# Virtual Network Peering

Virtual network peering enables you to seamlessly connect two virtual networks. Once peered, the virtual networks appear as one, for connectivity purposes. The traffic between virtual machines in the peered virtual networks is routed through the backbone infrastructure, much like traffic is routed between virtual machines in the same virtual network, through *private* IP addresses only.

The benefits of using virtual network peering include:

* Network traffic between peered virtual networks is private. Traffic between the virtual networks is kept on the backbone network. No public Internet or gateways are required in the communication between the virtual networks.
* A low-latency, high-bandwidth connection between resources in different virtual networks.
* The ability for resources in one virtual network to communicate with resources in a different virtual network once the virtual networks are peered.
* No downtime to resources in either virtual network when creating the peering, or after the peering is created.

## Requirements and constraints

* The peered virtual networks must have non-overlapping IP address spaces.
* The peered virtual networks must be managed by the same Network Controller.
* Address ranges cannot be added to or deleted from the address space of a virtual network once a virtual network is peered with another virtual network. If you need to add address ranges to the address space of a peered virtual network, you must remove the peering, add the address space, and then add the peering again.
* Virtual network peering is between two virtual networks. There is no derived transitive relationship across peerings. For example, if virtualNetworkA is peered with virtualNetworkB, and virtualNetworkB is peered with virtualNetworkC, virtualNetworkA is not peered to virtualNetworkC.

## Connectivity

After virtual networks are peered, resources in either virtual network can directly connect with resources in the peered virtual network.

The network latency between virtual machines in peered virtual networks is the same as the latency within a single virtual network. The network throughput is based on the bandwidth that's allowed for the virtual machine. There isn't any additional restriction on bandwidth within the peering.

The traffic between virtual machines in peered virtual networks is routed directly through the backbone infrastructure, not through a gateway or over the public Internet.

Virtual machines in a virtual network can access the internal load-balancer in the peered virtual network in the same region.

Access control lists (ACLs) can be applied in either virtual network to block access to other virtual networks or subnets, if desired. If you open full connectivity between peered virtual networks (which is the default option), you can apply ACLs to specific subnets or virtual machines to block or deny specific access. To learn more about ACLs, see [Use Access Control Lists (ACLs) to Manage Datacenter Network Traffic Flow](https://docs.microsoft.com/en-us/windows-server/networking/sdn/manage/use-acls-for-traffic-flow).

## Service chaining

You can configure user-defined routes that point to virtual machines in peered virtual networks as the next hop IP address, to enable service chaining. Service chaining enables you to direct traffic from one virtual network to a virtual appliance, in a peered virtual network, through user-defined routes.

You can deploy hub-and-spoke networks, where the hub virtual network can host infrastructure components such as a network virtual appliance. All the spoke virtual networks can then peer with the hub virtual network. Traffic can flow through network virtual appliances in the hub virtual network.

Virtual network peering enables the next hop in a user-defined route to be the IP address of a virtual machine in the peered virtual network. To learn more about user-defined routes, see [Use Network Virtual Appliances on a Virtual Network](https://docs.microsoft.com/en-us/windows-server/networking/sdn/manage/use-network-virtual-appliances-on-a-vn).

## Gateways and on-premises connectivity

Each virtual network, regardless of whether it is peered with another virtual network, can still have its own gateway and use it to connect to an on-premises network.

When virtual networks are peered, you can also configure the gateway in the peered virtual network as a transit point to an on-premises network. In this case, the virtual network that is using a remote gateway cannot have its own gateway. A virtual network can have only one gateway. The gateway can be either a local or remote gateway (in the peered virtual network).

## Monitor

When peering two virtual networks, a peering must be configured for each virtual network in the peering. You can monitor the status of your peering connection. The peering status is one of the following states:

* Initiated: The state shown when you create the peering from the first virtual network to the second virtual network.
* Connected: The state show once you've created the peering from the second virtual network to the first virtual network. The peering state for the first virtual network changes from Initiated to Connected. A virtual network peering is not successfully established until the state for both virtual network peerings is Connected.
* Disconnected: The state shown if a peering from one virtual network to another is deleted after a peering is established between two virtual networks.

For information on configuring Virtual Network Peering, see Configure Virtual Network Peering.

# Configure Virtual Network Peering

This topic contains the following sections which describe the steps required to enable peering between two virtual networks.

