Software Defined Networking (SDN) in Azure Stack HCI training: Lab for Module 6: Gateways

Microsoft Corporation
Published: June 18, 2024

Applies to

SDN training: Module 6: Gateways

Copyright

Revision History

Release Date	Changes
June 18, 2024	Initial release.

Contents

Overview 5

M6.1 Gateway overview and architecture 5

Lab 1: Examine the Gateway Manager service 5

Lab 2: Examine the RAS Gateways 6

Lab 3: Examine the Gateway Pools resource 7

M6.2 Virtual gateways 7

M6.3 Site-to-Site (S2S) Internet Protocol Security (IPsec) virtual private network (VPN) 7

Lab 1: Create virtual network and gateway subnet 7

Lab 2: Create the virtual gateway 8

Lab 3: Create the S2S IPsec connection 9

Lab 4: Configure the Fabrikam-GW01 gateway 13

Lab 5: Examine the virtual gateways resource 19

Lab 6: Examine the gateways and gateway pools 19

Lab 7: Examine Azure Virtual Filtering Platform (VFP) policies 19

Lab 8: Enable IPsec FastPath 20

M6.4 Site-to-Site (S2S) Generic Routing Encapsulation (GRE) virtual private network (VPN) 20

M6.5 L3 forwarding Gateway 20

Lab 1: Create the gateway subnet 20

Lab 2: Create the virtual gateway 20

Lab 3: Create the logical network and subnet 22

Lab 4: Create the L3 connection 22

Lab 5: Configure the Constoso-GW01 gateway 25

Lab 6: Examine the gateways and gateway pools 29

Lab 7: Examine VFP policies applied to the Fabrikam-VM1 VM 29

M6.6 Component tracing......29

Lab 1: Failover RAS Gateway......29

SDN LAB: M6 GATEWAYS

Overview

The SDN Gateway is a…

In this lab, you will learn about gateway infrastructure components, deploying tenant virtual gateways and connections and examining data plane configurations. By the end of this lab, you should have a better understanding of the gateway architecture and functionality.

M6.1 Gateway overview and architecture

Within the Gateway Architecture, we have the Gateway Manager, RAS Gateways and Gateway Pools that we will examine to understand the basic infrastructure that is deployed within our lab environment.

Lab 1: Examine the Gateway Manager service

1. Get the service health state by running:

```
Get-SdnServiceFabricService -ServiceTypeName 'GatewayManager'
```

```
PS C:\Users\Administrator> Get-SdnServiceFabricService -ServiceTypeName 'GatewayManager'


HasPersistedState      : True
ServiceKind            : Stateful
ServiceName            : fabric:/NetworkController/GatewayManager
ServiceTypeName        : GatewayManager
ServiceManifestVersion : 15.0.82
HealthState            : Ok
ServiceStatus          : Active
IsServiceGroup         : False
```

2. Get the process related to the Gateway Manager by running:

```
Get-Process -Name SDNGWM
```

```
PS C:\Users\Administrator> Get-Process -Name SDNGWM

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
   3216      71   193400     222664   2,008.94    2540   0 SDNGWM
```

3. Identify the primary replica for SDN Gateway Manager Service by running:

```
Get-SdnServiceFabricReplica -ServiceTypeName 'GatewayManager' -Primary
```

```
PS C:\Users\Administrator> Get-SdnServiceFabricReplica -ServiceTypeName 'GatewayManager' -Primary


ReplicaId           : 133132575062072549
ReplicaOrInstanceId : 133132575062072549
PartitionId         : 639e46b6-89e7-4ed3-9b67-84b9aca9c2dd
ReplicaRole         : Primary
ServiceKind         : Stateful
Id                  : 133132575062072549
ReplicaStatus       : Ready
HealthState         : Ok
ReplicaAddress      : SDN-NC01.SDN.LAB:0
NodeName            : SDN-NC01
LastInBuildDuration : 00:00:01
```

Lab 2: Examine the RAS Gateways

RAS Gateways when added to SDN Fabric will be represented under */networking/<api version>/gateways* within the NC NB API. Network Controller then leverages this information to program RRAS via WMI.

1. Get the RAS Gateway nodes within the environment:

   ```
   Get-SdnGateway -NcUri 'https://ncnorthbound.sdn.lab' | ConvertTo-Json
   ```

   Examine the properties to determine which gateway is active vs redundant.

2. Connect to the active RAS Gateway VM and check the status of the services. These services may be in a stopped state with the lab environment not having any previously configured Virtual Gateways. In this case, examine these services again after completing sections M6.3 S2S IPsec VPN and M6.5 L3 forwarding gateway:

   ```
   Get-Service -Name RemoteAccess

   Get-Service -Name GatewayService

   Get-Service -Name IKEEXT
   ```

Lab 3: Examine the Gateway Pools resource

RAS Gateways can be added into gateway pools to provide high availability for virtual gateways hosted on the RAS Gateways. There can be multiple pools configured with different RAS Gateways in each, provided services for all connection types or specific types.

1. Get the gateway pools:

```
Get-SdnResource -NcUri 'https://ncnorthbound.sdn.lab' -Resource GatewayPools | ConvertTo-Json
```

M6.2 Virtual gateways

No resources are available to examine until sections M6.3 S2S IPsec VPN and M6.5 L3 forwarding gateway have been completed.

M6.3 Site-to-Site (S2S) Internet Protocol Security (IPsec) virtual private network (VPN)

In this lab, we will create an S2S IPsec VPN for the Fabrikam tenant.

Lab 1: Create virtual network and gateway subnet

1. Navigate to **WAC → Virtual networks → Inventory → +New** and enter:
    a. **Name**: Fabrikam-VNET01
    b. **Address** Prefix: 172.16.0.0/16

c. **Subnets**:
    i. **Subnet01**: 172.16.0.0/24
    ii. **Gateway**-Subnet: 172.16.250.0/27
2. Select **Submit**.
3. Deploy a VM for Subnet01.

