

In this lab, you will connect two Azure VNets using Azure VNet peering.

Learning Objectives

In this lab, you will learn how to do the following:

Log into the Azure CLI

Understand Azure VNet peering

Configure Azure VNet peering using Azure CLI

Confirm VNet peering using the Azure CLI

Check VNet peering in the Azure Portal

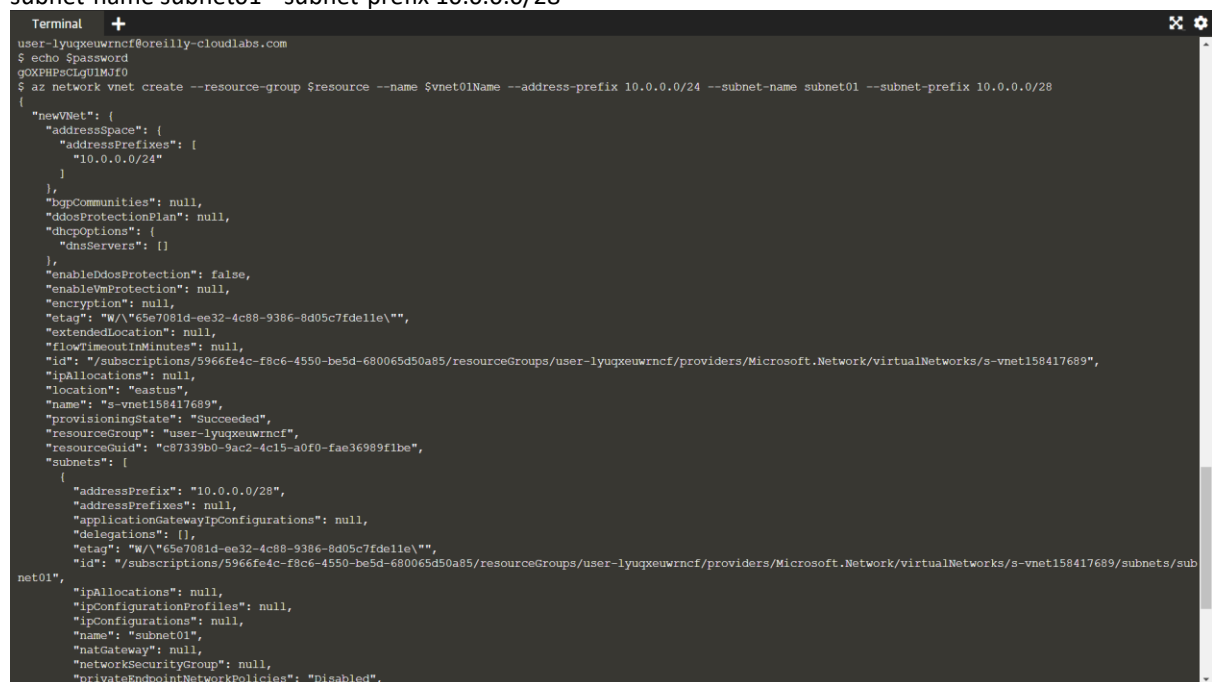
Remove the peering using the Azure CLI

To log into the Azure subscription:

```
az login -u $username -p $password
```

To create a new Azure Virtual Network and a subnet:

```
az network vnet create --resource-group $resource --name $vnet01Name --address-prefix 10.0.0.0/24 --  
subnet-name subnet01 --subnet-prefix 10.0.0.0/28
```

A terminal window with a dark background and light text. The prompt is 'user-lyuqxewrncf@oreilly-cloudlabs.com'. The user enters '\$ echo \$password' and 'gOXHPscLgUIMJf0'. Then they enter the 'az network vnet create' command with various parameters. The output is a large JSON object representing the created VNet and its subnets. The JSON includes fields like 'addressSpace', 'bgpCommunities', 'ddosProtectionPlan', 'dhcpOptions', 'dnsServers', 'enableDdosProtection', 'enableVmProtection', 'encryption', 'etag', 'extendedLocation', 'flowTimeoutInMinutes', 'id', 'ipAllocations', 'location', 'name', 'provisioningState', 'resourceGroup', 'resourceGuid', and 'subnets'. The 'subnets' array contains one object for 'subnet01' with its own 'addressPrefix', 'addressPrefixes', 'applicationGatewayIpConfigurations', 'delegations', 'etag', 'id', 'ipAllocations', 'ipConfigurationProfiles', 'ipConfigurations', 'name', 'natGateway', 'networkSecurityGroup', and 'privateEndpointNetworkPolicies'.

