks-gpupool-34613062-vmss000000     Ready    agent   20m   v1.18.14
aks-gpupool-34613062-vmss000001     Ready    agent   20m   v1.18.14
aks-gpupool-34613062-vmss000002     Ready    agent   20m   v1.18.14
verronmartina@verron-mac-0 ~ %
```

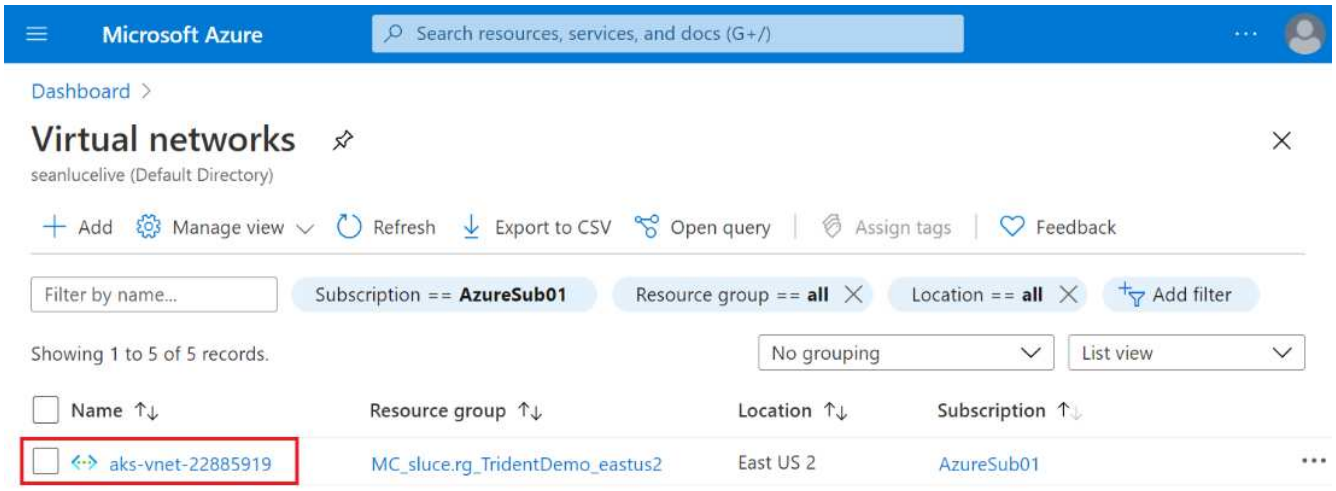
Next: Create a delegated subnet for Azure NetApp Files.

Create a delegated subnet for Azure NetApp Files

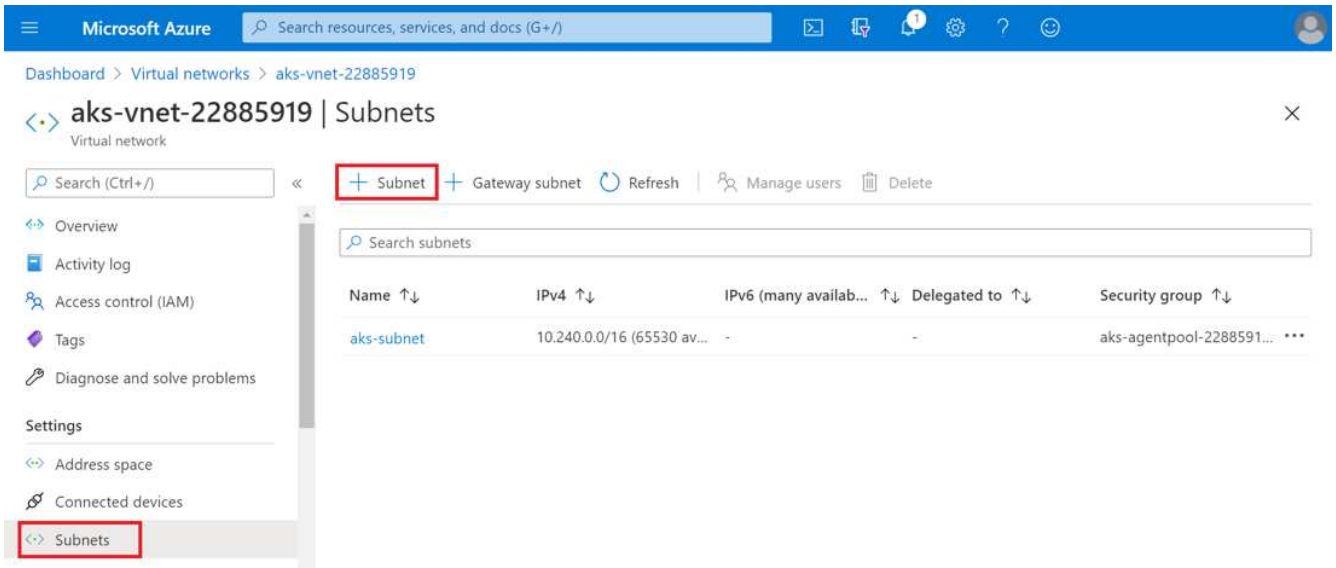
Previous: Install and set up the AKS cluster.

To create a delegated subnet for Azure NetApp Files, complete the following steps:

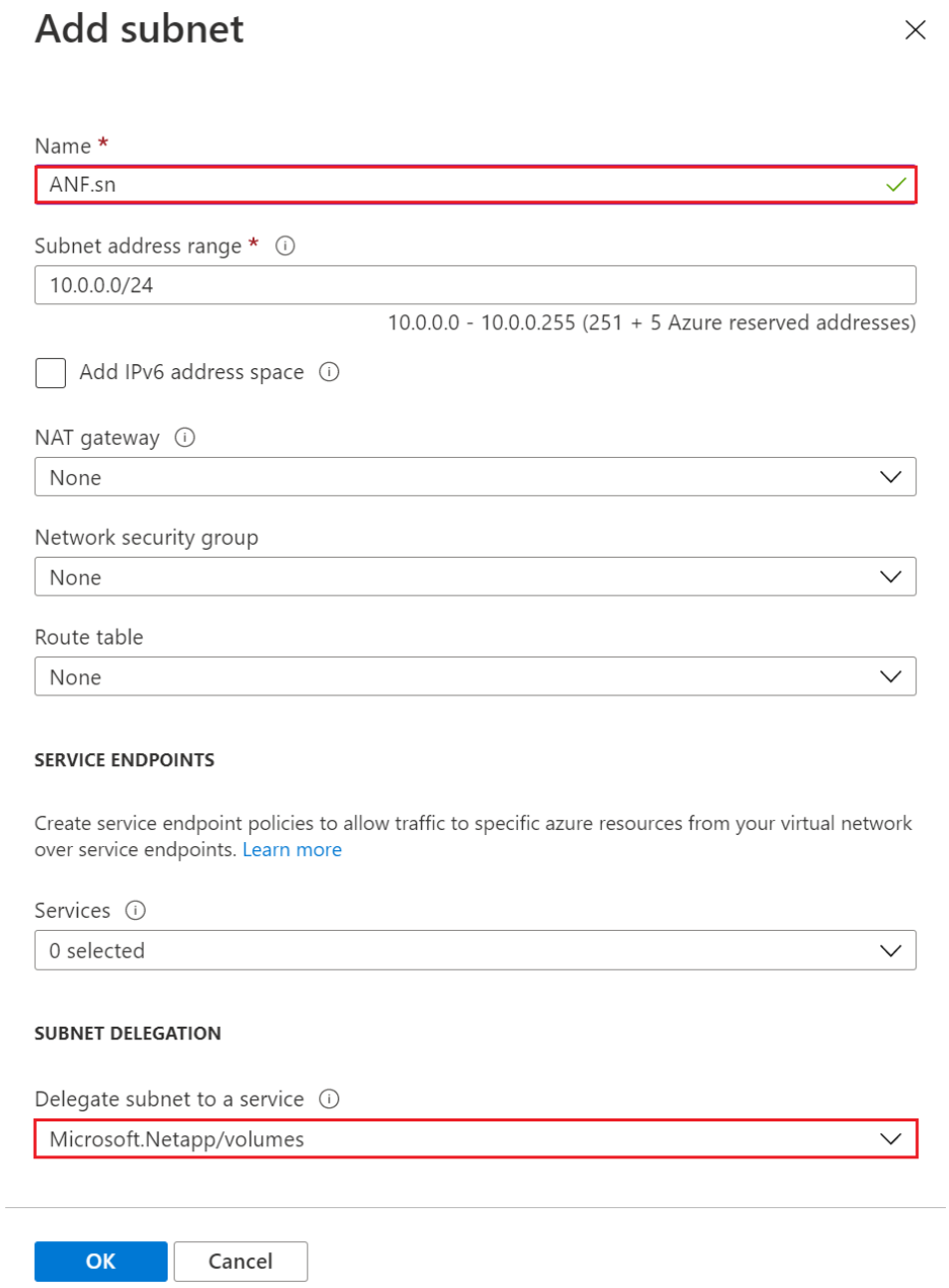
1. Navigate to Virtual Networks within the Azure portal. Find your newly created virtual network. It should have a prefix such as `aks-vnet`.
2. Click the name of the VNet.



3. Click Subnets and click +Subnet from the top toolbar.