Lab 2: Create the virtual gateway

Virtual Gateways are the parent resource in which Network Connections are placed. The Virtual Gateway is provisioned onto the active gateways, with the Network Connection being placed on same or different RAS Gateways. The Virtual Gateway contains information such as the Gateway Pool it's associated with, along with the Virtual Network it's associated with.

Option 1: Create a virtual gateway using Windows Admin Center

Windows Admin Center doesn't expose the ability to create the Virtual Gateway directly and instead you just create the Network Connection that gets inserted into a Virtual Gateway that is automatically created in the backend. Once you create a Network Connection (referred to as a Gateway Connection in WAC), the Virtual Gateway will be provisioned against the subnet specified. If you wish to change the subnet, you must manually delete the Virtual Gateway with PowerShell. Proceed to Lab 3: Create the S2S IPsec Connection.

Option 2: Create a virtual gateway using PowerShell

If you want to have more control regarding the Virtual Gateway creation, you can leverage PowerShell to create the Virtual Gateway.

1. Connect to SDN-DC01 and open PowerShell ISE console and run **Enter-PSSession -ComputerName SDN-NC01**
2. Run the following script to create the virtual network gateway resource:

```
Import-Module NetworkController

$uri = $uri = 'https://ncnorthbound.sdn.lab'

$gwPool = Get-NetworkControllerGatewayPool -ConnectionUri $uri -ResourceId
DefaultAll

$RoutingSubnet = Get-NetworkControllerVirtualSubnet -ConnectionUri $uri -
VirtualNetworkId 'Fabrikam-VNET01' -ResourceId 'Gateway-Subnet'



# Create a new object for tenant virtual gateway
```

```
$VirtualGWProperties = New-Object
Microsoft.Windows.NetworkController.VirtualGatewayProperties


# Update gateway pool reference

$VirtualGWProperties.GatewayPools = @()

$VirtualGWProperties.GatewayPools += $gwPool


# Specify the Virtual Subnet to use for routing between the gateway and virtual
network

$VirtualGWProperties.GatewaySubnets = @()

$VirtualGWProperties.GatewaySubnets += $RoutingSubnet


# Update the rest of the virtual gateway object properties

$VirtualGWProperties.RoutingType = "Dynamic"

$VirtualGWProperties.NetworkConnections = @()

$VirtualGWProperties.BgpRouters = @()


# Add the new virtual gateway for tenant

New-NetworkControllerVirtualGateway -ConnectionUri $uri -ResourceId
"Fabrikam_VirtualGW" -Properties $VirtualGWProperties
```