To create another Azure Virtual Network and a subnet:

```
az network vnet create --resource-group $resource --name $vnet02Name --address-prefix 10.0.1.0/24 --  
subnet-name subnet01 --subnet-prefix 10.0.1.0/28
```

```
Terminal +
{
  "type": "Microsoft.Network/virtualNetworks",
  "virtualNetworkPeerings": []
}
$ az network vnet create --resource-group $resource --name $vnet02Name --address-prefix 10.0.1.0/24 --subnet-name subnet01 --subnet-prefix 10.0.1.0/28
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
        "10.0.1.0/24"
      ]
    },
    "bgpCommunities": null,
    "ddosProtectionPlan": null,
    "dhcpOptions": {
      "dnsServers": []
    },
    "enableDdosProtection": false,
    "enableVmProtection": null,
    "encryption": null,
    "etag": "W/\"ba0750e6-9d1c-4356-88b9-273817183f21\"",
    "extendedLocation": null,
    "flowTimeoutInMinutes": null,
    "id": "/subscriptions/5966fe4c-f8c6-4550-be5d-680065d50a85/resourceGroups/user-lyuqxewrncf/providers/Microsoft.Network/virtualNetworks/d-vnet158417689",
    "ipAllocations": null,
    "location": "eastus",
    "name": "d-vnet158417689",
    "provisioningState": "Succeeded",
    "resourceGroup": "user-lyuqxewrncf",
    "resourceGuid": "35c8bee0-845c-4210-b669-b98b780aab51",
    "subnets": [
      {
        "addressPrefix": "10.0.1.0/28",
        "addressPrefixes": null,
        "applicationGatewayIpConfigurations": null,
        "delegations": [],
        "etag": "W/\"ba0750e6-9d1c-4356-88b9-273817183f21\"",
        "id": "/subscriptions/5966fe4c-f8c6-4550-be5d-680065d50a85/resourceGroups/user-lyuqxewrncf/providers/Microsoft.Network/virtualNetworks/d-vnet158417689/subnets/subnet01",
        "ipAllocations": null,
        "ipConfigurationProfiles": null,
        "ipConfigurations": null,
        "name": "subnet01",
        "natGateway": null,
        "networkSecurityGroup": null,

```

In this lab, we will use the Azure CLI to create an Azure VNet peering to connect two existing Azure Virtual Networks.

Understand Azure Virtual Network Peering

Virtual network peering allows you to connect two or more Virtual Networks in Azure.

Imagine that you have a virtual machine in VNet01. This VM needs to see another VM that is deployed to another Azure VNet. You can simply create VNet peering between these two VNets, allowing these VMs to communicate with each other.

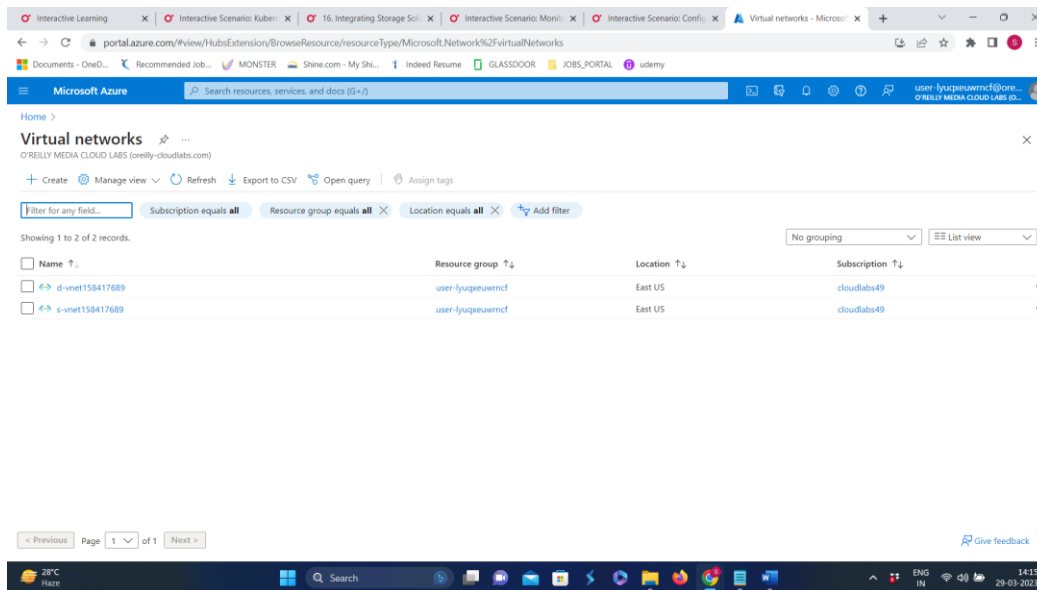
Important: The two peered VNets should not have any common IP addresses in their address spaces.