4. Provide the subnet with a name such as `ANF.sn` and, under the Subnet Delegation heading, select `Microsoft.Netapp/volumes`. Do not change anything else. Click OK.



Azure NetApp Files volumes are allocated to the application cluster and are consumed as persistent volume claims (PVCs) in Kubernetes. In turn, this process provides you the flexibility to map them to different services, such as Jupyter notebooks, serverless functions, and so on.

Users of services can consume storage from the platform in many ways. As this technical report discusses NFSs, the main benefits of Azure NetApp Files are:

- Providing users with the ability to use Snapshot copies.
- Enabling users to store large quantities of data on Azure NetApp Files volumes.
- Using the performance benefits of Azure NetApp Files volumes when running their models on large sets of files.

Next: Peer AKS vnet and Azure NetApp Files vnet.

Peer AKS VNet and Azure NetApp Files VNet

Previous: Create a delegated subnet for Azure NetApp Files.

To peer the AKS VNet to the Azure NetApp Files VNet, complete the following steps:

1. Enter Virtual Networks in the search field.
2. Select `vnet aks-vnet-name.` Click it and enter Peerings in the search field.
3. Click +Add.
4. Enter the following descriptors:
   a. The peering link name is `aks-vnet-name_to_anf`.
   b. subscriptionID and Azure NetApp Files VNet as the VNet peering partner.
   c. Leave all the nonasterisk sections with the default values.
5. Click Add.

For more information, see Create, change, or delete a virtual network peering.

Next: Install Trident.

Install Trident

Previous: Peer AKS VNet and Azure NetApp Files VNet.

To install Trident using Helm, complete the following steps:

1. Install Helm (for installation instructions, visit the source).
2. Download and extract the Trident 20.01.1 installer.