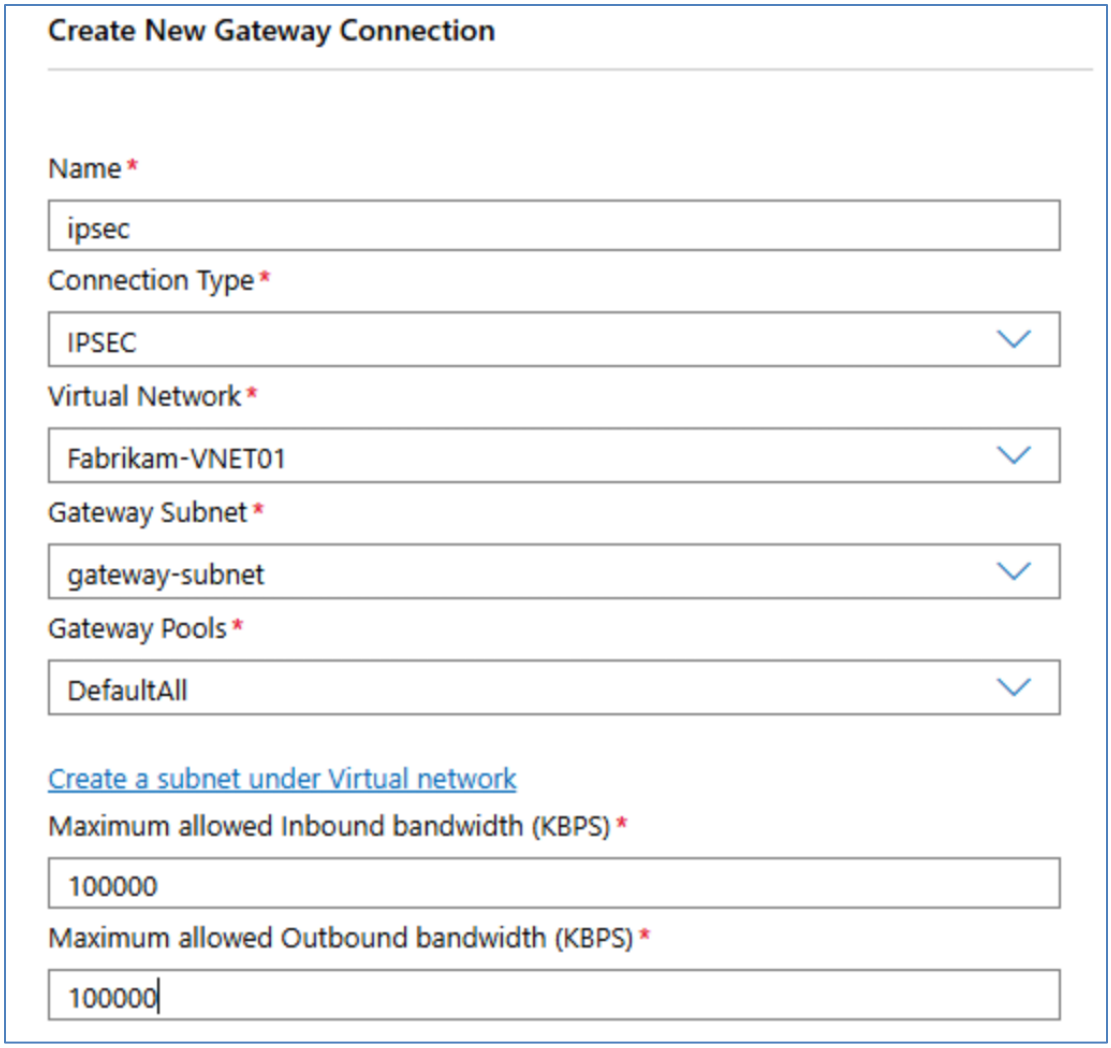
Lab 3: Create the S2S IPsec connection

The Network Connection (Gateway Connection) contains the programming information required to establish connectivity between the remote device and the SDN resources. In this lab, we will create a S2S IPsec connection which will contain the main mode (phase 1) and quick mode (phase 2) policies that will be used to establish the encrypted tunnel. This is ideal if your traffic must traverse the internet.

Option 1: Create a S2S connection using Windows Admin Center

1. Navigate to **Gateway Connections → +New** and enter the following:

a. **Name**: ipsec
b. **Connection Type**: IPSEC
c. **Virtual Network**: Fabrikam-VNET01
d. **Gateway Subnet**: gateway-subnet
e. **Gateway Pools**: DefaultAll
f. **Maximum Inbound/Outbound bandwidth (KBPS):** 100000
g. **Destination IP**: 3.3.3.3
h. **Routes**:
    i. **Route 1:**
        1. Route Metric: 10
        2. Destination Subnet Prefix: 192.168.254.0/24
    ii. **Route 2**:
        1. Route Metric: 10
        2. Destination Subnet Prefix: 4.4.4.4/32
i. **IPSEC Shared Secret**: Password1



Option 2: Create a S2S connection using PowerShell

Use the following script to create the S2S connection using PowerShell:

```
$uri = 'https://ncnorthbound.sdn.lab'

$VirtualGatewayId = Get-NetworkControllerVirtualGateway -ConnectionUri $uri -
ResourceId 'Fabrikam_VirtualGW'


# Create a new object for Tenant Network Connection

$nwConnectionProperties = New-Object
Microsoft.Windows.NetworkController.NetworkConnectionProperties


# Update the common object properties
```

```
$nwConnectionProperties.ConnectionType = "IPSec"

$nwConnectionProperties.OutboundKiloBitsPerSecond = 100000

$nwConnectionProperties.InboundKiloBitsPerSecond = 100000


# Update specific properties depending on the Connection Type

$nwConnectionProperties.IpSecConfiguration = New-Object
Microsoft.Windows.NetworkController.IpSecConfiguration

$nwConnectionProperties.IpSecConfiguration.AuthenticationMethod = "PSK"

$nwConnectionProperties.IpSecConfiguration.SharedSecret = "Password1"

$nwConnectionProperties.IpSecConfiguration.QuickMode = New-Object
Microsoft.Windows.NetworkController.QuickMode

$nwConnectionProperties.IpSecConfiguration.QuickMode.PerfectForwardSecrecy =
"PFS2048"

$nwConnectionProperties.IpSecConfiguration.QuickMode.AuthenticationTransformationC
onstant = "GCMAES256"

$nwConnectionProperties.IpSecConfiguration.QuickMode.CipherTransformationConstant
= "GCMAES256"

$nwConnectionProperties.IpSecConfiguration.QuickMode.SALifeTimeSeconds = 27000

$nwConnectionProperties.IpSecConfiguration.QuickMode.IdleDisconnectSeconds = 36000

$nwConnectionProperties.IpSecConfiguration.QuickMode.SALifeTimeKiloBytes =
33553408

$nwConnectionProperties.IpSecConfiguration.MainMode = New-Object
Microsoft.Windows.NetworkController.MainMode

$nwConnectionProperties.IpSecConfiguration.MainMode.DiffieHellmanGroup = "Group2"

$nwConnectionProperties.IpSecConfiguration.MainMode.IntegrityAlgorithm = "SHA384"

$nwConnectionProperties.IpSecConfiguration.MainMode.EncryptionAlgorithm = "AES256"

$nwConnectionProperties.IpSecConfiguration.MainMode.SALifeTimeSeconds = 28800

$nwConnectionProperties.IpSecConfiguration.MainMode.SALifeTimeKiloBytes = 33553408


# L3 specific configuration (leave blank for IPSec)

$nwConnectionProperties.IPAddresses = @()
```

```
$nwConnectionProperties.PeerIPAddresses = @()


# Tunnel Destination (Remote Endpoint) Address

$nwConnectionProperties.DestinationIPAddress = '3.3.3.3'


# Update the IPv4 Routes that are reachable over the site-to-site VPN Tunnel

$nwConnectionProperties.Routes = @()

$RouteDstPrefix = @( "192.168.254.0/24", "4.4.4.4/32" )

foreach ( $route in $RouteDstPrefix )

{

    $ipv4Route = New-Object Microsoft.Windows.NetworkController.RouteInfo

    $ipv4Route.DestinationPrefix = $route

    $ipv4Route.metric = 10

    $nwConnectionProperties.Routes += $ipv4Route

}


# Add the new Network Connection for the tenant

New-NetworkControllerVirtualGatewayNetworkConnection -ConnectionUri $uri -
VirtualGatewayId $VirtualGatewayId.ResourceId -ResourceId "ipsec" -Properties
$nwConnectionProperties -Force
```

Lab 4: Configure the Fabrikam-GW01 gateway

In this lab, we now need to simulate a remote VPN device outside of the SDN environment. Within the lab environment provisioned, there have been two RRAS VMs provisioned, one of them being for Fabrikam. In a customer environment, this may be any supported third-party VPN appliance but for our lab purpose, we will leverage Windows RRAS feature.