* Create the first virtual network
* Create the second virtual network
* Configure peering from the first virtual network to the second virtual network
* Configure peering from the second virtual network to the first virtual network

## Create the first virtual network

The following example script uses Windows Powershell commands to create Contoso’s virtual network with one subnet:

#Find the HNV Provider Logical Network

$logicalnetworks = Get-NetworkControllerLogicalNetwork -ConnectionUri $uri

foreach ($ln in $logicalnetworks) {

if ($ln.Properties.NetworkVirtualizationEnabled -eq "True") {

$HNVProviderLogicalNetwork = $ln

}

}

#Create the Virtual Subnet

$vsubnet = new-object Microsoft.Windows.NetworkController.VirtualSubnet

$vsubnet.ResourceId = "Contoso"

$vsubnet.Properties = new-object Microsoft.Windows.NetworkController.VirtualSubnetProperties

$vsubnet.Properties.AddressPrefix = "24.30.1.0/24"

$uri=”https://restserver”

#Create the Virtual Network

$vnetproperties = new-object Microsoft.Windows.NetworkController.VirtualNetworkProperties

$vnetproperties.AddressSpace = new-object Microsoft.Windows.NetworkController.AddressSpace

$vnetproperties.AddressSpace.AddressPrefixes = @("24.30.1.0/24")

$vnetproperties.LogicalNetwork = $HNVProviderLogicalNetwork

$vnetproperties.Subnets = @($vsubnet)

New-NetworkControllerVirtualNetwork -ResourceId "Contoso\_VNet1" -ConnectionUri $uri -Properties $vnetproperties

## Create the second virtual network

The following example script uses Windows Powershell commands to create Woodgrove’s virtual network with one subnet:

#Find the HNV Provider Logical Network

$logicalnetworks = Get-NetworkControllerLogicalNetwork -ConnectionUri $uri

foreach ($ln in $logicalnetworks) {

if ($ln.Properties.NetworkVirtualizationEnabled -eq "True") {

$HNVProviderLogicalNetwork = $ln

}

}

#Create the Virtual Subnet

$vsubnet = new-object Microsoft.Windows.NetworkController.VirtualSubnet

$vsubnet.ResourceId = "Woodgrove"

$vsubnet.Properties = new-object Microsoft.Windows.NetworkController.VirtualSubnetProperties

$vsubnet.Properties.AddressPrefix = "24.30.2.0/24"

$uri=”https://restserver”

#Create the Virtual Network

$vnetproperties = new-object Microsoft.Windows.NetworkController.VirtualNetworkProperties

$vnetproperties.AddressSpace = new-object Microsoft.Windows.NetworkController.AddressSpace

$vnetproperties.AddressSpace.AddressPrefixes = @("24.30.2.0/24")

$vnetproperties.LogicalNetwork = $HNVProviderLogicalNetwork

$vnetproperties.Subnets = @($vsubnet)

New-NetworkControllerVirtualNetwork -ResourceId "Woodgrove\_VNet1" -ConnectionUri $uri -Properties $vnetproperties

## Configure peering from the first virtual network to the second virtual network

The following example script establishes virtual network peering from Contoso\_vnet1 to Woodgrove\_vnet1.

*$peeringProperties = New-Object Microsoft.Windows.NetworkController.VirtualNetworkPeeringProperties*

*$vnet2=Get-NetworkControllerVirtualNetwork -ConnectionUri $uri -ResourceId “Woodgrove\_VNet1*"

*$peeringProperties.remoteVirtualNetwork = $vnet2*

# Indicates whether communication between the two virtual networks is allowed

*$peeringProperties.allowVirtualnetworkAccess = $true*

# Indicates whether forwarded traffic will be allowed across the vnets

*$peeringProperties.allowForwardedTraffic = $true*

# Indicates whether the peer virtual network can access this virtual network’s gateway

*$peeringProperties.allowGatewayTransit = $false*

# Indicates whether this virtual network will use peer virtual network’s gateway

*$peeringProperties.useRemoteGateways =$false*

*New-NetworkControllerVirtualNetworkPeering -ConnectionUri $uri -VirtualNetworkId “Contoso\_vnet1” -ResourceId “ContosotoWoodgrove” -Properties $peeringProperties*

After this is created, the vnet peering status shows “Initiated”

## Configure peering from the second virtual network to the first virtual network

The following example script establishes virtual network peering from Contoso\_vnet1 to Woodgrove\_vnet1.

*$peeringProperties = New-Object Microsoft.Windows.NetworkController.VirtualNetworkPeeringProperties*

*$vnet2=Get-NetworkControllerVirtualNetwork -ConnectionUri $uri -ResourceId “Contoso\_VNet1*"

*$peeringProperties.remoteVirtualNetwork = $vnet2*

# Indicates whether communication between the two virtual networks is allowed

*$peeringProperties.allowVirtualnetworkAccess = $true*

# Indicates whether forwarded traffic will be allowed across the vnets

*$peeringProperties.allowForwardedTraffic = $true*

# Indicates whether the peer virtual network can access this virtual network’s gateway

*$peeringProperties.allowGatewayTransit = $false*

# Indicates whether this virtual network will use peer virtual network’s gateway

*$peeringProperties.useRemoteGateways =$false*

*New-NetworkControllerVirtualNetworkPeering -ConnectionUri $uri -VirtualNetworkId “Woodgrove\_vnet1” -ResourceId “WoodgrovetoContoso” -Properties $peeringProperties*

After this is created, the vnet peering status shows “Connected” for both the peerings. Now, virtual machines in one virtual network will be able to communicate with virtual machines in the peered virtual network.