```
$wget
$tar -xf trident-installer-21.01.1.tar.gz
```

3. Change the directory to `trident-installer`.

```
$cd trident-installer
```

4. Copy `tridentctl` to a directory in your system `$PATH`.

```
$sudo cp ./tridentctl /usr/local/bin
```

5. Install Trident on the Kubernetes (K8s) cluster with Helm (source):
   a. Change the directory to the `helm` directory.

```
$cd helm
```

b. Install Trident.

```
$helm install trident trident-operator-21.01.1.tgz --namespace
trident --create-namespace
```

c. Check the status of Trident pods.

```
$kubectl -n trident get pods
```

If all the pods are up and running, then Trident is installed and you can move forward.

6. Set up the Azure NetApp Files backend and storage class for AKS.

a. Create an Azure Service Principle.

The service principal is how Trident communicates with Azure to manipulate your Azure NetApp Files resources.

```
$az ad sp create-for-rbac --name ""
```

The output should look like the following example:

```
{
"appId": "xxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
"displayName": "netapptrident",
"name": "",
"password": "xxxxxxxxxxxxxx.xxxxxxxxxxxxx",
"tenant": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"
}
```

7. Create a Trident backend json file, example name `anf-backend.json`.

8. Using your preferred text editor, complete the following fields inside the `anf-backend.json` file:

```
{
    "version": 1,
    "storageDriverName": "azure-netapp-files",
    "subscriptionID": "fakec765-4774-fake-ae98-a721add4fake",
    "tenantID": "fakef836-edc1-fake-bff9-b2d865eefake",
    "clientID": "fake0f63-bf8e-fake-8076-8de91e57fake",
    "clientSecret": "SECRET",
    "location": "westeurope",
    "serviceLevel": "Standard",
    "virtualNetwork": "anf-vnet",
    "subnet": "default",
    "nfsMountOptions": "vers=3,proto=tcp",
    "limitVolumeSize": "500Gi",
    "defaults": {
    "exportRule": "0.0.0.0/0",
    "size": "200Gi"
}
```

9. Substitute the following fields:
   - `subscriptionID`. Your Azure subscription ID.
   - `tenantID`. Your Azure Tenant ID from the output of `az ad sp` in the previous step.
   - `clientID`. Your appID from the output of `az ad sp` in the previous step.
   - `clientSecret`. Your password from the output of `az ad sp` in the previous step.
10. Instruct Trident to create the Azure NetApp Files backend in the `trident` namespace using `anf-backend.json` as the configuration file:

```
$tridentctl create backend -f anf-backend.json -n trident
```

```
+------------------------+-------------------+--------------------------------------+--------+---------+
|          NAME          |   STORAGE DRIVER  |                 UUID                 | STATE  | VOLUMES |
+------------------------+-------------------+--------------------------------------+--------+---------+
| azurenetappfiles_86181 | azure-netapp-files | 2ca85462-59ac-4946-be05-c03f5575a2ad | online |       0 |
+------------------------+-------------------+--------------------------------------+--------+---------+
```

11. Create a storage class. Kubernetes users provision volumes by using PVCs that specify a storage class by name. Instruct K8s to create a storage class `azurenetappfiles` that references the Trident backend created in the previous step.
12. Create a YAML (`anf-storage-class.yaml`) file for storage class and copy.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: azurenetappfiles
provisioner: netapp.io/trident
parameters:
backendType: "azure-netapp-files"
$kubectl create -f anf-storage-class.yaml
```

13. Verify that the storage class was created.

```
kubectl get sc azurenetappfiles
```

```
NAME              PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE   ALLOWVOLUMEEXPANSION   AGE
azurenetappfiles  csi.trident.netapp.io   Delete          Immediate           false                  98s
```

Next: Set up Dask with RAPIDS deployment on AKS using Helm.

Set up Dask with RAPIDS deployment on AKS using Helm

Previous: Install Trident.

To set up Dask with RAPIDS deployment on AKS using Helm, complete the following steps:

1. Create a namespace for installing Dask with RAPIDS.

```
kubectl create namespace rapids-dask
```

2. Create a PVC to store the click-through rate dataset:
   a. Save the following YAML content to a file to create a PVC.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-criteo-data
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1000Gi
  storageClassName: azurenetappfiles
```

b. Apply the YAML file to your Kubernetes cluster.

```
kubectl -n rapids-dask apply -f <your yaml file>
```

3. Clone the `rapidsai git` repository (https://github.com/rapidsai/helm-chart).

```
git clone https://github.com/rapidsai/helm-chart helm-chart
```

4. Modify `values.yaml` and include the PVC created earlier for workers and Jupyter workspace.

a. Go to the `rapidsai` directory of the repository.

```
cd helm-chart/rapidsai
```

b. Update the `values.yaml` file and mount the volume using PVC.