Use the following command to confirm that the Azure VNet is present:

`az network vnet list --resource-group $resource --query "[].{Name:name, AddressSpace: addressSpace.addressPrefixes[0]}"`

```
$ az network vnet list --resource-group $resource --query "[].{Name:name, AddressSpace: addressSpace.addressPrefixes[0]}"
[
  {
    "AddressSpace": "10.0.1.0/24",
    "Name": "d-vnet158417689"
  },
  {
    "AddressSpace": "10.0.0.0/24",
    "Name": "s-vnet158417689"
  }
]
$
```

Confirm that you can see two VNets in the command output with separate IP addressPrefixes.



In the next step, we will create a VNet Peering between these two VNets.

Create Azure Virtual Network Peering using Azure CLI

Use the following command to create a peering from \$vnet01Name to \$vnet02Name:

```
az network vnet peering create --resource-group $resource --name peering01 --vnet-name $vnet01Name --remote-vnet $vnet02Name --allow-vnet-access
```

Here are the command parameters:

--name: The peering name

--vnet-name: The source VNet name

--remote-vnet: The destination (remote) VNet name

--allow-vnet-access: Allows traffic from the source VNet to the remote VNet

```
Terminal +
{
  "AddressSpace": "10.0.1.0/24",
  "Name": "d-vnet158417689"
},
{
  "AddressSpace": "10.0.0.0/24",
  "Name": "s-vnet158417689"
}
}
$ az network vnet peering create --resource-group $resource --name peering01 --vnet-name $vnet01Name --remote-vnet $vnet02Name --allow-vnet-access
{
  "allowForwardedTraffic": false,
  "allowGatewayTransit": false,
  "allowVirtualNetworkAccess": true,
  "doNotVerifyRemoteGateways": false,
  "etag": "W/\"1df6097d-8e30-4f05-8c6c-9b401703228a\"",
  "id": "/subscriptions/5966fe4c-f8c6-4550-be5d-680065d50a85/resourceGroups/user-lyuqxewrncf/providers/Microsoft.Network/virtualNetworks/s-vnet158417689/virtualNetworkPeerings/peering01",
  "name": "peering01",
  "peeringState": "Initiated",
  "peeringSyncLevel": "RemoteNotInSync",
  "provisioningState": "Succeeded",
  "remoteAddressSpace": {
    "addressPrefixes": [
      "10.0.1.0/24"
    ]
  },
  "remoteBgpCommunities": null,
  "remoteVirtualNetwork": {
    "id": "/subscriptions/5966fe4c-f8c6-4550-be5d-680065d50a85/resourceGroups/user-lyuqxewrncf/providers/Microsoft.Network/virtualNetworks/d-vnet158417689",
    "resourceGroup": "user-lyuqxewrncf"
  },
  "remoteVirtualNetworkAddressSpace": {
    "addressPrefixes": [
      "10.0.1.0/24"
    ]
  },
  "remoteVirtualNetworkEncryption": null,
  "resourceGroup": "user-lyuqxewrncf",
  "resourceId": "f8c6-4550-be5d-680065d50a85/resourceGroups/user-lyuqxewrncf/providers/Microsoft.Network/virtualNetworks/virtualNetworkPeerings",
  "type": "Microsoft.Network/virtualNetworks/virtualNetworkPeerings",
  "useRemoteGateways": false
}
$
```

