

# Contents

## [Virtual Network Documentation](#)

### [Overview](#)

#### [About Virtual Network](#)

### [Quickstarts](#)

#### [Create virtual network - Portal](#)

#### [Create virtual network - PowerShell](#)

#### [Create virtual network - Azure CLI](#)

### [Tutorials](#)

#### [Filter network traffic](#)

#### [Route network traffic](#)

#### [Restrict network access to resources](#)

#### [Connect virtual networks](#)

### [Samples](#)

#### [Azure CLI](#)

#### [Azure PowerShell](#)

#### [Resource Manager templates](#)

#### [Azure Policy templates](#)

### [Concepts](#)

#### [Business continuity](#)

#### [Connectivity](#)

##### [NAT](#)

###### [Overview](#)

###### [Resource](#)

#### [Backend Connectivity Interoperability](#)

###### [Preface and Test Setup](#)

###### [Test Setup Configuration](#)

###### [Control Plane Analysis](#)

###### [Data Plane Analysis](#)

#### [Container networking](#)

- [Cross-network connectivity](#)
- [Peering](#)
- [Virtual network for Azure services](#)
- [IP services](#)
  - [IP addressing](#)
  - [Public IP prefix](#)
  - [IPv6 for Azure VNet](#)
  - [Public IPv6 address prefix](#)
- [Security](#)
  - [DDoS protection](#)
  - [Kubernetes network policies](#)
  - [Routing](#)
  - [Security groups](#)
  - [Service tags](#)
  - [Service endpoints](#)
  - [Service endpoint policies](#)
  - [Virtual network TAP](#)
- [Subnets](#)
  - [Subnet extension](#)
  - [Subnet delegation](#)
  - [Classic deployment](#)
- [How-to guides](#)
  - [Plan and configure](#)
    - [Virtual networks](#)
    - [Name resolution for VMs and cloud services](#)
    - [Use dynamic DNS with your own DNS server](#)
    - [Optimize network throughput](#)
    - [View and modify hostnames](#)
  - [Logs](#)
- [Connectivity](#)
  - [NAT](#)
    - [Create NAT gateway - Portal](#)

- [Create NAT gateway - PowerShell](#)
- [Create NAT gateway - CLI](#)
- [Create and validate NAT gateway - Portal](#)
- [Create and validate NAT gateway - PowerShell](#)
- [Create and validate NAT gateway - CLI](#)
- [Manage NICs](#)
- [Manage virtual network](#)
- [Manage subnets](#)
- [Manage subnet delegation](#)
- [Create VNet peering](#)
  - [Same deployment model - same subscription](#)
    - [Azure PowerShell](#)
    - [Azure CLI](#)
  - [Same deployment model - different subscriptions](#)
  - [Different deployment models - same subscription](#)
  - [Different deployment models - different subscriptions](#)
- [Manage VNet peering](#)
- [Connectivity scenarios](#)
  - [Virtual network to Virtual network](#)
  - [VNet \(Resource Manager\) to a VNet \(Classic\)](#)
  - [VNet to on-premises network \(VPN\)](#)
  - [VNet to on-premises network \(ExpressRoute\)](#)
  - [Highly available hybrid network architecture](#)
- [Configure Virtual network TAP](#)
- [Deploy Container networking](#)
- [IP services](#)
  - [IPv6 for VNet](#)
    - [Add IPv6 to IPv4 application](#)
      - [Azure PowerShell](#)
      - [Azure CLI](#)
  - [Basic Public Load Balancer](#)
    - [Azure PowerShell](#)

Azure CLI

Resource Manager template

Standard Public Load Balancer

Azure PowerShell

Azure CLI

Resource Manager template

Standard Internal Load Balancer

Virtual machine scale set with IPv6

Public IPv6 address prefix

IP addressing

Manage public IP addresses

Manage public IP prefix

Add or remove IP addresses from NICs

Security

Manage DDoS protection

Manage network security groups

Onboard partners to DDoS Protection Standard

Filter network traffic with network security groups

Azure PowerShell

Azure CLI

Route traffic using NVA

Azure PowerShell

Azure CLI

Ansible

Manage route tables

Use service endpoints

Azure PowerShell

Azure CLI

Create service endpoint policies - Portal

Security scenarios

Secure networks with virtual appliances

DMZ between Azure and the Internet

## VM networking

Create VM - static public IP

Azure portal

Azure PowerShell

Azure CLI

Add public IP address to existing VM

Dissociate public IP address from a VM

Create VM - static private IP

Azure portal

Azure PowerShell

Azure CLI

Create VM - multiple IPs

Azure portal

Azure PowerShell

Azure CLI

Add or remove network interfaces

Create VM - multiple NICs

Azure PowerShell

Azure CLI

Create VM - accelerated networking

Azure PowerShell

Azure CLI

Setup DPDK

TCP/IP Performance Tuning for Azure VMs

Virtual machine network throughput

Move across regions

Network security groups - Portal

Network security groups - PowerShell

Virtual networks - Portal

Virtual networks - PowerShell

Public IPs - Portal

Public IPs - PowerShell

## Classic deployment

### Reference

[Azure CLI](#)

[Azure PowerShell](#)

[.NET](#)

[Java](#)

[Node.js](#)

[Python](#)

[REST](#)

[Resource Manager template](#)

[Code samples](#)

### Resources

[Build your skills with Microsoft Learn](#)

[Azure roadmap](#)

[Networking blog](#)

[Networking forum](#)

[Networking feedback](#)

[Pricing](#)

[Pricing calculator](#)

[Stack Overflow](#)

[FAQ](#)

# What is Azure Virtual Network?

1/10/2020 • 5 minutes to read • [Edit Online](#)

Azure Virtual Network (VNet) is the fundamental building block for your private network in Azure. VNet enables many types of Azure resources, such as Azure Virtual Machines (VM), to securely communicate with each other, the internet, and on-premises networks. VNet is similar to a traditional network that you'd operate in your own data center, but brings with it additional benefits of Azure's infrastructure such as scale, availability, and isolation.

## VNet concepts

- **Address space:** When creating a VNet, you must specify a custom private IP address space using public and private (RFC 1918) addresses. Azure assigns resources in a virtual network a private IP address from the address space that you assign. For example, if you deploy a VM in a VNet with address space, 10.0.0.0/16, the VM will be assigned a private IP like 10.0.0.4.
- **Subnets:** Subnets enable you to segment the virtual network into one or more sub-networks and allocate a portion of the virtual network's address space to each subnet. You can then deploy Azure resources in a specific subnet. Just like in a traditional network, subnets allow you to segment your VNet address space into segments that are appropriate for the organization's internal network. This also improves address allocation efficiency. You can secure resources within subnets using Network Security Groups. For more information, see [Security groups](#).
- **Regions:** VNet is scoped to a single region/location; however, multiple virtual networks from different regions can be connected together using Virtual Network Peering.
- **Subscription:** VNet is scoped to a subscription. You can implement multiple virtual networks within each Azure [subscription](#) and Azure [region](#).

## Best practices

As you build your network in Azure, it is important to keep in mind the following universal design principles:

- Ensure non-overlapping address spaces. Make sure your VNet address space (CIDR block) does not overlap with your organization's other network ranges.
- Your subnets should not cover the entire address space of the VNet. Plan ahead and reserve some address space for the future.
- It is recommended you have fewer large VNets than multiple small VNets. This will prevent management overhead.
- Secure your VNet using Network Security Groups (NSGs).

## Communicate with the internet

All resources in a VNet can communicate outbound to the internet, by default. You can communicate inbound to a resource by assigning a public IP address or a public Load Balancer. You can also use public IP or public Load Balancer to manage your outbound connections. To learn more about outbound connections in Azure, see [Outbound connections](#), [Public IP addresses](#), and [Load Balancer](#).

### NOTE

When using only an internal [Standard Load Balancer](#), outbound connectivity is not available until you define how you want [outbound connections](#) to work with an instance-level public IP or a public Load Balancer.

## Communicate between Azure resources

Azure resources communicate securely with each other in one of the following ways:

- **Through a virtual network:** You can deploy VMs, and several other types of Azure resources to a virtual network, such as Azure App Service Environments, the Azure Kubernetes Service (AKS), and Azure Virtual Machine Scale Sets. To view a complete list of Azure resources that you can deploy into a virtual network, see [Virtual network service integration](#).
- **Through a virtual network service endpoint:** Extend your virtual network private address space and the identity of your virtual network to Azure service resources, such as Azure Storage accounts and Azure SQL databases, over a direct connection. Service endpoints allow you to secure your critical Azure service resources to only a virtual network. To learn more, see [Virtual network service endpoints overview](#).
- **Through VNet Peering:** You can connect virtual networks to each other, enabling resources in either virtual network to communicate with each other, using virtual network peering. The virtual networks you connect can be in the same, or different, Azure regions. To learn more, see [Virtual network peering](#).

## Communicate with on-premises resources

You can connect your on-premises computers and networks to a virtual network using any combination of the following options:

- **Point-to-site virtual private network (VPN):** Established between a virtual network and a single computer in your network. Each computer that wants to establish connectivity with a virtual network must configure its connection. This connection type is great if you're just getting started with Azure, or for developers, because it requires little or no changes to your existing network. The communication between your computer and a virtual network is sent through an encrypted tunnel over the internet. To learn more, see [Point-to-site VPN](#).
- **Site-to-site VPN:** Established between your on-premises VPN device and an Azure VPN Gateway that is deployed in a virtual network. This connection type enables any on-premises resource that you authorize to access a virtual network. The communication between your on-premises VPN device and an Azure VPN gateway is sent through an encrypted tunnel over the internet. To learn more, see [Site-to-site VPN](#).
- **Azure ExpressRoute:** Established between your network and Azure, through an ExpressRoute partner. This connection is private. Traffic does not go over the internet. To learn more, see [ExpressRoute](#).

## Filter network traffic

You can filter network traffic between subnets using either or both of the following options:

- **Security groups:** Network security groups and application security groups can contain multiple inbound and outbound security rules that enable you to filter traffic to and from resources by source and destination IP address, port, and protocol. To learn more, see [Network security groups](#) or [Application security groups](#).
- **Network virtual appliances:** A network virtual appliance is a VM that performs a network function, such as a firewall, WAN optimization, or other network function. To view a list of available network virtual appliances that you can deploy in a virtual network, see [Azure Marketplace](#).

## Route network traffic

Azure routes traffic between subnets, connected virtual networks, on-premises networks, and the Internet, by default. You can implement either or both of the following options to override the default routes Azure creates:

- **Route tables:** You can create custom route tables with routes that control where traffic is routed to for each subnet. Learn more about [route tables](#).
- **Border gateway protocol (BGP) routes:** If you connect your virtual network to your on-premises network using an Azure VPN Gateway or ExpressRoute connection, you can propagate your on-premises BGP routes to your virtual networks. Learn more about using BGP with [Azure VPN Gateway](#) and [ExpressRoute](#).

## Azure VNet limits

There are certain limits around the number of Azure resources you can deploy. Most Azure networking limits are at the maximum values. However, you can [increase certain networking limits](#) as specified on the [VNet limits page](#).

## Pricing

There is no charge for using Azure VNet, it is free of cost. Standard charges are applicable for resources, such as Virtual Machines (VMs) and other products. To learn more, see [VNet pricing](#) and the Azure [pricing calculator](#).

## Next steps

To get started using a virtual network, create one, deploy a few VMs to it, and communicate between the VMs. To learn how, see the [Create a virtual network](#) quickstart.

# Quickstart: Create a virtual network using the Azure portal

11/4/2019 • 5 minutes to read • [Edit Online](#)

A virtual network is the fundamental building block for your private network in Azure. It enables Azure resources, like virtual machines (VMs), to securely communicate with each other and with the internet. In this Quickstart, you will learn how to create a virtual network using the Azure portal. Then, you can deploy two VMs into the virtual network, securely communicate between the two VMs, and connect to the VMs from the internet.

If you don't have an Azure subscription, create a [free account](#) now.

## Sign in to Azure

Sign in to the [Azure portal](#).

## Create a virtual network

1. From the Azure portal menu, select **Create a resource**.
2. From the Azure Marketplace, select **Networking > Virtual network**.
3. In **Create virtual network**, enter or select this information:

SETTING	VALUE
Name	Enter <i>myVirtualNetwork</i> .
Address space	Enter <i>10.1.0.0/16</i> .
Subscription	Select your subscription.
Resource group	Select <b>Create new</b> , enter <i>myResourceGroup</i> , then select <b>OK</b> .
Location	Select <b>East US</b> .
Subnet - Name	Enter <i>myVirtualSubnet</i> .
Subnet - Address range	Enter <i>10.1.0.0/24</i> .

4. Leave the rest as default and select **Create**.

## Create virtual machines

Create two VMs in the virtual network:

### Create the first VM

1. From the Azure portal menu, select **Create a resource**.
2. From the Azure Marketplace, select **Compute > Windows Server 2019 Datacenter**.

3. In **Create a virtual machine - Basics**, enter or select this information:

SETTING	VALUE
<b>PROJECT DETAILS</b>	
Subscription	Select your subscription.
Resource group	Select <b>myResourceGroup</b> . You created this in the previous section.
<b>INSTANCE DETAILS</b>	
Virtual machine name	Enter <i>myVm1</i> .
Region	Select <b>East US</b> .
Availability options	Leave the default <b>No infrastructure redundancy required</b> .
Image	Leave the default <b>Windows Server 2019 Datacenter</b> .
Size	Leave the default <b>Standard DS1 v2</b> .
<b>ADMINISTRATOR ACCOUNT</b>	
Username	Enter a username of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the <a href="#">defined complexity requirements</a> .
Confirm Password	Reenter password.
<b>INBOUND PORT RULES</b>	
Public inbound ports	Leave the default <b>None</b> .
<b>SAVE MONEY</b>	
Already have a Windows license?	Leave the default <b>No</b> .

4. Select **Next : Disks**.

5. In **Create a virtual machine - Disks**, leave the defaults and select **Next : Networking**.

6. In **Create a virtual machine - Networking**, select this information:

SETTING	VALUE
Virtual network	Leave the default <b>myVirtualNetwork</b> .
Subnet	Leave the default <b>myVirtualSubnet (10.1.0.0/24)</b> .

SETTING	VALUE
Public IP	Leave the default ( <b>new</b> ) <b>myVm-ip</b> .
Public inbound ports	Select <b>Allow selected ports</b> .
Select inbound ports	Select <b>HTTP</b> and <b>RDP</b> .

7. Select **Next : Management**.

8. In **Create a virtual machine - Management**, for **Diagnostics storage account**, select **Create New**.

9. In **Create storage account**, enter or select this information:

SETTING	VALUE
Name	Enter <i>myvmstorageaccount</i> . If this name is taken, create a unique name.
Account kind	Leave the default <b>Storage (general purpose v1)</b> .
Performance	Leave the default <b>Standard</b> .
Replication	Leave the default <b>Locally-redundant storage (LRS)</b> .

10. Select **OK**

11. Select **Review + create**. You're taken to the **Review + create** page where Azure validates your configuration.

12. When you see the **Validation passed** message, select **Create**.

### Create the second VM

1. Complete steps 1 and 9 from above.

#### NOTE

In step 2, for the **Virtual machine name**, enter *myVm2*.

In step 7, for **Diagnosis storage account**, make sure you select **myvmstorageaccount**.

2. Select **Review + create**. You're taken to the **Review + create** page and Azure validates your configuration.

3. When you see the **Validation passed** message, select **Create**.

## Connect to a VM from the internet

After you've created *myVm1*, connect to the internet.

1. In the portal's search bar, enter *myVm1*.

2. Select the **Connect** button.

The screenshot shows the Azure portal interface for a virtual machine named 'myVm1'. At the top, there's a search bar labeled 'Search (Ctrl+ /)'. Below it, there are three main buttons: 'Connect' (highlighted with a red box), 'Start', and 'Restart'. To the right of these buttons, it says 'Resource group (change) myResourceGroup', 'Status Running', and 'Location East US'. On the left, there's a sidebar with navigation links: 'Overview' (which is selected and highlighted in blue), 'Activity log', 'Access control (IAM)', and 'Tags'.

After selecting the **Connect** button, **Connect to virtual machine** opens.

3. Select **Download RDP File**. Azure creates a Remote Desktop Protocol (.rdp) file and downloads it to your computer.
4. Open the downloaded .rdp file.
  - a. If prompted, select **Connect**.
  - b. Enter the username and password you specified when creating the VM.

**NOTE**

You may need to select **More choices > Use a different account**, to specify the credentials you entered when you created the VM.

5. Select **OK**.
6. You may receive a certificate warning during the sign in process. If you receive a certificate warning, select **Yes** or **Continue**.
7. Once the VM desktop appears, minimize it to go back to your local desktop.

## Communicate between VMs

1. In the Remote Desktop of *myVm1*, open PowerShell.
2. Enter `ping myVm2`.

You'll receive a message similar to this:

```
Pinging myVm2.0v0zze1s0uidpvtxz5z0r0cxg.bx.internal.clouda
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.1.0.5:
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

The `ping` fails, because `ping` uses the Internet Control Message Protocol (ICMP). By default, ICMP isn't allowed through the Windows firewall.

3. To allow *myVm2* to ping *myVm1* in a later step, enter this command:

```
New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4
```

This command allows ICMP inbound through the Windows firewall:

4. Close the remote desktop connection to *myVm1*.
5. Complete the steps in [Connect to a VM from the internet](#) again, but connect to *myVm2*.
6. From a command prompt, enter `ping myvm1`.

You'll get back something like this message:

```
Pinging myVm1.0v0zze1s0uiedpvtxz5z0r0cxg.bx.internal.cloudapp.net [10.1.0.4] with 32 bytes of data:  
Reply from 10.1.0.4: bytes=32 time=1ms TTL=128  
Reply from 10.1.0.4: bytes=32 time<1ms TTL=128  
Reply from 10.1.0.4: bytes=32 time<1ms TTL=128  
Reply from 10.1.0.4: bytes=32 time<1ms TTL=128  
  
Ping statistics for 10.1.0.4:  
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),  
    Approximate round trip times in milli-seconds:  
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

You receive replies from *myVm1*, because you allowed ICMP through the Windows firewall on the *myVm1* VM in step 3.

7. Close the remote desktop connection to *myVm2*.

## Clean up resources

When you're done using the virtual network and the VMs, delete the resource group and all of the resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal and select **myResourceGroup** from the search results.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME** and select **Delete**.

## Next steps

In this Quickstart, you created a default virtual network and two VMs. You connected to one VM from the internet and securely communicated between the two VMs. To learn more about virtual network settings, see [Manage a virtual network](#).

By default, Azure allows unrestricted secure communication between VMs. Conversely, it only allows inbound remote desktop connections to Windows VMs from the internet. To learn more about configuring different types of VM network communications, go to the [Filter network traffic](#) tutorial.

# Quickstart: Create a virtual network using PowerShell

10/30/2019 • 5 minutes to read • [Edit Online](#)

A virtual network lets Azure resources, like virtual machines (VMs), communicate privately with each other, and with the internet. In this quickstart, you learn how to create a virtual network. After creating a virtual network, you deploy two VMs into the virtual network. You then connect to the VMs from the internet, and communicate privately over the virtual network.

If you don't have an Azure subscription, create a [free account](#) now.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you decide to install and use PowerShell locally instead, this quickstart requires you to use Azure PowerShell module version 1.0.0 or later. To find the installed version, run `Get-Module -ListAvailable Az`. See [Install Azure PowerShell module](#) for install and upgrade info.

Finally, if you're running PowerShell locally, you'll also need to run `Connect-AzAccount`. That command creates a connection with Azure.

## Create a resource group and a virtual network

There are a handful of steps you have to walk through to get your resource group and virtual network configured.

### [Create the resource group](#)

Before you can create a virtual network, you have to create a resource group to host the virtual network. Create a resource group with [New-AzResourceGroup](#). This example creates a resource group named *myResourceGroup* in the *eastus* location:

```
New-AzResourceGroup -Name myResourceGroup -Location EastUS
```

## Create the virtual network

Create a virtual network with [New-AzVirtualNetwork](#). This example creates a default virtual network named *myVirtualNetwork* in the *EastUS* location:

```
$virtualNetwork = New-AzVirtualNetwork `  
    -ResourceGroupName myResourceGroup `  
    -Location EastUS `  
    -Name myVirtualNetwork `  
    -AddressPrefix 10.0.0.0/16
```

## Add a subnet

Azure deploys resources to a subnet within a virtual network, so you need to create a subnet. Create a subnet configuration named *default* with [Add-AzVirtualNetworkSubnetConfig](#):

```
$subnetConfig = Add-AzVirtualNetworkSubnetConfig `  
    -Name default `  
    -AddressPrefix 10.0.0.0/24 `  
    -VirtualNetwork $virtualNetwork
```

## Associate the subnet to the virtual network

You can write the subnet configuration to the virtual network with [Set-AzVirtualNetwork](#). This command creates the subnet:

```
$virtualNetwork | Set-AzVirtualNetwork
```

# Create virtual machines

Create two VMs in the virtual network.

## Create the first VM

Create the first VM with [New-AzVM](#). When you run the next command, you're prompted for credentials. Enter a user name and password for the VM:

```
New-AzVm `  
    -ResourceGroupName "myResourceGroup" `  
    -Location "East US" `  
    -VirtualNetworkName "myVirtualNetwork" `  
    -SubnetName "default" `  
    -Name "myVm1" `  
    -AsJob
```

The `-AsJob` option creates the VM in the background. You can continue to the next step.

When Azure starts creating the VM in the background, you'll get something like this back:

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
1	Long Running...	AzureLongRun...	Running	True	localhost	New-AzVM

## Create the second VM

Create the second VM with this command:

```
New-AzVm ` 
-ResourceGroupName "myResourceGroup" ` 
-VirtualNetworkName "myVirtualNetwork" ` 
-SubnetName "default" ` 
-Name "myVm2"
```

You'll have to create another user and password. Azure takes a few minutes to create the VM.

### IMPORTANT

Don't continue with the next step until Azure's finished. You'll know it's done when it returns output to PowerShell.

## Connect to a VM from the internet

Use [Get-AzPublicIpAddress](#) to return the public IP address of a VM. This example returns the public IP address of the *myVm1* VM:

```
Get-AzPublicIpAddress ` 
-Name myVm1 ` 
-ResourceGroupName myResourceGroup ` 
| Select IpAddress
```

Open a command prompt on your local computer. Run the `mstsc` command. Replace `<publicIpAddress>` with the public IP address returned from the last step:

### NOTE

If you've been running these commands from a PowerShell prompt on your local computer, and you're using the Az PowerShell module version 1.0 or later, you can continue in that interface.

```
mstsc /v:<publicIpAddress>
```

1. If prompted, select **Connect**.
2. Enter the user name and password you specified when creating the VM.

### NOTE

You may need to select **More choices > Use a different account**, to specify the credentials you entered when you created the VM.

3. Select **OK**.
4. You may receive a certificate warning. If you do, select **Yes** or **Continue**.

# Communicate between VMs

1. In the Remote Desktop of *myVm1*, open PowerShell.
2. Enter `ping myVm2`.

You'll get something like this back:

```
PS C:\Users\myVm1> ping myVm2

Pinging myVm2.ovvzzdcazhb5iczfvonhg2zrb.bx.internal.cloudapp.net
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 10.0.0.5:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

The ping fails, because it uses the Internet Control Message Protocol (ICMP). By default, ICMP isn't allowed through your Windows firewall.

3. To allow *myVm2* to ping *myVm1* in a later step, enter this command:

```
New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4
```

That command lets ICMP inbound through the Windows firewall.

4. Close the remote desktop connection to *myVm1*.
5. Repeat the steps in [Connect to a VM from the internet](#). This time, connect to *myVm2*.
6. From a command prompt on the *myVm2* VM, enter `ping myVm1`.

You'll get something like this back:

```
C:\windows\system32>ping myVm1

Pinging myVm1.e5p2dibbrqtejhq04lqrusvd4g.bx.internal.cloudapp.net [10.0.0.4] with 32 bytes of data:
Reply from 10.0.0.4: bytes=32 time=2ms TTL=128
Reply from 10.0.0.4: bytes=32 time<1ms TTL=128
Reply from 10.0.0.4: bytes=32 time<1ms TTL=128
Reply from 10.0.0.4: bytes=32 time<1ms TTL=128

Ping statistics for 10.0.0.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 2ms, Average = 0ms
```

You receive replies from *myVm1*, because you allowed ICMP through the Windows firewall on the *myVm1* VM in a previous step.

7. Close the remote desktop connection to *myVm2*.

## Clean up resources

When you're done with the virtual network and the VMs, use [Remove-AzResourceGroup](#) to remove the resource group and all the resources it has:

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

## Next steps

In this quickstart, you created a default virtual network and two VMs. You connected to one VM from the internet and communicated privately between the VM and another VM. To learn more about virtual network settings, see [Manage a virtual network](#).

Azure allows unrestricted private communication between virtual machines. By default, Azure only allows inbound remote desktop connections to Windows VMs from the internet. To learn more about configuring different types of VM network communications, go to the [Filter network traffic](#) tutorial.

# Quickstart: Create a virtual network using the Azure CLI

1/28/2020 • 3 minutes to read • [Edit Online](#)

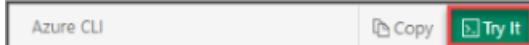
A virtual network enables Azure resources, like virtual machines (VMs), to communicate privately with each other, and with the internet. In this quickstart, you learn how to create a virtual network. After creating a virtual network, you deploy two VMs into the virtual network. You then connect to the VMs from the internet, and communicate privately over the new virtual network.

If you don't have an Azure subscription, create a [free account](#) now.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you decide to install and use Azure CLI locally instead, this quickstart requires you to use Azure CLI version 2.0.28 or later. To find your installed version, run `az --version`. See [Install Azure CLI](#) for install or upgrade info.

## Create a resource group and a virtual network

Before you can create a virtual network, you have to create a resource group to host the virtual network. Create a resource group with [az group create](#). This example creates a resource group named *myResourceGroup* in the *eastus* location:

```
az group create --name myResourceGroup --location eastus
```

Create a virtual network with [az network vnet create](#). This example creates a default virtual network named *myVirtualNetwork* with one subnet named *default*:

```
az network vnet create \
--name myVirtualNetwork \
--resource-group myResourceGroup \
--subnet-name default
```

## Create virtual machines

Create two VMs in the virtual network.

### Create the first VM

Create a VM with [az vm create](#). If SSH keys don't already exist in a default key location, the command creates them. To use a specific set of keys, use the `--ssh-key-value` option. The `--no-wait` option creates the VM in the background, so that you can continue to the next step. This example creates a VM named *myVm1*:

```
az vm create \
--resource-group myResourceGroup \
--name myVm1 \
--image UbuntuLTS \
--generate-ssh-keys \
--no-wait
```

### Create the second VM

Since you used the `--no-wait` option in the previous step, you can go ahead and create the second VM named *myVm2*.

```
az vm create \
--resource-group myResourceGroup \
--name myVm2 \
--image UbuntuLTS \
--generate-ssh-keys
```

### Azure CLI output message

The VMs take a few minutes to create. After Azure creates the VMs, the Azure CLI returns output like this:

```
{
  "fqdns": "",
  "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVm2",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.5",
  "publicIpAddress": "40.68.254.142",
  "resourceGroup": "myResourceGroup"
  "zones": ""
}
```

Take note of the **publicIpAddress**. You will use this address to connect to the VM from the internet in the next step.

# Connect to a VM from the internet

In this command, replace `<publicIpAddress>` with the public IP address of your *myVm2* VM:

```
ssh <publicIpAddress>
```

## Communicate between VMs

To confirm private communication between the *myVm2* and *myVm1* VMs, enter this command:

```
ping myVm1 -c 4
```

You'll receive four replies from *10.0.0.4*.

Exit the SSH session with the *myVm2* VM.

## Clean up resources

When no longer needed, you can use [az group delete](#) to remove the resource group and all the resources it has:

```
az group delete --name myResourceGroup --yes
```

## Next steps

In this quickstart, you created a default virtual network and two VMs. You connected to one VM from the internet and communicated privately between the two VMs. To learn more about virtual network settings, see [Manage a virtual network](#).

Azure lets unrestricted private communication between VMs. By default, Azure only lets inbound remote desktop connections to Windows VMs from the internet. To learn more about configuring different types of VM network communications, go to the [Filter network traffic](#) tutorial.

# Tutorial: Filter network traffic with a network security group using the Azure portal

12/20/2019 • 7 minutes to read • [Edit Online](#)

You can filter network traffic inbound to and outbound from a virtual network subnet with a network security group. Network security groups contain security rules that filter network traffic by IP address, port, and protocol. Security rules are applied to resources deployed in a subnet. In this tutorial, you learn how to:

- Create a network security group and security rules
- Create a virtual network and associate a network security group to a subnet
- Deploy virtual machines (VM) into a subnet
- Test traffic filters

If you prefer, you can complete this tutorial using the [Azure CLI](#) or [PowerShell](#).

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Sign in to Azure

Sign in to the Azure portal at <https://portal.azure.com>.

## Create a virtual network

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Select **Networking**, and then select **Virtual network**.
3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **Create**:

SETTING	VALUE
Name	myVirtualNetwork
Address space	10.0.0.0/16
Subscription	Select your subscription.
Resource group	Select <b>Create new</b> and enter <i>myResourceGroup</i> .
Location	Select <b>East US</b> .
Subnet- Name	mySubnet
Subnet - Address range	10.0.0.0/24

## Create application security groups

An application security group enables you to group together servers with similar functions, such as web servers.

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. In the **Search the Marketplace** box, enter *Application security group*. When **Application security group** appears in the search results, select it, select **Application security group** again under **Everything**, and then select **Create**.
3. Enter, or select, the following information, and then select **Create**:

SETTING	VALUE
Name	myAsgWebServers
Subscription	Select your subscription.
Resource group	Select <b>Use existing</b> and then select <b>myResourceGroup</b> .
Location	East US

4. Complete step 3 again, specifying the following values:

SETTING	VALUE
Name	myAsgMgmtServers
Subscription	Select your subscription.
Resource group	Select <b>Use existing</b> and then select <b>myResourceGroup</b> .
Location	East US

## Create a network security group

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Select **Networking**, and then select **Network security group**.
3. Enter, or select, the following information, and then select **Create**:

SETTING	VALUE
Name	myNsg
Subscription	Select your subscription.
Resource group	Select <b>Use existing</b> and then select <b>myResourceGroup</b> .
Location	East US

## Associate network security group to subnet

1. In the *Search resources, services, and docs* box at the top of the portal, begin typing *myNsg*. When **myNsg** appears in the search results, select it.
2. Under **SETTINGS**, select **Subnets** and then select **+ Associate**, as shown in the following picture:

The screenshot shows the 'Subnets' page for a Network Security Group named 'myNsg'. The left sidebar lists various settings like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and SETTINGS (Inbound security rules, Outbound security rules, Network interfaces). The 'Subnets' link is highlighted with a red box. The main pane shows a table with columns NAME and ADDRESS RANGE, with a message 'No results.'. A large red box highlights the '+ Associate' button at the top of the main pane.

- Under **Associate subnet**, select **Virtual network** and then select **myVirtualNetwork**. Select **Subnet**, select **mySubnet**, and then select **OK**.

## Create security rules

- Under **SETTINGS**, select **Inbound security rules** and then select **+ Add**, as shown in the following picture:

The screenshot shows the 'Inbound security rules' page for the 'myNsg' network security group. The left sidebar shows Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and SETTINGS (Inbound security rules). The 'Inbound security rules' link is highlighted with a red box. The main pane displays a table of existing rules with columns PRIORITY, NAME, PORT, PROTOCOL, SOURCE, DESTINATION, and ACTION. A new rule is being added, indicated by the '+ Add' button highlighted with a red box.

PRIORITY	NAME	PORT	PROTOCOL	SOURCE	DESTINATION	ACTION
65000	AllowVnetInBound	Any	Any	VirtualNet...	VirtualNet...	<span style="color: green;">Allow</span>
65001	AllowAzureLoadBalanc...	Any	Any	AzureLoad...	Any	<span style="color: green;">Allow</span>
65500	DenyAllInBound	Any	Any	Any	Any	<span style="color: red;">Deny</span>

- Create a security rule that allows ports 80 and 443 to the **myAsgWebServers** application security group. Under **Add inbound security rule**, enter, or select the following values, accept the remaining defaults, and then select **Add**:

SETTING	VALUE
Destination	Select <b>Application security group</b> , and then select <b>myAsgWebServers</b> for <b>Application security group</b> .
Destination port ranges	Enter 80,443

SETTING	VALUE
Protocol	Select TCP
Name	Allow-Web-All

3. Complete step 2 again, using the following values:

SETTING	VALUE
Destination	Select <b>Application security group</b> , and then select <b>myAsgMgmtServers</b> for <b>Application security group</b> .
Destination port ranges	Enter 3389
Protocol	Select TCP
Priority	Enter 110
Name	Allow-RDP-All

In this tutorial, RDP (port 3389) is exposed to the internet for the VM that is assigned to the *myAsgMgmtServers* application security group. For production environments, instead of exposing port 3389 to the internet, it's recommended that you connect to Azure resources that you want to manage using a VPN or private network connection.

Once you've completed steps 1-3, review the rules you created. Your list should look like the list in the following picture:

## Create virtual machines

Create two VMs in the virtual network.

### Create the first VM

- On the Azure portal menu or from the **Home** page, select **Create a resource**.
- Select **Compute**, and then select **Windows Server 2016 Datacenter**.
- Enter, or select, the following information, and accept the defaults for the remaining settings:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select <b>Use existing</b> and select <b>myResourceGroup</b> .

SETTING	VALUE
Name	myVmWeb
Location	Select <b>East US</b> .
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the <a href="#">defined complexity requirements</a> .

4. Select a size for the VM and then select **Select**.
5. Under **Networking**, select the following values, and accept the remaining defaults:

SETTING	VALUE
Virtual network	Select <b>myVirtualNetwork</b> .
NIC network security group	Select <b>None</b> .

6. Select **Review + Create** at the bottom, left corner, select **Create** to start VM deployment.

### Create the second VM

Complete steps 1-6 again, but in step 3, name the VM *myVmMgmt*. The VM takes a few minutes to deploy. Do not continue to the next step until the VM is deployed.

## Associate network interfaces to an ASG

When the portal created the VMs, it created a network interface for each VM, and attached the network interface to the VM. Add the network interface for each VM to one of the application security groups you created previously:

1. In the *Search resources, services, and docs* box at the top of the portal, begin typing *myVmWeb*. When the **myVmWeb** VM appears in the search results, select it.
2. Under **SETTINGS**, select **Networking**. Select **Configure the application security groups**, select **myAsgWebServers** for **Application security groups**, and then select **Save**, as shown in the following picture:

The screenshot shows the Azure portal interface for managing a virtual machine named 'myVmWeb'. On the left, there's a sidebar with a search bar and navigation links for Tags, Diagnose and solve problems, Networking (which is highlighted), Disks, and Size. The main area shows the 'Networking' settings for the VM. It includes sections for 'Attach network interface' and 'Detach network interface', a 'Network Interface' section (showing 'myVmWebVMNic' attached to 'myVirtualNetwork/mySubnet'), and an 'APPLICATION SECURITY GROUPS' section. This section has a button to 'Configure the application security groups' and a list where 'myAsgWebServers' is checked. A note says: 'Showing only application security groups in the same region as the network interface. If you choose more than one application security group, they must all exist in the same virtual network.' At the bottom right of the dialog is a 'Save' button.

3. Complete steps 1 and 2 again, searching for the **myVmMgmt** VM and selecting the **myAsgMgmtServers** ASG.

## Test traffic filters

1. Connect to the *myVmMgmt* VM. Enter *myVmMgmt* in the search box at the top of the portal. When **myVmMgmt** appears in the search results, select it. Select the **Connect** button.
2. Select **Download RDP file**.
3. Open the downloaded rdp file and select **Connect**. Enter the user name and password you specified when creating the VM. You may need to select **More choices**, then **Use a different account**, to specify the credentials you entered when you created the VM.
4. Select **OK**.
5. You may receive a certificate warning during the sign-in process. If you receive the warning, select **Yes** or **Continue**, to proceed with the connection.

The connection succeeds, because port 3389 is allowed inbound from the internet to the *myAsgMgmtServers* application security group that the network interface attached to the *myVmMgmt* VM is in.

6. Connect to the *myVmWeb* VM from the *myVmMgmt* VM by entering the following command in a PowerShell session:

