**How to connect VNets that are in the same subscription**

**Before you begin**

**Note**

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [**Introducing the new Azure PowerShell Az module**](https://docs.microsoft.com/en-us/powershell/azure/new-azureps-module-az). For Az module installation instructions, see [**Install Azure PowerShell**](https://docs.microsoft.com/en-us/powershell/azure/install-az-ps).

* Because it takes up to 45 minutes to create a gateway, Azure Cloud Shell will timeout periodically during this exercise. You can restart Cloud Shell by clicking in the upper left of the terminal. Be sure to redeclare any variables when you restart the terminal.
* If you would rather install latest version of the Azure PowerShell module locally, see [How to install and configure Azure PowerShell](https://docs.microsoft.com/en-us/powershell/azure/overview).

**Step 1 - Plan your IP address ranges**

In the following steps, you create two virtual networks along with their respective gateway subnets and configurations. You then create a VPN connection between the two VNets. It’s important to plan the IP address ranges for your network configuration. Keep in mind that you must make sure that none of your VNet ranges or local network ranges overlap in any way. In these examples, we do not include a DNS server. If you want name resolution for your virtual networks, see [Name resolution](https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-name-resolution-for-vms-and-role-instances).

We use the following values in the examples:

**Values for TestVNet1:**

* VNet Name: TestVNet1
* Resource Group: TestRG1
* Location: East US
* TestVNet1: 10.11.0.0/16 & 10.12.0.0/16
* FrontEnd: 10.11.0.0/24
* BackEnd: 10.12.0.0/24
* GatewaySubnet: 10.12.255.0/27
* GatewayName: VNet1GW
* Public IP: VNet1GWIP
* VPNType: RouteBased
* Connection(1to4): VNet1toVNet4
* Connection(1to5): VNet1toVNet5 (For VNets in different subscriptions)
* ConnectionType: VNet2VNet

**Values for TestVNet4:**

* VNet Name: TestVNet4
* TestVNet2: 10.41.0.0/16 & 10.42.0.0/16
* FrontEnd: 10.41.0.0/24
* BackEnd: 10.42.0.0/24
* GatewaySubnet: 10.42.255.0/27
* Resource Group: TestRG4
* Location: West US
* GatewayName: VNet4GW
* Public IP: VNet4GWIP
* VPNType: RouteBased
* Connection: VNet4toVNet1
* ConnectionType: VNet2VNet

**Step 2 - Create and configure TestVNet1**

1. Verify your subscription settings.

Connect to your account if you are running PowerShell locally on your computer. If you are using Azure Cloud Shell, you are connected automatically.

Azure PowerShellCopyTry It

Connect-AzAccount

Check the subscriptions for the account.

Azure PowerShellCopyTry It

Get-AzSubscription

If you have more than one subscription, specify the subscription that you want to use.

Azure PowerShellCopyTry It

Select-AzSubscription -SubscriptionName nameofsubscription

1. Declare your variables. This example declares the variables using the values for this exercise. In most cases, you should replace the values with your own. However, you can use these variables if you are running through the steps to become familiar with this type of configuration. Modify the variables if needed, then copy and paste them into your PowerShell console.

Azure PowerShellCopyTry It

$RG1 = "TestRG1"

$Location1 = "East US"

$VNetName1 = "TestVNet1"

$FESubName1 = "FrontEnd"

$BESubName1 = "Backend"

$GWSubName1 = "GatewaySubnet"

$VNetPrefix11 = "10.11.0.0/16"

$VNetPrefix12 = "10.12.0.0/16"

$FESubPrefix1 = "10.11.0.0/24"

$BESubPrefix1 = "10.12.0.0/24"

$GWSubPrefix1 = "10.12.255.0/27"

$GWName1 = "VNet1GW"

$GWIPName1 = "VNet1GWIP"

$GWIPconfName1 = "gwipconf1"

$Connection14 = "VNet1toVNet4"

$Connection15 = "VNet1toVNet5"

1. Create a resource group.

Azure PowerShellCopyTry It

New-AzResourceGroup -Name $RG1 -Location $Location1

1. Create the subnet configurations for TestVNet1. This example creates a virtual network named TestVNet1 and three subnets, one called GatewaySubnet, one called FrontEnd, and one called Backend. When substituting values, it's important that you always name your gateway subnet specifically GatewaySubnet. If you name it something else, your gateway creation fails.