1. From the lab host, open PowerShell ISE and run the following commands. This will automatically configure the appropriate settings to match what we configured for the Virtual Gateway above. If any errors happen, just re-run the script again:

```
$uri = 'https://ncnorthbound.sdn.lab'

$ncCreds = Get-Credential -Message 'Provide SDN.LAB\Administrator Credentials'

$fabrikamCreds = New-Object System.Management.Automation.PSCredential
('LOCALHOST\Administrator', $ncCreds.Password)



try {

    $dcVM = 'SDN-DC01.SDN.LAB'

    $fabrikamVM = 'Fabrikam-GW01'

    Invoke-Command -ComputerName $dcVM -ScriptBlock {

       "Successfully connected via WinRM using FQDN" | Write-Host -
ForegroundColor:Green

    } -ErrorAction Stop

}

catch {

    "Unable to connect via FQDN. Fallback to using IP address and configuring
TrustedHosts" | Write-Host -ForegroundColor Cyan

    $dcVM = '10.184.108.1'

    $fabrikamVM = '10.10.56.250'

    $stringValue = "{0},{1}" -f $dcVM, $fabrikamVM

    Set-Item -Path WSMan:localhost\client\trustedHosts -Value $stringValue

}



$virtualGateway = Invoke-Command -ComputerName  $dcVM -Credential $ncCreds -
Scriptblock {Get-SdnResource -NcUri $using:uri -Credential $using:ncCreds -
ResourceRef '/virtualGateways/Fabrikam_VirtualGW'}

$virtualNetwork = Invoke-Command -ComputerName  $dcVM  -Credential $ncCreds -
Scriptblock {Get-SdnResource -NcUri $using:uri -Credential $using:ncCreds -
ResourceRef '/virtualNetworks/Fabrikam-VNET01'}
```

```
Invoke-Command -ComputerName $fabrikamVM -Credential $fabrikamCreds -ScriptBlock {

    param ($vgw,$vnet)

    $ConnectionType =
$vgw.Properties.NetworkConnections.properties.ConnectionType

    $RemoteNetwork = $vnet.Properties.AddressSpace.AddressPrefixes

    if ($null -eq $ConnectionType -or $null -eq $RemoteNetwork) {

        "VirtualGateway or VirtualNetwork resource is null" | Write-Error

        return

    }

    $IntAlias = "Ethernet"


    #InstallLoopback

    Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force | Out-
Null

    Install-Module -Name LoopbackAdapter -MinimumVersion 1.2.0.0 -Force | Out-Null

    Import-Module -Name LoopbackAdapter

    $run = (Get-Service RemoteAccess).status

    Set-Service RemoteAccess -StartupType Automatic

    $tunnelMode = $true

    $RemoteAccessStatus = Get-RemoteAccess

    if(  $RemoteAccessStatus.VpnS2SStatus -ne "Installed")

    {

        Write-Host "Installing Remote Access VPN type VpnS2S on $env:COMPUTERNAME"

        Install-RemoteAccess -VpnType VpnS2S

    }

    $tunnelAdapter=$ConnectionType+"_LocalPeer"


    if ( !(Get-NetAdapter $tunnelAdapter -ea SilentlyContinue) )
```

```
    {
        New-LoopbackAdapter -Name $tunnelAdapter -Force | Out-Null
    }
    Write-Host "Configuring $($TenantvGW.Type) tunnel on $env:COMPUTERNAME"

    $LocalPeer =
$vgw.Properties.NetworkConnections.properties.DestinationIPAddress
    $DestinationPeer =
$vgw.Properties.NetworkConnections.properties.SourceIPAddress
    #Checking if GrePeer is plumbed
    if ( ! ((Get-NetIPAddress -AddressFamily IPv4).IPAddress -match $LocalPeer) )
{
        Write-Host  "IP Address $LocalPeer is missing so adding it"
        Get-NetAdapter $tunnelAdapter | New-NetIPAddress -IPAddress $LocalPeer -
PrefixLength 32 | Out-Null
    }
    $PSK = 'Password1'

    $PerfectForwardSecrecy =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.QuickMode.Perfect
ForwardSecrecy
    $AuthenticationTransformationConstant =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.QuickMode.Authent
icationTransformationConstant
    $CipherTransformationConstant =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.QuickMode.CipherT
ransformationConstant
    $SALifeTimeSeconds =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.QuickMode.SALifeT
imeSeconds
    $IdleDisconnectSeconds =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.QuickMode.IdleDis
connectSeconds
```

```
    $SALifeTimeKiloBytes =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.QuickMode.SALifeT
imeKiloBytes



    $DiffieHellmanGroup =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.MainMode.DiffieHe
llmanGroup

    $IntegrityAlgorithm =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.MainMode.Integrit
yAlgorithm

    $EncryptionAlgorithm =
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.MainMode.Encrypti
onAlgorithm



    if (
$vgw.Properties.NetworkConnections.properties.IpSecConfiguration.AuthenticationMet
hod -eq "PSK")

    {

        $AuthenticationMethod = "PSKOnly"

    }



    Add-VpnS2SInterface -CustomPolicy -Name $ConnectionType -Destination
$DestinationPeer -SourceIpAddress $LocalPeer `

        -EncryptionMethod $EncryptionAlgorithm -DhGroup $DiffieHellmanGroup  -
PfsGroup $PerfectForwardSecrecy `

        -CipherTransformConstants $CipherTransformationConstant  -
IntegrityCheckMethod $IntegrityAlgorithm `

        -AuthenticationTransformConstants $AuthenticationTransformationConstant -
Protocol "IKEv2" `

        -AuthenticationMethod $AuthenticationMethod -SharedSecret $PSK -
SALifeTimeSeconds $SALifeTimeSeconds `

        -IdleDisconnectSeconds $IdleDisconnectSeconds -
SADataSizeForRenegotiationKilobytes $SALifeTimeKiloBytes `

        -IPv4Subnet "$($RemoteNetwork):10"
```

```
    #Adding Local Loopback to simulate remote site

    $TenantRemoteSubnets =
$vgw.Properties.NetworkConnections.Properties.Routes.DestinationPrefix

    foreach ( $TenantRemoteSubnet in $TenantRemoteSubnets )

    {

        $Net    = $TenantRemoteSubnet.split("/")[0]

        [int]$Cidr   = $TenantRemoteSubnet.split("/")[1]

        $LocalLoopback=$ConnectionType+"_Dummy_Remote_"+$Net


        if ( $cidr -lt 32)

        {

            $Last = [int]$Net.split(".")[-1]

            $Last++

            $Ip = $Net -Replace
"(\d+).(\d+).(\d+).(\d+)",('$1'+"."+'$2'+"."+'$3'+"."+$Last)

            if ( !(Get-NetAdapter $LocalLoopback -ea SilentlyContinue) )

            {

                New-LoopbackAdapter -Name $LocalLoopback -Force | Out-Null

            }

                #Adding $Ip is not plumbed

            if ( ! ((Get-NetIPAddress -AddressFamily IPv4).IPAddress -match $Ip) )

            {

                Write-Host  "IP Address $Ip is missing so adding it"

                Get-NetAdapter $LocalLoopback | New-NetIPAddress -IPAddress $Ip -
PrefixLength 32 | Out-Null

            }

        }

    }
```