```
mstsc /v:myVmWeb
```

You are able to connect to the *myVmWeb* VM from the *myVmMgmt* VM because VMs in the same virtual network can communicate with each other over any port, by default. You can't however, create a remote desktop connection to the *myVmWeb* VM from the internet, because the security rule for the *myAsgWebServers* doesn't allow port 3389 inbound from the internet and inbound traffic from the Internet is denied to all resources, by default.

7. To install Microsoft IIS on the *myVmWeb* VM, enter the following command from a PowerShell session on the *myVmWeb* VM:

```
Install-WindowsFeature -name Web-Server -IncludeManagementTools
```

8. After the IIS installation is complete, disconnect from the *myVmWeb* VM, which leaves you in the *myVmMgmt* VM remote desktop connection.
9. Disconnect from the *myVmMgmt* VM.
10. In the *Search resources, services, and docs* box at the top of the Azure portal, begin typing *myVmWeb* from your computer. When **myVmWeb** appears in the search results, select it. Note the **Public IP address** for your VM. The address shown in the following picture is 137.135.84.74, but your address is different:

Setting	Value
Resource group (change)	myResourceGroup
Status	Running
Location	East US
Subscription (change)	
Subscription ID	
Computer name	myVmWeb
Operating system	Windows
Size	Standard DS1 v2 (1 vcpus, 3.5 GB memory)
Public IP address	137.135.84.74
Virtual network/subnet	myVirtualNetwork/mySubnet
DNS name	Configure

11. To confirm that you can access the *myVmWeb* web server from the internet, open an internet browser on your computer and browse to `http://<public-ip-address-from-previous-step>`. You see the IIS welcome screen, because port 80 is allowed inbound from the internet to the *myAsgWebServers* application security

group that the network interface attached to the *myVmWeb* VM is in.

## Clean up resources

When no longer needed, delete the resource group and all of the resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal. When you see **myResourceGroup** in the search results, select it.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

## Next steps

In this tutorial, you created a network security group and associated it to a virtual network subnet. To learn more about network security groups, see [Network security group overview](#) and [Manage a network security group](#).

Azure routes traffic between subnets by default. You may instead, choose to route traffic between subnets through a VM, serving as a firewall, for example. To learn how to create a route table, advance to the next tutorial.

[Create a route table](#)

# Tutorial: Route network traffic with a route table using the Azure portal

1/28/2020 • 9 minutes to read • [Edit Online](#)

Azure routes traffic between all subnets within a virtual network, by default. You can create your own routes to override Azure's default routing. The ability to create custom routes is helpful if, for example, you want to route traffic between subnets through a network virtual appliance (NVA). In this tutorial, you learn how to:

- Create a route table
- Create a route
- Create a virtual network with multiple subnets
- Associate a route table to a subnet
- Create an NVA that routes traffic
- Deploy virtual machines (VM) into different subnets
- Route traffic from one subnet to another through an NVA

If you prefer, you can finish this tutorial using the [Azure CLI](#) or [Azure PowerShell](#).

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Sign in to Azure

Sign in to the [Azure portal](#).

## Create a route table

1. From the Azure portal menu, select **Create a resource**.
2. In the search box, enter *Route table*. When **Route table** appears in the search results, select it.
3. In the **Route table** page, select **Create**.
4. In **Create route table**, enter or select this information:

SETTING	VALUE
Name	Enter <i>myRouteTablePublic</i> .
Subscription	Select your subscription.
Resource group	Select <b>Create new</b> , enter <i>myResourceGroup</i> , and select <b>OK</b> .
Location	Select <b>East US</b> .
Virtual Network Gateway route propagation	Leave the default <b>Enabled</b> .

5. Select **Create**.

## Create a route

1. In the portal's search bar, enter *myRouteTablePublic*.
2. When **myRouteTablePublic** appears in the search results, select it.
3. In **myRouteTablePublic** under **Settings**, select **Routes** > **+ Add**.

The screenshot shows the Azure portal interface for managing route tables. The main title is 'myRouteTablePublic - Routes'. On the left, there's a sidebar with links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (which is currently selected), Configuration, and Routes (which is highlighted with a red box). At the top right, there's a search bar labeled 'Search routes' and a large red-bordered button labeled '+ Add'. Below the search bar, there's a section titled 'NAME' with the message 'No results.'

4. In **Add route**, enter or select this information:

SETTING	VALUE
Route name	Enter <i>ToPrivateSubnet</i> .
Address prefix	Enter <i>10.0.1.0/24</i> .
Next hop type	Select <b>Virtual appliance</b> .
Next hop address	Enter <i>10.0.2.4</i> .

5. Select **OK**.

## Associate a route table to a subnet

Before you can associate a route table to a subnet, you have to create a virtual network and subnet.

### Create a virtual network

1. On the upper-left side of the screen, select **Create a resource** > **Networking** > **Virtual network**.
2. In **Create virtual network**, enter or select this information:

SETTING	VALUE
Name	Enter <i>myVirtualNetwork</i> .
Address space	Enter <i>10.0.0.0/16</i> .

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select <b>Select existing &gt; myResourceGroup</b> .
Location	Leave the default <b>East US</b> .
Subnet - Name	Enter <i>Public</i> .
Subnet - Address range	Enter <i>10.0.0.0/24</i> .

3. Leave the rest of the defaults and select **Create**.

#### Add subnets to the virtual network

1. In the portal's search bar, enter *myVirtualNetwork*.
2. When **myVirtualNetwork** appears in the search results, select it.
3. In **myVirtualNetwork**, under **Settings**, select **Subnets > + Subnet**.

Dashboard > myVirtualNetwork - Subnets

< > myVirtualNetwork - Subnets

Virtual network

Search (Ctrl+ /)

+ Subnet + Gateway subnet

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Address space

Connected devices

Subnets

NAME	ADDRESS RANGE
Public	10.0.0.0/24

4. In **Add subnet**, enter this information:

SETTING	VALUE
Name	Enter <i>Private</i> .
Address space	Enter <i>10.0.1.0/24</i> .

5. Leave the rest of the defaults and select **OK**.

6. Select **+ Subnet** again. This time, enter this information:

SETTING	VALUE
Name	Enter <b>DMZ</b> .
Address space	Enter <b>10.0.2.0/24</b> .

7. Like the last time, leave the rest of the defaults and select **OK**.

Azure shows the three subnets: **Public**, **Private**, and **DMZ**.

#### Associate myRouteTablePublic to your Public subnet

1. Select **Public**.
2. In **Public**, select **Route table** > **MyRouteTablePublic** > **Save**.

The screenshot shows the Azure portal interface for managing a subnet. On the left, there's a sidebar with 'Dashboard', 'myVirtualNetwork - Subnets', 'Public', and 'Resource'. Below that are buttons for 'Save' (highlighted with a red box), 'Discard', 'Delete', and 'Refresh'. Under 'Address range (CIDR block)', the value '10.0.0.0/24' is listed. 'Available addresses' are shown as 251. 'Network security group' is set to 'None'. The 'Route table' dropdown is open, showing 'None' and 'myRouteTablePublic' (which is also highlighted with a red box). On the right, a 'Resource' blade is open, displaying information about route tables in the 'East US' location.

## Create an NVA

NVAs are VMs that help with network functions like routing and firewall optimization. You can select a different operating system if you want. This tutorial assumes you're using **Windows Server 2016 Datacenter**.

1. On the upper-left side of the screen, select **Create a resource** > **Compute** > **Windows Server 2016 Datacenter**.
2. In **Create a virtual machine - Basics**, enter or select this information:

SETTING	VALUE
<b>PROJECT DETAILS</b>	
Subscription	Select your subscription.
Resource group	Select <b>myResourceGroup</b> .
<b>INSTANCE DETAILS</b>	
Virtual machine name	Enter <b>myVmNva</b> .
Region	Select <b>East US</b> .
Availability options	Leave the default <b>No infrastructure redundancy required</b> .
Image	Leave the default <b>Windows Server 2016 Datacenter</b> .

SETTING	VALUE
Size	Leave the default <b>Standard DS1 v2</b> .
<b>ADMINISTRATOR ACCOUNT</b>	
Username	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the <a href="#">defined complexity requirements</a> .
Confirm Password	Reenter password.
<b>INBOUND PORT RULES</b>	
Public inbound ports	Select <b>None</b> .
<b>SAVE MONEY</b>	
Already have a Windows license?	Leave the default <b>No</b> .

3. Select **Next : Disks**.
4. In **Create a virtual machine - Disks**, select the settings that are right for your needs.
5. Select **Next : Networking**.
6. In **Create a virtual machine - Networking**, select this information:

SETTING	VALUE
Virtual network	Leave the default <b>myVirtualNetwork</b> .
Subnet	Select <b>DMZ (10.0.2.0/24)</b> .
Public IP	Select <b>None</b> . You don't need a public IP address. The VM won't connect over the internet.

7. Leave the rest of the defaults and select **Next : Management**.
8. In **Create a virtual machine - Management**, for **Diagnostics storage account**, select **Create New**.
9. In **Create storage account**, enter or select this information:

SETTING	VALUE
Name	Enter <i>mynvastorageaccount</i> .
Account kind	Leave the default <b>Storage (general purpose v1)</b> .
Performance	Leave the default <b>Standard</b> .
Replication	Leave the default <b>Locally-redundant storage (LRS)</b> .

10. Select **OK**

11. Select **Review + create**. You're taken to the **Review + create** page and Azure validates your configuration.

12. When you see that **Validation passed**, select **Create**.

The VM takes a few minutes to create. Don't keep going until Azure finishes creating the VM. The **Your deployment is underway** page will show you deployment details.

13. When your VM is ready, select **Go to resource**.

## Turn on IP forwarding

Turn on IP forwarding for *myVmNva*. When Azure sends network traffic to *myVmNva*, if the traffic is destined for a different IP address, IP forwarding will send the traffic to the correct location.

1. On **myVmNva**, under **Settings**, select **Networking**.

2. Select **myvmnva123**. That's the network interface Azure created for your VM. It will have a string of numbers to make it unique for you.

The screenshot shows the Azure portal interface for managing a virtual machine named "myVmNva". The "Networking" tab is selected in the left sidebar. The main content area displays the "myVmNva - Networking" page. Key elements include:

- Network Interface:** myvmnva397 (highlighted with a red box)
- Virtual network/subnet:** myVirtualNetwork/DMZ
- APPLICATION SECURITY GROUPS:** A button to "Configure the application security groups".
- INBOUND PORT RULES:** A section showing a single rule for a Network security group named "myVmNva-nsg".

3. Under **Settings**, select **IP configurations**.

4. On **myvmnva123 - IP configurations**, for **IP forwarding**, select **Enabled** and then select **Save**.

Home > myVmNva - Networking > myvmnva397 - IP configurations

## myvmnva397 - IP configurations

Network interface

«

Add Save Discard

IP forwarding settings

IP forwarding

Virtual network myVirtualNetwork

IP configurations

\* Subnet DMZ (10.0.2.0/24)

Search IP configurations

- Overview
- Activity log
- Access control (IAM)
- Tags
- Settings
- IP configurations

## Create public and private virtual machines

Create a public VM and a private VM in the virtual network. Later, you'll use them to see that Azure routes the *Public* subnet traffic to the *Private* subnet through the NVA.

Complete steps 1-12 of [Create an NVA](#). Use most of the same settings. These values are the ones that have to be different:

SETTING	VALUE
<b>PUBLIC VM</b>	
BASICS	
Virtual machine name	Enter <i>myVmPublic</i> .
NETWORKING	
Subnet	Select <b>Public (10.0.0.0/24)</b> .
Public IP address	Accept the default.
Public inbound ports	Select <b>Allow selected ports</b> .
Select inbound ports	Select <b>HTTP and RDP</b> .
MANAGEMENT	
Diagnostics storage account	Leave the default <b>mynvastorageaccount</b> .
<b>PRIVATE VM</b>	
BASICS	
Virtual machine name	Enter <i>myVmPrivate</i> .

SETTING	VALUE
NETWORKING	
Subnet	Select <b>Private (10.0.1.0/24)</b> .
Public IP address	Accept the default.
Public inbound ports	Select <b>Allow selected ports</b> .
Select inbound ports	Select <b>HTTP and RDP</b> .
MANAGEMENT	
Diagnostics storage account	Leave the default <b>mynvastorageaccount</b> .

You can create the *myVmPrivate* VM while Azure creates the *myVmPublic* VM. Don't continue with the rest of the steps until Azure finishes creating both VMs.

## Route traffic through an NVA

### Sign in to myVmPrivate over remote desktop

1. In the portal's search bar, enter *myVmPrivate*.
2. When the **myVmPrivate** VM appears in the search results, select it.
3. Select **Connect** to create a remote desktop connection to the *myVmPrivate* VM.
4. In **Connect to virtual machine**, select **Download RDP File**. Azure creates a Remote Desktop Protocol (.rdp) file and downloads it to your computer.
5. Open the downloaded .rdp file.
  - a. If prompted, select **Connect**.
  - b. Enter the user name and password you specified when creating the Private VM.
  - c. You may need to select **More choices > Use a different account**, to use the Private VM credentials.
6. Select **OK**.

You may receive a certificate warning during the sign in process.

7. Select **Yes** to connect to the VM.

### Enable ICMP through the Windows firewall

In a later step, you'll use the trace route tool to test routing. Trace route uses the Internet Control Message Protocol (ICMP), which the Windows Firewall denies by default. Enable ICMP through the Windows firewall.

1. In the Remote Desktop of *myVmPrivate*, open PowerShell.
2. Enter this command:

```
New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4
```

You're using trace route to test routing in this tutorial. For production environments, we don't recommend

allowing ICMP through the Windows Firewall.

### Turn on IP forwarding within myVmNva

You [turned on IP forwarding](#) for the VM's network interface using Azure. The VM's operating system also has to forward network traffic. Turn on IP forwarding for *myVmNva* VM's operating system with these commands.

- From a command prompt on the *myVmPrivate* VM, open a remote desktop to the *myVmNva* VM:

```
mstsc /v:myvmnva
```

- From PowerShell on the *myVmNva*, enter this command to turn on IP forwarding:

```
Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters -Name IpEnableRouter -Value 1
```

- Restart the *myVmNva* VM. From the taskbar, select **Start button > Power button, Other (Planned) > Continue**.

That also disconnects the remote desktop session.

- After the *myVmNva* VM restarts, create a remote desktop session to the *myVmPublic* VM. While still connected to the *myVmPrivate* VM, open a command prompt and run this command:

```
mstsc /v:myVmPublic
```

- In the Remote Desktop of *myVmPublic*, open PowerShell.

- Enable ICMP through the Windows firewall by entering this command:

```
New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4
```

## Test the routing of network traffic

First, let's test routing of network traffic from the *myVmPublic* VM to the *myVmPrivate* VM.

- From PowerShell on the *myVmPublic* VM, enter this command:

```
tracert myVmPrivate
```

The response is similar to this example:

```
Tracing route to myVmPrivate.vpgub4nqnocezhjgurw44dnxrc.bx.internal.cloudapp.net [10.0.1.4]
over a maximum of 30 hops:

 1    <1 ms      *      1 ms  10.0.2.4
 2     1 ms      1 ms  1 ms  10.0.1.4

Trace complete.
```

You can see the first hop is to 10.0.2.4. It's NVA's private IP address. The second hop is to the private IP address of the *myVmPrivate* VM: 10.0.1.4. Earlier, you added the route to the *myRouteTablePublic* route table and associated it to the *Public* subnet. As a result, Azure sent the traffic through the NVA and not directly to the *Private* subnet.

2. Close the remote desktop session to the *myVmPublic* VM, which leaves you still connected to the *myVmPrivate* VM.

3. From a command prompt on the *myVmPrivate* VM, enter this command:

```
tracert myVmPublic
```

It tests the routing of network traffic from the *myVmPrivate* VM to the *myVmPublic* VM. The response is similar to this example:

```
Tracing route to myVmPublic.vpgub4nqnocezhjgurw44dnxrc.bx.internal.cloudapp.net [10.0.0.4]
over a maximum of 30 hops:

 1      1 ms      1 ms      1 ms  10.0.0.4

Trace complete.
```

You can see Azure routes traffic directly from the *myVmPrivate* VM to the *myVmPublic* VM. By default, Azure routes traffic directly between subnets.

4. Close the remote desktop session to the *myVmPrivate* VM.

## Clean up resources

When no longer needed, delete the resource group and all resources it has:

1. In the portal's search bar, enter *myResourceGroup*.
2. When you see **myResourceGroup** in the search results, select it.
3. Select **Delete resource group**.
4. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

## Next steps

In this tutorial, you created a route table and associated it to a subnet. You created a simple NVA that routed traffic from a public subnet to a private subnet. Now that you know how to do that, you can deploy different pre-configured NVAs from the [Azure Marketplace](#). They carry out many network functions you'll find useful. To learn more about routing, see [Routing overview](#) and [Manage a route table](#).

While you can deploy many Azure resources within a virtual network, Azure can't deploy resources for some PaaS services into a virtual network. It's possible to restrict access to the resources of some Azure PaaS services. The restriction must only be traffic from a virtual network subnet though. To learn how to restrict network access to Azure PaaS resources, advance to the next tutorial.

[Restrict network access to PaaS resources](#)

# Tutorial: Restrict network access to PaaS resources with virtual network service endpoints using the Azure portal

11/19/2019 • 9 minutes to read • [Edit Online](#)

Virtual network service endpoints enable you to limit network access to some Azure service resources to a virtual network subnet. You can also remove internet access to the resources. Service endpoints provide direct connection from your virtual network to supported Azure services, allowing you to use your virtual network's private address space to access the Azure services. Traffic destined to Azure resources through service endpoints always stays on the Microsoft Azure backbone network. In this tutorial, you learn how to:

- Create a virtual network with one subnet
- Add a subnet and enable a service endpoint
- Create an Azure resource and allow network access to it from only a subnet
- Deploy a virtual machine (VM) to each subnet
- Confirm access to a resource from a subnet
- Confirm access is denied to a resource from a subnet and the internet

If you prefer, you can complete this tutorial using the [Azure CLI](#) or [Azure PowerShell](#).

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Log in to Azure

Log in to the Azure portal at <https://portal.azure.com>.

## Create a virtual network

1. Select + **Create a resource** on the upper, left corner of the Azure portal.
2. Select **Networking**, and then select **Virtual network**.
3. Enter, or select, the following information, and then select **Create**:

SETTING	VALUE
Name	myVirtualNetwork
Address space	10.0.0.0/16
Subscription	Select your subscription
Resource group	Select <b>Create new</b> and enter <i>myResourceGroup</i> .
Location	Select <b>East US</b>
Subnet Name	Public
Subnet Address range	10.0.0.0/24

SETTING	VALUE
DDoS protection	Basic
Service endpoints	Disabled
Firewall	Disabled

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with 'Create a resource' and 'Resource groups'. The main content area has a 'New' tab selected. Under 'Azure Marketplace', 'Networking' is highlighted. On the right, a detailed configuration form for creating a virtual network is displayed. It includes fields for Name (set to 'myVirtualNetwork'), Address space (set to '10.0.0.0/16'), Subscription, Resource group (set to 'myResourceGroup'), Location (set to 'East US'), Subnet (set to 'Public'), and Address range (set to '10.0.0.0/24'). The 'Service endpoints' section shows 'Disabled' as the current state. At the bottom of the form are 'Create' and 'Automation options' buttons.

## Enable a service endpoint

Service endpoints are enabled per service, per subnet. Create a subnet and enable a service endpoint for the subnet.

1. In the **Search resources, services, and docs** box at the top of the portal, enter *myVirtualNetwork*. When **myVirtualNetwork** appears in the search results, select it.
2. Add a subnet to the virtual network. Under **SETTINGS**, select **Subnets**, and then select **+ Subnet**, as shown in the following picture:

- Under **Add subnet**, select or enter the following information, and then select **OK**:

SETTING	VALUE
Name	Private
Address range	10.0.1.0/24
Service endpoints	Select <b>Microsoft.Storage</b> under <b>Services</b>

#### Caution

Before enabling a service endpoint for an existing subnet that has resources in it, see [Change subnet settings](#).

## Restrict network access for a subnet

By default, all VMs in a subnet can communicate with all resources. You can limit communication to and from all resources in a subnet by creating a network security group, and associating it to the subnet.

- Select **+ Create a resource** on the upper, left corner of the Azure portal.
- Select **Networking**, and then select **Network security group**.
- Under **Create a network security group**, enter, or select, the following information, and then select **Create**:

SETTING	VALUE
Name	myNsgPrivate
Subscription	Select your subscription
Resource group	Select <b>Use existing</b> and select <i>myResourceGroup</i> .
Location	Select <b>East US</b>

4. After the network security group is created, enter *myNsgPrivate*, in the **Search resources, services, and docs** box at the top of the portal. When **myNsgPrivate** appears in the search results, select it.

5. Under **SETTINGS**, select **Outbound security rules**.

6. Select **+ Add**.

7. Create a rule that allows outbound communication to the Azure Storage service. Enter, or select, the following information, and then select **Add**:

SETTING	VALUE
Source	Select <b>VirtualNetwork</b>
Source port ranges	*
Destination	Select <b>Service Tag</b>
Destination service tag	Select <b>Storage</b>
Destination port ranges	*
Protocol	Any
Action	Allow
Priority	100
Name	Allow-Storage-All

8. Create another outbound security rule that denies communication to the internet. This rule overrides a default rule in all network security groups that allows outbound internet communication. Complete steps 5-7 again, using the following values:

SETTING	VALUE
Source	Select <b>VirtualNetwork</b>
Source port ranges	*
Destination	Select <b>Service Tag</b>
Destination service tag	Select <b>Internet</b>
Destination port ranges	*
Protocol	Any
Action	Deny
Priority	110
Name	Deny-Internet-All

9. Under **SETTINGS**, select **Inbound security rules**.

10. Select **+ Add**.

11. Create an inbound security rule that allows Remote Desktop Protocol (RDP) traffic to the subnet from anywhere. The rule overrides a default security rule that denies all inbound traffic from the internet. Remote desktop connections are allowed to the subnet so that connectivity can be tested in a later step. Under **SETTINGS**, select **Inbound security rules**, select **+Add**, enter the following values, and then select **Add**:

SETTING	VALUE
Source	Any
Source port ranges	*
Destination	Select <b>VirtualNetwork</b>
Destination port ranges	3389
Protocol	Any
Action	Allow
Priority	120
Name	Allow-RDP-All

12. Under **SETTINGS**, select **Subnets**.

13. Select **+ Associate**

14. Under **Associate subnet**, select **Virtual network** and then select **myVirtualNetwork** under **Choose a virtual network**.

15. Under **Choose subnet**, select **Private**, and then select **OK**.

## Restrict network access to a resource

The steps necessary to restrict network access to resources created through Azure services enabled for service endpoints varies across services. See the documentation for individual services for specific steps for each service. The remainder of this tutorial includes steps to restrict network access for an Azure Storage account, as an example.

### Create a storage account

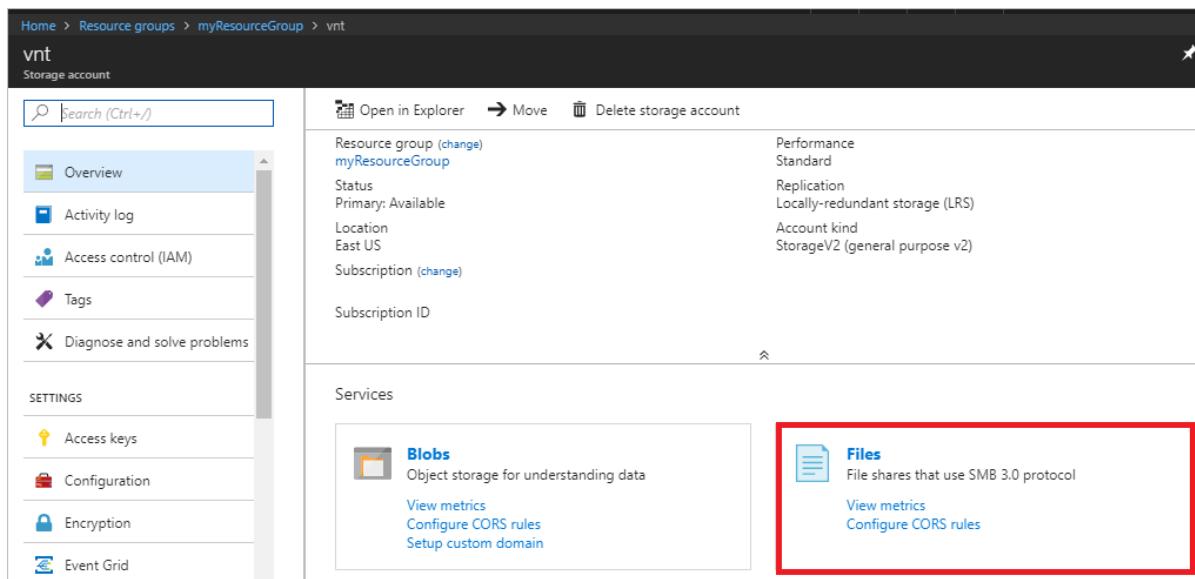
1. Select **+ Create a resource** on the upper, left corner of the Azure portal.
2. Select **Storage**, and then select **Storage account - blob, file, table, queue**.
3. Enter, or select, the following information, accept the remaining defaults, and then select **Create**:

SETTING	VALUE
Name	Enter a name that is unique across all Azure locations, between 3-24 characters in length, using only numbers and lower-case letters.

SETTING	VALUE
Account kind	StorageV2 (general purpose v2)
Location	Select <b>East US</b>
Replication	Locally-redundant storage (LRS)
Subscription	Select your subscription
Resource group	Select <b>Use existing</b> and select <i>myResourceGroup</i> .

### Create a file share in the storage account

1. After the storage account is created, enter the name of the storage account in the **Search resources, services, and docs** box, at the top of the portal. When the name of your storage account appears in the search results, select it.
2. Select **Files**, as shown in the following picture:



3. Select **+ File share**.
4. Enter *my-file-share* under **Name**, and then select **OK**.
5. Close the **File service** box.

### Restrict network access to a subnet

By default, storage accounts accept network connections from clients in any network, including the internet. Deny network access from the internet, and all other subnets in all virtual networks, except for the *Private* subnet in the *myVirtualNetwork* virtual network.

1. Under **SETTINGS** for the storage account, select **Firewalls and virtual networks**.
2. Select **Selected networks**.
3. Select **+Add existing virtual network**.
4. Under **Add networks**, select the following values, and then select **Add**:

Setting	Value
Subscription	Select your subscription.
Virtual networks	Select <b>myVirtualNetwork</b> , under <b>Virtual networks</b>
Subnets	Select <b>Private</b> , under <b>Subnets</b>

The screenshot shows the Azure portal interface for managing storage account firewalls and virtual networks. On the left, a sidebar lists various settings like Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The 'Firewalls and virtual networks' option is selected. The main content area displays the current configuration, which includes allowing access from 'Selected networks'. A 'Virtual networks' section shows a single entry: 'myVirtualNetwork' under 'Virtual networks' and 'Private' under 'Subnets'. A red box highlights this configuration. At the bottom, there are sections for Firewall (IP ranges) and Exceptions (checkboxes for Microsoft services and storage metrics). An 'Add' button is visible at the bottom right of the 'Add networks' dialog.

Home > Resource groups > myResourceGroup > vnt - Firewalls and virtual networks

vnt - Firewalls and virtual networks

Storage account

Search (Ctrl+)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

SETTINGS

Access keys

Configuration

Encryption

Event Grid

Shared access signature

Firewalls and virtual networks

Save Discard

Allow access from

All networks  Selected networks

Configure network security for your storage accounts. [Learn more](#).

Virtual networks

Secure your storage account with virtual networks. [+ Add existing virtual network](#)

VIRTUAL NETWORK	SUBNET	ADDRESS RANGE	ENDPOINT STATUS
No network selected.			

Firewall

Add IP ranges to allow access from the internet or your on-premises networks. [Learn more](#)

ADDRESS RANGE

IP address or CIDR

Exceptions

Allow trusted Microsoft services to access this storage account [?](#)

Allow read access to storage logging from any network

Allow read access to storage metrics from any network

Add networks

\* Subscription

\* Virtual networks [?](#)  
myVirtualNetwork

\* Subnets  
Private

Add

5. Select **Save**.
  6. Close the **Firewalls and virtual networks** box.
  7. Under **SETTINGS** for the storage account, select **Access keys**, as shown in the following picture:

The screenshot shows the Azure Storage account 'vnt' settings page. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and a SETTINGS section with 'Access keys' highlighted by a red box. The main content area displays instructions for using access keys, the storage account name 'vnt', and a table for managing access keys. The first key entry is shown with a yellow warning icon, labeled 'key1', and its value is partially visible as 'xy...'. A blue 'Edit' button is next to the key value.

Use access keys to authenticate your applications when making requests to this Azure storage account. Store your access keys securely - for example, using Azure Key Vault - and don't share them. We recommend regenerating your access keys regularly. You are provided two access keys so that you can maintain connections using one key while regenerating the other.

When you regenerate your access keys, you must update any Azure resources and applications that access this storage account to use the new keys. This action will not interrupt access to disks from your virtual machines. [Learn more](#)

Key	Value
key1	xy... rorNqclqJtaYwWaO4Grfw1UhwRld4FN8TqbYtO...

8. Note the **Key** value, as you'll have to manually enter it in a later step when mapping the file share to a drive letter in a VM.

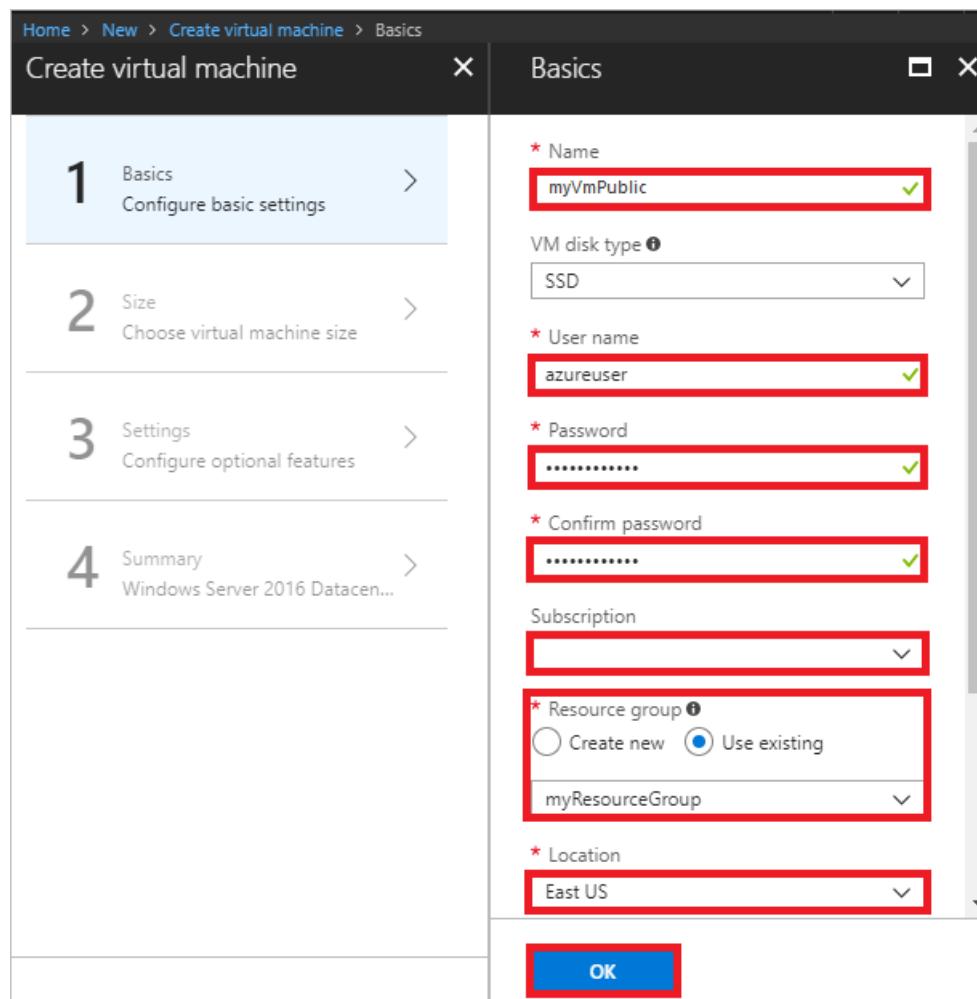
## Create virtual machines

To test network access to a storage account, deploy a VM to each subnet.

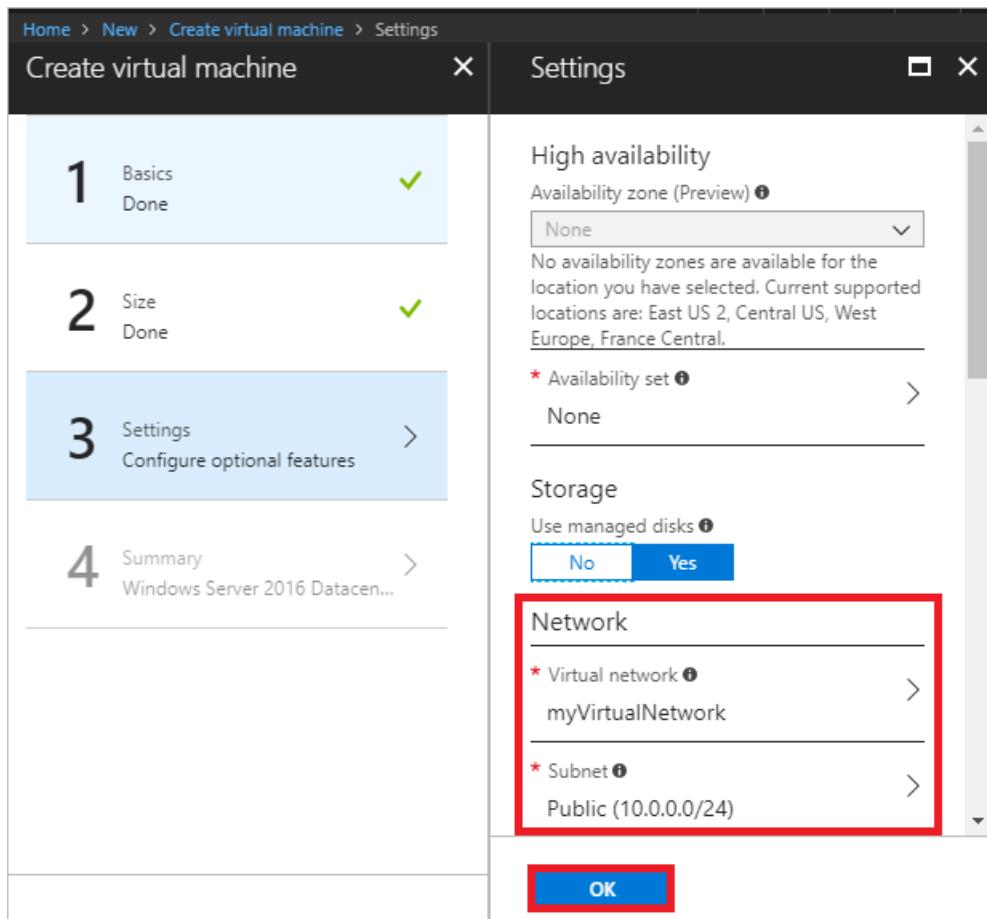
## Create the first virtual machine

1. Select + **Create a resource** found on the upper, left corner of the Azure portal.
2. Select **Compute**, and then select **Windows Server 2016 Datacenter**.
3. Enter, or select, the following information and then select **OK**:

SETTING	VALUE
Name	myVmPublic
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the <a href="#">defined complexity requirements</a> .
Subscription	Select your subscription.
Resource group	Select <b>Use existing</b> and select <b>myResourceGroup</b> .
Location	Select <b>East US</b> .



4. Select a size for the virtual machine and then select **Select**.
5. Under **Settings**, select **Network** and then select **myVirtualNetwork**. Then select **Subnet**, and select **Public**, as shown in the following picture:



6. Under **Network Security Group**, select **Advanced**. The portal automatically creates a network security group for you that allows port 3389, which you'll need open to connect to the virtual machine in a later step. Select **OK** on the **Settings** page.
7. On the **Summary** page, select **Create** to start the virtual machine deployment. The VM takes a few minutes to deploy, but you can continue to the next step while the VM is creating.

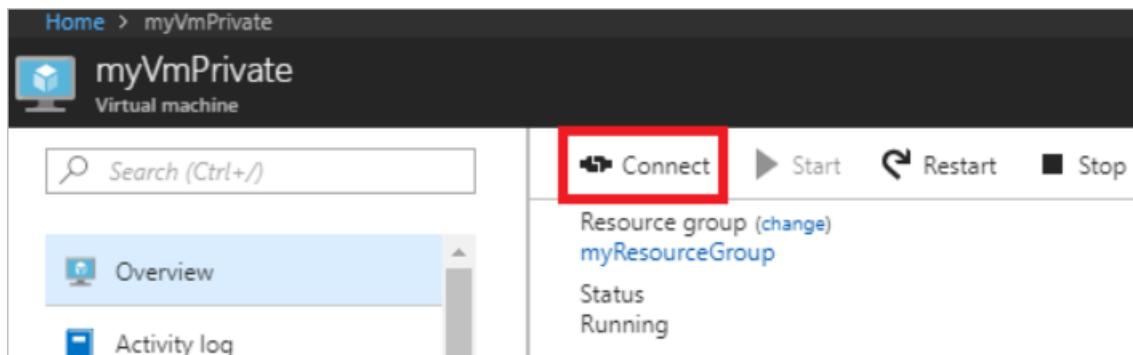
#### Create the second virtual machine

Complete steps 1-7 again, but in step 3, name the virtual machine *myVmPrivate* and in step 5, select the **Private** subnet.

The VM takes a few minutes to deploy. Do not continue to the next step until it finishes creating and its settings open in the portal.

## Confirm access to storage account

1. Once the *myVmPrivate* VM finishes creating, Azure opens the settings for it. Connect to the VM by selecting the **Connect** button, as shown in the following picture:



2. After selecting the **Connect** button, a Remote Desktop Protocol (.rdp) file is created and downloaded to

your computer.

3. Open the downloaded rdp file. If prompted, select **Connect**. Enter the user name and password you specified when creating the VM. You may need to select **More choices**, then **Use a different account**, to specify the credentials you entered when you created the VM.
4. Select **OK**.
5. You may receive a certificate warning during the sign-in process. If you receive the warning, select **Yes** or **Continue**, to proceed with the connection.
6. On the *myVmPrivate* VM, map the Azure file share to drive Z using PowerShell. Before running the commands that follow, replace `<storage-account-key>` and `<storage-account-name>` with values you supplied and retrieved in [Create a storage account](#).

