Setup

Install and set up the AKS cluster

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.

![Node Pool Table]

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 xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx
   aks get-credentials --resource-group resourcegroup --name aksclusternname
   ```

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

```
verrottmartina@verrott-mac-0 ~ % kubectl get nodes
NAME                     STATUS     ROLES   AGE       VERSION
aks-agentpool-36613062-vmss00000     Ready     agent   22m   v1.18.14
aks-agentpool-36613062-vmss00001     Ready     agent   22m   v1.18.14
aks-agentpool-36613062-vmss00002     Ready     agent   22m   v1.18.14
aks-gpupool-36613062-vmss00000     Ready     agent   20m   v1.18.14
aks-gpupool-36613062-vmss00001     Ready     agent   20m   v1.18.14
aks-gpupool-36613062-vmss00002     Ready     agent   20m   v1.18.14
verrottmartina@verrott-mac-0 ~
```
Create a delegated subnet for Azure NetApp Files

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.
Peer AKS VNet and Azure NetApp Files VNet

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.

Install Trident

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.


      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:

      
      
      
      
      

    

    

    


5. 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:
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:

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

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

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROVISIONER</th>
<th>RECLAIMPOLICY</th>
<th>VOLUMEBINDINGMODE</th>
<th>ALLOWVOLUMEEXPANSION</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>azurenetappfiles</td>
<td>csi.trident.netapp.io</td>
<td>Delete</td>
<td>Immediate</td>
<td>false</td>
<td>98s</td>
</tr>
</tbody>
</table>

Set up Dask with RAPIDS deployment on AKS using Helm

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.

```yaml
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>


```bash
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.

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

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

```yaml
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.
Azure NetApp Files performance tiers

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.

   ![Volumes page with volume options]

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:

  ```bash
  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