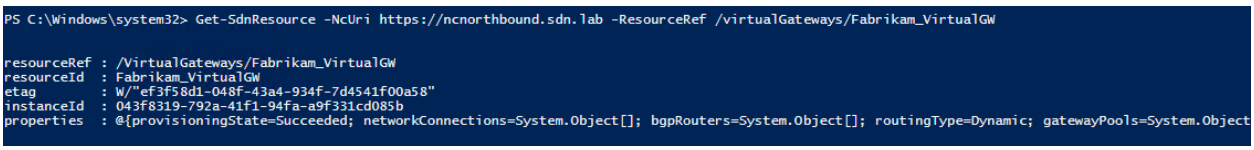
```
} -ArgumentList $virtualGateway, $virtualNetwork
```

2. If successful, you should be able to navigate to WAC → Gateway connections and see a connection with Connection State indicating Connected (this may take a couple minutes).
3. RDP into Fabrikam-VM that you deployed and perform a ping to 192.168.254.1. What are your results?

Lab 5: Examine the virtual gateways resource

1. Connect to DC01 and get the virtualGateways resource. Expand the object by using | ConvertTo-Json or by assigning the output to a variable and enumerating through the properties.

   ```
   Get-SdnResource -NcUri https://ncnorthbound.sdn.lab -ResourceRef
   /virtualGateways/Fabrikam_VirtualGW
   ```

2. If you created the S2S connection using WAC, you will need to determine the appropriate name by enumerating all Virtual Gateways first. Typically, the Virtual Gateway should be created with naming convention of GW-ipsec.

Lab 6: Examine the gateways and gateway pools

Now that there has been a virtual gateway deployed along with a connection, let us revisit M6.2: Examine the RAS Gateways and M6.2: Examine the Gateway Pools Resource.

a. Which gateway pool and gateway are actively hosting the connection recently created?
b. What other properties can you observe that are of interest?

Lab 7: Examine Azure Virtual Filtering Platform (VFP) policies

Examine the VFP policies applied to the VM Farbrikam-VM1:

1. From SDN-DC01, enter a PSSession to the SDN-HOST0# that is hosting the Fabrikam-VM you created in LAB 1: CREATE FABRIKAM VIRTUAL NETWORK AND GATEWAY SUBNET.
2. From there, run the following commands to enumerate the VFP port rules:

   ```
   $port = Get-SdnVMNetworkAdapterPortProfile -VMName 'Fabrikam-VM1'

   Show-SdnVfpPortConfig -PortId $port.PortId -Type IPv4 -Direction OUT
   ```

3. What rules have been added that will affect the outbound traffic? **Hint:** It is related to VNET (Virtual Network) mappings.
    a. What will happen to the traffic outbound?
    b. What do you see for Inbound rules?

Lab 8: Enable IPsec FastPath

For IPsec connections, by default, when you create the connection for your virtual networks, you get the Windows Server 2016 data path and performance numbers. To enable the Windows Server 2019 data path, do the following:

1. Open services.msc on SDN-DC01 and connect to SDN-GW01
2. Find the service named **Azure Gateway Service** and set startup type to Automatic.
3. Restart SDN-GW01. The active connections will failover to the redundant gateway.
4. Repeat steps 1 through 3 to enable **Azure Gateway Service** on SDN-GW02

M6.4 Site-to-Site (S2S) Generic Routing Encapsulation (GRE) virtual private network (VPN)

There are no labs associated with this feature.

M6.5 L3 forwarding Gateway

Lab 1: Create the gateway subnet

1. Navigate to **WAC → Virtual networks → Inventory → Contoso-VNET01**
2. Under **Subnets**, select **+New** and enter:
    a. **Name**: Gateway-Subnet
    b. **Address Prefix**: 172.16.250.0/27

Lab 2: Create the virtual gateway

Option 1: Create a virtual gateway using Windows Admin Center

Windows Admin Center (WAC) does not expose the ability to create the Virtual Gateway directly and instead you just create the network connection that gets inserted into a virtual gateway that is automatically created in the backend. Once you create a network connection (referred to as a Gateway Connection in Windows Admin Center), the virtual gateway is provisioned against the subnet specified. If you want to change the subnet, you must manually delete the virtual gateway using PowerShell.

Proceed to Lab 3: Create the Logical Network and Subnet and then complete Lab 4: Create L3 Connection.