The following example uses the variables that you set earlier. In this example, the gateway subnet is using a /27. While it is possible to create a gateway subnet as small as /29, we recommend that you create a larger subnet that includes more addresses by selecting at least /28 or /27. This will allow for enough addresses to accommodate possible additional configurations that you may want in the future.

Azure PowerShellCopyTry It

$fesub1 = New-AzVirtualNetworkSubnetConfig -Name $FESubName1 -AddressPrefix $FESubPrefix1

$besub1 = New-AzVirtualNetworkSubnetConfig -Name $BESubName1 -AddressPrefix $BESubPrefix1

$gwsub1 = New-AzVirtualNetworkSubnetConfig -Name $GWSubName1 -AddressPrefix $GWSubPrefix1

1. Create TestVNet1.

Azure PowerShellCopyTry It

New-AzVirtualNetwork -Name $VNetName1 -ResourceGroupName $RG1 `

-Location $Location1 -AddressPrefix $VNetPrefix11,$VNetPrefix12 -Subnet $fesub1,$besub1,$gwsub1

1. Request a public IP address to be allocated to the gateway you will create for your VNet. Notice that the AllocationMethod is Dynamic. You cannot specify the IP address that you want to use. It's dynamically allocated to your gateway.

Azure PowerShellCopyTry It

$gwpip1 = New-AzPublicIpAddress -Name $GWIPName1 -ResourceGroupName $RG1 `

-Location $Location1 -AllocationMethod Dynamic

1. Create the gateway configuration. The gateway configuration defines the subnet and the public IP address to use. Use the example to create your gateway configuration.

Azure PowerShellCopyTry It

$vnet1 = Get-AzVirtualNetwork -Name $VNetName1 -ResourceGroupName $RG1

$subnet1 = Get-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet1

$gwipconf1 = New-AzVirtualNetworkGatewayIpConfig -Name $GWIPconfName1 `

-Subnet $subnet1 -PublicIpAddress $gwpip1

1. Create the gateway for TestVNet1. In this step, you create the virtual network gateway for your TestVNet1. VNet-to-VNet configurations require a RouteBased VpnType. Creating a gateway can often take 45 minutes or more, depending on the selected gateway SKU.

Azure PowerShellCopyTry It

New-AzVirtualNetworkGateway -Name $GWName1 -ResourceGroupName $RG1 `

-Location $Location1 -IpConfigurations $gwipconf1 -GatewayType Vpn `

-VpnType RouteBased -GatewaySku VpnGw1

After you finish the commands, it will take up to 45 minutes to create this gateway. If you are using Azure Cloud Shell, you can restart your CloudShell session by clicking in the upper left of the Cloud Shell terminal, then configure TestVNet4. You don't need to wait until the TestVNet1 gateway completes.

**Step 3 - Create and configure TestVNet4**

Once you've configured TestVNet1, create TestVNet4. Follow the steps below, replacing the values with your own when needed.

1. Connect and declare your variables. Be sure to replace the values with the ones that you want to use for your configuration.

Azure PowerShellCopyTry It

$RG4 = "TestRG4"

$Location4 = "West US"

$VnetName4 = "TestVNet4"

$FESubName4 = "FrontEnd"

$BESubName4 = "Backend"

$GWSubName4 = "GatewaySubnet"

$VnetPrefix41 = "10.41.0.0/16"

$VnetPrefix42 = "10.42.0.0/16"

$FESubPrefix4 = "10.41.0.0/24"

$BESubPrefix4 = "10.42.0.0/24"

$GWSubPrefix4 = "10.42.255.0/27"

$GWName4 = "VNet4GW"

$GWIPName4 = "VNet4GWIP"

$GWIPconfName4 = "gwipconf4"

$Connection41 = "VNet4toVNet1"

1. Create a resource group.

Azure PowerShellCopyTry It

New-AzResourceGroup -Name $RG4 -Location $Location4

1. Create the subnet configurations for TestVNet4.

Azure PowerShellCopyTry It

$fesub4 = New-AzVirtualNetworkSubnetConfig -Name $FESubName4 -AddressPrefix $FESubPrefix4

$besub4 = New-AzVirtualNetworkSubnetConfig -Name $BESubName4 -AddressPrefix $BESubPrefix4