```
dask:
  …
  worker:
    name: worker
    …
    mounts:
      volumes:
        - name: data
          persistentVolumeClaim:
            claimName: pvc-criteo-data
      volumeMounts:
        - name: data
          mountPath: /data
    …
  jupyter:
    name: jupyter
    …
    mounts:
      volumes:
        - name: data
          persistentVolumeClaim:
            claimName: pvc-criteo-data
      volumeMounts:
        - name: data
          mountPath: /data
    …
```

5. Go to the repository's home directory and deploy Dask with three worker nodes on AKS using Helm.

```
cd ..
helm dep update rapidsai
helm install rapids-dask --namespace rapids-dask rapidsai
```

Next: Azure NetApp Files performance tiers.

Azure NetApp Files performance tiers

Previous: Set up Dask with RAPIDS deployment on AKS using Helm.

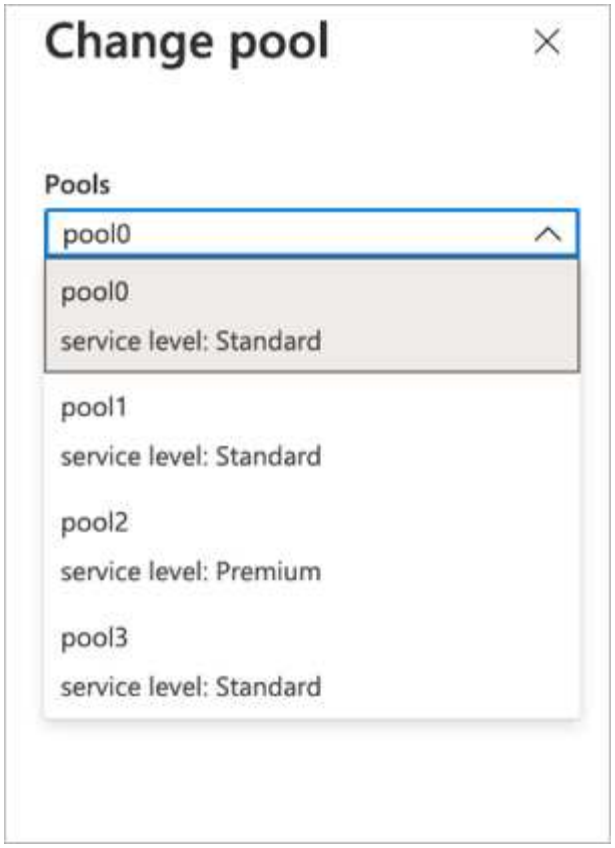
You can change the service level of an existing volume by moving the volume to another capacity pool that uses the service level you want for the volume. This solution enables customers to start with a small dataset and small number of GPUs in Standard Tier and scale out or scale up to Premium Tier as the amount of data and GPUs increase. The Premium Tier offers four times the throughput per terabyte as the Standard Tier, and scale up is performed without having to move any data to change the service level of a volume.

Dynamically change the service level of a volume

To dynamically change the service level of a volume, complete the following steps:

1. On the Volumes page, right-click the volume whose service level you want to change. Select Change Pool.

2. In the Change Pool window, select the capacity pool to which you want to move the volume.



3. Click OK.

Automate performance tier change

The following options are available to automate performance tier changes:

- Dynamic Service Level change is still in Public Preview at this time and not enabled by default. To enable this feature on the Azure Subscription, see this documentation about how to Dynamically change the service level of a volume.
- Azure CLI volume pool change commands are provided in volume pool change documentation and in the following example:

```
az netappfiles volume pool-change -g mygroup --account-name myaccname
--pool-name mypoolname --name myvolname --new-pool-resource-id
mynewresourceid
```

- PowerShell: The Set-AzNetAppFilesVolumePool cmdlet changes the pool of an Azure NetApp Files volume and is shown in the following example:

```
Set-AzNetAppFilesVolumePool
-ResourceGroupName "MyRG"
-AccountName "MyAnfAccount"
-PoolName "MyAnfPool"
-Name "MyAnfVolume"
-NewPoolResourceId 7d6e4069-6c78-6c61-7bf6-c60968e45fbf
```

Next: Libraries for data processing and model training.

Copyright information

Trademark information