Option 2: Create a virtual gateway using PowerShell

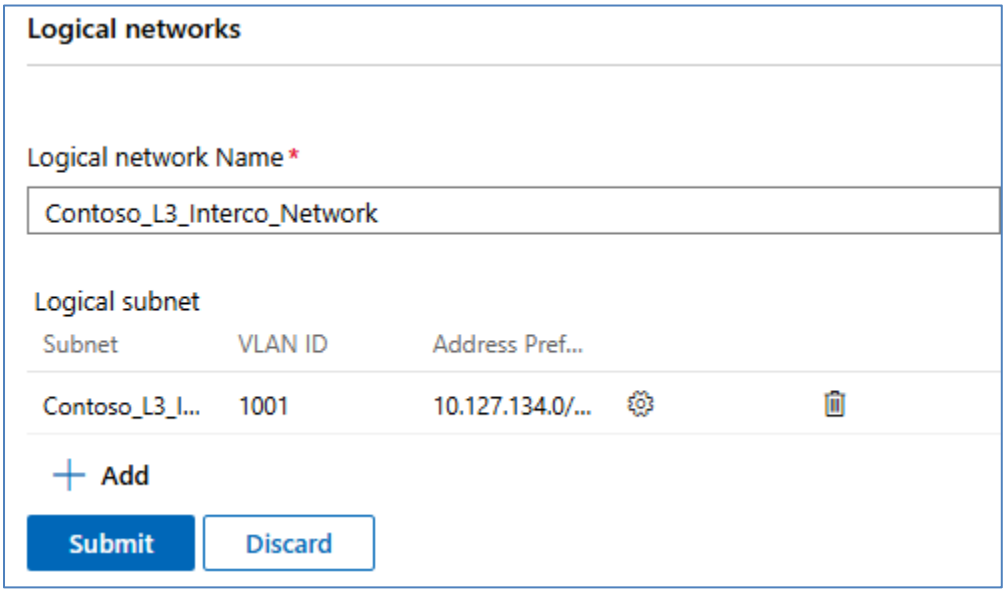
1. Connect to SDN-DC01 and open PowerShell ISE console and run **Enter-PSSession -ComputerName SDN-NC01**
2. Run the following script to create the Virtual Network Gateway resource for Contoso:

```
Import-Module NetworkController


$uri = $uri = 'https://ncnorthbound.sdn.lab'

$gwPool = Get-NetworkControllerGatewayPool -ConnectionUri $uri -ResourceId
DefaultAll

$RoutingSubnet = Get-NetworkControllerVirtualSubnet -ConnectionUri $uri -
VirtualNetworkId 'Contoso-VNET01' -ResourceId 'gateway-subnet'


# Create a new object for tenant virtual gateway

$VirtualGWProperties = New-Object
Microsoft.Windows.NetworkController.VirtualGatewayProperties


# Update gateway pool reference

$VirtualGWProperties.GatewayPools = @()

$VirtualGWProperties.GatewayPools += $gwPool


# Specify the Virtual Subnet to use for routing between the gateway and virtual
network

$VirtualGWProperties.GatewaySubnets = @()

$VirtualGWProperties.GatewaySubnets += $RoutingSubnet
```

```
# Update the rest of the virtual gateway object properties

$VirtualGWProperties.RoutingType = "Dynamic"

$VirtualGWProperties.NetworkConnections = @()

$VirtualGWProperties.BgpRouters = @()


# Add the new virtual gateway for tenant

New-NetworkControllerVirtualGateway -ConnectionUri $uri -ResourceId
"Contoso_VirtualGW" -Properties $VirtualGWProperties
```

Lab 3: Create the logical network and subnet

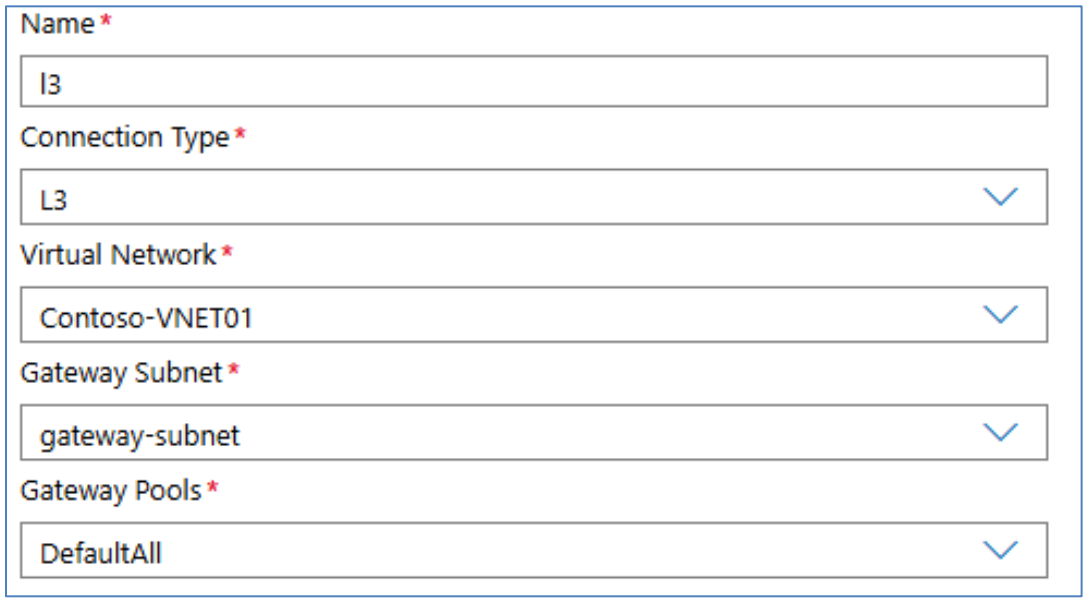
1. Navigate to Windows Admin Center, select **Logical networks,** select **+New**, and then enter the following:
    a. **Logical Network Name**: Contoso_L3_Interco_Network
    b. **Logical Subnet**:
        i. **Logical Subnet Name**: Contoso_L3_Interco_Subnet
        ii. **VLAN ID**: 1001
        iii. **Address Prefix**: 10.127.134.0/25
        iv. **Default Gateway**: 10.127.134.1