$gwsub4 = New-AzVirtualNetworkSubnetConfig -Name $GWSubName4 -AddressPrefix $GWSubPrefix4

1. Create TestVNet4.

Azure PowerShellCopyTry It

New-AzVirtualNetwork -Name $VnetName4 -ResourceGroupName $RG4 `

-Location $Location4 -AddressPrefix $VnetPrefix41,$VnetPrefix42 -Subnet $fesub4,$besub4,$gwsub4

1. Request a public IP address.

Azure PowerShellCopyTry It

$gwpip4 = New-AzPublicIpAddress -Name $GWIPName4 -ResourceGroupName $RG4 `

-Location $Location4 -AllocationMethod Dynamic

1. Create the gateway configuration.

Azure PowerShellCopyTry It

$vnet4 = Get-AzVirtualNetwork -Name $VnetName4 -ResourceGroupName $RG4

$subnet4 = Get-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet4

$gwipconf4 = New-AzVirtualNetworkGatewayIpConfig -Name $GWIPconfName4 -Subnet $subnet4 -PublicIpAddress $gwpip4

1. Create the TestVNet4 gateway. Creating a gateway can often take 45 minutes or more, depending on the selected gateway SKU.

Azure PowerShellCopyTry It

New-AzVirtualNetworkGateway -Name $GWName4 -ResourceGroupName $RG4 `

-Location $Location4 -IpConfigurations $gwipconf4 -GatewayType Vpn `

-VpnType RouteBased -GatewaySku VpnGw1

**Step 4 - Create the connections**

Wait until both gateways are completed. Restart your Azure Cloud Shell session and copy and paste the variables from the beginning of Step 2 and Step 3 into the console to redeclare values.

1. Get both virtual network gateways.

Azure PowerShellCopyTry It

$vnet1gw = Get-AzVirtualNetworkGateway -Name $GWName1 -ResourceGroupName $RG1

$vnet4gw = Get-AzVirtualNetworkGateway -Name $GWName4 -ResourceGroupName $RG4

1. Create the TestVNet1 to TestVNet4 connection. In this step, you create the connection from TestVNet1 to TestVNet4. You'll see a shared key referenced in the examples. You can use your own values for the shared key. The important thing is that the shared key must match for both connections. Creating a connection can take a short while to complete.

Azure PowerShellCopyTry It

New-AzVirtualNetworkGatewayConnection -Name $Connection14 -ResourceGroupName $RG1 `

-VirtualNetworkGateway1 $vnet1gw -VirtualNetworkGateway2 $vnet4gw -Location $Location1 `

-ConnectionType Vnet2Vnet -SharedKey 'AzureA1b2C3'

1. Create the TestVNet4 to TestVNet1 connection. This step is similar to the one above, except you are creating the connection from TestVNet4 to TestVNet1. Make sure the shared keys match. The connection will be established after a few minutes.

Azure PowerShellCopyTry It

New-AzVirtualNetworkGatewayConnection -Name $Connection41 -ResourceGroupName $RG4 `

-VirtualNetworkGateway1 $vnet4gw -VirtualNetworkGateway2 $vnet1gw -Location $Location4 `

-ConnectionType Vnet2Vnet -SharedKey 'AzureA1b2C3'

1. Verify your connection. See the section [How to verify your connection](https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-vnet-vnet-rm-ps#verify).

**How to verify a connection**

**Important**

When working with gateway subnets, avoid associating a network security group (NSG) to the gateway subnet. Associating a network security group to this subnet may cause your VPN gateway to stop functioning as expected. For more information about network security groups, see [**What is a network security group?**](https://docs.microsoft.com/en-us/azure/virtual-network/security-overview)

You can verify that your connection succeeded by using the 'Get-AzVirtualNetworkGatewayConnection' cmdlet, with or without '-Debug'.

1. Use the following cmdlet example, configuring the values to match your own. If prompted, select 'A' in order to run 'All'. In the example, '-Name' refers to the name of the connection that you want to test.

Azure PowerShellCopyTry It

Get-AzVirtualNetworkGatewayConnection -Name VNet1toSite1 -ResourceGroupName TestRG1

1. After the cmdlet has finished, view the values. In the example below, the connection status shows as 'Connected' and you can see ingress and egress bytes.

Copy

"connectionStatus": "Connected",

"ingressBytesTransferred": 33509044,

"egressBytesTransferred": 4142431