```
$acctKey = ConvertTo-SecureString -String "<storage-account-key>" -AsPlainText -Force  
$credential = New-Object System.Management.Automation.PSCredential -ArgumentList "Azure\<storage-account-name>", $acctKey  
New-PSDrive -Name Z -PSPrinter FileSystem -Root "\\\<storage-account-name>.file.core.windows.net\my-file-share" -Credential $credential
```

PowerShell returns output similar to the following example output:

Name	Used (GB)	Free (GB)	Provider	Root
---	-----	-----	-----	-----
Z			FileSystem	\\\vnt.file.core.windows.net\my-f...

The Azure file share successfully mapped to the Z drive.

7. Confirm that the VM has no outbound connectivity to the internet from a command prompt:

```
ping bing.com
```

You receive no replies, because the network security group associated to the *Private* subnet does not allow outbound access to the internet.

8. Close the remote desktop session to the *myVmPrivate* VM.

## Confirm access is denied to storage account

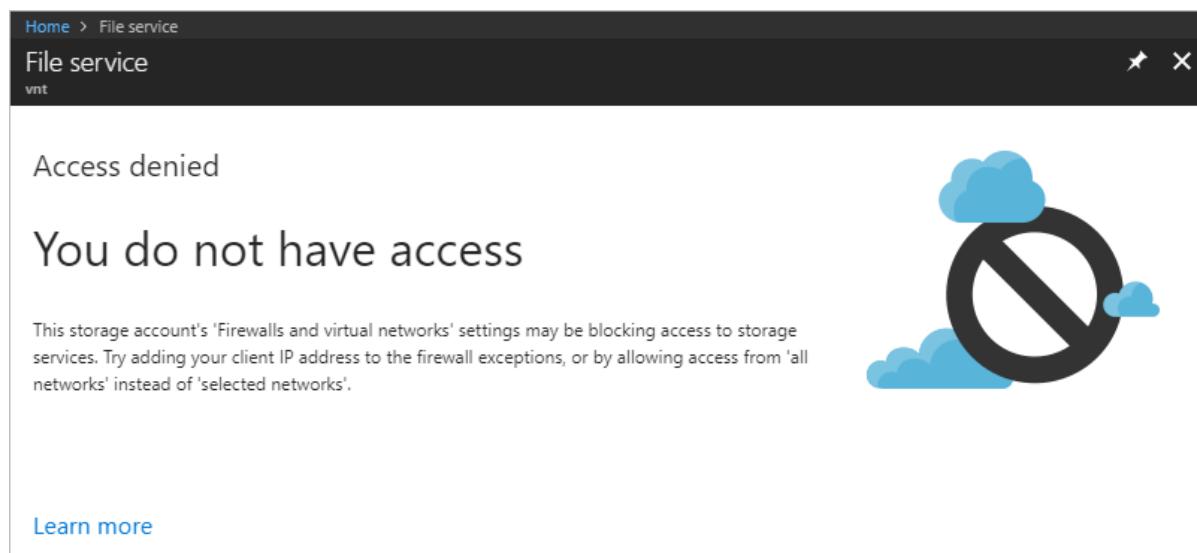
1. Enter *myVmPublic* In the **Search resources, services, and docs** box at the top of the portal.
2. When **myVmPublic** appears in the search results, select it.
3. Complete steps 1-6 in [Confirm access to storage account](#) for the *myVmPublic* VM.

After a short wait, you receive a `New-PSDrive : Access is denied` error. Access is denied because the *myVmPublic* VM is deployed in the *Public* subnet. The *Public* subnet does not have a service endpoint enabled for Azure Storage. The storage account only allows network access from the *Private* subnet, not the *Public* subnet.

4. Close the remote desktop session to the *myVmPublic* VM.
5. From your computer, browse to the Azure [portal](#).
6. Enter the name of the storage account you created in the **Search resources, services, and docs** box. When the name of your storage account appears in the search results, select it.

7. Select **Files**.

8. You receive the error shown in the following picture:



Access is denied, because your computer is not in the *Private* subnet of the *MyVirtualNetwork* virtual network.

## Clean up resources

When no longer needed, delete the resource group and all resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal. When you see **myResourceGroup** in the search results, select it.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

## Next steps

In this tutorial, you enabled a service endpoint for a virtual network subnet. You learned that you can enable service endpoints for resources deployed from multiple Azure services. You created an Azure Storage account and restricted network access to the storage account to only resources within a virtual network subnet. To learn more about service endpoints, see [Service endpoints overview](#) and [Manage subnets](#).

If you have multiple virtual networks in your account, you may want to connect two virtual networks together so the resources within each virtual network can communicate with each other. To learn how to connect virtual networks, advance to the next tutorial.

[Connect virtual networks](#)

# Tutorial: Connect virtual networks with virtual network peering using the Azure portal

2/13/2020 • 5 minutes to read • [Edit Online](#)

You can connect virtual networks to each other with virtual network peering. These virtual networks can be in the same region or different regions (also known as Global VNet peering). Once virtual networks are peered, resources in both virtual networks are able to communicate with each other, with the same latency and bandwidth as if the resources were in the same virtual network. In this tutorial, you learn how to:

- Create two virtual networks
- Connect two virtual networks with a virtual network peering
- Deploy a virtual machine (VM) into each virtual network
- Communicate between VMs

If you prefer, you can complete this tutorial using the [Azure CLI](#) or [Azure PowerShell](#).

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Log in to Azure

Log in to the Azure portal at <https://portal.azure.com>.

## Create virtual networks

1. On the Azure portal, select **Create a resource**.
2. Select **Networking**, and then select **Virtual network**.
3. On the **Basics** tab, enter or select the following information and accept the defaults for the remaining settings:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select <b>Create new</b> and enter <i>myResourceGroup</i> .
Region	Select <b>East US</b> .
Name	myVirtualNetwork1

4. On the **IP Addresses** tab, enter 10.0.0.0/16 for the **Address Space** field. Click the **Add subnet** button below and enter *Subnet1* for **Subnet Name** and 10.0.0.0/24 for **Subnet Address range**.
5. Select **Review + Create** and then select **Create**.
6. Complete steps 1-5 again, with the following changes:

SETTING	VALUE
Name	myVirtualNetwork2

SETTING	VALUE
Address space	10.1.0.0/16
Resource group	Select <b>Use existing</b> and then select <b>myResourceGroup</b> .
Subnet name	Subnet2
Subnet Address range	10.1.0.0/24

## Peer virtual networks

1. In the Search box at the top of the Azure portal, begin typing *MyVirtualNetwork1*. When **myVirtualNetwork1** appears in the search results, select it.
2. Select **Peerings**, under **Settings**, and then select **Add**, as shown in the following picture:

3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **OK**.

SETTING	VALUE
Name of the peering from myVirtualNetwork1 to remote virtual network	myVirtualNetwork1-myVirtualNetwork2 - When the page first loads, you'll see the phrase "remote virtual network" here. After you choose the remote virtual network, the phrase "remote virtual network" will be replaced with the name of the remote virtual network.
Subscription	Select your subscription.
Virtual network	myVirtualNetwork2 - To select the <i>myVirtualNetwork2</i> virtual network, select <b>Virtual network</b> , then select <b>myVirtualNetwork2 (myResourceGroup)</b> . You can select a virtual network in the same region or in a different region.
Name of the peering from myVirtualNetwork2 to myVirtualNetwork1	myVirtualNetwork2-myVirtualNetwork1

Home > myVirtualNetwork1 - Peerings > Add peering

### Add peering

myVirtualNetwork1

**For peering to work, a peering link must be created from myVirtualNetwork1 to myVirtualNetwork2 as well as from myVirtualNetwork2 to myVirtualNetwork1.**

\* Name of the peering from myVirtualNetwork1 to myVirtualNetwork2  
myVirtualNetwork1-myVirtualNetwork2

Peer details

Virtual network deployment model [?](#)  
 Resource manager  Classic

I know my resource ID [?](#)

\* Subscription [?](#)  
 mySubscription (myResourceGroup)

\* Virtual network  
myVirtualNetwork2 (myResourceGroup)

\* Name of the peering from myVirtualNetwork2 to myVirtualNetwork1  
myVirtualNetwork2-myVirtualNetwork1

**OK**

The **PEERING STATUS** is *Connected*, as shown in the following picture:

Home > myVirtualNetwork1 - Peerings

### myVirtualNetwork1 - Peerings

Virtual network

Search (Ctrl+/  
)

[Add](#) [Refresh](#)

NAME	PEERING STATUS	PEER	GATEWAY TRANSIT
myVirtualNetwork1-myVi... myVirtualNetwork2	Connected	myVirtualNetwork2	Disabled

If you don't see the status, refresh your browser.

## Create virtual machines

Create a VM in each virtual network so that you can communicate between them in a later step.

### Create the first VM

1. On the Azure portal, select **Create a resource**.
2. Select **Compute**, and then select **Windows Server 2016 Datacenter**. You can select a different operating system, but the remaining steps assume you selected **Windows Server 2016 Datacenter**.
3. Enter, or select, the following information for **Basics**, accept the defaults for the remaining settings, and then select **Create**:

SETTING	VALUE
Resource group	Select <b>Use existing</b> and then select <b>myResourceGroup</b> .
Name	myVm1
Location	Select <b>East US</b> .
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the <a href="#">defined complexity requirements</a> .

- Select a VM size for the **Size** option.
- Select the following values for under **Networking**:

SETTING	VALUE
Virtual network	myVirtualNetwork1 - If it's not already selected, select <b>Virtual network</b> and then select <b>myVirtualNetwork1</b> .
Subnet	Subnet1 - If it's not already selected, select <b>Subnet</b> and then select <b>Subnet1</b> .

- Select **Networking**. Choose **Allow selected ports** for the **Public inbound ports** option. Choose **RDP** for the **Select inbound ports** option below this.

- Select the **Review + Create** button in the lower, left-hand corner to start the VM deployment.

### Create the second VM

Complete steps 1-6 again, with the following changes:

SETTING	VALUE
Name	myVm2
Virtual network	myVirtualNetwork2

The VMs take a few minutes to create. Do not continue with the remaining steps until both VMs are created.

## Communicate between VMs

- In the *Search* box at the top of the portal, begin typing *myVm1*. When **myVm1** appears in the search results, select it.
- Create a remote desktop connection to the *myVm1* VM by selecting **Connect**, as shown in the following picture:

The screenshot shows the Azure portal interface for a virtual machine named 'myVm1'. The top navigation bar includes 'Home > myResourceGroup > myVm1'. Below the navigation bar is a toolbar with buttons for 'Start', 'Restart', 'Stop', 'Capture', 'Move', 'Delete', and 'Refresh'. A 'Connect' button is highlighted with a red box. On the left, there is a sidebar with links for 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', and 'Diagnose and solve problems'. The main content area displays the VM's properties: Resource group (myResourceGroup), Status (Running), Location (East US), Subscription (change), Computer name (myVm1), Operating system (Windows), Size (Standard DS1 v2 (1 vcpu, 3.5 GB memory)), Public IP address (40.121.220.97), Virtual network/subnet (myVirtualNetwork1/Subnet1), and DNS name (myvm1-efb1f0.eastus.cloudapp.azure.com).

- To connect to the VM, open the downloaded RDP file. If prompted, select **Connect**.
- Enter the user name and password you specified when creating the VM (you may need to select **More choices**, then **Use a different account**, to specify the credentials you entered when you created the VM), then select **OK**.
- You may receive a certificate warning during the sign-in process. Select **Yes** to proceed with the connection.
- In a later step, ping is used to communicate with the *myVm2* VM from the *myVm1* VM. Ping uses the

Internet Control Message Protocol (ICMP), which is denied through the Windows Firewall, by default. On the *myVm1* VM, enable ICMP through the Windows firewall, so that you can ping this VM from *myVm2* in a later step, using PowerShell:

```
New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4
```

Though ping is used to communicate between VMs in this tutorial, allowing ICMP through the Windows Firewall for production deployments is not recommended.

7. To connect to the *myVm2* VM, enter the following command from a command prompt on the *myVm1* VM:

```
mstsc /v:10.1.0.4
```

8. Since you enabled ping on *myVm1*, you can now ping it by IP address:

```
ping 10.0.0.4
```

9. Disconnect your RDP sessions to both *myVm1* and *myVm2*.

## Clean up resources

When no longer needed, delete the resource group and all resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal. When you see **myResourceGroup** in the search results, select it.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

## Next steps

In this tutorial, you learned how to connect two networks in the same Azure region, with virtual network peering. You can also peer virtual networks in different [supported regions](#) and in [different Azure subscriptions](#), as well as create [hub and spoke network designs](#) with peering. To learn more about virtual network peering, see [Virtual network peering overview](#) and [Manage virtual network peerings](#).

To connect your own computer to a virtual network through a VPN, and interact with resources in a virtual network, or in peered virtual networks, see [Connect your computer to a virtual network](#).

# Azure CLI samples for virtual network

11/14/2019 • 2 minutes to read • [Edit Online](#)

The following table includes links to bash scripts with Azure CLI commands:

<a href="#">Create a virtual network for multi-tier applications</a>	Creates a virtual network with front-end and back-end subnets. Traffic to the front-end subnet is limited to HTTP and SSH, while traffic to the back-end subnet is limited to MySQL, port 3306.
<a href="#">Peer two virtual networks</a>	Creates and connects two virtual networks in the same region.
<a href="#">Route traffic through a network virtual appliance</a>	Creates a virtual network with front-end and back-end subnets and a VM that is able to route traffic between the two subnets.
<a href="#">Filter inbound and outbound VM network traffic</a>	Creates a virtual network with front-end and back-end subnets. Inbound network traffic to the front-end subnet is limited to HTTP, HTTPS, and SSH. Outbound traffic to the internet from the back-end subnet is not permitted.
<a href="#">Configure IPv4 + IPv6 dual stack virtual network with Basic Load Balancer</a>	Deploys dual-stack (IPv4+IPv6) virtual network with two VMs and an Azure Basic Load Balancer with IPv4 and IPv6 public IP addresses.
<a href="#">Configure IPv4 + IPv6 dual stack virtual network with Standard Load Balancer</a>	Deploys dual-stack (IPv4+IPv6) virtual network with two VMs and an Azure Standard Load Balancer with IPv4 and IPv6 public IP addresses.

# Azure PowerShell samples for virtual network

11/14/2019 • 2 minutes to read • [Edit Online](#)

The following table includes links to Azure Powershell scripts:

<a href="#">Create a virtual network for multi-tier applications</a>	Creates a virtual network with front-end and back-end subnets. Traffic to the front-end subnet is limited to HTTP, while traffic to the back-end subnet is limited to SQL, port 1433.
<a href="#">Peer two virtual networks</a>	Creates and connects two virtual networks in the same region.
<a href="#">Route traffic through a network virtual appliance</a>	Creates a virtual network with front-end and back-end subnets and a VM that is able to route traffic between the two subnets.
<a href="#">Filter inbound and outbound VM network traffic</a>	Creates a virtual network with front-end and back-end subnets. Inbound network traffic to the front-end subnet is limited to HTTP and HTTPS. Outbound traffic to the internet from the back-end subnet is not permitted.
<a href="#">Configure IPv4 + IPv6 dual stack virtual network with Basic Load Balancer</a>	Deploys dual-stack (IPv4+IPv6) virtual network with two VMs and an Azure Basic Load Balancer with IPv4 and IPv6 public IP addresses.
<a href="#">Configure IPv4 + IPv6 dual stack virtual network with Standard Load Balancer</a>	Deploys dual-stack (IPv4+IPv6) virtual network with two VMs and an Azure Standard Load Balancer with IPv4 and IPv6 public IP addresses.

# Azure Resource Manager template samples for virtual network

1/14/2020 • 2 minutes to read • [Edit Online](#)

The following table includes links to Azure Resource Manager template samples. You can deploy templates using the Azure [portal](#), Azure [CLI](#), or Azure [PowerShell](#). To learn how to author your own templates, see [Create your first template](#) and [Understand the structure and syntax of Azure Resource Manager templates](#).

For the JSON syntax and properties to use in templates, see [Microsoft.Network resource types](#).

TASK	DESCRIPTION
<a href="#">Create a virtual network with two subnets</a>	Creates a virtual network with two subnets.
<a href="#">Route traffic through a network virtual appliance</a>	Creates a virtual network with three subnets. Deploys a virtual machine into each of the subnets. Creates a route table containing routes to direct traffic from one subnet to another through the virtual machine in the third subnet. Associates the route table to one of the subnets.
<a href="#">Create a virtual network service endpoint for Azure Storage</a>	Creates a new virtual network with two subnets, and a network interface in each subnet. Enables a service endpoint to Azure Storage for one of the subnets and secures a new storage account to that subnet.
<a href="#">Connect two virtual networks</a>	Creates two virtual networks and a virtual network peering between them.
<a href="#">Create a virtual machine with multiple IP addresses</a>	Creates a Windows or Linux VM with multiple IP addresses.
<a href="#">Configure IPv4 + IPv6 dual stack virtual network</a>	Deploys dual-stack (IPv4+IPv6) virtual network with two VMs and an Azure Basic Load Balancer with IPv4 and IPv6 public IP addresses.

# Azure policy sample built-ins for virtual network

2/12/2020 • 7 minutes to read • [Edit Online](#)

The following table includes links to [Azure Policy](#) samples. The samples are found in the [Azure Policy samples repository](#).

## Network

NAME	DESCRIPTION	EFFECT(S)	VERSION	SOURCE
<a href="#">A custom IPsec/IKE policy must be applied to all Azure virtual network gateway connections</a>	This policy ensures that all Azure virtual network gateway connections use a custom Internet Protocol Security(IPsec)/Internet Key Exchange(IKE) policy. Supported algorithms and key strengths - <a href="https://aka.ms/AA62kb0">https://aka.ms/AA62kb0</a>	Audit, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">App Service should use a virtual network service endpoint</a>	This policy audits any App Service not configured to use a virtual network service endpoint.	AuditIfNotExists, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Azure VPN gateways should not use 'basic' SKU</a>	This policy ensures that VPN gateways do not use 'basic' SKU.	Audit, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Container Registry should use a virtual network service endpoint</a>	This policy audits any Container Registry not configured to use a virtual network service endpoint.	Audit, Disabled	1.0.0-preview	<a href="#">GitHub</a>
<a href="#">Cosmos DB should use a virtual network service endpoint</a>	This policy audits any Cosmos DB not configured to use a virtual network service endpoint.	Audit, Disabled	1.0.0	<a href="#">GitHub</a>

<b>NAME</b>	<b>DESCRIPTION</b>	<b>EFFECT(S)</b>	<b>VERSION</b>	<b>SOURCE</b>
<a href="#">Deploy network watcher when virtual networks are created</a>	This policy creates a network watcher resource in regions with virtual networks. You need to ensure existence of a resource group named networkWatcherRG, which will be used to deploy network watcher instances.	DeployIfNotExists	1.0.0	<a href="#">GitHub</a>
<a href="#">Event Hub should use a virtual network service endpoint</a>	This policy audits any Event Hub not configured to use a virtual network service endpoint.	AuditIfNotExists, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Gateway subnets should not be configured with a network security group</a>	This policy denies if a gateway subnet is configured with a network security group. Assigning a network security group to a gateway subnet will cause the gateway to stop functioning.	deny	1.0.0	<a href="#">GitHub</a>
<a href="#">Key Vault should use a virtual network service endpoint</a>	This policy audits any Key Vault not configured to use a virtual network service endpoint.	Audit, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Network interfaces should disable IP forwarding</a>	This policy denies the network interfaces which enabled IP forwarding. The setting of IP forwarding disables Azure's check of the source and destination for a network interface. This should be reviewed by the network security team.	deny	1.0.0	<a href="#">GitHub</a>

NAME	DESCRIPTION	EFFECT(S)	VERSION	SOURCE
<a href="#">Network interfaces should not have public IPs</a>	This policy denies the network interfaces which are configured with any public IP. Public IP addresses allow internet resources to communicate inbound to Azure resources, and Azure resources to communicate outbound to the internet. This should be reviewed by the network security team.	deny	1.0.0	<a href="#">GitHub</a>
<a href="#">Network Watcher should be enabled</a>	Network Watcher is a regional service that enables you to monitor and diagnose conditions at a network scenario level in, to, and from Azure. Scenario level monitoring enables you to diagnose problems at an end to end network level view. Network diagnostic and visualization tools available with Network Watcher help you understand, diagnose, and gain insights to your network in Azure.	auditIfNotExists	1.0.0	<a href="#">GitHub</a>
<a href="#">RDP access from the Internet should be blocked</a>	This policy audits any network security rule that allows RDP access from Internet	Audit, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Service Bus should use a virtual network service endpoint</a>	This policy audits any Service Bus not configured to use a virtual network service endpoint.	AuditIfNotExists, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">SQL Server should use a virtual network service endpoint</a>	This policy audits any SQL Server not configured to use a virtual network service endpoint.	AuditIfNotExists, Disabled	1.0.0	<a href="#">GitHub</a>

NAME	DESCRIPTION	EFFECT(S)	VERSION	SOURCE
<a href="#">SSH access from the Internet should be blocked</a>	This policy audits any network security rule that allows SSH access from Internet	Audit, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Storage Accounts should use a virtual network service endpoint</a>	This policy audits any Storage Account not configured to use a virtual network service endpoint.	Audit, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Virtual machines should be connected to an approved virtual network</a>	This policy audits any virtual machine connected to a virtual network that is not approved.	Audit, Deny, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Virtual networks should use specified virtual network gateway</a>	This policy audits any virtual network if the default route does not point to the specified virtual network gateway.	AuditIfExists, Disabled	1.0.0	<a href="#">GitHub</a>

## Tags

NAME	DESCRIPTION	EFFECT(S)	VERSION	SOURCE
<a href="#">Add a tag to resource groups</a>	Adds the specified tag and value when any resource group missing this tag is created or updated. Existing resource groups can be remediated by triggering a remediation task. If the tag exists with a different value it will not be changed.	modify	1.0.0	<a href="#">GitHub</a>
<a href="#">Add a tag to resources</a>	Adds the specified tag and value when any resource missing this tag is created or updated. Existing resources can be remediated by triggering a remediation task. If the tag exists with a different value it will not be changed. Does not modify tags on resource groups.	modify	1.0.0	<a href="#">GitHub</a>

NAME	DESCRIPTION	EFFECT(S)	VERSION	SOURCE
<a href="#">Add or replace a tag on resource groups</a>	<p>Adds or replaces the specified tag and value when any resource group is created or updated. Existing resource groups can be remediated by triggering a remediation task.</p>	modify	1.0.0	<a href="#">GitHub</a>
<a href="#">Add or replace a tag on resources</a>	<p>Adds or replaces the specified tag and value when any resource is created or updated. Existing resources can be remediated by triggering a remediation task. Does not modify tags on resource groups.</p>	modify	1.0.0	<a href="#">GitHub</a>
<a href="#">Append tag and its default value</a>	<p>Appends the specified tag and value when any resource which is missing this tag is created or updated. Does not modify the tags of resources created before this policy was applied until those resources are changed. Does not apply to resource groups. New 'modify' effect policies are available that support remediation of tags on existing resources (see <a href="https://aka.ms/modify_doc">https://aka.ms/modify doc</a>).</p>	append	1.0.0	<a href="#">GitHub</a>

NAME	DESCRIPTION	EFFECT(S)	VERSION	SOURCE
<a href="#">Append tag and its default value to resource groups</a>	<p>Appends the specified tag and value when any resource group which is missing this tag is created or updated. Does not modify the tags of resource groups created before this policy was applied until those resource groups are changed. New 'modify' effect policies are available that support remediation of tags on existing resources (see <a href="https://aka.ms/modifydoc">https://aka.ms/modifydoc</a>).</p>	append	1.0.0	<a href="#">GitHub</a>
<a href="#">Append tag and its value from the resource group</a>	<p>Appends the specified tag with its value from the resource group when any resource which is missing this tag is created or updated. Does not modify the tags of resources created before this policy was applied until those resources are changed. New 'modify' effect policies are available that support remediation of tags on existing resources (see <a href="https://aka.ms/modifydoc">https://aka.ms/modifydoc</a>).</p>	append	1.0.0	<a href="#">GitHub</a>
<a href="#">Inherit a tag from the resource group</a>	<p>Adds or replaces the specified tag and value from the parent resource group when any resource is created or updated. Existing resources can be remediated by triggering a remediation task.</p>	modify	1.0.0	<a href="#">GitHub</a>

NAME	DESCRIPTION	EFFECT(S)	VERSION	SOURCE
<a href="#">Inherit a tag from the resource group if missing</a>	Adds the specified tag with its value from the parent resource group when any resource missing this tag is created or updated. Existing resources can be remediated by triggering a remediation task. If the tag exists with a different value it will not be changed.	modify	1.0.0	<a href="#">GitHub</a>
<a href="#">Inherit a tag from the subscription</a>	Adds or replaces the specified tag and value from the containing subscription when any resource is created or updated. Existing resources can be remediated by triggering a remediation task.	modify	1.0.0	<a href="#">GitHub</a>
<a href="#">Inherit a tag from the subscription if missing</a>	Adds the specified tag with its value from the containing subscription when any resource missing this tag is created or updated. Existing resources can be remediated by triggering a remediation task. If the tag exists with a different value it will not be changed.	modify	1.0.0	<a href="#">GitHub</a>
<a href="#">Require specified tag</a>	Enforces existence of a tag. Does not apply to resource groups.	deny	1.0.0	<a href="#">GitHub</a>
<a href="#">Require specified tag on resource groups</a>	Enforces existence of a tag on resource groups.	deny	1.0.0	<a href="#">GitHub</a>
<a href="#">Require tag and its value</a>	Enforces a required tag and its value. Does not apply to resource groups.	deny	1.0.0	<a href="#">GitHub</a>
<a href="#">Require tag and its value on resource groups</a>	Enforces a required tag and its value on resource groups.	deny	1.0.0	<a href="#">GitHub</a>

## General

Name	Description	Effect(s)	Version	Source
Allowed locations	This policy enables you to restrict the locations your organization can specify when deploying resources. Use to enforce your geo-compliance requirements. Excludes resource groups, Microsoft.AzureActive Directory/b2cDirectories, and resources that use the 'global' region.	deny	1.0.0	<a href="#">GitHub</a>
Allowed locations for resource groups	This policy enables you to restrict the locations your organization can create resource groups in. Use to enforce your geo-compliance requirements.	deny	1.0.0	<a href="#">GitHub</a>
Allowed resource types	This policy enables you to specify the resource types that your organization can deploy. Only resource types that support 'tags' and 'location' will be affected by this policy. To restrict all resources please duplicate this policy and change the 'mode' to 'All'.	deny	1.0.0	<a href="#">GitHub</a>
Audit resource location matches resource group location	Audit that the resource location matches its resource group location	audit	1.0.0	<a href="#">GitHub</a>

NAME	DESCRIPTION	EFFECT(S)	VERSION	SOURCE
<a href="#">Audit usage of custom RBAC rules</a>	Audit built-in roles such as 'Owner', 'Contributer', 'Reader' instead of custom RBAC roles, which are error prone. Using custom roles is treated as an exception and requires a rigorous review and threat modeling	Audit, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Custom subscription owner roles should not exist</a>	This policy ensures that no custom subscription owner roles exist.	Audit, Disabled	1.0.0	<a href="#">GitHub</a>
<a href="#">Not allowed resource types</a>	This policy enables you to specify the resource types that your organization cannot deploy.	Deny	1.0.0	<a href="#">GitHub</a>

# Virtual Network – Business Continuity

7/14/2019 • 2 minutes to read • [Edit Online](#)

## Overview

A Virtual Network (VNet) is a logical representation of your network in the cloud. It allows you to define your own private IP address space and segment the network into subnets. VNets serve as a trust boundary to host your compute resources such as Azure Virtual Machines and Cloud Services (web/worker roles). A VNet allows direct private IP communication between the resources hosted in it. You can link a virtual network to an on-premises network through a VPN Gateway, or ExpressRoute.

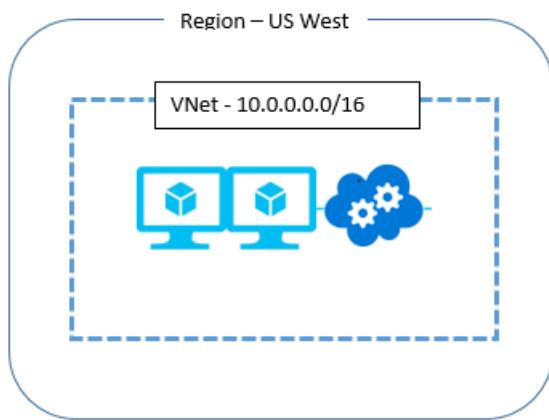
A VNet is created within the scope of a region. You can *create* VNets with same address space in two different regions (For example, US East and US West), but because they have the same address space, you can't connect them together.

## Business Continuity

There could be several different ways that your application could be disrupted. A region could be completely cut off due to a natural disaster, or a partial disaster, due to a failure of multiple devices or services. The impact on the VNet service is different in each of these situations.

**Q: If an outage occurs for an entire region, what do I do? For example, if a region is completely cut off due to a natural disaster? What happens to the virtual networks hosted in the region?**

A: The virtual network and the resources in the affected region remains inaccessible during the time of the service disruption.



**Q: What can I do to re-create the same virtual network in a different region?**

A: Virtual networks are fairly lightweight resources. You can invoke Azure APIs to create a VNet with the same address space in a different region. To recreate the same environment that was present in the affected region, you make API calls to redeploy the Cloud Services web and worker roles, and the virtual machines that you had. If you have on-premises connectivity, such as in a hybrid deployment, you have to deploy a new VPN Gateway, and connect to your on-premises network.

To create a virtual network, see [Create a virtual network](#).

**Q: Can a replica of a VNet in a given region be re-created in another region ahead of time?**

A: Yes, you can create two VNets using the same private IP address space and resources in two different regions ahead of time. If you are hosting internet-facing services in the VNet, you could have set up Traffic Manager to

geo-route traffic to the region that is active. However, you cannot connect two VNets with the same address space to your on-premises network, as it would cause routing issues. At the time of a disaster and loss of a VNet in one region, you can connect the other VNet in the available region, with the matching address space to your on-premises network.

To create a virtual network, see [Create a virtual network](#).

# What is Virtual Network NAT (Public Preview)?

2/27/2020 • 5 minutes to read • [Edit Online](#)

Virtual Network NAT (network address translation) simplifies outbound-only Internet connectivity for virtual networks. When configured on a subnet, all outbound connectivity uses your specified static public IP addresses. Outbound connectivity is possible without load balancer or public IP addresses directly attached to virtual machines. NAT is fully managed and highly resilient.

*Figure: Virtual Network NAT*

## NOTE

Virtual Network NAT is available as public preview at this time. Currently it's only available in a limited set of [regions](#). This preview is provided without a service level agreement and isn't recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Static IP addresses for outbound-only

Outbound connectivity can be defined for each subnet with NAT. Multiple subnets within the same virtual network can have different NATs. A subnet is configured by specifying which [NAT gateway resource](#) to use. All UDP and TCP outbound flows from any virtual machine instance will use NAT.

NAT is compatible with standard SKU [public IP address resources](#) or [public IP prefix resources](#) or a combination of both. You can use a public IP prefix directly or distribute the public IP addresses of the prefix across multiple NAT gateway resources. NAT will groom all traffic to the range of IP addresses of the prefix. Any IP whitelisting of your deployments is now easy.

All outbound traffic for the subnet is processed by NAT automatically without any customer configuration. User-defined routes aren't necessary. NAT takes precedence over other [outbound scenarios](#) and replaces the default Internet destination of a subnet.

## On-demand SNAT with multiple IP addresses for scale

NAT uses "port network address translation" (PNAT or PAT) and is recommended for most workloads. Dynamic or divergent workloads can be easily accommodated with on-demand outbound flow allocation. Extensive pre-planning, pre-allocation, and ultimately overprovisioning of outbound resources is avoided. SNAT port resources are shared and available across all subnets using a specific NAT gateway resource and are provided when needed.

A public IP address attached to NAT provides up to 64,000 concurrent flows for UDP and TCP. You can start with a single IP address and scale up to 16 public IP addresses.

NAT allows flows to be created from the virtual network to the Internet. Return traffic from the Internet is only allowed in response to an active flow.

Unlike load balancer outbound SNAT, NAT has no restrictions on which private IP of a virtual machine instance can make outbound connections. Secondary IP configurations can create outbound Internet connection with NAT.

## Coexistence of inbound and outbound

NAT is compatible with the following standard SKU resources:

- [Load balancer](#)
- [Public IP address](#)
- [Public IP prefix](#)

When used together with NAT, these resources provide inbound Internet connectivity to your subnet(s). NAT provides all outbound Internet connectivity from your subnet(s).

NAT and compatible Standard SKU features are aware of the direction the flow was started. Inbound and outbound scenarios can coexist. These scenarios will receive the correct network address translations because these features are aware of the flow direction.

*Figure: Virtual Network NAT flow direction*

## Fully managed, highly resilient

NAT is fully scaled out from the start. There's no ramp up or scale-out operation required. Azure manages the operation of NAT for you. NAT always has multiple fault domains and can sustain multiple failures without service outage.

## TCP Reset for unrecognized flows

The private side of NAT sends TCP Reset packets for attempts to communicate on a TCP connection that doesn't exist. One example is connections that have reached idle timeout. The next packet received will return a TCP Reset to the private IP address to signal and force connection closure.

The public side of NAT doesn't generate TCP Reset packets or any other traffic. Only traffic produced by the customer's virtual network is emitted.

## Configurable idle timeout

A default idle timeout of 4 minutes is used and can be increased to up to 120 minutes. Any activity on a flow can also reset the idle timer, including TCP keepalives.

## Regional or zone isolation with availability zones

NAT is regional by default. When creating [availability zones](#) scenarios, NAT can be isolated in a specific zone (zonal deployment).

*Figure: Virtual Network NAT with availability zones*

## Multi-dimensional metrics for observability

You can monitor the operation of your NAT through multi-dimensional metrics exposed in Azure Monitor. These metrics can be used to observe the usage and for troubleshooting. NAT gateway resources expose the following metrics:

- Bytes
- Packets
- Dropped Packets
- Total SNAT connections

- SNAT connection state transitions per interval.

## SLA

At general availability, NAT data path is at least 99.9% available.

## Region availability

NAT is currently available in these regions:

- Europe West
- Japan East
- US East 2
- US West
- US West 2
- US West Central

## Public Preview participation

Subscriptions must be registered to allow participation in the Public Preview. Participation requires a two-step process and instructions are provided below for Azure CLI and Azure PowerShell. The activation may take several minutes to complete.

### Azure CLI

1. register subscription for Public Preview

```
az feature register --namespace Microsoft.Network --name AllowNatGateway
```

2. activate registration

```
az provider register --namespace Microsoft.Network
```

### Azure PowerShell

1. register subscription for Public Preview

```
Register-AzProviderFeature -ProviderNamespace Microsoft.Network -FeatureName AllowNatGateway
```

2. activate registration

```
Register-AzResourceProvider -ProviderNamespace Microsoft.Network
```

## Pricing

NAT gateway is billed with two separate meters:

METER	RATE
Resource hours	\$0.045/hour
Data processed	\$0.045/GB

Resource hours accounts for the duration during which a NAT gateway resource exists. Data processed accounts for all traffic processed by a NAT gateway resource.

During public preview, pricing is discounted at 50%.

## Support

NAT is supported through normal support channels.

## Feedback

We want to know how we can improve the service. Share your [feedback on the Public Preview](#) with us. And you can propose and vote on what we should build next at [UserVoice for NAT](#).

## Limitations

- NAT is compatible with standard SKU public IP, public IP prefix, and load balancer resources. Basic resources (for example basic load balancer) and any products derived from them aren't compatible with NAT. Basic resources must be placed on a subnet not configured with NAT.
- IPv4 address family is supported. NAT doesn't interact with IPv6 address family.
- NSG flow logging isn't supported when using NAT.
- NAT can't span multiple virtual networks.

## Next steps

- Learn about [NAT gateway resource](#).
- [Tell us what to build next in UserVoice](#).
- [Provide feedback on the Public Preview](#).

# Designing virtual networks with NAT gateway resources (Public Preview)

2/27/2020 • 12 minutes to read • [Edit Online](#)

NAT gateway resources are part of [Virtual Network NAT](#) and provide outbound Internet connectivity for one or more subnets of a virtual network. The subnet of the virtual network states which NAT gateway will be used. NAT provides source network address translation (SNAT) for a subnet. NAT gateway resources specify which static IP addresses virtual machines use when creating outbound flows. Static IP addresses come from public IP address resources, public IP prefix resources, or both. A NAT gateway resource can use up to 16 static IP addresses from either.

*Figure: Virtual Network NAT for outbound to Internet*

## NOTE

Virtual Network NAT is available as public preview at this time. Currently it's only available in a limited set of [regions](#). This preview is provided without a service level agreement and isn't recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## How to deploy NAT

Configuring and using NAT gateway is intentionally made simple:

NAT gateway resource:

- Create regional or zonal (zone-isolated) NAT gateway resource,
- Assign IP addresses,
- Modify idle timeout (optional).

Virtual network:

- Configure virtual network subnet to use a NAT gateway.

User-defined routes aren't necessary.

## Resource

The resource is designed to be simple as you can see from the following Azure Resource Manager example in a template-like format. This template-like format is shown here to illustrate the concepts and structure. Modify the example for your needs. This document isn't intended as a tutorial.

The following diagram shows the writeable references between the different Azure Resource Manager resources. The arrow indicates the direction of the reference, originating from where it's writeable. Review

*Figure: Virtual Network NAT object model*

NAT is recommended for most workloads unless you have a specific dependency on [pool-based Load Balancer](#)

## outbound connectivity.

You can migrate from standard load balancer scenarios, including [outbound rules](#), to NAT gateway. To migrate, move the public ip and public ip prefix resources from load balancer frontends to NAT gateway. New IP addresses for NAT gateway aren't required. Standard public IP and prefix can be reused as long as the total doesn't exceed 16 IP addresses. Plan for migration with service interruption in mind during the transition. You can minimize the interruption by automating the process. Test the migration in a staging environment first. During the transition, inbound originated flows aren't affected.

The following example would create a NAT gateway resource called *myNATGateway* is created in region *East US 2, AZ 1* with a *4-minutes* idle timeout. The outbound IP addresses provided are:

- A set of public IP address resources *myIP1* and *myIP2* and
- A set of public IP prefix resources *myPrefix1* and *myPrefix2*.

The total number of IP addresses provided by all four IP address resources can't exceed 16 IP addresses total. Any number of IP addresses between 1 and 16 is allowed.