Lab 4: Create the L3 connection

Option 1: Create a virtual gateway using Windows Admin Center

1. Navigate to **Gateway Connections → +New** and enter the following:
    a. **Name**: l3
    b. **Connection Type**: L3
    c. Virtual Network**: CONTOSO-VNET01**
    d. **Gateway Subnet**: gateway-subnet
    e. **Gateway Pools**: DefaultAll
    f. **Maximum Inbound/Outbound bandwidth (KBPS):** 100000
    g. **Routes**:
        i. Route 1:
            1. Route Metric: 10
            2. Destination Subnet Prefix: 192.168.254.0/24
        ii. Route 2:
            1. Route Metric: 10
            2. Destination Subnet Prefix: 4.4.4.4/32
    h. **L3 Logical Network**: Contoso_L3_Interco_Network
    i. **L3 Logical Subnet**: Contoso_L3_Interco_Subnet
    j. **L3 IP Address**: 10.127.134.55
    k. **L3 Peer IP**: 10.127.134.65



Option 2: Create a virtual gateway using PowerShell

1. Connect to SDN-DC01 and open PowerShell ISE console and run **Enter-PSSession -ComputerName SDN-NC01**
2. Run the following script to create the L3 connection:

```
Import-Module NetworkController

$uri = 'https://ncnorthbound.sdn.lab'

$LogicalNetwork = Get-NetworkControllerLogicalNetwork -ConnectionUri $uri | ?
ResourceId -eq 'Contoso_L3_Interco_Network'

$nwConnectionProperties = New-Object
Microsoft.Windows.NetworkController.NetworkConnectionProperties

# Update the common object properties

$nwConnectionProperties.ConnectionType = 'L3'

$nwConnectionProperties.OutboundKiloBitsPerSecond = 100000

$nwConnectionProperties.InboundKiloBitsPerSecond = 100000


# Update specific properties depending on the Connection Type

$nwConnectionProperties.L3Configuration = New-Object
Microsoft.Windows.NetworkController.L3Configuration

$nwConnectionProperties.L3Configuration.VlanSubnet =
$LogicalNetwork.properties.Subnets[0]

$nwConnectionProperties.IPAddresses = @()

$localIPAddress = New-Object Microsoft.Windows.NetworkController.CidrIPAddress

$localIPAddress.IPAddress = '10.127.134.55'

$localIPAddress.PrefixLength = ('10.127.134.0/25').split("/")[1]

$nwConnectionProperties.IPAddresses += $localIPAddress

$nwConnectionProperties.PeerIPAddresses = @( "10.127.134.65" )


# Update the IPv4 Routes that are reachable over the site-to-site VPN Tunnel

$nwConnectionProperties.Routes = @()

foreach ( $RouteDstPrefix in @( "172.15.254.0/24", "2.2.2.2/32" ))

{

    $ipv4Route = New-Object Microsoft.Windows.NetworkController.RouteInfo

    $ipv4Route.DestinationPrefix = $RouteDstPrefix
```

```
    <#
    if ( $gw.Type -eq "L3")
    {
        $ipv4Route.NextHop = $Gw.PeerIpAddrGW[0]
    }
    #>
    $ipv4Route.metric = 10
    $nwConnectionProperties.Routes += $ipv4Route
}

# Add the new Network Connection for the tenant
New-NetworkControllerVirtualGatewayNetworkConnection -ConnectionUri $uri -
VirtualGatewayId 'Contoso_VirtualGW' `
    -ResourceId "l3" -Properties $nwConnectionProperties -Force
```