Now you need to create the next peering leg from \$vnet02Name to \$vnet01Name:

```
az network vnet peering create --resource-group $resource --name peering02 --vnet-name $vnet02Name --remote-vnet $vnet01Name --allow-vnet-access
```

```
Terminal +
"addressPrefixes": [
  "10.0.1.0/24"
],
"remoteVirtualNetworkEncryption": null,
"resourceGroup": "user-lyuqxewrncf",
"resourceGuid": "fdbb8750-1e9e-0e05-1699-436811835aef",
"type": "Microsoft.Network/virtualNetworks/virtualNetworkPeerings",
"useRemoteGateways": false
}
$ az network vnet peering create --resource-group $resource --name peering02 --vnet-name $vnet02Name --remote-vnet $vnet01Name --allow-vnet-access
{
  "allowForwardedTraffic": false,
  "allowGatewayTransit": false,
  "allowVirtualNetworkAccess": true,
  "doNotVerifyRemoteGateways": false,
  "etag": "W/\"484a5dc5-bfc0-4e21-b1a8-32eaeab1bd2a\"",
  "id": "/subscriptions/5966fe4c-f8c6-4550-be5d-680065d50a85/resourceGroups/user-lyuqxewrncf/providers/Microsoft.Network/virtualNetworks/d-vnet158417689/virtualNetworkPeerings/peering02",
  "name": "peering02",
  "peeringState": "Connected",
  "peeringSyncLevel": "FullyInSync",
  "provisioningState": "Succeeded",
  "remoteAddressSpace": {
    "addressPrefixes": [
      "10.0.0.0/24"
    ]
  },
  "remoteBgpCommunities": null,
  "remoteVirtualNetwork": {
    "id": "/subscriptions/5966fe4c-f8c6-4550-be5d-680065d50a85/resourceGroups/user-lyuqxewrncf/providers/Microsoft.Network/virtualNetworks/s-vnet158417689",
    "resourceGroup": "user-lyuqxewrncf"
  },
  "remoteVirtualNetworkAddressSpace": {
    "addressPrefixes": [
      "10.0.0.0/24"
    ]
  },
  "remoteVirtualNetworkEncryption": null,
  "resourceGroup": "user-lyuqxewrncf",
  "resourceGuid": "fdbb8750-1e9e-0e05-1699-436811835aef",
  "type": "Microsoft.Network/virtualNetworks/virtualNetworkPeerings",
  "useRemoteGateways": false
}
$
```

Important: As you saw, to connect two VNETs, you need to create two peering resources to cover connections from both networks.

In the next step, we will confirm that the peering resources are created.

Confirm the Peering Creation using Azure CLI

Use the following command to get the list of created peering resources for the first VNet:

```
az network vnet peering list --resource-group $resource --vnet-name $vnet01Name --query "[].{Name:name}"
```

In the command output, confirm that you can see one peering named peering01.

Now run the following command to get the list of created peering resources for the second VNet:

```
az network vnet peering list --resource-group $resource --vnet-name $vnet02Name --query "[].{Name:name}"
```

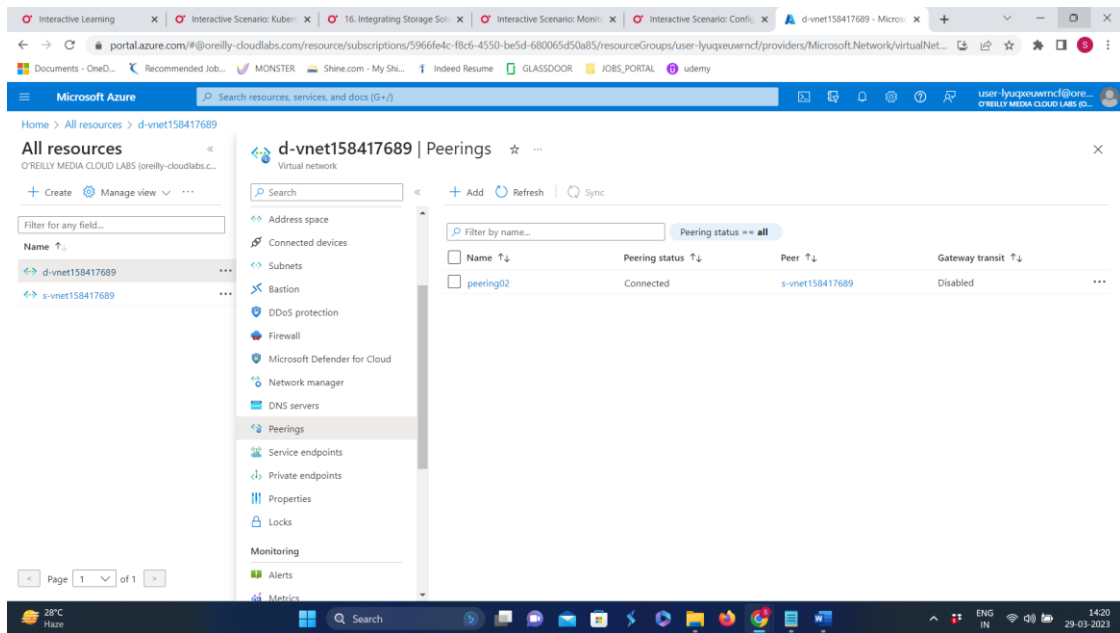
In the command output, confirm that you can see one peering named peering02

```
}
$ az network vnet peering list --resource-group $resource --vnet-name $vnet01Name --query "[].{Name:name}"
[
  {
    "Name": "peering01"
  }
]
$ az network vnet peering list --resource-group $resource --vnet-name $vnet02Name --query "[].{Name:name}"
[
  {
    "Name": "peering02"
  }
]
$
```