```
{
  "name": "myNATGateway",
  "type": "Microsoft.Network/natGateways",
  "apiVersion": "2018-11-01",
  "location": "East US 2",
  "sku": { "name": "Standard" },
  "zones": [ "1" ],
  "properties": {
    "idleTimeoutInMinutes": 4,
    "publicIPPrefixes": [
      {
        "id": "ref to myPrefix1"
      },
      {
        "id": "ref to myPrefix2"
      }
    ],
    "publicIPAddresses": [
      {
        "id": "ref to myIP1"
      },
      {
        "id": "ref to myIP2"
      }
    ]
  }
}
```

When the NAT gateway resource has been created, it can be used on one or more subnets of a virtual network. Specify which subnets use this NAT gateway resource. A NAT gateway isn't able to span more than one virtual network. It isn't required to assign the same NAT gateway to all subnets of a virtual network. Individual subnets can be configured with different NAT gateway resources.

Scenarios that don't use availability zones will be regional (no zone specified). If you're using availability zones, you can specify a zone to isolate NAT to a specific zone. Zone-redundancy isn't supported. Review NAT [availability zones](#).

```
{
  "name": "myVNet",
  "apiVersion": "2018-11-01",
  "type": "Microsoft.Network/virtualNetworks",
  "location": "myRegion",
  "properties": {
    "addressSpace": {
      "addressPrefixes": [
        "192.168.0.0/16"
      ]
    },
    "subnets": [
      {
        "name": "mySubnet1",
        "properties": {
          "addressPrefix": "192.168.0.0/24",
          "natGateway": {
            "id": "ref to myNATGateway"
          }
        }
      }
    ]
  }
}
```

NAT gateways are defined with a property on a subnet within a virtual network. Flows created by virtual machines on subnet *mySubnet1* of virtual network *myVNet* will use the NAT gateway. All outbound connectivity will use the IP addresses associated with *myNatGateway* as the source IP address.

## Design Guidance

Review this section to familiarize yourself with considerations for designing virtual networks with NAT.

1. [Cost optimization](#)
2. [Coexistence of inbound and outbound](#)
3. [Managing Basic resources](#)
4. [Availability Zones](#)

### **Cost optimization**

[Service endpoints](#) and [private link](#) are two options to consider for optimizing cost where NAT isn't needed. Any traffic directed to service endpoints or private link is not processed by the virtual network's NAT.

Service endpoints tie Azure service resources to your virtual network and control access to your Azure service resources. For example, when you access Azure storage, use a service endpoint for storage to avoid data processed NAT charges. Service endpoints are free.

Private link exposes Azure PaaS service (or other services hosted with private link) as a private endpoint inside a virtual network. Private link is billed based on duration and data processed.

Evaluate if either or both of these approaches are a good fit for your scenario and use as needed.

### **Coexistence of inbound and outbound**

NAT gateway is compatible with:

- Standard load balancer
- Standard public IP
- Standard public IP prefix

When developing a new deployment, start with standard SKUs.

*Figure: Virtual Network NAT for outbound to Internet*

The Internet outbound only scenario provided by NAT gateway can be expanded with inbound from Internet functionality. Each resource is aware of the direction in which a flow is originated. On a subnet with a NAT gateway, all outbound to Internet scenarios are superseded by the NAT gateway. Inbound from Internet scenarios are provided by the respective resource.

#### **NAT and VM with instance-level Public IP**

*Figure: Virtual Network NAT and VM with instance-level Public IP*

DIRECTION	RESOURCE
Inbound	VM with instance-level Public IP
Outbound	NAT gateway

VM will use NAT gateway for outbound. Inbound originated isn't affected.

#### **NAT and VM with public Load Balancer**

*Figure: Virtual Network NAT and VM with public Load Balancer*

DIRECTION	RESOURCE
Inbound	public Load Balancer
Outbound	NAT gateway

Any outbound configuration from a load-balancing rule or outbound rules is superseded by NAT gateway. Inbound originated isn't affected.

#### **NAT and VM with instance-level public IP and public Load Balancer**

*Figure: Virtual Network NAT and VM with instance-level public IP and public Load Balancer*

DIRECTION	RESOURCE
Inbound	VM with instance-level public IP and public Load Balancer
Outbound	NAT gateway

Any outbound configuration from a load-balancing rule or outbound rules is superseded by NAT gateway. The VM will also use NAT gateway for outbound. Inbound originated isn't affected.

#### **Managing Basic resources**

Standard load balancer, public IP, and public IP prefix are compatible with NAT gateway. NAT gateways operate in the scope of a subnet. The basic SKU of these services must be deployed on a subnet without a NAT gateway. This separation allows both SKU variants to coexist in the same virtual network.

NAT gateways take precedence over outbound scenarios of the subnet. Basic load balancer or public IP (and any managed service built with them) is unable to be adjusted with the correct translations. NAT gateway takes

control over outbound to Internet traffic on a subnet. Inbound traffic to basic load balancer and public ip is unavailable. Inbound traffic to a basic load balancer and, or a public ip configured on a VM won't be available.

## Availability Zones

Even without availability zones, NAT is resilient and can survive multiple infrastructure component failures. When availability zones are part of your scenario, you should configure NAT for a specific zone. The control plane operations and data plane are constrained to the specified zone. Failure in a zone other than where your scenario exists is expected to be without impact to NAT. Outbound traffic from virtual machines in the same zone will fail because of zone isolation.

*Figure: Virtual Network NAT with availability zones*

A zone-isolated NAT gateway requires IP addresses to match the zone of the NAT gateway. NAT gateway resources with IP addresses from a different zone or without a zone are unsupported.

Virtual networks and subnets are regional and not zonal aligned. A VM must be in the same zone as NAT gateway for a zonal promise of outbound connections. Zone isolation is created by creating a zonal "stack" per availability zone. A zonal promise won't exist when crossing zones of a zonal NAT gateway or using a regional NAT gateway with zonal VMs.

When you deploy virtual machine scale sets to use with NAT, you deploy a zonal scale set on its own subnet and attach the matching zone NAT gateway to that subnet. If you use zone-spanning scale sets (a scale set in two or more zones), NAT won't provide a zonal promise. NAT doesn't support zone-redundancy. Only regional or zone-isolation is supported.

*Figure: Zone-spanning Virtual Network NAT*

The zones property isn't mutable. Redeploy NAT gateway resource with the intended regional or zone preference.

### NOTE

IP addresses by themselves aren't zone-redundant if no zone is specified. The frontend of a [Standard Load Balancer](#) is [zone-redundant](#) if an IP address isn't created in a specific zone. This doesn't apply to NAT. Only regional or zone-isolation is supported.

## Source Network Address Translation

Source network address translation (SNAT) rewrites the source of a flow to originate from a different IP address. NAT gateway resources use a variant of SNAT commonly referred to port address translation (PAT). PAT rewrites the source address and source port. With SNAT, there's no fixed relationship between the number of private addresses and their translated public addresses.

### Fundamentals

Let's look at an example of four flows to explain the basic concept. The NAT gateway is using public IP address resource 65.52.0.2.

FLOW	SOURCE TUPLE	DESTINATION TUPLE
1	192.168.0.16:4283	65.52.0.1:80
2	192.168.0.16:4284	65.52.0.1:80

FLOW	SOURCE TUPLE	DESTINATION TUPLE
3	192.168.0.17.5768	65.52.0.1:80
4	192.168.0.16:4285	65.52.0.2:80

These flows might look like this after PAT has taken place:

FLOW	SOURCE TUPLE	SNAT'ED SOURCE TUPLE	DESTINATION TUPLE
1	192.168.0.16:4283	65.52.0.2:234	65.52.0.1:80
2	192.168.0.16:4284	65.52.0.2:235	65.52.0.1:80
3	192.168.0.17.5768	65.52.0.2:236	65.52.0.1:80
4	192.168.0.16:4285	65.52.0.2:237	65.52.0.2:80

The destination will see the source of the flow as 65.52.0.2 (SNAT source tuple) with the assigned port shown. PAT as shown in the preceding table is also called port masquerading SNAT. Multiple private sources are masqueraded behind an IP and port.

Don't take a dependency on the specific way source ports are assigned. The preceding is an illustration of the fundamental concept only.

SNAT provided by NAT is different from [Load Balancer](#) in several aspects.

### On-demand

NAT provides on-demand SNAT ports for new outbound traffic flows. All available SNAT ports in inventory are used by any virtual machine on subnets configured with NAT.

*Figure: Virtual Network NAT on-demand outbound SNAT*

Any IP configuration of a virtual machine can create outbound flows on-demand as needed. Pre-allocation, per instance planning including per instance worst case overprovisioning, isn't required.

*Figure: Differences in exhaustion scenarios*

Once a SNAT port releases, it's available for use by any virtual machine on subnets configured with NAT. On-demand allocation allows dynamic and divergent workloads on subnet(s) to use SNAT ports as they need. As long as there's SNAT port inventory available, SNAT flows will succeed. SNAT port hot spots benefit from the larger inventory instead. SNAT ports aren't left unused for virtual machines not actively needing them.

### Scaling

NAT needs sufficient SNAT port inventory for the complete outbound scenario. Scaling NAT is primarily a function of managing the shared, available SNAT port inventory. Sufficient inventory needs to exist to address the peak outbound flow for all subnets attached to a NAT gateway resource.

SNAT maps multiple private addresses to one public address and uses multiple public IPs to scale.

A NAT gateway resource will use 64,000 ports (SNAT ports) of a public IP address. These SNAT ports become the available inventory for the private to public flow mapping. And adding more public IP addresses increases the available inventory SNAT ports. NAT gateway resources can scale up to 16 IP addresses and 1M SNAT ports.

TCP and UDP are separate SNAT port inventories and unrelated.

NAT gateway resources opportunistically reuse source ports. For scaling purposes, you should assume each flow requires a new SNAT port and scale the total number of available IP addresses for outbound traffic.

## Protocols

NAT gateway resources interact with IP and IP transport headers of UDP and TCP flows and are agnostic to application layer payloads. Other IP protocols aren't supported.

## Timers

Idle timeout can be adjusted from 4 minutes (default) to 120 minutes (2 hours) for all flows. Additionally, you can reset the idle timer with traffic on the flow. A recommended pattern for refreshing long idle connections and endpoint liveness detection is TCP keepalives. TCP keepalives appear as duplicate ACKs to the endpoints, are low overhead, and invisible to the application layer.

The following timers are used for SNAT port release:

TIMER	VALUE
TCP FIN	60 seconds
TCP RST	10 seconds
TCP half open	30 seconds

A SNAT port is available for reuse to the same destination IP address and destination port after 5 seconds.

### NOTE

These timer settings are subject to change. The values are provided to help troubleshooting and you shouldn't take a dependency on specific timers at this time.

## Limitations

- NAT is compatible with standard SKU public IP, public IP prefix, and load balancer resources. Basic resources (for example basic load balancer) and any products derived from them aren't compatible with NAT. Basic resources must be placed on a subnet not configured with NAT.
- IPv4 address family is supported. NAT doesn't interact with IPv6 address family.
- NSG flow logging isn't supported when using NAT.
- NAT can't span multiple virtual networks.

## Preview participation

Follow [instructions to enable your subscription](#).

## Feedback

We want to know how we can improve the service. Share your [feedback on the Public Preview](#) with us. And you can propose and vote on what we should build next at [UserVoice for NAT](#).

## Next steps

- Learn more about [virtual network NAT](#).
- Tutorial for validating NAT Gateway

- [Azure CLI](#),
- [PowerShell](#),
- [Portal](#)
- Quickstart for deploying a NAT gateway resource
  - [Azure CLI](#),
  - [PowerShell](#),
  - [Portal](#).
- Learn more about [availability zones](#).
- Learn more about [standard load balancer](#).
- Learn more about [availability zones and standard load balancer](#).
- Learn more about NAT gateway resource API
  - [REST API](#),
  - [Azure CLI](#),
  - [PowerShell](#).
- [Tell us what to build next in UserVoice](#).
- [Provide feedback on the Public Preview](#).

# Interoperability in Azure back-end connectivity features: Test setup

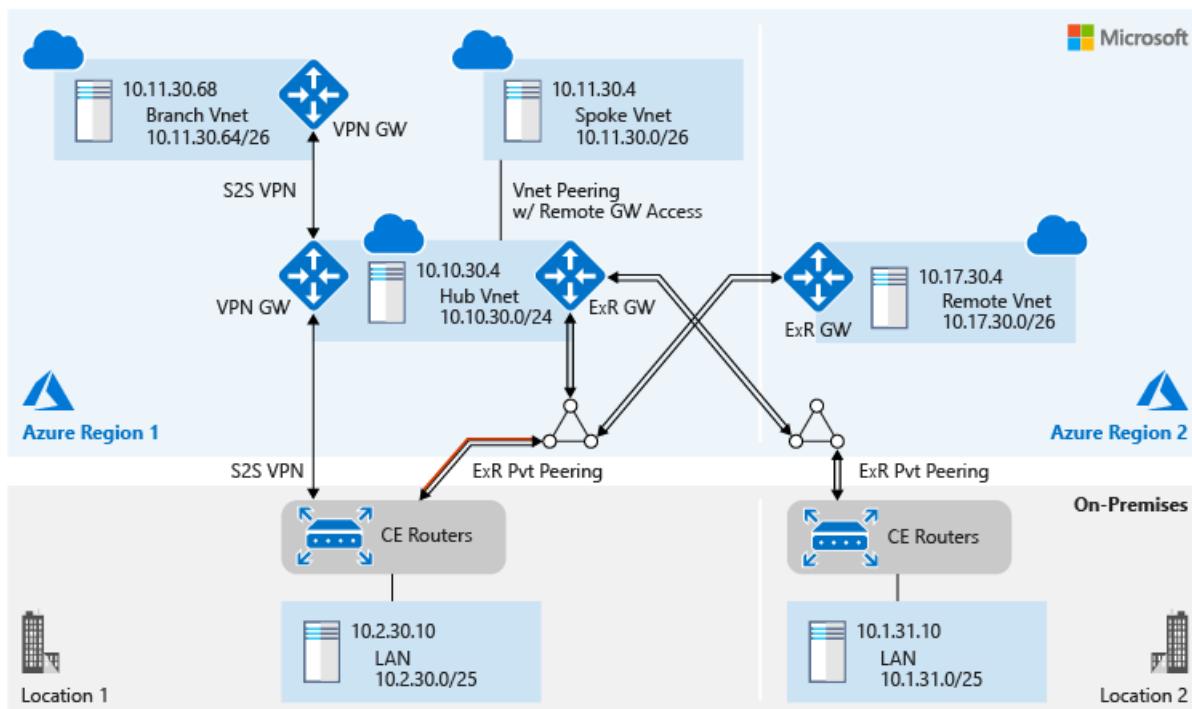
12/5/2019 • 4 minutes to read • [Edit Online](#)

This article describes a test setup you can use to analyze how Azure networking services interoperate at the control plane level and data plane level. Let's look briefly at the Azure networking components:

- **Azure ExpressRoute:** Use private peering in Azure ExpressRoute to directly connect private IP spaces in your on-premises network to your Azure Virtual Network deployments. ExpressRoute can help you achieve higher bandwidth and a private connection. Many ExpressRoute eco partners offer ExpressRoute connectivity with SLAs. To learn more about ExpressRoute and to learn how to configure ExpressRoute, see [Introduction to ExpressRoute](#).
- **Site-to-site VPN:** You can use Azure VPN Gateway as a site-to-site VPN to securely connect an on-premises network to Azure over the internet or by using ExpressRoute. To learn how to configure a site-to-site VPN to connect to Azure, see [Configure VPN Gateway](#).
- **VNet peering:** Use virtual network (VNet) peering to establish connectivity between VNets in Azure Virtual Network. To learn more about VNet peering, see the [tutorial on VNet peering](#).

## Test setup

The following figure illustrates the test setup:



The centerpiece of the test setup is the hub VNet in Azure Region 1. The hub VNet is connected to different networks in the following ways:

- The hub VNet is connected to the spoke VNet by using VNet peering. The spoke VNet has remote access to both gateways in the hub VNet.
- The hub VNet is connected to the branch VNet by using site-to-site VPN. The connectivity uses eBGP to exchange routes.
- The hub VNet is connected to the on-premises Location 1 network by using ExpressRoute private peering as

the primary path. It uses site-to-site VPN connectivity as the backup path. In the rest of this article, we refer to this ExpressRoute circuit as ExpressRoute 1. By default, ExpressRoute circuits provide redundant connectivity for high availability. On ExpressRoute 1, the secondary customer edge (CE) router's subinterface that faces the secondary Microsoft Enterprise Edge Router (MSEE) is disabled. A red line over the double-line arrow in the preceding figure represents the disabled CE router subinterface.

- The hub VNet is connected to the on-premises Location 2 network by using another ExpressRoute private peering. In the rest of this article, we refer to this second ExpressRoute circuit as ExpressRoute 2.
- ExpressRoute 1 also connects both the hub VNet and the on-premises Location 1 network to a remote VNet in Azure Region 2.

## ExpressRoute and site-to-site VPN connectivity in tandem

### Site-to-site VPN over ExpressRoute

You can configure a site-to-site VPN by using ExpressRoute Microsoft peering to privately exchange data between your on-premises network and your Azure VNets. With this configuration, you can exchange data with confidentiality, authenticity, and integrity. The data exchange also is anti-replay. For more information about how to configure a site-to-site IPsec VPN in tunnel mode by using ExpressRoute Microsoft peering, see [Site-to-site VPN over ExpressRoute Microsoft peering](#).

The primary limitation of configuring a site-to-site VPN that uses Microsoft peering is throughput. Throughput over the IPsec tunnel is limited by the VPN gateway capacity. The VPN gateway throughput is lower than ExpressRoute throughput. In this scenario, using the IPsec tunnel for highly secure traffic and using private peering for all other traffic helps optimize the ExpressRoute bandwidth utilization.

### Site-to-site VPN as a secure failover path for ExpressRoute

ExpressRoute serves as a redundant circuit pair to ensure high availability. You can configure geo-redundant ExpressRoute connectivity in different Azure regions. Also, as demonstrated in our test setup, within an Azure region, you can use a site-to-site VPN to create a failover path for your ExpressRoute connectivity. When the same prefixes are advertised over both ExpressRoute and a site-to-site VPN, Azure prioritizes ExpressRoute. To avoid asymmetrical routing between ExpressRoute and the site-to-site VPN, on-premises network configuration should also reciprocate by using ExpressRoute connectivity before it uses site-to-site VPN connectivity.

For more information about how to configure coexisting connections for ExpressRoute and a site-to-site VPN, see [ExpressRoute and site-to-site coexistence](#).

## Extend back-end connectivity to spoke VNets and branch locations

### Spoke VNet connectivity by using VNet peering

Hub and spoke VNet architecture is widely used. The hub is a VNet in Azure that acts as a central point of connectivity between your spoke VNets and to your on-premises network. The spokes are VNets that peer with the hub, and which you can use to isolate workloads. Traffic flows between the on-premises datacenter and the hub through an ExpressRoute or VPN connection. For more information about the architecture, see [Implement a hub-spoke network topology in Azure](#).

In VNet peering within a region, spoke VNets can use hub VNet gateways (both VPN and ExpressRoute gateways) to communicate with remote networks.

### Branch VNet connectivity by using site-to-site VPN

You might want branch VNets, which are in different regions, and on-premises networks to communicate with each other via a hub VNet. The native Azure solution for this configuration is site-to-site VPN connectivity by using a VPN. An alternative is to use a network virtual appliance (NVA) for routing in the hub.

For more information, see [What is VPN Gateway?](#) and [Deploy a highly available NVA](#).

## Next steps

Learn about [configuration details](#) for the test topology.

Learn about [control plane analysis](#) of the test setup and the views of different VNets or VLANs in the topology.

Learn about the [data plane analysis](#) of the test setup and Azure network monitoring feature views.

See the [ExpressRoute FAQ](#) to:

- Learn how many ExpressRoute circuits you can connect to an ExpressRoute gateway.
- Learn how many ExpressRoute gateways you can connect to an ExpressRoute circuit.
- Learn about other scale limits of ExpressRoute.

# Interoperability in Azure back-end connectivity features: Test configuration details

7/19/2019 • 6 minutes to read • [Edit Online](#)

This article describes the configuration details of the [test setup](#). The test setup helps you analyze how Azure networking services interoperate at the control plane level and data plane level.

## Spoke VNet connectivity by using VNet peering

The following figure shows the Azure Virtual Network peering details of a spoke virtual network (VNet). To learn how to set up peering between two VNets, see [Manage VNet peering](#). If you want the spoke VNet to use the gateways that are connected to the hub VNet, select **Use remote gateways**.

The screenshot shows the configuration details for a VNet peering named "Spoke01-VNet01-peering".

**General Information:**

- Name: Spoke01-VNet01-peering
- Peering status: Connected
- Provisioning state: Succeeded

**Peer details:**

- Address space: 10.10.30.0/24
- Virtual network: vNet01

**Configuration:**

- Allow virtual network access: Enabled
- Allow forwarded traffic:
- Allow gateway transit:
- Use remote gateways:

The following figure shows the VNet peering details of the hub VNet. If you want the hub VNet to permit the spoke VNet to use the hub's gateways, select **Allow gateway transit**.

VNet01-Spoke01-Peering

VNet01

Save Discard Delete

Name  
VNet01-Spoke01-Peering

Peering status  
Connected

Provisioning state  
Succeeded

**Peer details**

Address space  
10.11.30.0/26

Virtual network  
**Spoke01-VNet**

**Configuration**

Allow virtual network access ⓘ

Disabled  Enabled

Allow forwarded traffic ⓘ

Allow gateway transit ⓘ

Use remote gateways ⓘ

## Branch VNet connectivity by using a site-to-site VPN

Set up site-to-site VPN connectivity between the hub and branch VNets by using VPN gateways in Azure VPN Gateway. By default, VPN gateways and Azure ExpressRoute gateways use a private autonomous system number (ASN) value of **65515**. You can change the ASN value in VPN Gateway. In the test setup, the ASN value of the branch VNet VPN gateway is changed to **65516** to support eBGP routing between the hub and branch VNets.

## On-premises Location 1 connectivity by using ExpressRoute and a site-to-site VPN

### ExpressRoute 1 configuration details

The following figure shows the Azure Region 1 ExpressRoute circuit configuration toward on-premises Location 1 customer edge (CE) routers:

Type	Status	Primary Subnet	Secondary Subnet	Last Modified By
Azure private	Provisioned	192.168.30.16/30	192.168.30.20/30	Customer

The following figure shows the connection configuration between the ExpressRoute 1 circuit and the hub VNet:

[Move](#) [Delete](#)

Resource group ( <a href="#">change</a> )	Data in
ASH-Cust30	0 B
Status	Data out
Succeeded	0 B
Location	Virtual network
East US	VNet01
Subscription ( <a href="#">change</a> )	Virtual network gateway
ExpressRoute-Lab	ASH-Cust30-gw (13.90.87.1)
Subscription ID	Circuit
	ASH-Cust30-ER
Tags ( <a href="#">change</a> )	
<a href="#">Click here to add tags</a>	

The following list shows the primary CE router configuration for ExpressRoute private peering connectivity. (Cisco ASR1000 routers are used as CE routers in the test setup.) When site-to-site VPN and ExpressRoute circuits are configured in parallel to connect an on-premises network to Azure, Azure prioritizes the ExpressRoute circuit by default. To avoid asymmetrical routing, the on-premises network also should prioritize ExpressRoute connectivity over site-to-site VPN connectivity. The following configuration establishes prioritization by using the BGP **local-preference** attribute:

```
interface TenGigabitEthernet0/0/0.300
description Customer 30 private peering to Azure
encapsulation dot1Q 30 second-dot1q 300
ip vrf forwarding 30
ip address 192.168.30.17 255.255.255.252
!
interface TenGigabitEthernet1/0/0.30
description Customer 30 to south bound LAN switch
encapsulation dot1Q 30
ip vrf forwarding 30
ip address 192.168.30.0 255.255.255.254
ip ospf network point-to-point
!
router ospf 30 vrf 30
router-id 10.2.30.253
redistribute bgp 65021 subnets route-map BGP20SPF
network 192.168.30.0 0.0.0.1 area 0.0.0.0
default-information originate always
default-metric 10
!
router bgp 65021
!
address-family ipv4 vrf 30
network 10.2.30.0 mask 255.255.255.128
neighbor 192.168.30.18 remote-as 12076
neighbor 192.168.30.18 activate
neighbor 192.168.30.18 next-hop-self
neighbor 192.168.30.18 soft-reconfiguration inbound
neighbor 192.168.30.18 route-map prefer-ER-over-VPN in
neighbor 192.168.30.18 prefix-list Cust30_to_Private out
exit-address-family
!
route-map prefer-ER-over-VPN permit 10
set local-preference 200
!
ip prefix-list Cust30_to_Private seq 10 permit 10.2.30.0/25
!
```

## Site-to-site VPN configuration details

The following list shows the primary CE router configuration for site-to-site VPN connectivity:

```

crypto ikev2 proposal Cust30-azure-proposal
  encryption aes-cbc-256 aes-cbc-128 3des
  integrity sha1
  group 2
!
crypto ikev2 policy Cust30-azure-policy
  match address local 66.198.12.106
  proposal Cust30-azure-proposal
!
crypto ikev2 keyring Cust30-azure-keyring
  peer azure
  address 52.168.162.84
  pre-shared-key local IamSecure123
  pre-shared-key remote IamSecure123
!
crypto ikev2 profile Cust30-azure-profile
  match identity remote address 52.168.162.84 255.255.255.255
  identity local address 66.198.12.106
  authentication local pre-share
  authentication remote pre-share
  keyring local Cust30-azure-keyring
!
crypto ipsec transform-set Cust30-azure-ipsec-proposal-set esp-aes 256 esp-sha-hmac
  mode tunnel
!
crypto ipsec profile Cust30-azure-ipsec-profile
  set transform-set Cust30-azure-ipsec-proposal-set
  set ikev2-profile Cust30-azure-profile
!
interface Loopback30
  ip address 66.198.12.106 255.255.255.255
!
interface Tunnel30
  ip vrf forwarding 30
  ip address 10.2.30.125 255.255.255.255
  tunnel source Loopback30
  tunnel mode ipsec ipv4
  tunnel destination 52.168.162.84
  tunnel protection ipsec profile Cust30-azure-ipsec-profile
!
router bgp 65021
!
address-family ipv4 vrf 30
  network 10.2.30.0 mask 255.255.255.128
  neighbor 10.10.30.254 remote-as 65515
  neighbor 10.10.30.254 ebgp-multihop 5
  neighbor 10.10.30.254 update-source Tunnel30
  neighbor 10.10.30.254 activate
  neighbor 10.10.30.254 soft-reconfiguration inbound
exit-address-family
!
ip route vrf 30 10.10.30.254 255.255.255.255 Tunnel30

```

## On-premises Location 2 connectivity by using ExpressRoute

A second ExpressRoute circuit, in closer proximity to on-premises Location 2, connects on-premises Location 2 to the hub VNet. The following figure shows the second ExpressRoute configuration:

		Move	Delete	Refresh
Resource group <a href="#">(change)</a>	ASH-Cust30	Provider	Equinix	
Circuit status	Enabled	Provider status	Provisioned	
Location	East US	Peering location	Seattle	
Subscription <a href="#">(change)</a>	ExpressRoute-Lab	Bandwidth	50 Mbps	
Subscription ID		Service key		
Tags <a href="#">(change)</a>				
	<a href="#">Click here to add tags</a>			
<hr/>				
Peerings				
Type	Status	Primary Subnet	Secondary Subnet	Last modified by
Azure private	Provisioned	192.168.31.16/30	192.168.31.20/30	Customer
				...

The following figure shows the connection configuration between the second ExpressRoute circuit and the hub VNet:

Resource group <a href="#">(change)</a>	ASH-Cust30	Data in 0 B
Status	Succeeded	Data out 0 B
Location	East US	Virtual network <a href="#">VNet01</a>
Subscription <a href="#">(change)</a>	ExpressRoute-Lab	Virtual network gateway <a href="#">ASH-Cust30-gw (13.90.87.1)</a>
Subscription ID		Circuit <a href="#">SEA-Cust31-ER</a>
Tags <a href="#">(change)</a>		
	<a href="#">Click here to add tags</a>	

ExpressRoute 1 connects both the hub VNet and on-premises Location 1 to a remote VNet in a different Azure region:

		Move	Delete
Resource group <a href="#">(change)</a>	ASH-Cust30	Data in 0 B	
Status	Succeeded	Data out 0 B	
Location	West US 2	Virtual network <a href="#">USWst2-VNet</a>	
Subscription <a href="#">(change)</a>	ExpressRoute-Lab	Virtual network gateway <a href="#">ASH30-USWst2-ERGW (52.175.245.182)</a>	
Subscription ID		Circuit <a href="#">ASH-Cust30-ER</a>	
Tags <a href="#">(change)</a>			
	<a href="#">Click here to add tags</a>		

## ExpressRoute and site-to-site VPN connectivity in tandem

### Site-to-site VPN over ExpressRoute

You can configure a site-to-site VPN by using ExpressRoute Microsoft peering to privately exchange data between your on-premises network and your Azure VNets. With this configuration, you can exchange data with confidentiality, authenticity, and integrity. The data exchange also is anti-replay. For more information about how to configure a site-to-site IPsec VPN in tunnel mode by using ExpressRoute Microsoft peering, see [Site-to-site VPN over ExpressRoute Microsoft peering](#).

The primary limitation of configuring a site-to-site VPN that uses Microsoft peering is throughput. Throughput over the IPsec tunnel is limited by the VPN gateway capacity. The VPN gateway throughput is lower than

ExpressRoute throughput. In this scenario, using the IPsec tunnel for highly secure traffic and using private peering for all other traffic helps optimize the ExpressRoute bandwidth utilization.

### **Site-to-site VPN as a secure failover path for ExpressRoute**

ExpressRoute serves as a redundant circuit pair to ensure high availability. You can configure geo-redundant ExpressRoute connectivity in different Azure regions. Also, as demonstrated in our test setup, within an Azure region, you can use a site-to-site VPN to create a failover path for your ExpressRoute connectivity. When the same prefixes are advertised over both ExpressRoute and a site-to-site VPN, Azure prioritizes ExpressRoute. To avoid asymmetrical routing between ExpressRoute and the site-to-site VPN, on-premises network configuration should also reciprocate by using ExpressRoute connectivity before it uses site-to-site VPN connectivity.

For more information about how to configure coexisting connections for ExpressRoute and a site-to-site VPN, see [ExpressRoute and site-to-site coexistence](#).

## Extend back-end connectivity to spoke VNets and branch locations

### **Spoke VNet connectivity by using VNet peering**

Hub and spoke VNet architecture is widely used. The hub is a VNet in Azure that acts as a central point of connectivity between your spoke VNets and to your on-premises network. The spokes are VNets that peer with the hub, and which you can use to isolate workloads. Traffic flows between the on-premises datacenter and the hub through an ExpressRoute or VPN connection. For more information about the architecture, see [Implement a hub-spoke network topology in Azure](#).

In VNet peering within a region, spoke VNets can use hub VNet gateways (both VPN and ExpressRoute gateways) to communicate with remote networks.

### **Branch VNet connectivity by using site-to-site VPN**

You might want branch VNets, which are in different regions, and on-premises networks to communicate with each other via a hub VNet. The native Azure solution for this configuration is site-to-site VPN connectivity by using a VPN. An alternative is to use a network virtual appliance (NVA) for routing in the hub.

For more information, see [What is VPN Gateway?](#) and [Deploy a highly available NVA](#).

## Next steps

Learn about [control plane analysis](#) of the test setup and the views of different VNets or VLANs in the topology.

Learn about [data plane analysis](#) of the test setup and Azure network monitoring feature views.

See the [ExpressRoute FAQ](#) to:

- Learn how many ExpressRoute circuits you can connect to an ExpressRoute gateway.
- Learn how many ExpressRoute gateways you can connect to an ExpressRoute circuit.
- Learn about other scale limits of ExpressRoute.

# Interoperability in Azure back-end connectivity features: Control plane analysis

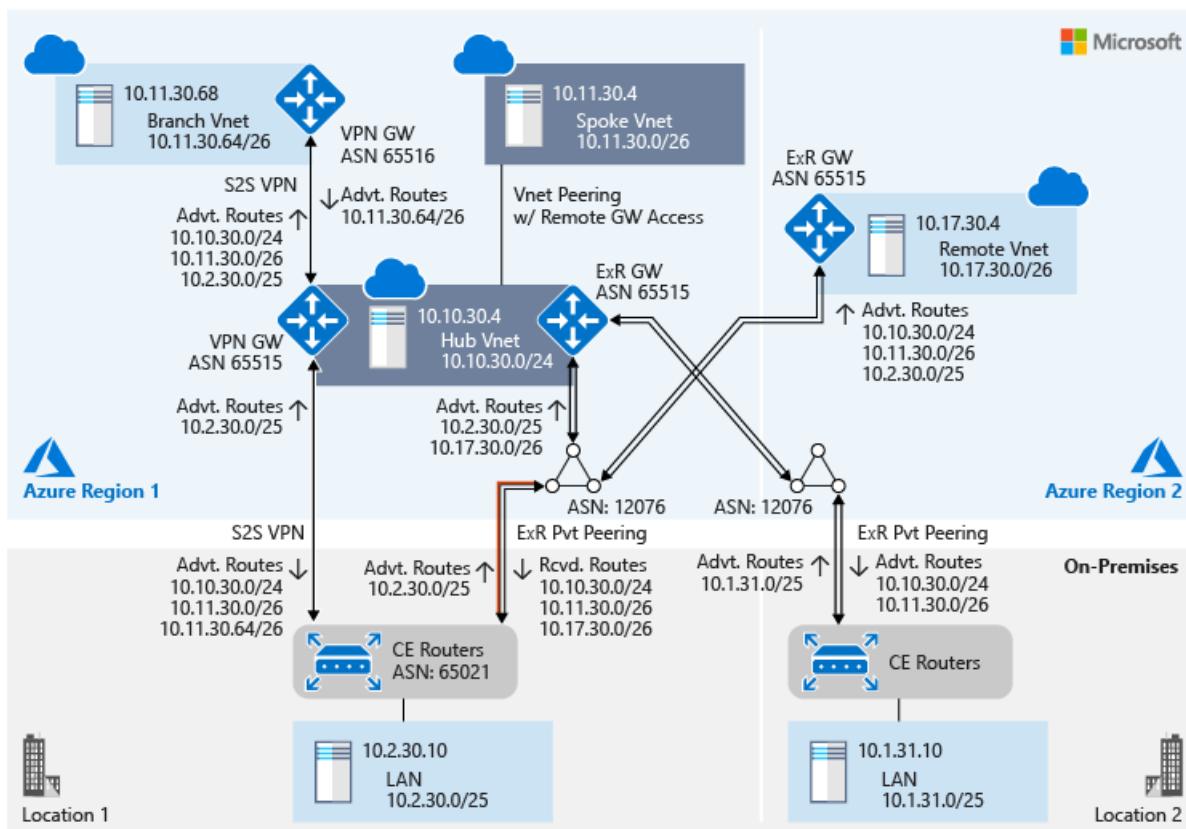
12/5/2019 • 4 minutes to read • [Edit Online](#)

This article describes the control plane analysis of the [test setup](#). You can also review the [test setup configuration](#) and the [data plane analysis](#) of the test setup.

Control plane analysis essentially examines routes that are exchanged between networks within a topology. Control plane analysis can help you understand how different networks view the topology.

## Hub and spoke VNet perspective

The following figure illustrates the network from the perspective of a hub virtual network (VNet) and a spoke VNet (highlighted in blue). The figure also shows the autonomous system number (ASN) of different networks and routes that are exchanged between different networks:



The ASN of the VNet's Azure ExpressRoute gateway is different from the ASN of Microsoft Enterprise Edge Routers (MSEEs). An ExpressRoute gateway uses a private ASN (a value of **65515**) and MSEEs use public ASN (a value of **12076**) globally. When you configure ExpressRoute peering, because MSEE is the peer, you use **12076** as the peer ASN. On the Azure side, MSEE establishes eBGP peering with the ExpressRoute gateway. The dual eBGP peering that the MSEE establishes for each ExpressRoute peering is transparent at the control plane level. Therefore, when you view an ExpressRoute route table, you see the VNet's ExpressRoute gateway ASN for the VNet's prefixes.

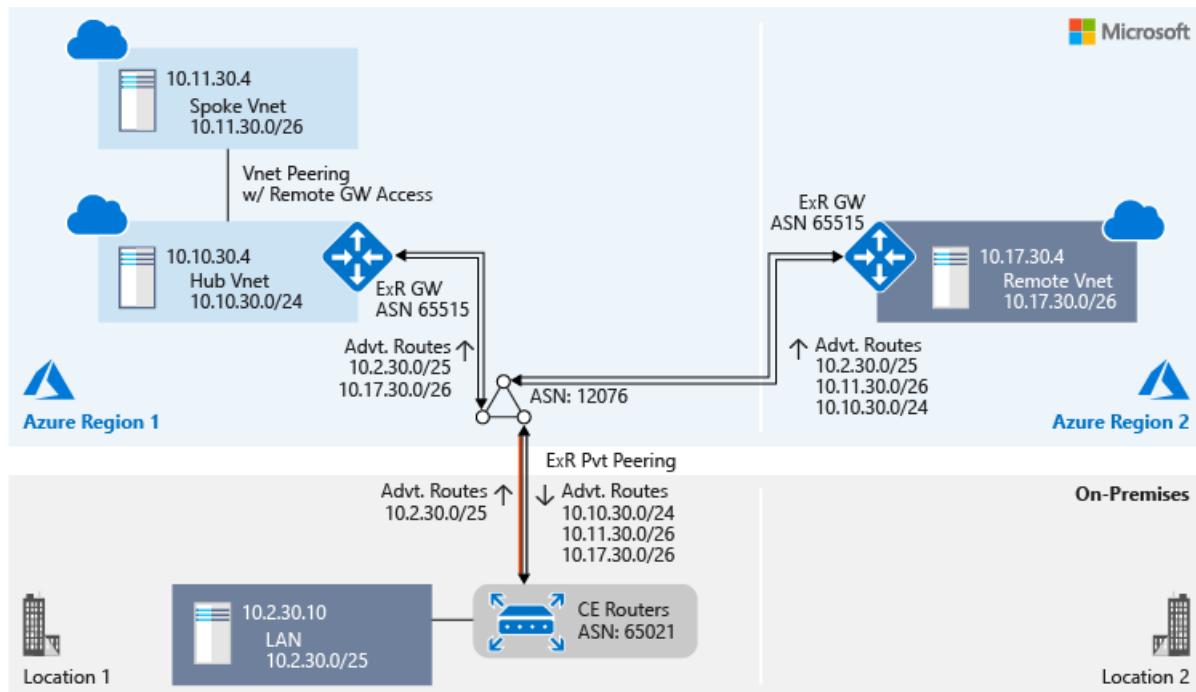
The following figure shows a sample ExpressRoute route table:

Route table (Primary)					
AzurePrivatePeering - ASH-Cust30-ER					
Download	Show secondary				
<b>i</b> Showing only top 200 primary records, click Download above to see all.					
NETWORK	NEXT HOP	LOCPRF	WEIGHT	PATH	
10.2.30.0/25	192.168.30.17		0	65021	
10.2.30.125/32	192.168.30.17		0	65021 65515	
10.10.30.0/24	10.10.30.141		0	65515	
	10.10.30.140		0	65515	
10.11.30.0/26	10.10.30.141		0	65515	
	10.10.30.140		0	65515	

Within Azure, the ASN is significant only from a peering perspective. By default, the ASN of both the ExpressRoute gateway and the VPN gateway in Azure VPN Gateway is **65515**.

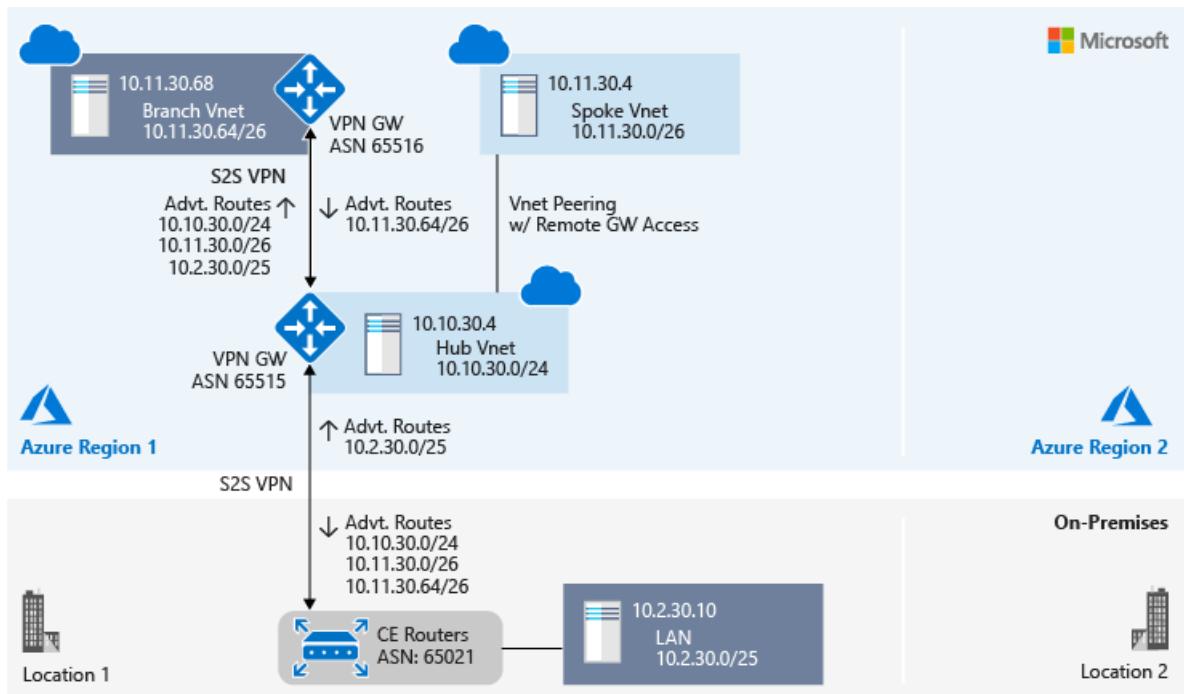
## On-premises Location 1 and the remote VNet perspective via ExpressRoute 1

Both on-premises Location 1 and the remote VNet are connected to the hub VNet via ExpressRoute 1. They share the same perspective of the topology, as shown in the following diagram:



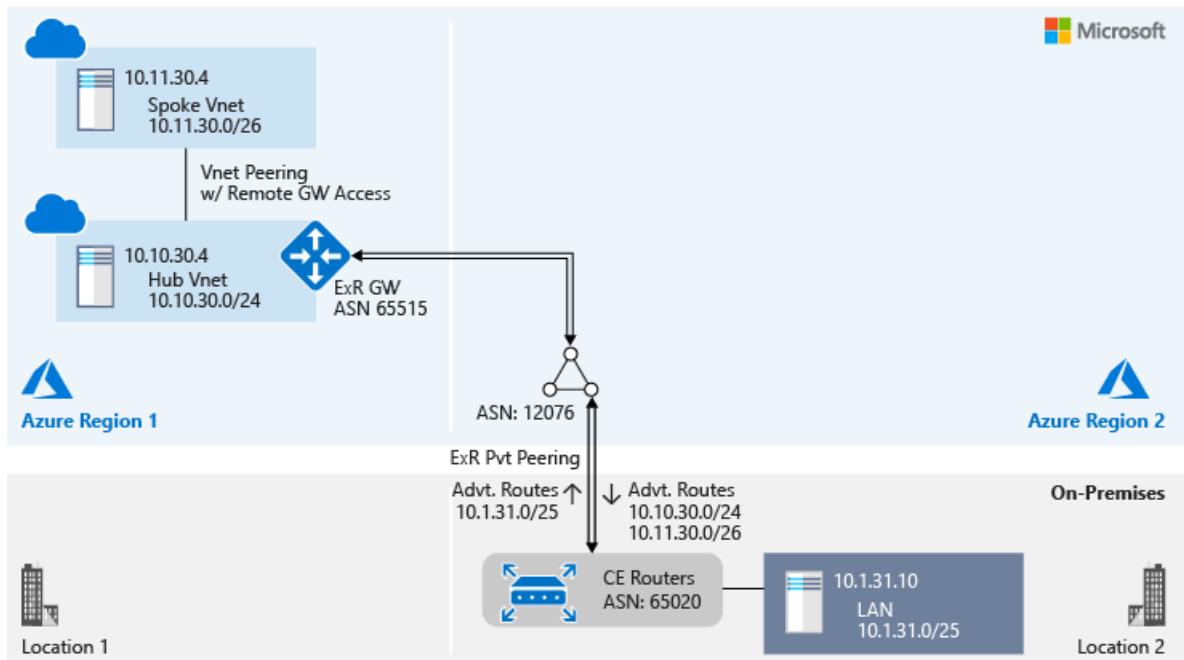
## On-premises Location 1 and the branch VNet perspective via a site-to-site VPN

Both on-premises Location 1 and the branch VNet are connected to a hub VNet's VPN gateway via a site-to-site VPN connection. They share the same perspective of the topology, as shown in the following diagram:



## On-premises Location 2 perspective

On-premises Location 2 is connected to a hub VNet via private peering of ExpressRoute 2:



## ExpressRoute and site-to-site VPN connectivity in tandem

### Site-to-site VPN over ExpressRoute

You can configure a site-to-site VPN by using ExpressRoute Microsoft peering to privately exchange data between your on-premises network and your Azure VNets. With this configuration, you can exchange data with confidentiality, authenticity, and integrity. The data exchange also is anti-replay. For more information about how to configure a site-to-site IPsec VPN in tunnel mode by using ExpressRoute Microsoft peering, see [Site-to-site VPN over ExpressRoute Microsoft peering](#).

The primary limitation of configuring a site-to-site VPN that uses Microsoft peering is throughput. Throughput over the IPsec tunnel is limited by the VPN gateway capacity. The VPN gateway throughput is lower than ExpressRoute throughput. In this scenario, using the IPsec tunnel for highly secure traffic and using private peering for all other traffic helps optimize the ExpressRoute bandwidth utilization.

## **Site-to-site VPN as a secure failover path for ExpressRoute**

ExpressRoute serves as a redundant circuit pair to ensure high availability. You can configure geo-redundant ExpressRoute connectivity in different Azure regions. Also, as demonstrated in our test setup, within an Azure region, you can use a site-to-site VPN to create a failover path for your ExpressRoute connectivity. When the same prefixes are advertised over both ExpressRoute and a site-to-site VPN, Azure prioritizes ExpressRoute. To avoid asymmetrical routing between ExpressRoute and the site-to-site VPN, on-premises network configuration should also reciprocate by using ExpressRoute connectivity before it uses site-to-site VPN connectivity.

For more information about how to configure coexisting connections for ExpressRoute and a site-to-site VPN, see [ExpressRoute and site-to-site coexistence](#).

## Extend back-end connectivity to spoke VNets and branch locations

### **Spoke VNet connectivity by using VNet peering**

Hub and spoke VNet architecture is widely used. The hub is a VNet in Azure that acts as a central point of connectivity between your spoke VNets and to your on-premises network. The spokes are VNets that peer with the hub, and which you can use to isolate workloads. Traffic flows between the on-premises datacenter and the hub through an ExpressRoute or VPN connection. For more information about the architecture, see [Implement a hub-spoke network topology in Azure](#).

In VNet peering within a region, spoke VNets can use hub VNet gateways (both VPN and ExpressRoute gateways) to communicate with remote networks.

### **Branch VNet connectivity by using site-to-site VPN**

You might want branch VNets, which are in different regions, and on-premises networks to communicate with each other via a hub VNet. The native Azure solution for this configuration is site-to-site VPN connectivity by using a VPN. An alternative is to use a network virtual appliance (NVA) for routing in the hub.

For more information, see [What is VPN Gateway?](#) and [Deploy a highly available NVA](#).

## Next steps

Learn about [data plane analysis](#) of the test setup and Azure network monitoring feature views.

See the [ExpressRoute FAQ](#) to:

- Learn how many ExpressRoute circuits you can connect to an ExpressRoute gateway.
- Learn how many ExpressRoute gateways you can connect to an ExpressRoute circuit.
- Learn about other scale limits of ExpressRoute.

# Interoperability in Azure back-end connectivity features: Data plane analysis

2/21/2020 • 16 minutes to read • [Edit Online](#)

This article describes the data plane analysis of the [test setup](#). You can also review the [test setup configuration](#) and the [control plane analysis](#) of the test setup.

Data plane analysis examines the path taken by packets that traverse from one local network (LAN or virtual network) to another within a topology. The data path between two local networks isn't necessarily symmetrical. Therefore, in this article, we analyze a forwarding path from a local network to another network that's separate from the reverse path.

## Data path from the hub VNet

### Path to the spoke VNet

Virtual network (VNet) peering emulates network bridge functionality between the two VNets that are peered. Traceroute output from a hub VNet to a VM in the spoke VNet is shown here:

```
C:\Users\rb>tracert 10.11.30.4

Tracing route to 10.11.30.4 over a maximum of 30 hops

 1      2 ms      1 ms      1 ms  10.11.30.4

Trace complete.
```

The following figure shows the graphical connection view of the hub VNet and the spoke VNet from the perspective of Azure Network Watcher:



### Path to the branch VNet

Traceroute output from a hub VNet to a VM in the branch VNet is shown here:

```
C:\Users\rb>tracert 10.11.30.68

Tracing route to 10.11.30.68 over a maximum of 30 hops

 1      1 ms      1 ms      1 ms  10.10.30.142
 2      *          *          *      Request timed out.
 3      2 ms      2 ms      2 ms  10.11.30.68

Trace complete.
```

In this traceroute, the first hop is the VPN gateway in Azure VPN Gateway of the hub VNet. The second hop is the VPN gateway of the branch VNet. The IP address of the VPN gateway of the branch VNet isn't advertised in the hub VNet. The third hop is the VM on the branch VNet.

The following figure shows the graphical connection view of the hub VNet and the branch VNet from the perspective of Network Watcher:



For the same connection, the following figure shows the grid view in Network Watcher:

Hops				
NAME	IP ADDRESS	STATUS	NEXT HOP IP ADDRESS	
ash-cust30-vm1195	10.10.30.4	✓	13.90.87.1	
ASH-Cust30-gw	13.90.87.1	✓	10.11.30.68	
ash-c30-sk2-vm1619	10.11.30.68	✓	-	

Average Latency in milliseconds

3

Minimum Latency in milliseconds

2

Maximum Latency in milliseconds

6

Probes Sent

66

Probes Failed

0

## Path to on-premises Location 1

Traceroute output from a hub VNet to a VM in on-premises Location 1 is shown here:

```
C:\Users\rb>tracert 10.2.30.10

Tracing route to 10.2.30.10 over a maximum of 30 hops

 1  2 ms    2 ms    2 ms  10.10.30.132
 2  *        *        *      Request timed out.
 3  *        *        *      Request timed out.
 4  2 ms    2 ms    2 ms  10.2.30.10

Trace complete.
```

In this traceroute, the first hop is the Azure ExpressRoute gateway tunnel endpoint to a Microsoft Enterprise Edge Router (MSEE). The second and third hops are the customer edge (CE) router and the on-premises Location 1 LAN IPs. These IP addresses aren't advertised in the hub VNet. The fourth hop is the VM in the on-premises Location 1.

## Path to on-premises Location 2

Traceroute output from a hub VNet to a VM in on-premises Location 2 is shown here:

```
C:\Users\rb>tracert 10.1.31.10

Tracing route to 10.1.31.10 over a maximum of 30 hops

 1  76 ms   75 ms   75 ms  10.10.30.134
 2  *         *         *      Request timed out.
 3  *         *         *      Request timed out.
 4  75 ms   75 ms   75 ms  10.1.31.10

Trace complete.
```

In this traceroute, the first hop is the ExpressRoute gateway tunnel endpoint to an MSEE. The second and third hops are the CE router and the on-premises Location 2 LAN IPs. These IP addresses aren't advertised in the hub VNet. The fourth hop is the VM on the on-premises Location 2.

### Path to the remote VNet

Traceroute output from a hub VNet to a VM in the remote VNet is shown here:

```
C:\Users\rb>tracert 10.17.30.4

Tracing route to 10.17.30.4 over a maximum of 30 hops

 1  2 ms   2 ms   2 ms  10.10.30.132
 2  *         *         *      Request timed out.
 3  69 ms   68 ms   69 ms  10.17.30.4

Trace complete.
```

In this traceroute, the first hop is the ExpressRoute gateway tunnel endpoint to an MSEE. The second hop is the remote VNet's gateway IP. The second hop IP range isn't advertised in the hub VNet. The third hop is the VM on the remote VNet.

## Data path from the spoke VNet

The spoke VNet shares the network view of the hub VNet. Through VNet peering, the spoke VNet uses the remote gateway connectivity of the hub VNet as if it's directly connected to the spoke VNet.

### Path to the hub VNet

Traceroute output from the spoke VNet to a VM in the hub VNet is shown here:

```
C:\Users\rb>tracert 10.10.30.4

Tracing route to 10.10.30.4 over a maximum of 30 hops

 1  <1 ms   <1 ms   <1 ms  10.10.30.4

Trace complete.
```

### Path to the branch VNet

Traceroute output from the spoke VNet to a VM in the branch VNet is shown here:

```
C:\Users\rb>tracert 10.11.30.68

Tracing route to 10.11.30.68 over a maximum of 30 hops

 1  1 ms    <1 ms    <1 ms  10.10.30.142
 2  *         *         *      Request timed out.
 3  3 ms    2 ms    2 ms  10.11.30.68

Trace complete.
```

In this traceroute, the first hop is the VPN gateway of the hub VNet. The second hop is the VPN gateway of the branch VNet. The IP address of the VPN gateway of the branch VNet isn't advertised within the hub/spoke VNet. The third hop is the VM on the branch VNet.

### Path to on-premises Location 1

Traceroute output from the spoke VNet to a VM in on-premises Location 1 is shown here:

```
C:\Users\rb>tracert 10.2.30.10

Tracing route to 10.2.30.10 over a maximum of 30 hops

 1  24 ms    2 ms    3 ms  10.10.30.132
 2  *         *         *      Request timed out.
 3  *         *         *      Request timed out.
 4  3 ms    2 ms    2 ms  10.2.30.10

Trace complete.
```

In this traceroute, the first hop is the hub VNet's ExpressRoute gateway tunnel endpoint to an MSEE. The second and third hops are the CE router and the on-premises Location 1 LAN IPs. These IP addresses aren't advertised in the hub/spoke VNet. The fourth hop is the VM in the on-premises Location 1.

### Path to on-premises Location 2

Traceroute output from the spoke VNet to a VM in on-premises Location 2 is shown here:

```
C:\Users\rb>tracert 10.1.31.10

Tracing route to 10.1.31.10 over a maximum of 30 hops

 1  76 ms    75 ms    76 ms  10.10.30.134
 2  *         *         *      Request timed out.
 3  *         *         *      Request timed out.
 4  75 ms    75 ms    75 ms  10.1.31.10

Trace complete.
```

In this traceroute, the first hop is the hub VNet's ExpressRoute gateway tunnel endpoint to an MSEE. The second and third hops are the CE router and the on-premises Location 2 LAN IPs. These IP addresses aren't advertised in the hub/spoke VNets. The fourth hop is the VM in the on-premises Location 2.

### Path to the remote VNet

Traceroute output from the spoke VNet to a VM in the remote VNet is shown here:

```
C:\Users\rb>tracert 10.17.30.4

Tracing route to 10.17.30.4 over a maximum of 30 hops

 1  2 ms    1 ms    1 ms  10.10.30.133
 2  *        *        *      Request timed out.
 3  71 ms   70 ms   70 ms  10.17.30.4

Trace complete.
```

In this traceroute, the first hop is the hub VNet's ExpressRoute gateway tunnel endpoint to an MSEE. The second hop is the remote VNet's gateway IP. The second hop IP range isn't advertised in the hub/spoke VNet. The third hop is the VM on the remote VNet.

## Data path from the branch VNet

### Path to the hub VNet

Traceroute output from the branch VNet to a VM in the hub VNet is shown here:

```
C:\Windows\system32>tracert 10.10.30.4

Tracing route to 10.10.30.4 over a maximum of 30 hops

 1  <1 ms    <1 ms    <1 ms  10.11.30.100
 2  *        *        *      Request timed out.
 3  4 ms    3 ms    3 ms  10.10.30.4

Trace complete.
```

In this traceroute, the first hop is the VPN gateway of the branch VNet. The second hop is the VPN gateway of the hub VNet. The IP address of the VPN gateway of the hub VNet isn't advertised in the remote VNet. The third hop is the VM on the hub VNet.

### Path to the spoke VNet

Traceroute output from the branch VNet to a VM in the spoke VNet is shown here:

```
C:\Users\rb>tracert 10.11.30.4

Tracing route to 10.11.30.4 over a maximum of 30 hops

 1  1 ms    <1 ms    1 ms  10.11.30.100
 2  *        *        *      Request timed out.
 3  4 ms    3 ms    2 ms  10.11.30.4

Trace complete.
```

In this traceroute, the first hop is the VPN gateway of the branch VNet. The second hop is the VPN gateway of the hub VNet. The IP address of the VPN gateway of the hub VNet isn't advertised in the remote VNet. The third hop is the VM on the spoke VNet.

### Path to on-premises Location 1

Traceroute output from the branch VNet to a VM in on-premises Location 1 is shown here:

```
C:\Users\rb>tracert 10.2.30.10

Tracing route to 10.2.30.10 over a maximum of 30 hops

 1  1 ms    <1 ms    <1 ms  10.11.30.100
 2  *         *         *      Request timed out.
 3  3 ms    2 ms    2 ms  10.2.30.125
 4  *         *         *      Request timed out.
 5  3 ms    3 ms    3 ms  10.2.30.10

Trace complete.
```

In this traceroute, the first hop is the VPN gateway of the branch VNet. The second hop is the VPN gateway of the hub VNet. The IP address of the VPN gateway of the hub VNet isn't advertised in the remote VNet. The third hop is the VPN tunnel termination point on the primary CE router. The fourth hop is an internal IP address of on-premises Location 1. This LAN IP address isn't advertised outside the CE router. The fifth hop is the destination VM in the on-premises Location 1.

### **Path to on-premises Location 2 and the remote VNet**

As we discussed in the control plane analysis, the branch VNet has no visibility either to on-premises Location 2 or to the remote VNet per the network configuration. The following ping results confirm:

```
C:\Users\rb>ping 10.1.31.10

Pinging 10.1.31.10 with 32 bytes of data:

Request timed out.
...
Request timed out.

Ping statistics for 10.1.31.10:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\Users\rb>ping 10.17.30.4

Pinging 10.17.30.4 with 32 bytes of data:

Request timed out.
...
Request timed out.

Ping statistics for 10.17.30.4:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

## **Data path from on-premises Location 1**

### **Path to the hub VNet**

Traceroute output from on-premises Location 1 to a VM in the hub VNet is shown here:

```
C:\Users\rb>tracert 10.10.30.4

Tracing route to 10.10.30.4 over a maximum of 30 hops

 1 <1 ms    <1 ms    <1 ms  10.2.30.3
 2 <1 ms    <1 ms    <1 ms  192.168.30.0
 3 <1 ms    <1 ms    <1 ms  192.168.30.18
 4 *        *        * Request timed out.
 5 2 ms     2 ms     2 ms  10.10.30.4

Trace complete.
```

In this traceroute, the first two hops are part of the on-premises network. The third hop is the primary MSEE interface that faces the CE router. The fourth hop is the ExpressRoute gateway of the hub VNet. The IP range of the ExpressRoute gateway of the hub VNet isn't advertised to the on-premises network. The fifth hop is the destination VM.

Network Watcher provides only an Azure-centric view. For an on-premises perspective, we use Azure Network Performance Monitor. Network Performance Monitor provides agents that you can install on servers in networks outside Azure for data path analysis.

The following figure shows the topology view of the on-premises Location 1 VM connectivity to the VM on the hub VNet via ExpressRoute:



As discussed earlier, the test setup uses a site-to-site VPN as backup connectivity for ExpressRoute between the on-premises Location 1 and the hub VNet. To test the backup data path, let's induce an ExpressRoute link failure between the on-premises Location 1 primary CE router and the corresponding MSEE. To induce an ExpressRoute link failure, shut down the CE interface that faces the MSEE:

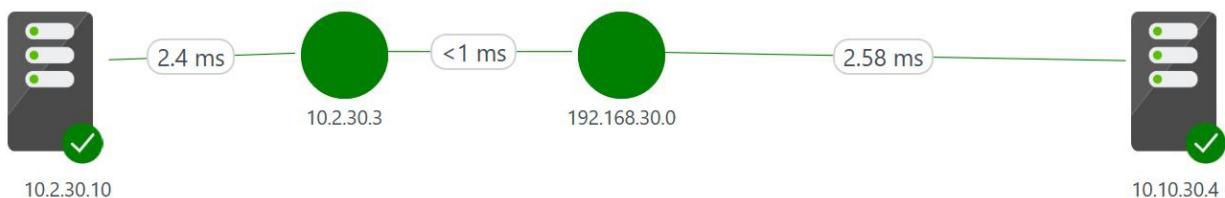
```
C:\Users\rb>tracert 10.10.30.4

Tracing route to 10.10.30.4 over a maximum of 30 hops

 1 <1 ms    <1 ms    <1 ms  10.2.30.3
 2 <1 ms    <1 ms    <1 ms  192.168.30.0
 3 3 ms     2 ms     3 ms  10.10.30.4

Trace complete.
```

The following figure shows the topology view of the on-premises Location 1 VM connectivity to the VM on the hub VNet via site-to-site VPN connectivity when ExpressRoute connectivity is down:



## Path to the spoke VNet

Traceroute output from on-premises Location 1 to a VM in the spoke VNet is shown here:

Let's bring back the ExpressRoute primary connectivity to do the data path analysis toward the spoke VNet:

```
C:\Users\rb>tracert 10.11.30.4

Tracing route to 10.11.30.4 over a maximum of 30 hops

 1  <1 ms    <1 ms    <1 ms  10.2.30.3
 2  <1 ms    <1 ms    <1 ms  192.168.30.0
 3  <1 ms    <1 ms    <1 ms  192.168.30.18
 4  *         *         *      Request timed out.
 5  3 ms     2 ms     2 ms  10.11.30.4

Trace complete.
```

Bring up the primary ExpressRoute 1 connectivity for the remainder of the data path analysis.

## Path to the branch VNet

Traceroute output from on-premises Location 1 to a VM in the branch VNet is shown here:

```
C:\Users\rb>tracert 10.11.30.68

Tracing route to 10.11.30.68 over a maximum of 30 hops

 1  <1 ms    <1 ms    <1 ms  10.2.30.3
 2  <1 ms    <1 ms    <1 ms  192.168.30.0
 3  3 ms     2 ms     2 ms  10.11.30.68

Trace complete.
```

## Path to on-premises Location 2

As we discuss in the [control plane analysis](#), the on-premises Location 1 has no visibility to on-premises Location 2 per the network configuration. The following ping results confirm:

```
C:\Users\rb>ping 10.1.31.10

Pinging 10.1.31.10 with 32 bytes of data:

Request timed out.
...
Request timed out.

Ping statistics for 10.1.31.10:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

## Path to the remote VNet

Traceroute output from on-premises Location 1 to a VM in the remote VNet is shown here:

```
C:\Users\rb>tracert 10.17.30.4

Tracing route to 10.17.30.4 over a maximum of 30 hops

 1  <1 ms    <1 ms    <1 ms  10.2.30.3
 2  2 ms     5 ms     7 ms   192.168.30.0
 3  <1 ms    <1 ms    <1 ms  192.168.30.18
 4  *         *         *      Request timed out.
 5  69 ms    70 ms    69 ms  10.17.30.4

Trace complete.
```

## Data path from on-premises Location 2

### Path to the hub VNet

Traceroute output from on-premises Location 2 to a VM in the hub VNet is shown here:

```
C:\Windows\system32>tracert 10.10.30.4

Tracing route to 10.10.30.4 over a maximum of 30 hops

 1  <1 ms    <1 ms    <1 ms  10.1.31.3
 2  <1 ms    <1 ms    <1 ms  192.168.31.4
 3  <1 ms    <1 ms    <1 ms  192.168.31.22
 4  *         *         *      Request timed out.
 5  75 ms    74 ms    74 ms  10.10.30.4

Trace complete.
```

### Path to the spoke VNet

Traceroute output from on-premises Location 2 to a VM in the spoke VNet is shown here:

```
C:\Windows\system32>tracert 10.11.30.4

Tracing route to 10.11.30.4 over a maximum of 30 hops

 1  <1 ms    <1 ms    1 ms   10.1.31.3
 2  <1 ms    <1 ms    <1 ms  192.168.31.0
 3  <1 ms    <1 ms    <1 ms  192.168.31.18
 4  *         *         *      Request timed out.
 5  75 ms    74 ms    74 ms  10.11.30.4

Trace complete.
```

### Path to the branch VNet, on-premises Location 1, and the remote VNet

As we discuss in the [control plane analysis](#), the on-premises Location 1 has no visibility to the branch VNet, to on-premises Location 1, or to the remote VNet per the network configuration.

## Data path from the remote VNet

### Path to the hub VNet

Traceroute output from the remote VNet to a VM in the hub VNet is shown here:

```
C:\Users\rb>tracert 10.10.30.4

Tracing route to 10.10.30.4 over a maximum of 30 hops

 1  65 ms    65 ms    65 ms  10.17.30.36
 2  *         *         *      Request timed out.
 3  69 ms    68 ms    68 ms  10.10.30.4

Trace complete.
```

## Path to the spoke VNet

Traceroute output from the remote VNet to a VM in the spoke VNet is shown here:

```
C:\Users\rb>tracert 10.11.30.4

Tracing route to 10.11.30.4 over a maximum of 30 hops

 1  67 ms    67 ms    67 ms  10.17.30.36
 2  *         *         *      Request timed out.
 3  71 ms    69 ms    69 ms  10.11.30.4

Trace complete.
```

## Path to the branch VNet and on-premises Location 2

As we discuss in the [control plane analysis](#), the remote VNet has no visibility to the branch VNet or to on-premises Location 2 per the network configuration.

## Path to on-premises Location 1

Traceroute output from the remote VNet to a VM in on-premises Location 1 is shown here:

```
C:\Users\rb>tracert 10.2.30.10

Tracing route to 10.2.30.10 over a maximum of 30 hops

 1  67 ms    67 ms    67 ms  10.17.30.36
 2  *         *         *      Request timed out.
 3  *         *         *      Request timed out.
 4  69 ms    69 ms    69 ms  10.2.30.10