Lab 5: Configure the Constoso-GW01 gateway

1. From the lab host, open PowerShell ISE and run the following commands. This will automatically configure the appropriate settings to match what we configured for the virtual gateway above. If any errors happen, just re-run the script again.

```
    $uri = 'https://ncnorthbound.sdn.lab'
    $ncCreds = Get-Credential -Message 'Provide SDN.LAB\Administrator Credentials'
    $contosoCreds = New-Object System.Management.Automation.PSCredential
    ('LOCALHOST\Administrator', $ncCreds.Password)

try {
    $dcVM = 'SDN-DC01.SDN.LAB'
    $contosoVM = 'Contoso-GW01'
    Invoke-Command -ComputerName $dcVM -ScriptBlock {
```

```
        "Successfully connected via WinRM using FQDN" | Write-Host -
    ForegroundColor:Green

    } -ErrorAction Stop

}

catch {

    "Unable to connect via FQDN. Fallback to using IP address and configuring
TrustedHosts" | Write-Host -ForegroundColor Cyan

    $dcVM = '10.184.108.1'

    $contosoVM = '10.184.108.200'

    $stringValue = "{0},{1}" -f $dcVM, $contosoVM

    Set-Item -Path WSMan:localhost\client\trustedHosts -Value $stringValue

}



$virtualGateway = Invoke-Command -ComputerName $dcVM -Credential $ncCreds -
Scriptblock {Get-SdnResource -NcUri $using:uri -Credential $using:ncCreds -
ResourceRef '/virtualGateways/Contoso_VirtualGW'}

$virtualNetwork = Invoke-Command -ComputerName $dcVM -Credential $ncCreds -
Scriptblock {Get-SdnResource -NcUri $using:uri -Credential $using:ncCreds -
ResourceRef '/virtualNetworks/Contoso-VNET01'}



Invoke-Command -ComputerName $contosoVM -Credential $contosoCreds -ScriptBlock {

    param ($vgw,$vnet)

    $ConnectionType =
$vgw.Properties.NetworkConnections.properties.ConnectionType

    $RemoteNetwork = $vnet.Properties.AddressSpace.AddressPrefixes

    if ($null -eq $ConnectionType -or $null -eq $RemoteNetwork) {

        "VirtualGateway or VirtualNetwork resource is null" | Write-Error

        return

    }

    $IntAlias = "Ethernet"
```

```
    #InstallLoopback

    Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force | Out-
Null

    Install-Module -Name LoopbackAdapter -MinimumVersion 1.2.0.0 -Force | Out-Null

    Import-Module -Name LoopbackAdapter

    $run = (Get-Service RemoteAccess).status

    Set-Service RemoteAccess -StartupType Automatic

    $tunnelMode = $false

    if ( $ConnectionType -eq "L3" )

    {

        $VpnType = "RoutingOnly"

    }


    $RemoteAccessStatus = Get-RemoteAccess

    if ( $ConnectionType -eq "L3" )

    {

        if(  $RemoteAccessStatus.RoutingStatus -ne "Installed")

        {

            Write-Host "Installing Remote Access VPNtype=$VpnType on
$env:COMPUTERNAME"

            Install-RemoteAccess -VpnType $VpnType

        }

    }


    #Adding Local Loopback to simulate remote site

    $TenantRemoteSubnets =
$vgw.Properties.NetworkConnections.Properties.Routes.DestinationPrefix

    foreach ( $TenantRemoteSubnet in $TenantRemoteSubnets )

    {
```

```
        $Net    = $TenantRemoteSubnet.split("/")[0]
        [int]$Cidr   = $TenantRemoteSubnet.split("/")[1]
        $LocalLoopback=$ConnectionType+"_Dummy_Remote_"+$Net
        if ( $cidr -lt 32)
        {
            $Last = [int]$Net.split(".")[-1]
            $Last++
            $Ip = $Net -Replace
"(\d+).(\d+).(\d+).(\d+)",('$1'+"."+'$2'+"."+'$3'+"."+$Last)
            if ( !(Get-NetAdapter $LocalLoopback -ea SilentlyContinue) )
            {
                New-LoopbackAdapter -Name $LocalLoopback -Force | Out-Null
            }
                #Adding $Ip is not plumbed
            if ( ! ((Get-NetIPAddress -AddressFamily IPv4).IPAddress -match $Ip) )
            {
                Write-Host  "IP Address $Ip is missing so adding it"
                Get-NetAdapter $LocalLoopback | New-NetIPAddress -IPAddress $Ip -
PrefixLength 32 | Out-Null
            }
        }
    }

} -ArgumentList $virtualGateway, $virtualNetwork
```

2. If successful, you should be able to navigate to WAC → Gateway connections and see a connection with Connection State indicating Connected (this may take a couple minutes).
3. RDP into Contoso-VM1 that you deployed and perform a ping to 172.15.254.1. What are your results?

Try This:

- From Contoso-VM2 which is deployed against Contoso-VNET02, are you able to ping the 172.15.254.1 IP address?
    - Why can VMs (Virtual Machines) on Contoso-VNET01 ping resources through the L3 gateway but VMs on Contoso-VNET02 cannot, even though VNet (Virtual Network) Peering is established?
    - What modifications must be made to enable routing for Contoso-VNET02 to leverage Contoso-VNET01's gateway?
- Is BGP (Border Gateway Protocol) enabled for this L3 connection?

Lab 6: Examine the gateways and gateway pools

1. Now that there has been a virtual gateway deployed along with a connection, let us revisit M6.2: Examine the RAS Gateways and M6.2: Examine the Gateway Pools Resource.
    a. Which gateway pool and gateway are actively hosting the connection recently created?
    b. What other properties can you observe that are of interest?

Lab 7: Examine VFP policies applied to the Fabrikam-VM1 VM

1. From SDN-DC01, enter a PSSession to the SDN-HOST0# that is hosting the Contoso-VM1 you created previously during this week's training.
2. From there, run the following commands to enumerate the VFP port rules:

```
$port = Get-SdnVMNetworkAdapterPortProfile -VMName 'Contoso-VM1'

Show-SdnVfpPortConfig -PortId $port.PortId -Type IPv4 -Direction OUT
```

3. What rules have been added that will affect the outbound traffic? Hint: It is related to VNET mappings.
    a. What will happen to the traffic outbound?
    b. What do you see for Inbound rules?

M6.6 Component tracing

Lab 1: Failover RAS Gateway

Troubleshooting why resources went offline temporarily is a common thing that operators and/or tenants may request a root cause analysis for. In scenarios where traffic may have temporarily been impacted, it is vital to being able to identify if issue was related to unexpected failure with a RAS gateway that may have caused a temporary disruption to connectivity while the tunnels were being moved to new active RAS gateway.

1. To simulate a failure for RAS Gateway, let us shut down the current active gateway. To determine the active gateway within the environment, you can query the gateways resource. Leverage the command from M6.2: Examine the RAS Gateways to query Network Controller.
2. Once the Active gateway has been identified, shut it down via Hyper-V Manager
3. Perform the steps in M6.2: Examine the RAS Gateways and M6.3: Examine the Virtual Gateways Resource to observe what happens to the resources.
4. Once you are done examining the resources, start the gateway back up.
5. Identify the primary replica for SDN Gateway Manager service using steps from M6.2: Examine Gateway Manager Service.
6. From DC01, enter PSSession into the Network Controller VM that is primary replica.
7. Change your current directory by running **Set-Location C:\Windows\Tracing\SdnDiagnostics\Logs** and enumerate the current directory by running **Get-ChildItem**
8. Locate the most recent .etl file, and convert the .etl file into .txt by running **Convert-SdnEtwTraceToTxt -FileName .\{FileName}.etl**
9. Copy the file to the desktop of labhost, and open with notepad or Visual Studio Code
    a. Can you locate the GWM (Gateway Manager) events related to RAS Gateway offline?
    b. Can you locate the GWM events related to failover of the virtual gateways?