Check the VNet Peering in the Azure Portal

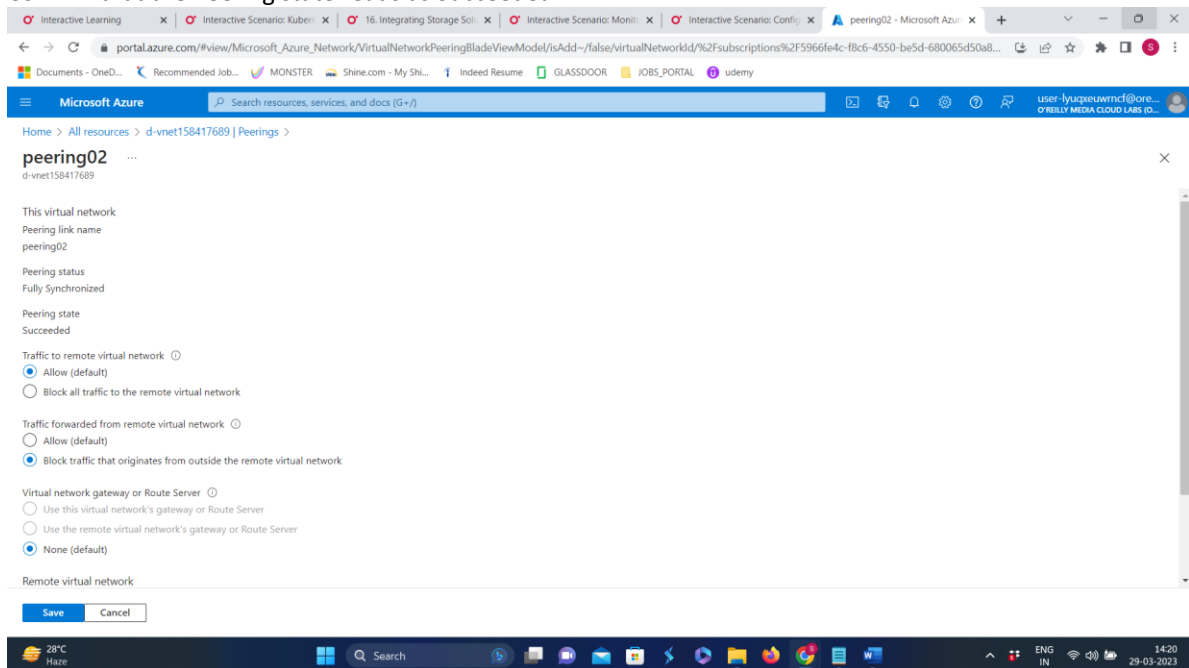
Click on the first VNet.

Under Settings, click on Peerings.



Click on the peering in the list.

Confirm that the Peering state reads as Succeeded



Delete Azure Virtual Network Peerings

Use the following two commands to remove the VNet peering:

```
az network vnet peering delete --resource-group $resource --name peering01 --vnet-name $vnet01Name
az network vnet peering delete --resource-group $resource --name peering02 --vnet-name $vnet02Name
```

Wait for the command to execute. Now use the following commands to get the list of available peerings in both VNets:

```
az network vnet peering list --resource-group $resource --vnet-name $vnet01Name --query "[].{Name:name}"
az network vnet peering list --resource-group $resource --vnet-name $vnet02Name --query "[].{Name:name}"
```