Trace complete.
```

# ExpressRoute and site-to-site VPN connectivity in tandem

## Site-to-site VPN over ExpressRoute

You can configure a site-to-site VPN by using ExpressRoute Microsoft peering to privately exchange data between your on-premises network and your Azure VNets. With this configuration, you can exchange data with confidentiality, authenticity, and integrity. The data exchange also is anti-replay. For more information about how to configure a site-to-site IPsec VPN in tunnel mode by using ExpressRoute Microsoft peering, see [Site-to-site VPN over ExpressRoute Microsoft peering](#).

The primary limitation of configuring a site-to-site VPN that uses Microsoft peering is throughput. Throughput over the IPsec tunnel is limited by the VPN gateway capacity. The VPN gateway throughput is lower than ExpressRoute throughput. In this scenario, using the IPsec tunnel for highly secure traffic and using private peering for all other traffic helps optimize the ExpressRoute bandwidth utilization.

## Site-to-site VPN as a secure failover path for ExpressRoute

ExpressRoute serves as a redundant circuit pair to ensure high availability. You can configure geo-redundant ExpressRoute connectivity in different Azure regions. Also, as demonstrated in our test setup, within an Azure region, you can use a site-to-site VPN to create a failover path for your ExpressRoute connectivity. When the same prefixes are advertised over both ExpressRoute and a site-to-site VPN, Azure prioritizes ExpressRoute. To avoid asymmetrical routing between ExpressRoute and the site-to-site VPN, on-premises network configuration should also reciprocate by using ExpressRoute connectivity before it uses site-to-site VPN connectivity.

For more information about how to configure coexisting connections for ExpressRoute and a site-to-site VPN, see [ExpressRoute and site-to-site coexistence](#).

## Extend back-end connectivity to spoke VNets and branch locations

### **Spoke VNet connectivity by using VNet peering**

Hub and spoke VNet architecture is widely used. The hub is a VNet in Azure that acts as a central point of connectivity between your spoke VNets and to your on-premises network. The spokes are VNets that peer with the hub, and which you can use to isolate workloads. Traffic flows between the on-premises datacenter and the hub through an ExpressRoute or VPN connection. For more information about the architecture, see [Implement a hub-spoke network topology in Azure](#).

In VNet peering within a region, spoke VNets can use hub VNet gateways (both VPN and ExpressRoute gateways) to communicate with remote networks.

### **Branch VNet connectivity by using site-to-site VPN**

You might want branch VNets, which are in different regions, and on-premises networks to communicate with each other via a hub VNet. The native Azure solution for this configuration is site-to-site VPN connectivity by using a VPN. An alternative is to use a network virtual appliance (NVA) for routing in the hub.

For more information, see [What is VPN Gateway?](#) and [Deploy a highly available NVA](#).

## Next steps

See the [ExpressRoute FAQ](#) to:

- Learn how many ExpressRoute circuits you can connect to an ExpressRoute gateway.
- Learn how many ExpressRoute gateways you can connect to an ExpressRoute circuit.
- Learn about other scale limits of ExpressRoute.

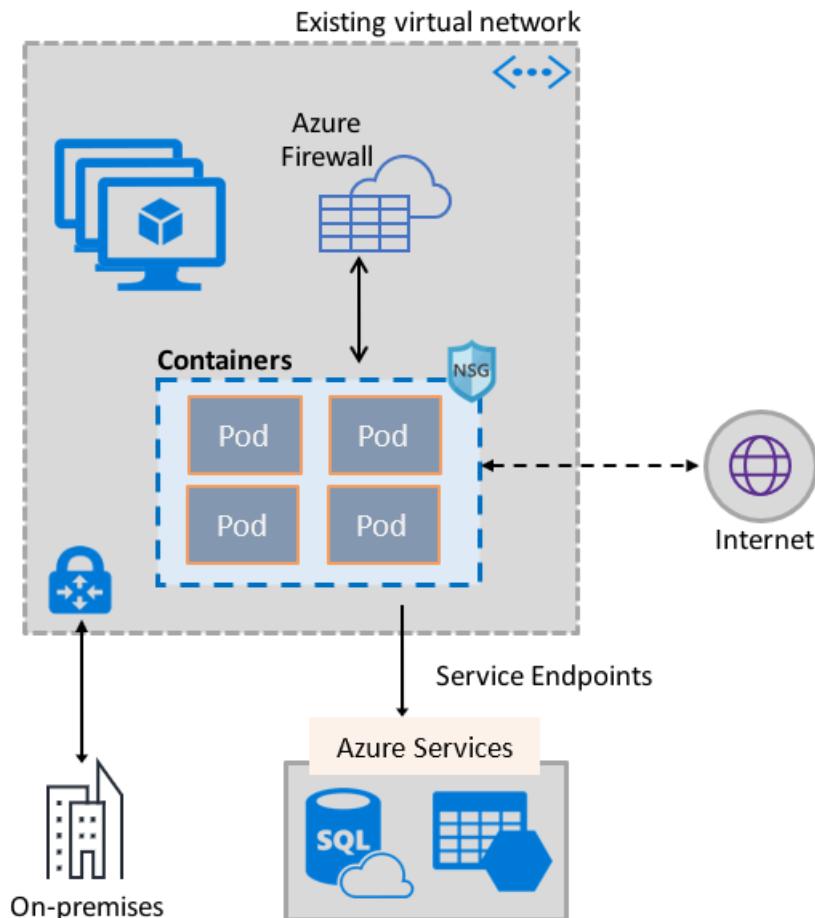
# Enable containers to use Azure Virtual Network capabilities

1/3/2020 • 3 minutes to read • [Edit Online](#)

Bring the rich set of Azure network capabilities to containers, by utilizing the same software defined networking stack that powers virtual machines. The Azure Virtual Network container network interface (CNI) plug-in installs in an Azure Virtual Machine. The plug-in assigns IP addresses from a virtual network to containers brought up in the virtual machine, attaching them to the virtual network, and connecting them directly to other containers and virtual network resources. The plug-in doesn't rely on overlay networks, or routes, for connectivity, and provides the same performance as virtual machines. At a high level, the plug-in provides the following capabilities:

- A virtual network IP address is assigned to every Pod, which could consist of one or more containers.
- Pods can connect to peered virtual networks and to on-premises over ExpressRoute or a site-to-site VPN. Pods are also reachable from peered and on-premises networks.
- Pods can access services such as Azure Storage and Azure SQL Database, that are protected by virtual network service endpoints.
- Network security groups and routes can be applied directly to Pods.
- Pods can be placed directly behind an Azure internal or public Load Balancer, just like virtual machines
- Pods can be assigned a public IP address, which makes them directly accessible from the internet. Pods can also access the internet themselves.
- Works seamlessly with Kubernetes resources such as Services, Ingress controllers, and Kube DNS. A Kubernetes Service can also be exposed internally or externally through the Azure Load Balancer.

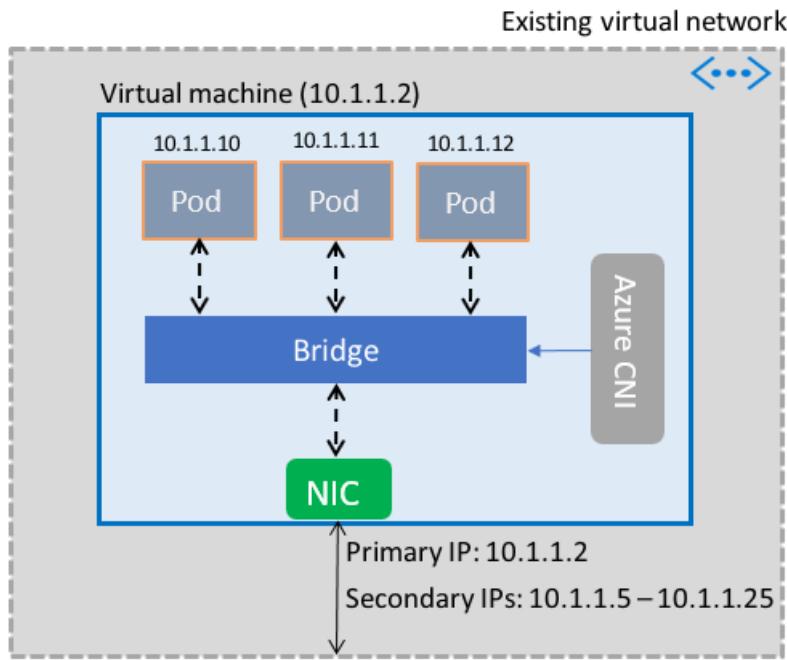
The following picture shows how the plug-in provides Azure Virtual Network capabilities to Pods:



The plug-in supports both Linux and Windows platforms.

## Connecting Pods to a virtual network

Pods are brought up in a virtual machine that is part of a virtual network. A pool of IP addresses for the Pods is configured as secondary addresses on a virtual machine's network interface. Azure CNI sets up the basic Network connectivity for Pods and manages the utilization of the IP addresses in the pool. When a Pod comes up in the virtual machine, Azure CNI assigns an available IP address from the pool and connects the Pod to a software bridge in the virtual machine. When the Pod terminates, the IP address is added back to the pool. The following picture shows how Pods connect to a virtual network:



## Internet access

To enable Pods to access the internet, the plug-in configures *iptables* rules to network address translate (NAT) the internet bound traffic from Pods. The source IP address of the packet is translated to the primary IP address on the virtual machine's network interface. Windows virtual machines automatically source NAT (SNAT) traffic destined to IP addresses outside the subnet the virtual machine is in. Typically, all traffic destined to an IP address outside of the IP range of the virtual network is translated.

## Limits

The plug-in supports up to 250 Pods per virtual machine and up to 16,000 Pods in a virtual network. These limits are different for the [Azure Kubernetes Service](#).

## Using the plug-in

The plug-in can be used in the following ways, to provide basic virtual network attach for Pods or Docker containers:

- **Azure Kubernetes Service:** The plug-in is integrated into the Azure Kubernetes Service (AKS), and can be used by choosing the *Advanced Networking* option. Advanced Networking lets you deploy a Kubernetes cluster in an existing, or a new, virtual network. To learn more about Advanced Networking and the steps to set it up, see [Network configuration in AKS](#).
- **AKS-Engine:** AKS-Engine is a tool that generates an Azure Resource Manager template for the deployment of a Kubernetes cluster in Azure. For detailed instructions, see [Deploy the plug-in for AKS-Engine Kubernetes clusters](#).

- **Creating your own Kubernetes cluster in Azure:** The plug-in can be used to provide basic networking for Pods in Kubernetes clusters that you deploy yourself, without relying on AKS, or tools like the AKS-Engine. In this case, the plug-in is installed and enabled on every virtual machine in a cluster. For detailed instructions, see [Deploy the plug-in for a Kubernetes cluster that you deploy yourself](#).
- **Virtual network attach for Docker containers in Azure:** The plug-in can be used in cases where you don't want to create a Kubernetes cluster, and would like to create Docker containers with virtual network attach, in virtual machines. For detailed instructions, see [Deploy the plug-in for Docker](#).

## Next steps

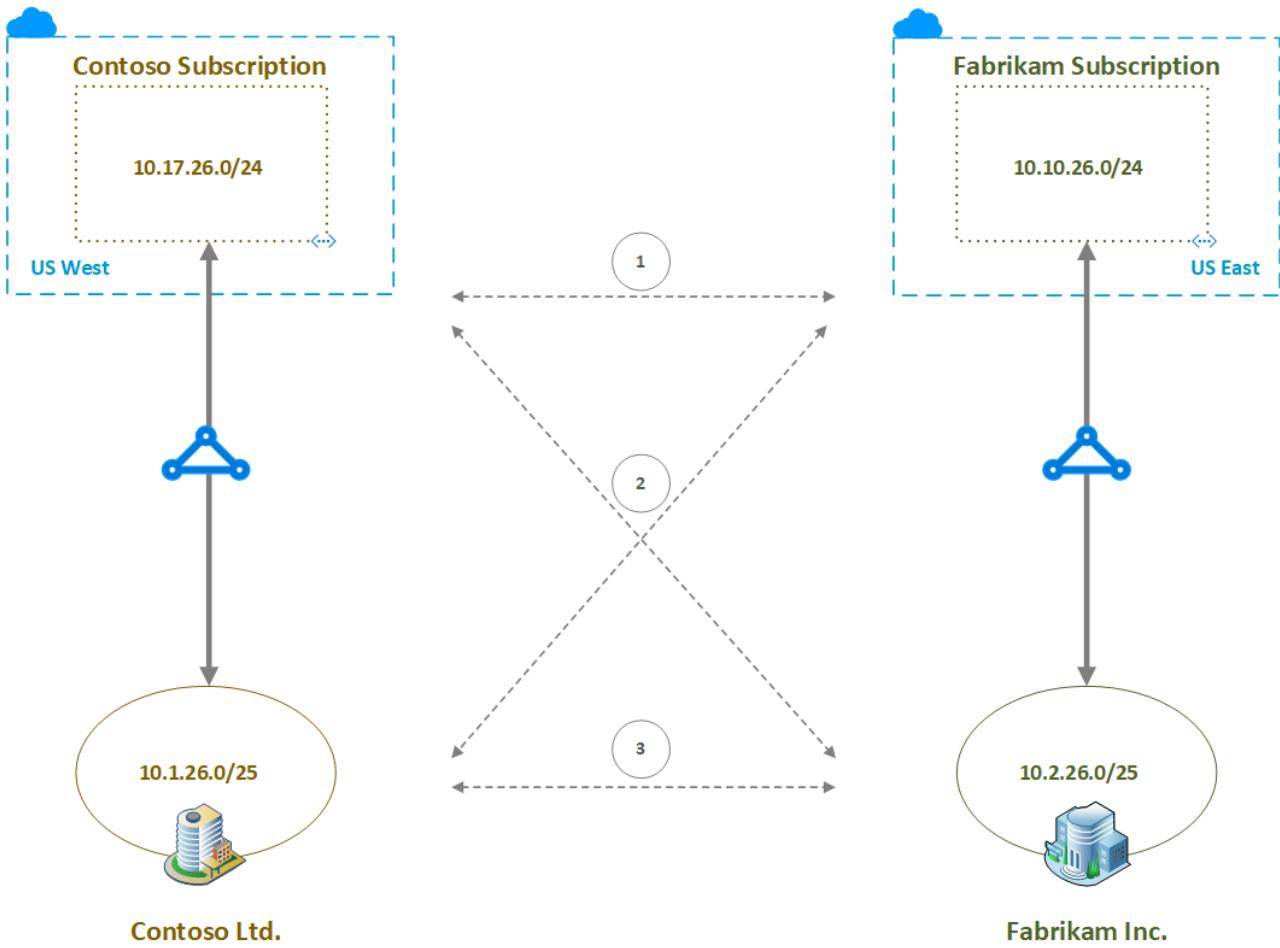
[Deploy the plug-in](#) for Kubernetes clusters or Docker containers

# Cross-network connectivity

1/3/2020 • 4 minutes to read • [Edit Online](#)

Fabrikam Inc. has a large physical presence and Azure deployment in East US. Fabrikam has back-end connectivity between its on-premises and Azure deployments via ExpressRoute. Similarly, Contoso Ltd. has a presence and Azure deployment in West US. Contoso has back-end connectivity between its on-premises and Azure deployments via ExpressRoute.

Fabrikam Inc. acquires Contoso Ltd. Following the merger, Fabrikam wants to interconnect the networks. The following figure illustrates the scenario:



The dashed arrows in the middle of the above figure indicate the desired network interconnections. Specifically, there are three types cross connections desired: 1) Fabrikam and Contoso VNets cross connect, 2) Cross regional on-premises and VNets cross connects (that is, connecting Fabrikam on-premises network to Contoso VNet and connecting Contoso on-premises network to Fabrikam VNet), and 3) Fabrikam and Contoso on-premises network cross connect.

The following table shows the route table of the private peering of the ExpressRoute of Contoso Ltd., before the merger.

## Route table (Primary)

AzurePrivatePeering - Contoso-ER



[Download](#) [Show secondary](#)

**i** Showing only top 200 primary records, click Download above to see all.

NETWORK	NEXT HOP	LOCPRF	WEIGHT	PATH
10.1.26.0/25	192.168.26.17		0	65020
10.17.26.0/24	10.17.26.140		0	65515
	10.17.26.141		0	65515

The following table shows the effective routes of a VM in the Contoso subscription, before the merger. Per the table, the VM on the VNet is aware of the VNet address space and the Contoso on-premises network, apart from the default ones.

[Download](#) [Refresh](#)

**i** Showing only top 200 records, click Download above to see all.

Scope Network interface (Contoso-VM01-nic)

### Effective routes

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP TYPE IP ADDRESS
Default	Active	10.17.26.0/24	Virtual network	-
Virtual network gateway	Active	10.1.26.0/25	Virtual network gateway	10.3.129.53
Virtual network gateway	Active	10.1.26.0/25	Virtual network gateway	10.3.129.52
Default	Active	0.0.0.0/0	Internet	-
Default	Active	10.0.0.0/8	None	-
Default	Active	100.64.0.0/10	None	-
Default	Active	192.168.0.0/16	None	-

The following table shows the route table of the private peering of the ExpressRoute of Fabrikam Inc., before the merger.

## Route table (Primary)

AzurePrivatePeering - Fabrikam-ER



[Download](#) [Show secondary](#)

**i** Showing only top 200 primary records, click Download above to see all.

NETWORK	NEXT HOP	LOCPRF	WEIGHT	PATH
10.2.26.0/25	192.168.26.17		0	65021
10.10.26.0/24	10.10.26.140		0	65515
	10.10.26.141		0	65515

The following table shows the effective routes of a VM in the Fabrikam subscription, before the merger. Per the table, the VM on the VNet is aware of the VNet address space and the Fabrikam on-premises network, apart from the default ones.

[Download](#)  [Refresh](#)

Showing only top 200 records, click Download above to see all.

Scope

Network interface (Fabrikam-VM01-nic)

## Effective routes

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP TYPE IP ADDRESS
Default	Active	10.10.26.0/24	Virtual network	-
Virtual network gateway	Active	10.2.26.0/25	Virtual network gateway	10.3.129.24
Virtual network gateway	Active	10.2.26.0/25	Virtual network gateway	10.3.129.25
Default	Active	0.0.0.0/0	Internet	-
Default	Active	10.0.0.0/8	None	-
Default	Active	100.64.0.0/10	None	-
Default	Active	192.168.0.0/16	None	-

In this article, let's go through step by step and discuss how to achieve the desired cross connections using the following Azure network features:

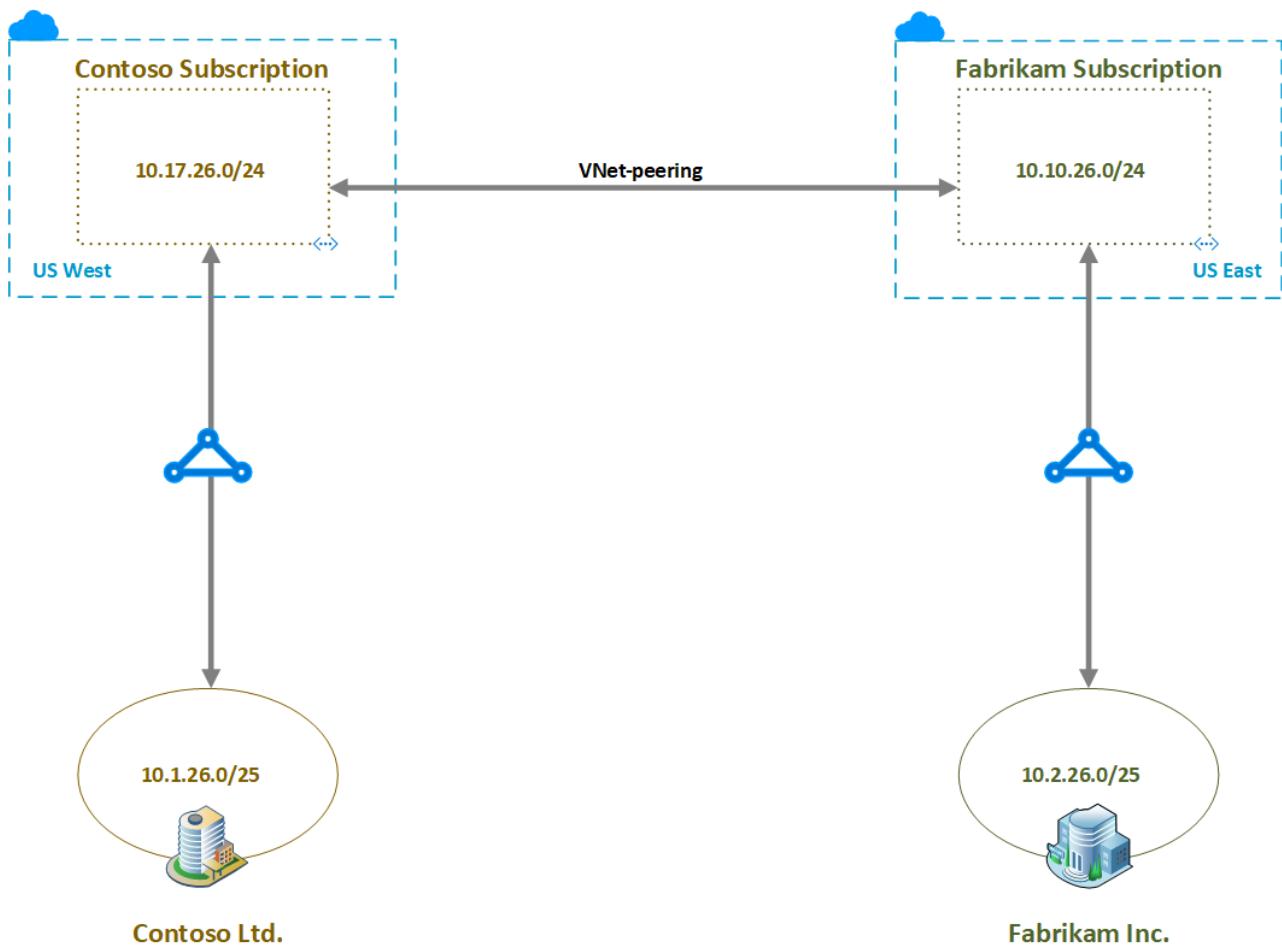
- [Virtual network peering](#)
- [Virtual network ExpressRoute connection](#)
- [Global Reach](#)

## Cross connecting VNets

Virtual network peering (VNet peering) provides the most optimal and the best network performance when connecting two virtual networks. VNet peering supports peering two VNets both within the same Azure region (commonly called VNet peering) and in two different Azure regions (commonly called Global VNet peering).

Let's configure Global VNet peering between the VNets in Contoso and Fabrikam Azure subscriptions. For how to create the virtual network peering between two the virtual networks, see [Create a virtual network peering](#) article.

The following picture shows the network architecture after configuring Global VNet peering.



The following table shows the routes known to the Contoso subscription VM. Pay attention to the last entry of the table. This entry is the result of cross connecting the virtual networks.

Effective routes					
Source	State	Address prefixes	Next hop type	Next hop type IP address	
Default	Active	10.17.26.0/24	Virtual network	-	
Virtual network gateway	Active	10.1.26.0/25	Virtual network gateway	10.3.129.53	
Virtual network gateway	Active	10.1.26.0/25	Virtual network gateway	10.3.129.52	
Default	Active	0.0.0.0/0	Internet	-	
Default	Active	10.0.0.0/8	None	-	
Default	Active	100.64.0.0/10	None	-	
Default	Active	192.168.0.0/16	None	-	
Default	Active	10.10.26.0/24	VNetGlobalPeering	-	

The following table shows the routes known to the Fabrikam subscription VM. Pay attention to the last entry of the table. This entry is the result of cross connecting the virtual networks.

[Download](#) [Refresh](#)

Showing only top 200 records, click Download above to see all.

Scope

Network interface (Fabrikam-VM01-nic)

## Effective routes

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP TYPE IP ADDRESS
Default	Active	10.10.26.0/24	Virtual network	-
Virtual network gateway	Active	10.2.26.0/25	Virtual network gateway	10.3.129.24
Virtual network gateway	Active	10.2.26.0/25	Virtual network gateway	10.3.129.25
Default	Active	0.0.0.0/0	Internet	-
Default	Active	10.0.0.0/8	None	-
Default	Active	100.64.0.0/10	None	-
Default	Active	192.168.0.0/16	None	-
Default	Active	10.17.26.0/24	VNetGlobalPeering	-

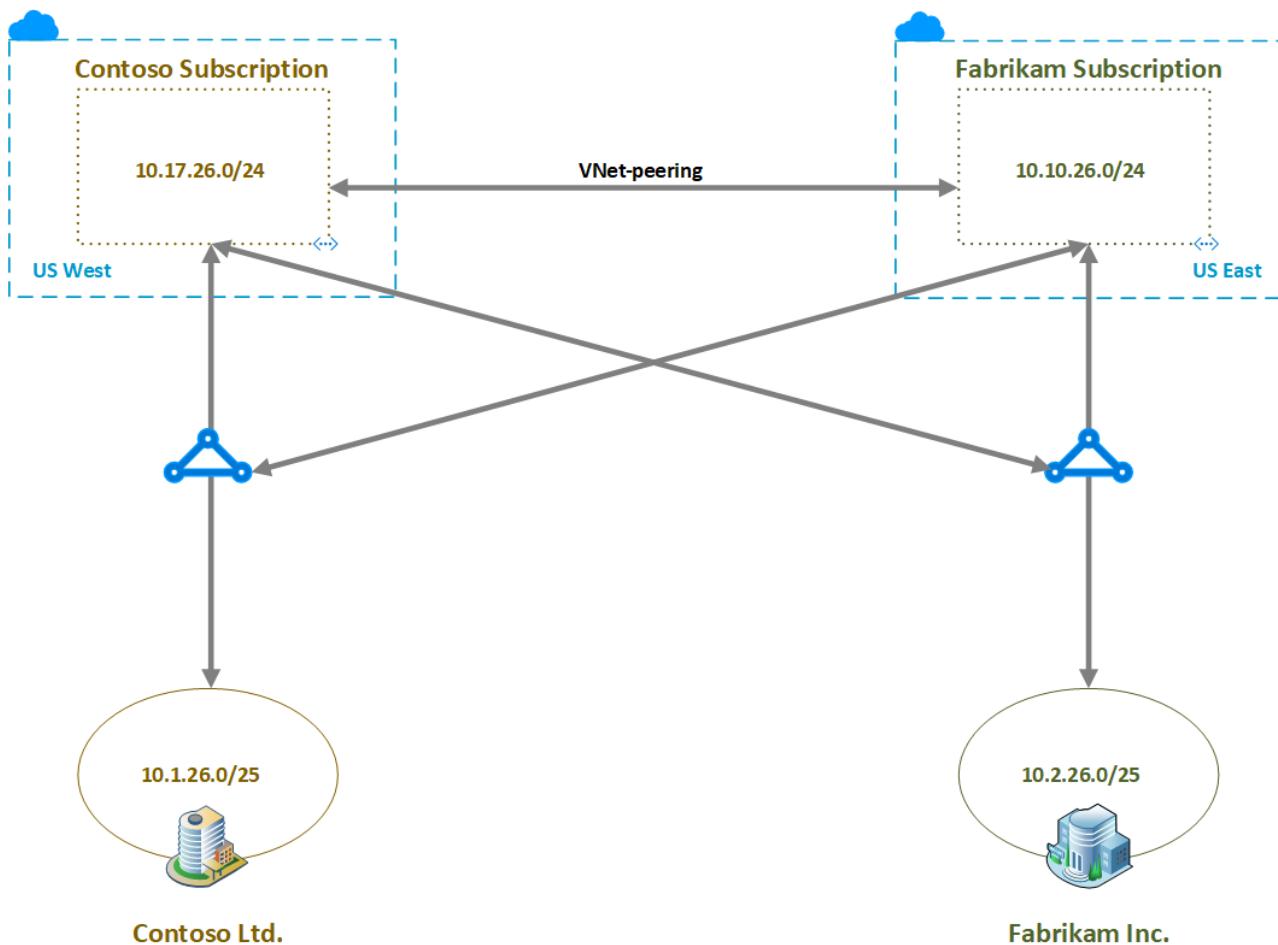
VNet peering directly links two virtual networks (see there are no next hop for *VNetGlobalPeering* entry in the above two tables)

## Cross connecting VNets to the on-premises networks

We can connect an ExpressRoute circuit to multiple virtual networks. See [Subscription and service limits](#) for the maximum number of virtual networks that can be connected to an ExpressRoute circuit.

Let's connect Fabrikam ExpressRoute circuit to Contoso subscription VNet and similarly Contoso ExpressRoute circuit to Fabrikam subscription VNet to enable cross connectivity between virtual networks and the on-premises networks. To connect a virtual network to an ExpressRoute circuit in a different subscription, we need to create and use an authorization. See the article: [Connect a virtual network to an ExpressRoute circuit](#).

The following picture shows the network architecture after configuring the ExpressRoute cross connectivity to the virtual networks.



The following table shows the route table of the private peering of the ExpressRoute of Contoso Ltd., after cross connecting virtual networks to the on-premises networks via ExpressRoute. See that the route table has routes belonging to both the virtual networks.

Route table (Primary)						
AzurePrivatePeering - Contoso-ER						
NETWORK	NEXT HOP	LOCPRF	WEIGHT	PATH		
10.1.26.0/25	192.168.26.17		0	65020		
10.10.26.0/24	10.10.26.141		0	65515		
	10.10.26.140		0	65515		
10.17.26.0/24	10.17.26.140		0	65515		
	10.17.26.141		0	65515		

The following table shows the route table of the private peering of the ExpressRoute of Fabrikam Inc., after cross connecting virtual networks to the on-premises networks via ExpressRoute. See that the route table has routes belonging to both the virtual networks.

## Route table (Primary)

AzurePrivatePeering - Fabrikam-ER



[Download](#)

[Show secondary](#)

**i** Showing only top 200 primary records, click Download above to see all.

NETWORK	NEXT HOP	LOCPRF	WEIGHT	PATH
10.2.26.0/25	192.168.26.17	0	65021	
10.10.26.0/24	10.10.26.140	0	65515	
	10.10.26.141	0	65515	
10.17.26.0/24	10.17.26.140	0	65515	
	10.17.26.141	0	65515	

The following table shows the routes known to the Contoso subscription VM. Pay attention to *Virtual network gateway* entries of the table. The VM sees routes for both the on-premises networks.

[Download](#) [Refresh](#)

**i** Showing only top 200 records, click Download above to see all.

Scope

Network interface (Contoso-VM01-nic)

### Effective routes

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP TYPE IP ADDRESS
Default	Active	10.17.26.0/24	Virtual network	-
Virtual network gateway	Active	10.2.26.0/25	Virtual network gateway	10.3.129.24
Virtual network gateway	Active	10.2.26.0/25	Virtual network gateway	10.3.129.25
Virtual network gateway	Active	10.1.26.0/25	Virtual network gateway	10.3.129.53
Virtual network gateway	Active	10.1.26.0/25	Virtual network gateway	10.3.129.52
Default	Active	0.0.0.0/0	Internet	-
Default	Active	10.0.0.0/8	None	-
Default	Active	100.64.0.0/10	None	-
Default	Active	192.168.0.0/16	None	-
Default	Active	10.10.26.0/24	VNetGlobalPeering	-

The following table shows the routes known to the Fabrikam subscription VM. Pay attention to *Virtual network gateway* entries of the table. The VM sees routes for both the on-premises networks.

[Download](#) [Refresh](#)

Showing only top 200 records, click Download above to see all.

Scope

Network interface (Fabrikam-VM01-nic)

## Effective routes

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP TYPE IP ADDRESS
Default	Active	10.10.26.0/24	Virtual network	-
Virtual network gateway	Active	10.1.26.0/25	Virtual network gateway	10.3.129.53
Virtual network gateway	Active	10.1.26.0/25	Virtual network gateway	10.3.129.52
Virtual network gateway	Active	10.2.26.0/25	Virtual network gateway	10.3.129.24
Virtual network gateway	Active	10.2.26.0/25	Virtual network gateway	10.3.129.25
Default	Active	0.0.0.0/0	Internet	-
Default	Active	10.0.0.0/8	None	-
Default	Active	100.64.0.0/10	None	-
Default	Active	192.168.0.0/16	None	-
Default	Active	10.17.26.0/24	VNetGlobalPeering	-

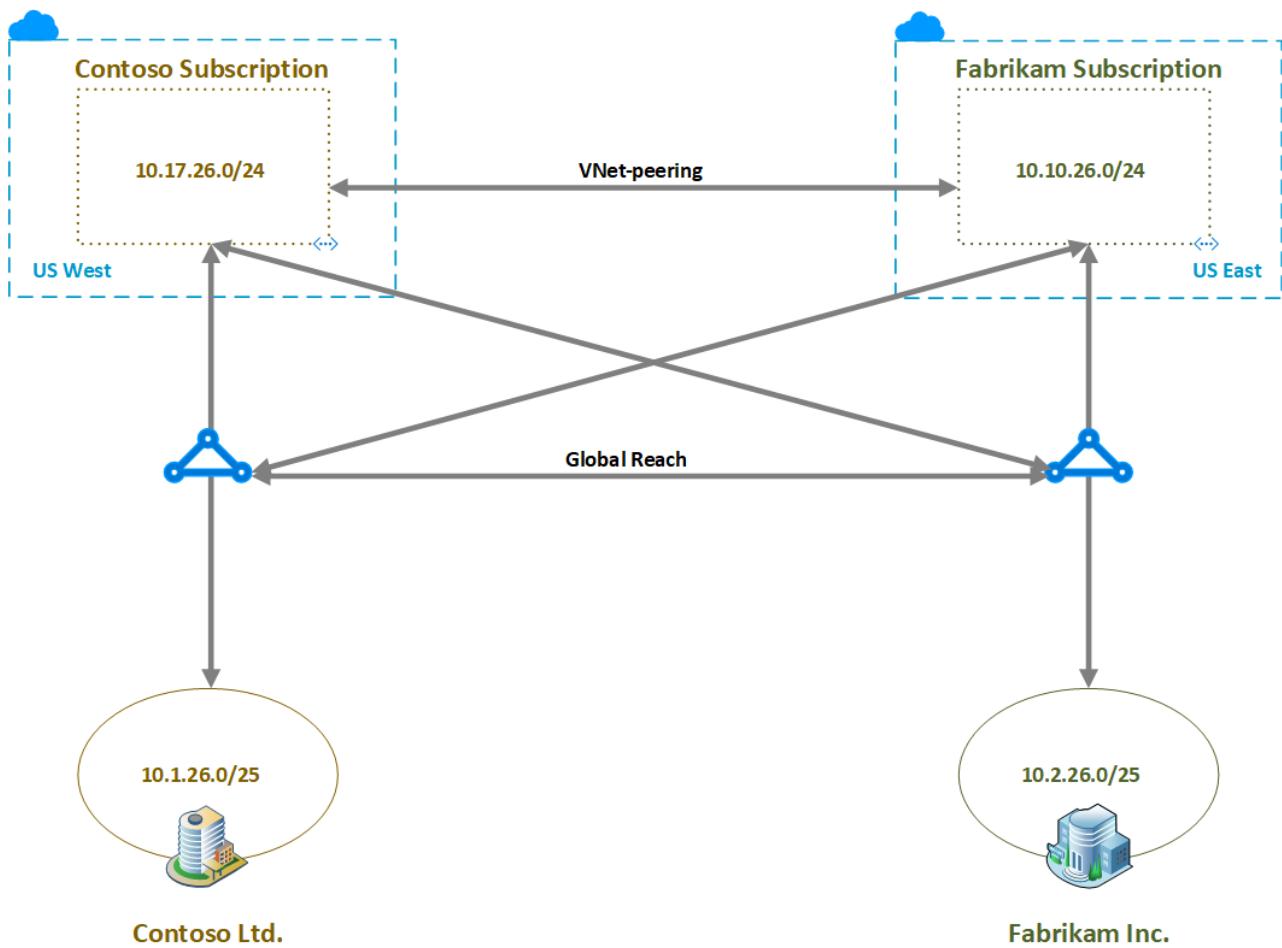
## NOTE

In either the Fabrikam and/or Contoso subscriptions you can also have spoke VNets to the respective hub VNet (a hub and spoke design is not illustrated in the architecture diagrams in this article). The cross connections between the hub VNet gateways to ExpressRoute will also allow communication between East and West hubs and spokes.

## Cross connecting on-premises networks

ExpressRoute Global Reach provides connectivity between on-premises networks that are connected to different ExpressRoute circuits. Let's configure Global Reach between Contoso and Fabrikam ExpressRoute circuits. Because the ExpressRoute circuits are in different subscriptions, we need to create and use an authorization. See [Configure ExpressRoute Global Reach](#) article for step by step guidance.

The following picture shows the network architecture after configuring Global Reach.



The following table shows the route table of the private peering of the ExpressRoute of Contoso Ltd., after configuring Global Reach. See that the route table has routes belonging to both the on-premises networks.

Route table (Primary)							
AzurePrivatePeering - Contoso-ER							
<a href="#">Download</a>		<a href="#">Show secondary</a>					
Showing only top 200 primary records, click Download above to see all.							
NETWORK	NEXT HOP	LOCPRF	WEIGHT	PATH			
10.1.26.0/25	192.168.26.17		0	65020			
10.2.26.0/25	192.168.26.51	10	0	65021			
10.10.26.0/24	192.168.26.51	10	0	65515			
	10.10.26.141		0	65515			
	10.10.26.140		0	65515			
10.17.26.0/24	192.168.26.51	10	0	65515			
	10.17.26.140		0	65515			
	10.17.26.141		0	65515			

The following table shows the route table of the private peering of the ExpressRoute of Fabrikam Inc., after configuring Global Reach. See that the route table has routes belonging to both the on-premises networks.

## Route table (Primary)

AzurePrivatePeering - Fabrikam-ER



[Download](#)

[Show secondary](#)

**i** Showing only top 200 primary records, click Download above to see all.

NETWORK	↑↓	NEXT HOP	↑↓	LOCPRF	↑↓	WEIGHT	↑↓	PATH	↑↓
10.1.26.0/25		192.168.26.49		10		0		65020	
10.2.26.0/25		192.168.26.17				0		65021	
10.10.26.0/24		192.168.26.49		10		0		65515	
		10.10.26.140				0		65515	
		10.10.26.141				0		65515	
10.17.26.0/24		192.168.26.49		10		0		65515	
		10.17.26.140				0		65515	
		10.17.26.141				0		65515	

## Next steps

See [virtual network FAQ](#), for any further questions on VNet and VNet-peering. See [ExpressRoute FAQ](#) for any further questions on ExpressRoute and virtual network connectivity.

Global Reach is rolled out on a country/region by country/region basis. To see if Global Reach is available in the countries/regions that you want, see [ExpressRoute Global Reach](#).

# Virtual network peering

2/28/2020 • 6 minutes to read • [Edit Online](#)

Virtual network peering enables you to seamlessly connect networks in [Azure Virtual Network](#). The virtual networks appear as one for connectivity purposes. The traffic between virtual machines uses the Microsoft backbone infrastructure. Like traffic between virtual machines in the same network, traffic is routed through Microsoft's *private* network only.

Azure supports the following types of peering:

- Virtual network peering: Connect virtual networks within the same Azure region.
- Global virtual network peering: Connecting virtual networks across Azure regions.

The benefits of using virtual network peering, whether local or global, include:

- 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.
- The ability to transfer data between virtual networks across Azure subscriptions, Azure Active Directory tenants, deployment models, and Azure regions.
- The ability to peer virtual networks created through the Azure Resource Manager.
- The ability to peer a virtual network created through Resource Manager to one created through the classic deployment model. To learn more about Azure deployment models, see [Understand Azure deployment models](#).
- No downtime to resources in either virtual network when creating the peering, or after the peering is created.

Network traffic between peered virtual networks is private. Traffic between the virtual networks is kept on the Microsoft backbone network. No public Internet, gateways, or encryption is required in the communication between the virtual networks.

## Connectivity

For peered virtual networks, 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 in the same region 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, proportionate to its size. 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 Microsoft backbone infrastructure, not through a gateway or over the public Internet.

You can apply network security groups in either virtual network to block access to other virtual networks or subnets. When configuring virtual network peering, either open or close the network security group rules between the virtual networks. If you open full connectivity between peered virtual networks, you can apply network security groups to block or deny specific access. Full connectivity is the default option. To learn more about network security groups, see [Security groups](#).

## Service chaining

Service chaining enables you to direct traffic from one virtual network to a virtual appliance or gateway in a peered network through user-defined routes.

To enable service chaining, configure user-defined routes that point to virtual machines in peered virtual networks as the *next hop* IP address. User-defined routes could also point to virtual network gateways to enable service chaining.

You can deploy hub-and-spoke networks, where the hub virtual network hosts infrastructure components such as a network virtual appliance or VPN gateway. All the spoke virtual networks can then peer with the hub virtual network. Traffic flows through network virtual appliances or VPN gateways in the hub virtual network.

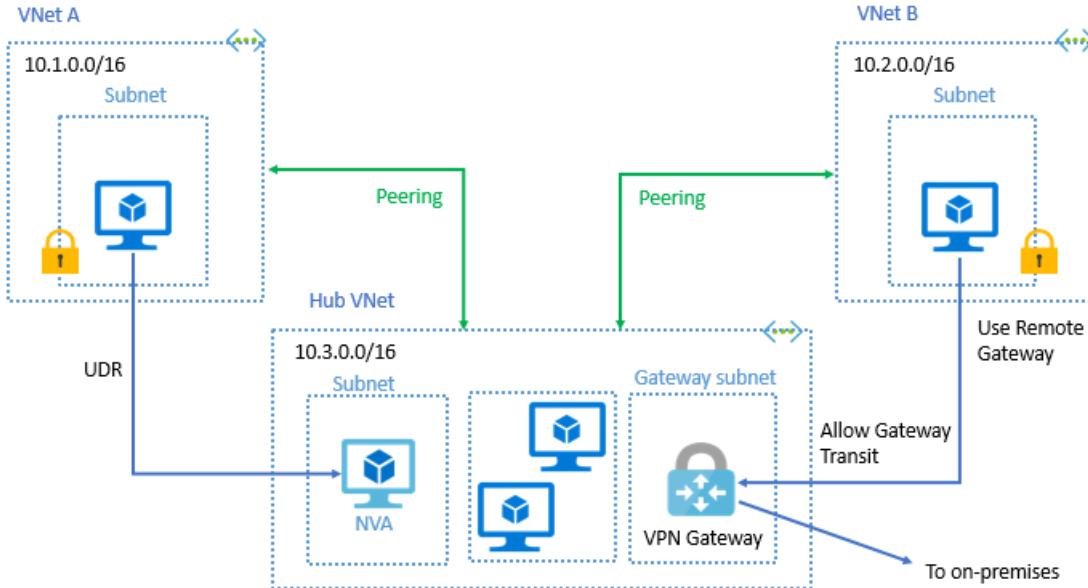
Virtual network peering enables the next hop in a user-defined route to be the IP address of a virtual machine in the peered virtual network, or a VPN gateway. You can't route between virtual networks with a user-defined route that specifies an Azure ExpressRoute gateway as the next hop type. To learn more about user-defined routes, see [User-defined routes overview](#). To learn how to create a hub and spoke network topology, see [Hub-spoke network topology in Azure](#).

## Gateways and on-premises connectivity

Each virtual network, including a peered virtual network, can have its own gateway. A virtual network can use its gateway to connect to an on-premises network. You can also configure [virtual network-to-virtual network connections](#) by using gateways, even for peered virtual networks.

When you configure both options for virtual network interconnectivity, the traffic between the virtual networks flows through the peering configuration. The traffic uses the Azure backbone.

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 can't have its own gateway. A virtual network has only one gateway. The gateway is either a local or remote gateway in the peered virtual network, as shown in the following diagram:



Both virtual network peering and global virtual network peering support gateway transit.

Gateway transit between virtual networks created through different deployment models is supported. The gateway must be in the virtual network in the Resource Manager model. To learn more about using a gateway for transit, see [Configure a VPN gateway for transit in a virtual network peering](#).

When you peer virtual networks that share a single Azure ExpressRoute connection, the traffic between them goes through the peering relationship. That traffic uses the Azure backbone network. You can still use local gateways in each virtual network to connect to the on-premises circuit. Otherwise, you can use a shared gateway and configure transit for on-premises connectivity.

## Troubleshoot

To confirm that virtual networks are peered, you can check effective routes. Check routes for a network interface in any subnet in a virtual network. If a virtual network peering exists, all subnets within the virtual network have routes with next hop type *VNet peering*, for each address space in each peered virtual network. For more information, see [Diagnose a virtual machine routing problem](#).

You can also troubleshoot connectivity to a virtual machine in a peered virtual network using Azure Network Watcher. A connectivity check lets you see how traffic is routed from a source virtual machine's network interface to a destination virtual machine's network interface. For more information, see [Troubleshoot connections with Azure Network Watcher using the Azure portal](#).

You can also try the [Troubleshoot virtual network peering issues](#).

## Constraints for peered virtual networks

The following constraints apply only when virtual networks are globally peered:

- Resources in one virtual network can't communicate with the front-end IP address of a Basic Internal Load Balancer (ILB) in a globally peered virtual network.
- Some services that use a Basic load balancer don't work over global virtual network peering. For more information, see [What are the constraints related to Global VNet Peering and Load Balancers?](#).

For more information, see [Requirements and constraints](#). To learn more about the supported number of peerings, see [Networking limits](#).

## Permissions

To learn about permissions required to create a virtual network peering, see [Permissions](#).

## Pricing

There's a nominal charge for ingress and egress traffic that uses a virtual network peering connection. For more information, see [Virtual Network pricing](#).

Gateway Transit is a peering property that enables a virtual network to utilize a VPN/ExpressRoute gateway in a peered virtual network. Gateway transit works for both cross premises and network-to-network connectivity. Traffic to the gateway (ingress or egress) in the peered virtual network incurs virtual network peering charges on the spoke VNet (or non-gateway VNet). For more information, see [VPN Gateway pricing](#) for VPN gateway charges and ExpressRoute Gateway pricing for ExpressRoute gateway charges.

### NOTE

A previous version of this document stated that virtual network peering charges would not apply on the spoke VNet (or non-gateway VNet) with Gateway Transit. It now reflects accurate pricing per the pricing page.

## Next steps

- You can create a peering between two virtual networks. The networks can belong to the same subscription, different deployment models in the same subscription, or different subscriptions. Complete a tutorial for one of the following scenarios:

AZURE DEPLOYMENT MODEL	SUBSCRIPTION
Both Resource Manager	Same
	Different
One Resource Manager, one classic	Same
	Different

- To learn how to create a hub and spoke network topology, see [Hub-spoke network topology in Azure](#).
- To learn about all virtual network peering settings, see [Create, change, or delete a virtual network peering](#).
- For answers to common virtual network peering and global virtual network peering questions, see [VNet Peering](#).

# Virtual network integration for Azure services

2/6/2020 • 2 minutes to read • [Edit Online](#)

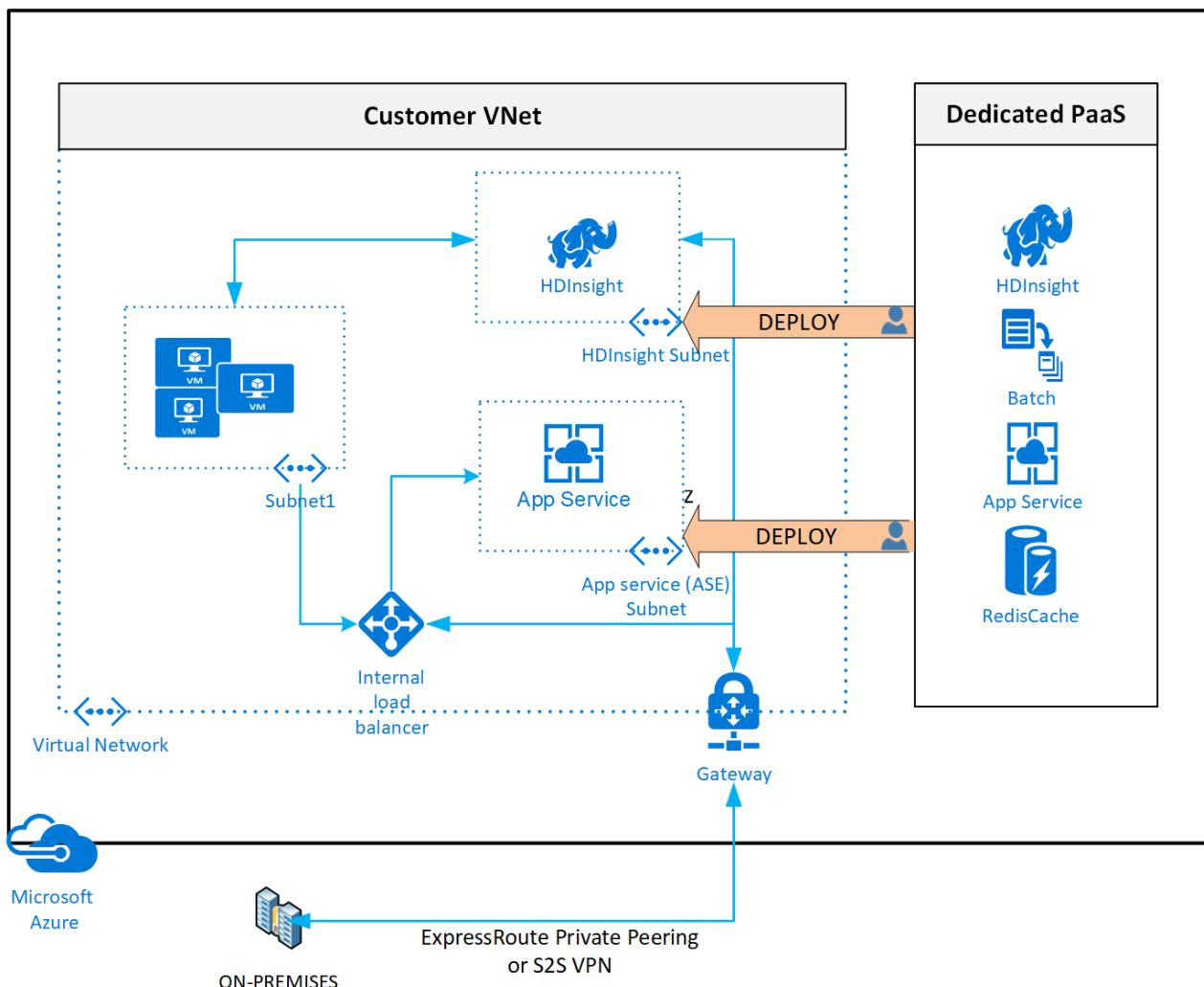
Integrating Azure services to an Azure virtual network enables private access to the service from virtual machines or compute resources in the virtual network. You can integrate Azure services in your virtual network with the following options:

- Deploying dedicated instances of the service into a virtual network. The services can then be privately accessed within the virtual network and from on-premises networks.
- Using [Private Link](#) to access privately a specific instance of the service from your virtual network and from on-premises networks.

You can also access the service using public endpoints by extending a virtual network to the service, through [service endpoints](#). Service endpoints allow service resources to be secured to the virtual network.

## Deploy Azure services into virtual networks

When you deploy dedicated Azure services in a [virtual network](#), you can communicate with the service resources privately, through private IP addresses.



Deploying services within a virtual network provides the following capabilities:

- Resources within the virtual network can communicate with each other privately, through private IP addresses.

Example, directly transferring data between HDInsight and SQL Server running on a virtual machine, in the virtual network.

- On-premises resources can access resources in a virtual network using private IP addresses over a [Site-to-Site VPN \(VPN Gateway\)](#) or [ExpressRoute](#).
- Virtual networks can be [peered](#) to enable resources in the virtual networks to communicate with each other, using private IP addresses.
- Service instances in a virtual network are typically fully managed by the Azure service. This includes monitoring the health of the resources and scaling with load.
- Service instances are deployed into a subnet in a virtual network. Inbound and outbound network access for the subnet must be opened through [network security groups](#), per guidance provided by the service.
- Certain services also impose restrictions on the subnet they are deployed in, limiting the application of policies, routes or combining VMs and service resources within the same subnet. Check with each service on the specific restrictions as they may change over time. Examples of such services are Azure NetApp Files, Dedicated HSM, Azure Container Instances, App Service.
- Optionally, services might require a [delegated subnet](#) as an explicit identifier that a subnet can host a particular service. By delegating, services get explicit permissions to create service-specific resources in the delegated subnet.
- See an example of a REST API response on a [virtual network with a delegated subnet](#). A comprehensive list of services that are using the delegated subnet model can be obtained via the [Available Delegations API](#).

### Services that can be deployed into a virtual network

CATEGORY	SERVICE	DEDICATED <sup>1</sup> SUBNET
Compute	Virtual machines: <a href="#">Linux</a> or <a href="#">Windows</a> <a href="#">Virtual machine scale sets</a> <a href="#">Cloud Service</a> : Virtual network (classic) only <a href="#">Azure Batch</a>	No No No No <sup>2</sup>
Network	<a href="#">Application Gateway - WAF</a> <a href="#">VPN Gateway</a> <a href="#">Azure Firewall</a> <a href="#">Network Virtual Appliances</a>	Yes Yes Yes No
Data	<a href="#">RedisCache</a> <a href="#">Azure SQL Database Managed Instance</a>	Yes Yes
Analytics	<a href="#">Azure HDInsight</a> <a href="#">Azure Databricks</a>	No <sup>2</sup> No <sup>2</sup>
Identity	<a href="#">Azure Active Directory Domain Services</a>	No
Containers	<a href="#">Azure Kubernetes Service (AKS)</a> <a href="#">Azure Container Instance (ACI)</a> <a href="#">Azure Container Service Engine</a> with Azure Virtual Network CNI <a href="#">plug-in</a>	No <sup>2</sup> Yes No
Web	<a href="#">API Management</a> <a href="#">App Service Environment</a> <a href="#">Azure Logic Apps</a>	Yes Yes Yes
Hosted	<a href="#">Azure Dedicated HSM</a> <a href="#">Azure NetApp Files</a>	Yes Yes

<sup>1</sup> 'Dedicated' implies that only service specific resources can be deployed in this subnet and cannot be combined with customer VM/VMSSs

<sup>2</sup> It is recommended as a best practice to have these services in a dedicated subnet, but not a mandatory requirement imposed by the service.

# IP address types and allocation methods in Azure

2/4/2020 • 12 minutes to read • [Edit Online](#)

You can assign IP addresses to Azure resources to communicate with other Azure resources, your on-premises network, and the Internet. There are two types of IP addresses you can use in Azure:

- **Public IP addresses:** Used for communication with the Internet, including Azure public-facing services.
- **Private IP addresses:** Used for communication within an Azure virtual network (VNet), and your on-premises network, when you use a VPN gateway or ExpressRoute circuit to extend your network to Azure.

You can also create a contiguous range of static public IP addresses through a public IP prefix. [Learn about a public IP prefix](#).

## NOTE

Azure has two different deployment models for creating and working with resources: [Resource Manager](#) and [classic](#). This article covers using the Resource Manager deployment model, which Microsoft recommends for most new deployments instead of the [classic deployment model](#).

If you are familiar with the classic deployment model, check the [differences in IP addressing between classic and Resource Manager](#).

## Public IP addresses

Public IP addresses allow Internet resources to communicate inbound to Azure resources. Public IP addresses also enable Azure resources to communicate outbound to Internet and public-facing Azure services with an IP address assigned to the resource. The address is dedicated to the resource, until it is unassigned by you. If a public IP address is not assigned to a resource, the resource can still communicate outbound to the Internet, but Azure dynamically assigns an available IP address that is not dedicated to the resource. For more information about outbound connections in Azure, see [Understand outbound connections](#).

In Azure Resource Manager, a [public IP](#) address is a resource that has its own properties. Some of the resources you can associate a public IP address resource with are:

- Virtual machine network interfaces
- Internet-facing load balancers
- VPN gateways
- Application gateways
- Azure Firewall

## IP address version

Public IP addresses are created with an IPv4 or IPv6 address.

## SKU

Public IP addresses are created with one of the following SKUs:

## **IMPORTANT**

Matching SKUs must be used for load balancer and public IP resources. You can't have a mixture of basic SKU resources and standard SKU resources. You can't attach standalone virtual machines, virtual machines in an availability set resource, or a virtual machine scale set resources to both SKUs simultaneously. New designs should consider using Standard SKU resources. Please review [Standard Load Balancer](#) for details.

### **Basic**

All public IP addresses created before the introduction of SKUs are Basic SKU public IP addresses. With the introduction of SKUs, you have the option to specify which SKU you would like the public IP address to be. Basic SKU addresses are:

- Assigned with the static or dynamic allocation method.
- Have an adjustable inbound originated flow idle timeout of 4-30 minutes, with a default of 4 minutes, and fixed outbound originated flow idle timeout of 4 minutes.
- Are open by default. Network security groups are recommended but optional for restricting inbound or outbound traffic.
- Assigned to any Azure resource that can be assigned a public IP address, such as network interfaces, VPN Gateways, Application Gateways, and Internet-facing load balancers.
- Do not support Availability Zone scenarios. You need to use Standard SKU public IP for Availability Zone scenarios. To learn more about availability zones, see [Availability zones overview](#) and [Standard Load Balancer and Availability Zones](#).

### **Standard**

Standard SKU public IP addresses are:

- Always use static allocation method.
- Have an adjustable inbound originated flow idle timeout of 4-30 minutes, with a default of 4 minutes, and fixed outbound originated flow idle timeout of 4 minutes.
- Are secure by default and closed to inbound traffic. You must explicit whitelist allowed inbound traffic with a [network security group](#).
- Assigned to network interfaces, Standard public Load Balancers, or Application Gateways. For more information about Standard Load Balancer, see [Azure Standard Load Balancer](#).
- Zone redundant by default and optionally zonal (can be created zonal and guaranteed in a specific availability zone). To learn more about availability zones, see [Availability zones overview](#) and [Standard Load Balancer and Availability Zones](#).

### **NOTE**

Inbound communication with a Standard SKU resource fails until you create and associate a [network security group](#) and explicitly allow the desired inbound traffic.

### **NOTE**

Only Public IP addresses with basic SKU are available when using [instance metadata service IMDS](#). Standard SKU is not supported.

## **Allocation method**

Both basic and standard SKU public IP addresses support the *static* allocation method. The resource is assigned an IP address at the time it is created and the IP address is released when the resource is deleted.

Basic SKU public IP addresses also support a *dynamic* allocation method, which is the default if allocation method

is not specified. Selecting *dynamic* allocation method for a basic public IP address resource means the IP address is **not** allocated at the time of the resource creation. The public IP address is allocated when you associate the public IP address with a virtual machine or when you place the first virtual machine instance into the backend pool of a basic load balancer. The IP address is released when you stop (or delete) the resource. After being released from resource A, for example, the IP address can be assigned to a different resource. If the IP address is assigned to a different resource while resource A is stopped, when you restart resource A, a different IP address is assigned. If you change the allocation method of a basic public IP address resource from *static* to *dynamic*, the address is released. To ensure the IP address for the associated resource remains the same, you can set the allocation method explicitly to *static*. A static IP address is assigned immediately.

#### NOTE

Even when you set the allocation method to *static*, you cannot specify the actual IP address assigned to the public IP address resource. Azure assigns the IP address from a pool of available IP addresses in the Azure location the resource is created in.

Static public IP addresses are commonly used in the following scenarios:

- When you must update firewall rules to communicate with your Azure resources.
- DNS name resolution, where a change in IP address would require updating A records.
- Your Azure resources communicate with other apps or services that use an IP address-based security model.
- You use SSL certificates linked to an IP address.

#### NOTE

Azure allocates public IP addresses from a range unique to each region in each Azure cloud. You can download the list of ranges (prefixes) for the Azure [Public](#), [US government](#), [China](#), and [Germany](#) clouds.

### DNS hostname resolution

You can specify a DNS domain name label for a public IP resource, which creates a mapping for `domainnamelabel.location.cloudapp.azure.com` to the public IP address in the Azure-managed DNS servers. For instance, if you create a public IP resource with **contoso** as a `domainnamelabel` in the **West US** Azure *location*, the fully qualified domain name (FQDN) **contoso.westus.cloudapp.azure.com** resolves to the public IP address of the resource.

#### IMPORTANT

Each domain name label created must be unique within its Azure location.

### DNS Best Practices

If you ever need to migrate to a different region, you cannot migrate the FQDN of your public IP Address. As a best practice, you can use the FQDN to create a custom domain CNAME record pointing to the public IP address in Azure. If you need to move to a different public IP, it will require an update to the CNAME record instead of having to manually update the FQDN to the new address. You can use [Azure DNS](#) or an external DNS provider for your DNS Record.

### Virtual machines

You can associate a public IP address with a [Windows](#) or [Linux](#) virtual machine by assigning it to its **network interface**. You can assign either a dynamic or a static public IP address to a virtual machine. Learn more about [assigning IP addresses to network interfaces](#).

### Internet-facing load balancers

You can associate a public IP address created with either [SKU](#) with an [Azure Load Balancer](#), by assigning it to the load balancer **frontend** configuration. The public IP address serves as a load-balanced virtual IP address (VIP). You can assign either a dynamic or a static public IP address to a load balancer front-end. You can also assign multiple public IP addresses to a load balancer front-end, which enables [multi-VIP](#) scenarios like a multi-tenant environment with SSL-based websites. For more information about Azure load balancer SKUs, see [Azure load balancer standard SKU](#).

## VPN gateways

An [Azure VPN Gateway](#) connects an Azure virtual network to other Azure virtual networks, or to an on-premises network. A public IP address is assigned to the VPN Gateway to enable it to communicate with the remote network. You can only assign a *dynamic* basic public IP address to a VPN gateway.

## Application gateways

You can associate a public IP address with an Azure [Application Gateway](#), by assigning it to the gateway's **frontend** configuration. This public IP address serves as a load-balanced VIP. You can only assign a *dynamic* basic public IP address to an application gateway V1 front-end configuration, and only a *static* standard SKU address to a V2 front-end configuration.

## At-a-glance

The following table shows the specific property through which a public IP address can be associated to a top-level resource, and the possible allocation methods (dynamic or static) that can be used.

TOP-LEVEL RESOURCE	IP ADDRESS ASSOCIATION	DYNAMIC	STATIC
Virtual machine	Network interface	Yes	Yes
Internet-facing Load balancer	Front-end configuration	Yes	Yes
VPN gateway	Gateway IP configuration	Yes	No
Application gateway	Front-end configuration	Yes (V1 only)	Yes (V2 only)

## Private IP addresses

Private IP addresses allow Azure resources to communicate with other resources in a [virtual network](#) or an on-premises network through a VPN gateway or ExpressRoute circuit, without using an Internet-reachable IP address.

In the Azure Resource Manager deployment model, a private IP address is associated to the following types of Azure resources:

- Virtual machine network interfaces
- Internal load balancers (ILBs)
- Application gateways

## Allocation method

A private IP address is allocated from the address range of the virtual network subnet a resource is deployed in. Azure reserves the first four addresses in each subnet address range, so the addresses cannot be assigned to resources. For example, if the subnet's address range is 10.0.0.0/16, addresses 10.0.0.0-10.0.0.3 and 10.0.255.255 cannot be assigned to resources. IP addresses within the subnet's address range can only be assigned to one resource at a time.

There are two methods in which a private IP address is allocated:

- **Dynamic:** Azure assigns the next available unassigned or unreserved IP address in the subnet's address range. For example, Azure assigns 10.0.0.10 to a new resource, if addresses 10.0.0.4-10.0.0.9 are already assigned to other resources. Dynamic is the default allocation method. Once assigned, dynamic IP addresses are only released if a network interface is deleted, assigned to a different subnet within the same virtual network, or the allocation method is changed to static, and a different IP address is specified. By default, Azure assigns the previous dynamically assigned address as the static address when you change the allocation method from dynamic to static.
- **Static:** You select and assign any unassigned or unreserved IP address in the subnet's address range. For example, if a subnet's address range is 10.0.0.0/16 and addresses 10.0.0.4-10.0.0.9 are already assigned to other resources, you can assign any address between 10.0.0.10 - 10.0.255.254. Static addresses are only released if a network interface is deleted. If you change the allocation method to dynamic, Azure dynamically assigns the previously assigned static IP address as the dynamic address, even if the address isn't the next available address in the subnet's address range. The address also changes if the network interface is assigned to a different subnet within the same virtual network, but to assign the network interface to a different subnet, you must first change the allocation method from static to dynamic. Once you've assigned the network interface to a different subnet, you can change the allocation method back to static, and assign an IP address from the new subnet's address range.

## Virtual machines

One or more private IP addresses are assigned to one or more **network interfaces** of a [Windows](#) or [Linux](#) virtual machine. You can specify the allocation method as either dynamic or static for each private IP address.

### Internal DNS hostname resolution (for virtual machines)

All Azure virtual machines are configured with [Azure-managed DNS servers](#) by default, unless you explicitly configure custom DNS servers. These DNS servers provide internal name resolution for virtual machines that reside within the same virtual network.

When you create a virtual machine, a mapping for the hostname to its private IP address is added to the Azure-managed DNS servers. If a virtual machine has multiple network interfaces, or multiple IP configurations for a network interface the hostname is mapped to the private IP address of the primary IP configuration of the primary network interface.

Virtual machines configured with Azure-managed DNS servers are able to resolve the hostnames of all virtual machines within the same virtual network to their private IP addresses. To resolve host names of virtual machines in connected virtual networks, you must use a custom DNS server.

### Internal load balancers (ILB) & Application gateways

You can assign a private IP address to the **front-end** configuration of an [Azure Internal Load Balancer](#) (ILB) or an [Azure Application Gateway](#). This private IP address serves as an internal endpoint, accessible only to the resources within its virtual network and the remote networks connected to the virtual network. You can assign either a dynamic or static private IP address to the front-end configuration.

## At-a-glance

The following table shows the specific property through which a private IP address can be associated to a top-level resource, and the possible allocation methods (dynamic or static) that can be used.

TOP-LEVEL RESOURCE	IP ADDRESS ASSOCIATION	DYNAMIC	STATIC
Virtual machine	Network interface	Yes	Yes
Load balancer	Front-end configuration	Yes	Yes
Application gateway	Front-end configuration	Yes	Yes

## Limits

The limits imposed on IP addressing are indicated in the full set of [limits for networking](#) in Azure. The limits are per region and per subscription. You can [contact support](#) to increase the default limits up to the maximum limits based on your business needs.

## Pricing

Public IP addresses may have a nominal charge. To learn more about IP address pricing in Azure, review the [IP address pricing](#) page.

## Next steps

- [Deploy a VM with a static public IP using the Azure portal](#)
- [Deploy a VM with a static private IP address using the Azure portal](#)

# Public IP address prefix

1/3/2020 • 5 minutes to read • [Edit Online](#)

A public IP address prefix is a reserved range of IP addresses for your public endpoints in Azure. Azure allocates a contiguous range of addresses to your subscription based on how many you specify. If you're not familiar with public addresses, see [Public IP addresses](#).

Public IP addresses are assigned from a pool of addresses in each Azure region. You can [download](#) the list of ranges Azure uses for each region. For example, 40.121.0.0/16 is one of over 100 ranges Azure uses in the East US region. The range includes the usable addresses of 40.121.0.1 - 40.121.255.254.

You create a public IP address prefix in an Azure region and subscription by specifying a name, and how many addresses you want the prefix to include. For example, If you create a public IP address prefix of /28, Azure allocates 16 addresses from one of its ranges for you. You don't know which range Azure will assign until you create the range, but the addresses are contiguous. Public IP address prefixes have a fee. For details, see [public IP address pricing](#).

## Why create a public IP address prefix?

When you create public IP address resources, Azure assigns an available public IP address from any of the ranges used in the region. Once Azure assigns the address, you know what the address is, but until Azure assigns the address, you don't know what address might be assigned. This can be problematic when, for example, you, or your business partners, setup firewall rules that allow specific IP addresses. Each time you assign a new public IP address to a resource, the address has to be added to the firewall rule. When you assign addresses to your resources from a public IP address prefix, firewall rules don't need to be updated each time you assign one of the addresses, because the whole range could be added to a rule.

## Benefits

- You can create public IP address resources from a known range.
- You, or your business partners can create firewall rules with ranges that include public IP addresses you've currently assigned, as well as addresses you haven't assigned yet. This eliminates the need to change firewall rules as you assign IP addresses to new resources.
- The default size of a range you can create is /28 or 16 IP addresses.
- There are no limits as to how many ranges you can create, however, there are limits on the maximum number of static public IP addresses you can have in an Azure subscription. As a result, the number of ranges you create can't encompass more static public IP addresses than you can have in your subscription. For more information, see [Azure limits](#).
- The addresses that you create using addresses from the prefix can be assigned to any Azure resource that you can assign a public IP address to.
- You can easily see which IP addresses that are allocated and not yet allocated within the range.

## Scenarios

You can associate the following resources to a static public IP address from a prefix:

RESOURCE	SCENARIO	STEPS
----------	----------	-------

RESOURCE	SCENARIO	STEPS
Virtual Machines	Associating public IPs from a prefix to your virtual machines in Azure reduces management overhead when it comes to whitelisting IPs in a firewall. You can simply whitelist an entire prefix with a single firewall rule. As you scale with virtual machines in Azure, you can associate IPs from the same prefix saving cost, time, and management overhead.	To associate IPs from a prefix to your virtual machine: 1. <a href="#">Create a prefix</a> . 2. <a href="#">Create an IP from the prefix</a> . 3. <a href="#">Associate the IP to your virtual machine's network interface</a> . You can also <a href="#">associate the IPs to a Virtual Machine Scale Set</a> .
Standard Load Balancers	Associating public IPs from a prefix to your frontend IP configuration or outbound rule of a Load Balancer ensures simplification of your Azure public IP address space. You can simplify your scenario by grooming outbound connections to be originated from a range of contiguous IP addresses defined by public IP prefix.	To associate IPs from a prefix to your Load balancer: 1. <a href="#">Create a prefix</a> . 2. <a href="#">Create an IP from the prefix</a> . 3. When creating the Load Balancer, select or update the IP created in step 2 above as the frontend IP of your Load Balancer.
Azure Firewall	You can use a public IP from a prefix for outbound SNAT. This means all outbound virtual network traffic is translated to the <a href="#">Azure Firewall</a> public IP. Since this IP comes from a predetermined prefix, it is very easy to know ahead of time what your public IP footprint in Azure will look like.	1. <a href="#">Create a prefix</a> . 2. <a href="#">Create an IP from the prefix</a> . 3. When you <a href="#">deploy the Azure Firewall</a> , be sure to select the IP you previously allocated from the prefix.
Application Gateway v2	You can use a public IP from a prefix for your autoscaling and zone-redundant Application gateway v2. Since this IP comes from a predetermined prefix, it is very easy to know ahead of time what your public IP footprint in Azure will look like.	1. <a href="#">Create a prefix</a> . 2. <a href="#">Create an IP from the prefix</a> . 3. When you <a href="#">deploy the Application Gateway</a> , be sure to select the IP you previously allocated from the prefix.

## Constraints

- You can't specify the IP addresses for the prefix. Azure allocates the IP addresses for the prefix, based on the size that you specify.
- You can create a prefix of up to 16 IP addresses or a /28. For more information, see [Azure limits](#).
- You can't change the range, once you've created the prefix.
- Only static public IP addresses created with the Standard SKU can be assigned from the prefix's range. To learn more about public IP address SKUs, see [public IP address](#).
- Addresses from the range can only be assigned to Azure Resource Manager resources. Addresses cannot be assigned to resources in the classic deployment model.
- All public IP addresses created from the prefix must exist in the same Azure region and subscription as the prefix, and must be assigned to resources in the same region and subscription.
- You can't delete a prefix if any addresses within it are assigned to public IP address resources associated to a resource. Dissociate all public IP address resources that are assigned IP addresses from the prefix first.

## Next steps

- [Create](#) a public IP address prefix

# What is IPv6 for Azure Virtual Network? (Preview)

1/9/2020 • 5 minutes to read • [Edit Online](#)

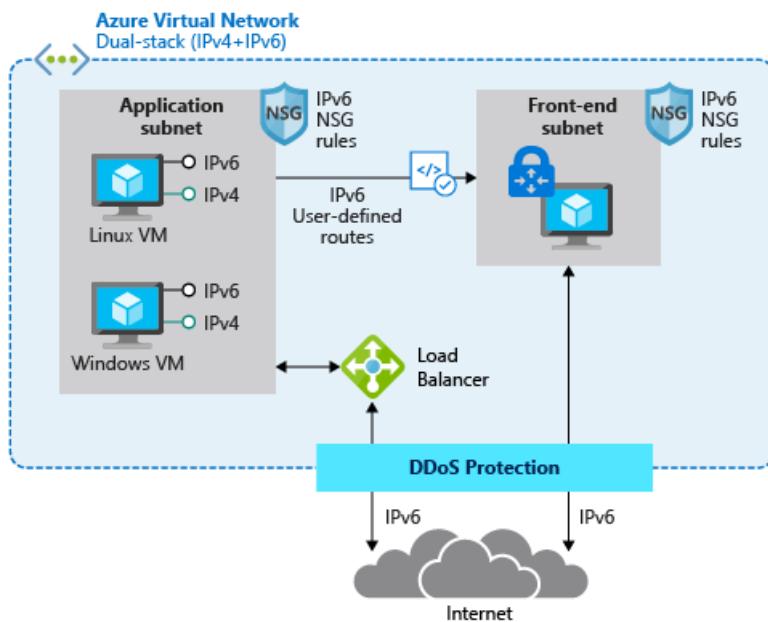
IPv6 for Azure Virtual Network (VNet) enables you to host applications in Azure with IPv6 and IPv4 connectivity both within a virtual network and to and from the Internet. Due to the exhaustion of public IPv4 addresses, new networks for mobility and Internet of Things (IoT) are often built on IPv6. Even long established ISP and mobile networks are being transformed to IPv6. IPv4-only services can find themselves at a real disadvantage in both existing and emerging markets. Dual stack IPv4/IPv6 connectivity enables Azure-hosted services to traverse this technology gap with globally available, dual-stacked services that readily connect with both the existing IPv4 and these new IPv6 devices and networks.

Azure's original IPv6 connectivity makes it easy to provide dual stack (IPv4/IPv6) Internet connectivity for applications hosted in Azure. It allows for simple deployment of VMs with load balanced IPv6 connectivity for both inbound and outbound initiated connections. This feature is still available and more information is available [here](#). IPv6 for Azure virtual network is much more full featured- enabling full IPv6 solution architectures to be deployed in Azure.

## IMPORTANT

IPv6 for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

The following diagram depicts a simple dual stack (IPv4/IPv6) deployment in Azure:



## Benefits

IPv6 for Azure VNET benefits:

- Helps expand the reach of your Azure-hosted applications into the growing mobile and Internet of Things markets.
- Dual stacked IPv4/IPv6 VMs provide maximum service deployment flexibility. A single service instance can connect with both IPv4 and IPv6-capable Internet clients.

- Builds on long-established, stable Azure VM-to-Internet IPv6 connectivity.
- Secure by default since IPv6 connectivity to the Internet is only established when you explicitly request it in your deployment.

## Capabilities

IPv6 for Azure VNet includes the following capabilities:

- Azure customers can define their own IPv6 virtual network address space to meet the needs of their applications, customers, or seamlessly integrate into their on-premises IP space.
- Dual stack (IPv4 and IPv6) virtual networks with dual stack subnets enable applications to connect with both IPv4 and IPv6 resources in their virtual network or - the Internet.

### IMPORTANT

The subnets for IPv6 must be exactly /64 in size. This ensures future compatibility should you decide to enable routing of the subnet to an on-premises network since some routers can only accept /64 IPv6 routes.

- Protect your resources with IPv6 rules for Network Security Groups.
  - And the Azure platform's Distributed Denial of Service (DDoS) protections are extended to Internet-facing Public IP's
- Customize the routing of IPv6 traffic in your virtual network with User-Defined Routes- especially when leveraging Network Virtual Appliances to augment your application.
- Linux and Windows Virtual Machines can all use IPv6 for Azure VNET
- [Standard IPv6 public Load Balancer](#) support to create resilient, scalable applications, which include:
  - Optional IPv6 health probe to determine which backend pool instances are healthy and thus can receive new flows.
  - Optional outbound rules which provide full declarative control over outbound connectivity to scale and tune this ability to your specific needs.
  - Optional multiple front-end configurations which enable a single load balancer to use multiple IPv6 public IP addresses- the same frontend protocol and port can be reused across frontend addresses.
  - Optional IPv6 ports can be reused on backend instances using the *Floating IP* feature of load-balancing rules
- [Standard IPv6 internal Load Balancer](#) support to create resilient multi-tier applications within Azure VNets.
- Basic IPv6 public Load Balancer support for compatibility with legacy deployments
- [Reserved IPv6 Public IP addresses and address ranges](#) provide stable, predictable IPv6 addresses which ease whitelisting of your azure-hosted applications for your company and your customers.
- Instance-level Public IP provides IPv6 Internet connectivity directly to individual VMs.
- [Add IPv6 to Existing IPv4-only deployments](#)- this feature enables you to easily add IPv6 connectivity to existing IPv4-only deployments without the need to recreate deployments. The IPv4 network traffic is unaffected during this process so depending on your application and OS you may be able to add IPv6 even to live services.
- Let Internet clients seamlessly access your dual stack application using their protocol of choice with Azure DNS support for IPv6 (AAAA) records.
- Create dual stack applications that automatically scale to your load with virtual machine scale sets with IPv6.
- [Virtual Network \(VNET\) Peering](#) - both within-regional and global peering - enables you to seamlessly connect dual stack VNets- both the IPv4 and IPv6 endpoints on VMs in the peered networks will be able to communicate with each other. You can even peer dual stack with IPv4-only VNets as you are transitioning your deployments to dual stack.
- IPv6 Troubleshooting and Diagnostics are available with load balancer metrics/alerting and Network Watcher

features such as packet capture, NSG flow logs, connection troubleshooting and connection monitoring.

## Scope

IPv6 for Azure VNET is a foundational feature set which enables customers to host dual stack (IPv4+IPv6) applications in Azure. We intend to add IPv6 support to more Azure networking features over time and eventually to offer dual stack versions of Azure PaaS services but in the meantime all Azure PaaS services can be accessed via the IPv4 endpoints on dual stack Virtual Machines.

## Limitations

The current IPv6 for Azure virtual network release has the following limitations:

- IPv6 for Azure virtual network (Preview) is available in all global Azure regions, but only in Global Azure- not yet in government clouds.
- ExpressRoute and VPN gateways cannot be used in a VNET with IPv6 enabled, either directly or peered with "UseRemoteGateway".
- The Azure platform (AKS, etc.) does not support IPv6 communication for Containers.

## Pricing

IPv6 Azure resources and bandwidth are charged at the same rates as IPv4. There are no additional or different charges for IPv6. You can find details about pricing for [public IP addresses](#), [network bandwidth](#), or [Load Balancer](#).

## Next steps

- Learn how to [deploy an IPv6 dual stack application using Azure PowerShell](#).
- Learn how to [deploy an IPv6 dual stack application using Azure CLI](#).
- Learn how to [deploy an IPv6 dual stack application using Resource Manager Templates \(JSON\)](#)

# Reserved public IPv6 address prefix (Preview)

10/27/2019 • 2 minutes to read • [Edit Online](#)

In Azure, dual stack (IPv4+IPv6) virtual networks (VNet) and virtual machines (VMs) are secure by default since they have no Internet connectivity. You can easily add an IPv6 Internet connectivity to your Azure Load Balancers and VMs with public IPv6 addresses that you obtain from Azure.

Any public IPs that you reserve are associated with an Azure region of your choice and with your Azure subscription. You may move a reserved (static) IPv6 public IP between any of the Azure Load Balancers or VMs in your subscription. You may dissociate the IPv6 public IP entirely and it will be held for your use when you're ready.

## WARNING

Use caution to not delete your public IP addresses accidentally. Deleting a public IP removes it from your subscription and you will not be able to recover it (not even with the help of Azure support).

In addition to reserving individual IPv6 addresses, you can reserve contiguous ranges of Azure IPv6 addresses (known as IP prefix) for your use. Similar to individual IP addresses, reserved prefixes are associated with an Azure region of your choice and with your Azure subscription. Reserving a predictable, contiguous range of addresses has many uses. For example, you can greatly simplify IP *whitelisting* of your Azure-hosted applications by your company and your customers as your static IP ranges can be readily programmed into on-premises firewalls. You can create individual public IPs from your IP prefix as needed and when you delete those individual Public IPs they are *returned* to your reserved range so that you can reuse them later. All the IP addresses in your IP Prefix are reserved for your exclusive use until such time as you delete your Prefix.

## IMPORTANT

IPv6 for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## IPv6 prefix sizes

The following public IP prefix sizes are available:

- Minimum IPv6 Prefix size: /127 = 2 addresses
- Maximum IPv6 Prefix size: /124 = 16 addresses

Prefix size is specified as a Classless Inter-Domain Routing (CIDR) mask size. For example, a mask of /128 represents an individual IPv6 address as IPv6 addresses are composed of 128 bits.

## Pricing

For costs associated with using Azure Public IPs, both individual IP addresses and IP ranges, see [Public IP Address pricing](#).

## Limitations

IPv6 is supported on Basic Public IPs only with "dynamic" allocation that means that the IPv6 address will change if you delete and redeploy your application (VM's or load balancers) in Azure. Standard IPv6 Public IP's support

solely static (reserved) allocation though Standard INTERNAL load balancers can also support dynamic allocation from within the subnet to which they are assigned.

As a best practice, we recommend that you use Standard Public IPs and Standard Load Balancers for your IPv6 applications.

## Next steps

- Reserve a public [IPv6 address prefix](#).
- Learn more about [IPv6 addresses](#).
- Learn about [how to create and use public IPs](#) (both IPv4 and IPv6) in Azure.

# Azure DDoS Protection Standard overview

1/29/2020 • 5 minutes to read • [Edit Online](#)

Distributed denial of service (DDoS) attacks are some of the largest availability and security concerns facing customers that are moving their applications to the cloud. A DDoS attack attempts to exhaust an application's resources, making the application unavailable to legitimate users. DDoS attacks can be targeted at any endpoint that is publicly reachable through the internet.

Azure DDoS protection, combined with application design best practices, provide defense against DDoS attacks. Azure DDoS protection provides the following service tiers:

- **Basic:** Automatically enabled as part of the Azure platform. Always-on traffic monitoring, and real-time mitigation of common network-level attacks, provide the same defenses utilized by Microsoft's online services. The entire scale of Azure's global network can be used to distribute and mitigate attack traffic across regions. Protection is provided for IPv4 and IPv6 Azure [public IP addresses](#).
- **Standard:** Provides additional mitigation capabilities over the Basic service tier that are tuned specifically to Azure Virtual Network resources. DDoS Protection Standard is simple to enable, and requires no application changes. Protection policies are tuned through dedicated traffic monitoring and machine learning algorithms. Policies are applied to public IP addresses associated to resources deployed in virtual networks, such as Azure Load Balancer, Azure Application Gateway, and Azure Service Fabric instances, but this protection does not apply to App Service Environments. Real-time telemetry is available through Azure Monitor views during an attack, and for history. Rich attack mitigation analytics are available via diagnostic settings. Application layer protection can be added through the [Azure Application Gateway Web Application Firewall](#) or by installing a 3rd party firewall from Azure Marketplace. Protection is provided for IPv4 and IPv6 Azure [public IP addresses](#).

FEATURE	DDOS PROTECTION BASIC	DDOS PROTECTION STANDARD
Active traffic monitoring & always on detection	Yes	Yes
Automatic attack mitigations	Yes	Yes
Availability guarantee	Azure Region	Application
Mitigation policies	Tuned for Azure traffic region volume	Tuned for application traffic volume
Metrics & alerts	No	Real time attack metrics & diagnostic logs via Azure monitor
Mitigation reports	No	Post attack mitigation reports
Mitigation flow logs	No	NRT log stream for SIEM integration
Migration policy customizations	No	Engage DDoS Experts
Support	Best effort	Access to DDoS Experts during an active attack
SLA	Azure Region	Application guarantee & cost protection

FEATURE	DDOS PROTECTION BASIC	DDOS PROTECTION STANDARD
Pricing	Free	Monthly & usage based

## Types of DDoS attacks that DDoS Protection Standard mitigates

DDoS Protection Standard can mitigate the following types of attacks:

- **Volumetric attacks:** The attack's goal is to flood the network layer with a substantial amount of seemingly legitimate traffic. It includes UDP floods, amplification floods, and other spoofed-packet floods. DDoS Protection Standard mitigates these potential multi-gigabyte attacks by absorbing and scrubbing them, with Azure's global network scale, automatically.
- **Protocol attacks:** These attacks render a target inaccessible, by exploiting a weakness in the layer 3 and layer 4 protocol stack. It includes, SYN flood attacks, reflection attacks, and other protocol attacks. DDoS Protection Standard mitigates these attacks, differentiating between malicious and legitimate traffic, by interacting with the client, and blocking malicious traffic.
- **Resource (application) layer attacks:** These attacks target web application packets, to disrupt the transmission of data between hosts. The attacks include HTTP protocol violations, SQL injection, cross-site scripting, and other layer 7 attacks. Use a Web Application Firewall, such as the Azure [Application Gateway web application firewall](#), as well as DDoS Protection Standard to provide defense against these attacks. There are also third-party web application firewall offerings available in the [Azure Marketplace](#).

DDoS Protection Standard protects resources in a virtual network including public IP addresses associated with virtual machines, load balancers, and application gateways. When coupled with the Application Gateway web application firewall, or a third-party web application firewall deployed in a virtual network with a public IP, DDoS Protection Standard can provide full layer 3 to layer 7 mitigation capability.

## DDoS Protection Standard features



Reliable  
proven success



Scalable  
global capacity



Automatic  
simple, automated



Adaptive  
real time tuning

DDoS Protection Standard features include:

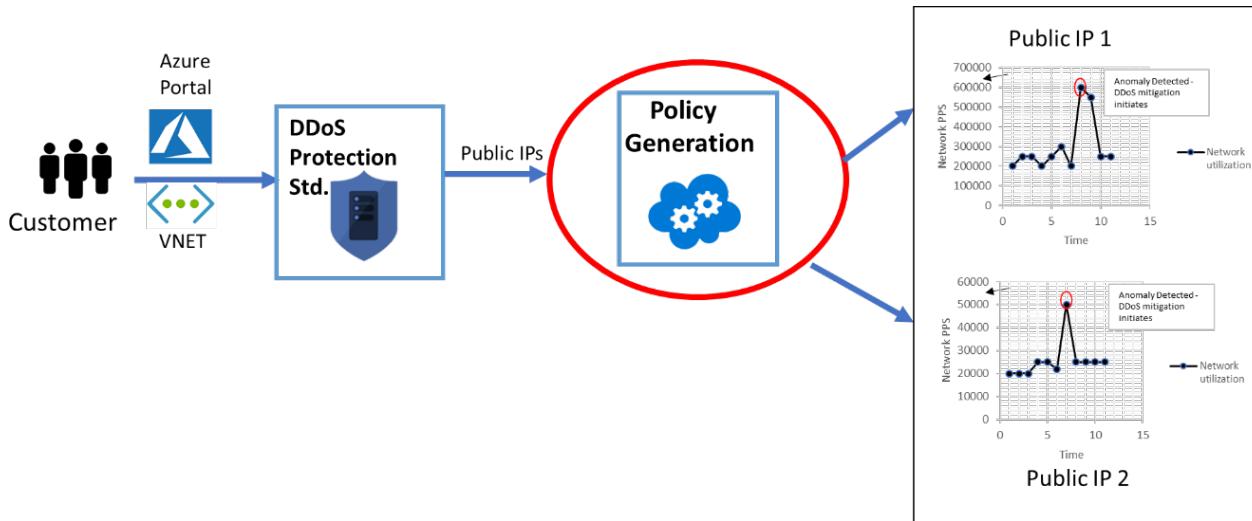
- **Native platform integration:** Natively integrated into Azure. Includes configuration through the Azure portal. DDoS Protection Standard understands your resources and resource configuration.
- **Turn-key protection:** Simplified configuration immediately protects all resources on a virtual network as soon as DDoS Protection Standard is enabled. No intervention or user definition is required. DDoS Protection Standard instantly and automatically mitigates the attack, once it is detected.
- **Always-on traffic monitoring:** Your application traffic patterns are monitored 24 hour a day, 7 days a week, looking for indicators of DDoS attacks. Mitigation is performed when protection policies are exceeded.
- **Adaptive tuning:** Intelligent traffic profiling learns your application's traffic over time, and selects and updates the profile that is the most suitable for your service. The profile adjusts as traffic changes over time.
- **Multi-Layered protection:** Provides full stack DDoS protection, when used with a web application firewall.
- **Extensive mitigation scale:** Over 60 different attack types can be mitigated, with global capacity, to protect against the largest known DDoS attacks.
- **Attack analytics:** Get detailed reports in five-minute increments during an attack, and a complete summary

after the attack ends. Stream mitigation flow logs to an offline security information and event management (SIEM) system for near real-time monitoring during an attack.

- **Attack metrics:** Summarized metrics from each attack are accessible through Azure Monitor.
- **Attack alerting:** Alerts can be configured at the start and stop of an attack, and over the attack's duration, using built-in attack metrics. Alerts integrate into your operational software like Microsoft Azure Monitor logs, Splunk, Azure Storage, Email, and the Azure portal.
- **Cost guarantee:** Data-transfer and application scale-out service credits for documented DDoS attacks.

## DDoS Protection Standard mitigation

DDoS Protection Standard monitors actual traffic utilization and constantly compares it against the thresholds defined in the DDoS Policy. When the traffic threshold is exceeded, DDoS mitigation is initiated automatically. When traffic returns below the threshold, the mitigation is removed.



During mitigation, traffic sent to the protected resource is redirected by the DDoS protection service and several checks are performed, such as the following checks:

- Ensure packets conform to internet specifications and are not malformed.
- Interact with the client to determine if the traffic is potentially a spoofed packet (e.g: SYN Auth or SYN Cookie or by dropping a packet for the source to retransmit it).
- Rate-limit packets, if no other enforcement method can be performed.

DDoS protection blocks attack traffic and forwards the remaining traffic to its intended destination. Within a few minutes of attack detection, you are notified using Azure Monitor metrics. By configuring logging on DDoS Protection Standard telemetry, you can write the logs to available options for future analysis. Metric data in Azure Monitor for DDoS Protection Standard is retained for 30 days.

Microsoft has partnered with [BreakingPoint Cloud](#) to build an interface where you can generate traffic against DDoS Protection-enabled public IP addresses for simulations. The BreakPoint Cloud simulation allows you to:

- Validate how Microsoft Azure DDoS Protection Standard protects your Azure resources from DDoS attacks
- Optimize your incident response process while under DDoS attack
- Document DDoS compliance
- Train your network security teams

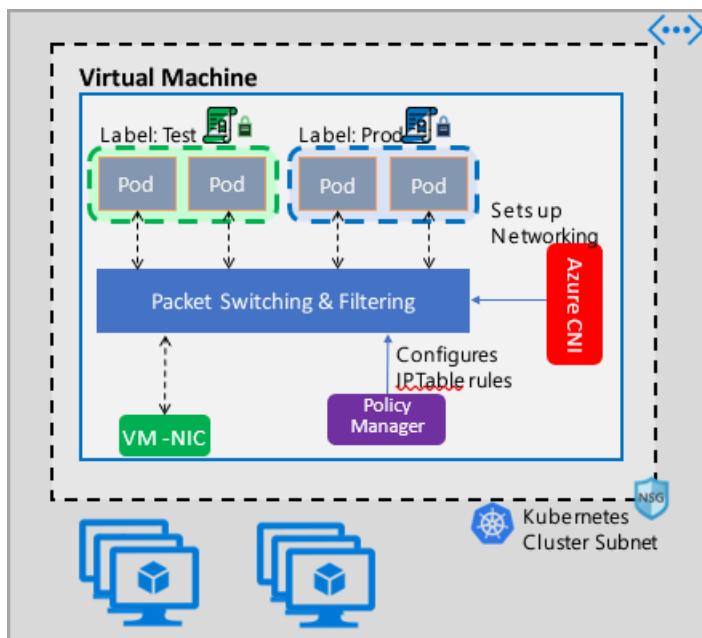
## Next steps

- [Configure DDoS Protection Standard](#)

# Azure Kubernetes network policies overview

10/30/2019 • 2 minutes to read • [Edit Online](#)

Network Policies provide micro-segmentation for pods just like Network Security Groups (NSGs) provide micro-segmentation for VMs. The Azure Network Policy implementation supports the standard Kubernetes Network Policy specification. You can use labels to select a group of pods and define a list of ingress and egress rules that specify the kind of traffic that is allowed to and from these pods. Learn more about the Kubernetes network policies in the [Kubernetes documentation](#).



Azure network policies work in conjunction with the Azure CNI that provides VNet integration for containers. It is supported only on Linux nodes today. The implementations configure Linux IP Table rules based on the defined policies to enforce traffic filtering.

## Planning security for your Kubernetes cluster

When implementing security for your cluster, use network security groups (NSGs) to filter North-South traffic, that is, traffic entering and leaving your cluster subnet, and use Kubernetes network policies for East-West traffic, that is, traffic between pods in your cluster.

## Using Azure Kubernetes network policies

Azure Network Policies can be used in the following ways to provide micro-segmentation for pods.

### ACS-engine

ACS-Engine is a tool that generates an Azure Resource Manager template for the deployment of a Kubernetes cluster in Azure. The cluster configuration is specified in a JSON file that is passed to the tool when generating the template. To learn more about the entire list of supported cluster settings and their descriptions, see Microsoft Azure Container Service Engine - Cluster Definition.

To enable policies on clusters deployed using acs-engine, specify the value of the `networkPolicy` setting in the cluster definition file to be "azure".

### Example configuration

The below JSON example configuration creates a new virtual network and subnet, and deploys a Kubernetes

cluster in it with Azure CNI. We recommend that you use "Notepad" to edit the JSON file.

```
{
  "apiVersion": "vlabs",
  "properties": {
    "orchestratorProfile": {
      "orchestratorType": "Kubernetes",
      "kubernetesConfig": {
        "networkPolicy": "azure"
      }
    },
    "masterProfile": {
      "count": 1,
      "dnsPrefix": "<specify a cluster name>",
      "vmSize": "Standard_D2s_v3"
    },
    "agentPoolProfiles": [
      {
        "name": "agentpool",
        "count": 2,
        "vmSize": "Standard_D2s_v3",
        "availabilityProfile": "AvailabilitySet"
      }
    ],
    "linuxProfile": {
      "adminUsername": "<specify admin username>",
      "ssh": {
        "publicKeys": [
          {
            "keyData": "<cut and paste your ssh key here>"
          }
        ]
      }
    },
    "servicePrincipalProfile": {
      "clientId": "<enter the client ID of your service principal here >",
      "secret": "<enter the password of your service principal here>"
    }
  }
}
```

## Creating your own Kubernetes cluster in Azure

The implementation can be used to provide Network Policies for Pods in Kubernetes clusters that you deploy yourself, without relying on tools like the ACS-Engine. In this case, you first install the CNI plug-in and enable it on every virtual machine in a cluster. For detailed instructions, see [Deploy the plug-in for a Kubernetes cluster that you deploy yourself](#).

Once the cluster is deployed run the following `kubectl` command to download and apply the Azure network policy *daemonset* to the cluster.

```
kubectl apply -f https://raw.githubusercontent.com/Azure/acs-engine/master/parts/k8s/addons/kubernetesmasteraddons-azure-npm-daemonset.yaml
```

The solution is also open source and the code is available on the [Azure Container Networking repository](#).

## Next steps

- Learn about [Azure Kubernetes Service](#).
- Learn about [container networking](#).

- Deploy the [plug-in](#) for Kubernetes clusters or Docker containers.

# Virtual network traffic routing

2/21/2020 • 24 minutes to read • [Edit Online](#)

Learn about how Azure routes traffic between Azure, on-premises, and Internet resources. Azure automatically creates a route table for each subnet within an Azure virtual network and adds system default routes to the table. To learn more about virtual networks and subnets, see [Virtual network overview](#). You can override some of Azure's system routes with [custom routes](#), and add additional custom routes to route tables. Azure routes outbound traffic from a subnet based on the routes in a subnet's route table.

## System routes

Azure automatically creates system routes and assigns the routes to each subnet in a virtual network. You can't create system routes, nor can you remove system routes, but you can override some system routes with [custom routes](#). Azure creates default system routes for each subnet, and adds additional [optional default routes](#) to specific subnets, or every subnet, when you use specific Azure capabilities.

### Default

Each route contains an address prefix and next hop type. When traffic leaving a subnet is sent to an IP address within the address prefix of a route, the route that contains the prefix is the route Azure uses. Learn more about [how Azure selects a route](#) when multiple routes contain the same prefixes, or overlapping prefixes. Whenever a virtual network is created, Azure automatically creates the following default system routes for each subnet within the virtual network:

SOURCE	ADDRESS PREFIXES	NEXT HOP TYPE
Default	Unique to the virtual network	Virtual network
Default	0.0.0.0/0	Internet
Default	10.0.0.0/8	None
Default	192.168.0.0/16	None
Default	100.64.0.0/10	None

The next hop types listed in the previous table represent how Azure routes traffic destined for the address prefix listed. Explanations for the next hop types follow:

- **Virtual network:** Routes traffic between address ranges within the [address space](#) of a virtual network. Azure creates a route with an address prefix that corresponds to each address range defined within the address space of a virtual network. If the virtual network address space has multiple address ranges defined, Azure creates an individual route for each address range. Azure automatically routes traffic between subnets using the routes created for each address range. You don't need to define gateways for Azure to route traffic between subnets. Though a virtual network contains subnets, and each subnet has a defined address range, Azure does *not* create default routes for subnet address ranges, because each subnet address range is within an address range of the address space of a virtual network.
- **Internet:** Routes traffic specified by the address prefix to the Internet. The system default route specifies the 0.0.0.0/0 address prefix. If you don't override Azure's default routes, Azure routes traffic for any address not specified by an address range within a virtual network, to the Internet, with one exception. If

the destination address is for one of Azure's services, Azure routes the traffic directly to the service over Azure's backbone network, rather than routing the traffic to the Internet. Traffic between Azure services does not traverse the Internet, regardless of which Azure region the virtual network exists in, or which Azure region an instance of the Azure service is deployed in. You can override Azure's default system route for the 0.0.0.0/0 address prefix with a [custom route](#).

- **None:** Traffic routed to the **None** next hop type is dropped, rather than routed outside the subnet. Azure automatically creates default routes for the following address prefixes:

- **10.0.0.0/8 and 192.168.0.0/16:** Reserved for private use in RFC 1918.
- **100.64.0.0/10:** Reserved in RFC 6598.

If you assign any of the previous address ranges within the address space of a virtual network, Azure automatically changes the next hop type for the route from **None** to **Virtual network**. If you assign an address range to the address space of a virtual network that includes, but isn't the same as, one of the four reserved address prefixes, Azure removes the route for the prefix and adds a route for the address prefix you added, with **Virtual network** as the next hop type.

### Optional default routes

Azure adds additional default system routes for different Azure capabilities, but only if you enable the capabilities. Depending on the capability, Azure adds optional default routes to either specific subnets within the virtual network, or to all subnets within a virtual network. The additional system routes and next hop types that Azure may add when you enable different capabilities are:

SOURCE	ADDRESS PREFIXES	NEXT HOP TYPE	SUBNET WITHIN VIRTUAL NETWORK THAT ROUTE IS ADDED TO
Default	Unique to the virtual network, for example: 10.1.0.0/16	VNet peering	All
Virtual network gateway	Prefixes advertised from on-premises via BGP, or configured in the local network gateway	Virtual network gateway	All
Default	Multiple	VirtualNetworkServiceEndpoint	Only the subnet a service endpoint is enabled for.

- **Virtual network (VNet) peering:** When you create a virtual network peering between two virtual networks, a route is added for each address range within the address space of each virtual network a peering is created for. Learn more about [virtual network peering](#).
- **Virtual network gateway:** One or more routes with *Virtual network gateway* listed as the next hop type are added when a virtual network gateway is added to a virtual network. The source is also *virtual network gateway*, because the gateway adds the routes to the subnet. If your on-premises network gateway exchanges border gateway protocol (**BGP**) routes with an Azure virtual network gateway, a route is added for each route propagated from the on-premises network gateway. It's recommended that you summarize on-premises routes to the largest address ranges possible, so the fewest number of routes are propagated to an Azure virtual network gateway. There are limits to the number of routes you can propagate to an Azure virtual network gateway. For details, see [Azure limits](#).
- **VirtualNetworkServiceEndpoint:** The public IP addresses for certain services are added to the route table by Azure when you enable a service endpoint to the service. Service endpoints are enabled for individual subnets within a virtual network, so the route is only added to the route table of a subnet a service endpoint is enabled for. The public IP addresses of Azure services change periodically. Azure

manages the addresses in the route table automatically when the addresses change. Learn more about [virtual network service endpoints](#), and the services you can create service endpoints for.

#### NOTE

The **VNet peering** and **VirtualNetworkServiceEndpoint** next hop types are only added to route tables of subnets within virtual networks created through the Azure Resource Manager deployment model. The next hop types are not added to route tables that are associated to virtual network subnets created through the classic deployment model. Learn more about Azure [deployment models](#).

## Custom routes

You create custom routes by either creating [user-defined](#) routes, or by exchanging [border gateway protocol](#) (BGP) routes between your on-premises network gateway and an Azure virtual network gateway.

### User-defined

You can create custom, or user-defined(static), routes in Azure to override Azure's default system routes, or to add additional routes to a subnet's route table. In Azure, you create a route table, then associate the route table to zero or more virtual network subnets. Each subnet can have zero or one route table associated to it. To learn about the maximum number of routes you can add to a route table and the maximum number of user-defined route tables you can create per Azure subscription, see [Azure limits](#). If you create a route table and associate it to a subnet, the routes within it are combined with, or override, the default routes Azure adds to a subnet by default.

You can specify the following next hop types when creating a user-defined route:

- **Virtual appliance:** A virtual appliance is a virtual machine that typically runs a network application, such as a firewall. To learn about a variety of pre-configured network virtual appliances you can deploy in a virtual network, see the [Azure Marketplace](#). When you create a route with the **virtual appliance** hop type, you also specify a next hop IP address. The IP address can be:

- The [private IP address](#) of a network interface attached to a virtual machine. Any network interface attached to a virtual machine that forwards network traffic to an address other than its own must have the Azure *Enable IP forwarding* option enabled for it. The setting disables Azure's check of the source and destination for a network interface. Learn more about how to [enable IP forwarding for a network interface](#). Though *Enable IP forwarding* is an Azure setting, you may also need to enable IP forwarding within the virtual machine's operating system for the appliance to forward traffic between private IP addresses assigned to Azure network interfaces. If the appliance must route traffic to a public IP address, it must either proxy the traffic, or network address translate the private IP address of the source's private IP address to its own private IP address, which Azure then network address translates to a public IP address, before sending the traffic to the Internet. To determine required settings within the virtual machine, see the documentation for your operating system or network application. To understand outbound connections in Azure, see [Understanding outbound connections](#).

#### NOTE

Deploy a virtual appliance into a different subnet than the resources that route through the virtual appliance are deployed in. Deploying the virtual appliance to the same subnet, then applying a route table to the subnet that routes traffic through the virtual appliance, can result in routing loops, where traffic never leaves the subnet.

- The private IP address of an Azure [internal load balancer](#). A load balancer is often used as part of a [high availability strategy for network virtual appliances](#).

You can define a route with 0.0.0.0/0 as the address prefix and a next hop type of virtual appliance, enabling the appliance to inspect the traffic and determine whether to forward or drop the traffic. If you intend to create a user-defined route that contains the 0.0.0.0/0 address prefix, read [0.0.0.0/0 address prefix](#) first.

- **Virtual network gateway:** Specify when you want traffic destined for specific address prefixes routed to a virtual network gateway. The virtual network gateway must be created with type **VPN**. You cannot specify a virtual network gateway created as type **ExpressRoute** in a user-defined route because with ExpressRoute, you must use BGP for custom routes. You can define a route that directs traffic destined for the 0.0.0.0/0 address prefix to a [route-based](#) virtual network gateway. On your premises, you might have a device that inspects the traffic and determines whether to forward or drop the traffic. If you intend to create a user-defined route for the 0.0.0.0/0 address prefix, read [0.0.0.0/0 address prefix](#) first. Instead of configuring a user-defined route for the 0.0.0.0/0 address prefix, you can advertise a route with the 0.0.0.0/0 prefix via BGP, if you've [enabled BGP for a VPN virtual network gateway](#).
- **None:** Specify when you want to drop traffic to an address prefix, rather than forwarding the traffic to a destination. If you haven't fully configured a capability, Azure may list *None* for some of the optional system routes. For example, if you see *None* listed as the **Next hop IP address** with a **Next hop type** of *Virtual network gateway* or *Virtual appliance*, it may be because the device isn't running, or isn't fully configured. Azure creates system [default routes](#) for reserved address prefixes with **None** as the next hop type.
- **Virtual network:** Specify when you want to override the default routing within a virtual network. See [Routing example](#), for an example of why you might create a route with the **Virtual network** hop type.
- **Internet:** Specify when you want to explicitly route traffic destined to an address prefix to the Internet, or if you want traffic destined for Azure services with public IP addresses kept within the Azure backbone network.

You cannot specify **VNet peering** or **VirtualNetworkServiceEndpoint** as the next hop type in user-defined routes. Routes with the **VNet peering** or **VirtualNetworkServiceEndpoint** next hop types are only created by Azure, when you configure a virtual network peering, or a service endpoint.

## Next hop types across Azure tools

The name displayed and referenced for next hop types is different between the Azure portal and command-line tools, and the Azure Resource Manager and classic deployment models. The following table lists the names used to refer to each next hop type with the different tools and [deployment models](#):

NEXT HOP TYPE	AZURE CLI AND POWERSHELL (RESOURCE MANAGER)	AZURE CLASSIC CLI AND POWERSHELL (CLASSIC)
Virtual network gateway	VirtualNetworkGateway	VPNGateway
Virtual network	VNetLocal	VNETLocal (not available in the classic CLI in asm mode)
Internet	Internet	Internet (not available in the classic CLI in asm mode)
Virtual appliance	VirtualAppliance	VirtualAppliance
None	None	Null (not available in the classic CLI in asm mode)

Next hop type	Azure CLI and PowerShell (Resource Manager)	Azure Classic CLI and PowerShell (Classic)
Virtual network peering	VNet peering	Not applicable
Virtual network service endpoint	VirtualNetworkServiceEndpoint	Not applicable

## Border gateway protocol

An on-premises network gateway can exchange routes with an Azure virtual network gateway using the border gateway protocol (BGP). Using BGP with an Azure virtual network gateway is dependent on the type you selected when you created the gateway. If the type you selected were:

- **ExpressRoute:** You must use BGP to advertise on-premises routes to the Microsoft Edge router. You cannot create user-defined routes to force traffic to the ExpressRoute virtual network gateway if you deploy a virtual network gateway deployed as type: ExpressRoute. You can use user-defined routes for forcing traffic from the Express Route to, for example, a Network Virtual Appliance.
- **VPN:** You can, optionally use BGP. For details, see [BGP with site-to-site VPN connections](#).

When you exchange routes with Azure using BGP, a separate route is added to the route table of all subnets in a virtual network for each advertised prefix. The route is added with *Virtual network gateway* listed as the source and next hop type.

ER and VPN Gateway route propagation can be disabled on a subnet using a property on a route table. When you exchange routes with Azure using BGP, routes are not added to the route table of all subnets with Virtual network gateway route propagation disabled. Connectivity with VPN connections is achieved using [custom routes](#) with a next hop type of *Virtual network gateway*. For details, see [How to disable Virtual network gateway route propagation](#).

## How Azure selects a route

When outbound traffic is sent from a subnet, Azure selects a route based on the destination IP address, using the longest prefix match algorithm. For example, a route table has two routes: One route specifies the 10.0.0.0/24 address prefix, while the other route specifies the 10.0.0.0/16 address prefix. Azure routes traffic destined for 10.0.0.5, to the next hop type specified in the route with the 10.0.0.0/24 address prefix, because 10.0.0.0/24 is a longer prefix than 10.0.0.0/16, even though 10.0.0.5 is within both address prefixes. Azure routes traffic destined to 10.0.1.5, to the next hop type specified in the route with the 10.0.0.0/16 address prefix, because 10.0.1.5 isn't included in the 10.0.0.0/24 address prefix, therefore the route with the 10.0.0.0/16 address prefix is the longest prefix that matches.

If multiple routes contain the same address prefix, Azure selects the route type, based on the following priority:

1. User-defined route
2. BGP route
3. System route

### NOTE

System routes for traffic related to virtual network, virtual network peerings, or virtual network service endpoints, are preferred routes, even if BGP routes are more specific.

For example, a route table contains the following routes:

SOURCE	ADDRESS PREFIXES	NEXT HOP TYPE
Default	0.0.0.0/0	Internet
User	0.0.0.0/0	Virtual network gateway

When traffic is destined for an IP address outside the address prefixes of any other routes in the route table, Azure selects the route with the **User** source, because user-defined routes are higher priority than system default routes.

See [Routing example](#) for a comprehensive routing table with explanations of the routes in the table.

## 0.0.0.0/0 address prefix

A route with the 0.0.0.0/0 address prefix instructs Azure how to route traffic destined for an IP address that is not within the address prefix of any other route in a subnet's route table. When a subnet is created, Azure creates a [default](#) route to the 0.0.0.0/0 address prefix, with the **Internet** next hop type. If you don't override this route, Azure routes all traffic destined to IP addresses not included in the address prefix of any other route, to the Internet. The exception is that traffic to the public IP addresses of Azure services remains on the Azure backbone network, and is not routed to the Internet. If you override this route, with a [custom](#) route, traffic destined to addresses not within the address prefixes of any other route in the route table is sent to a network virtual appliance or virtual network gateway, depending on which you specify in a custom route.

When you override the 0.0.0.0/0 address prefix, in addition to outbound traffic from the subnet flowing through the virtual network gateway or virtual appliance, the following changes occur with Azure's default routing:

- Azure sends all traffic to the next hop type specified in the route, including traffic destined for public IP addresses of Azure services. When the next hop type for the route with the 0.0.0.0/0 address prefix is **Internet**, traffic from the subnet destined to the public IP addresses of Azure services never leaves Azure's backbone network, regardless of the Azure region the virtual network or Azure service resource exist in. When you create a user-defined or BGP route with a **Virtual network gateway** or **Virtual appliance** next hop type however, all traffic, including traffic sent to public IP addresses of Azure services you haven't enabled [service endpoints](#) for, is sent to the next hop type specified in the route. If you've enabled a service endpoint for a service, traffic to the service is not routed to the next hop type in a route with the 0.0.0.0/0 address prefix, because address prefixes for the service are specified in the route that Azure creates when you enable the service endpoint, and the address prefixes for the service are longer than 0.0.0.0/0.
- You are no longer able to directly access resources in the subnet from the Internet. You can indirectly access resources in the subnet from the Internet, if inbound traffic passes through the device specified by the next hop type for a route with the 0.0.0.0/0 address prefix before reaching the resource in the virtual network. If the route contains the following values for next hop type:
  - **Virtual appliance:** The appliance must:
    - Be accessible from the Internet
    - Have a public IP address assigned to it,
    - Not have a network security group rule associated to it that prevents communication to the device
    - Not deny the communication
    - Be able to network address translate and forward, or proxy the traffic to the destination resource in the subnet, and return the traffic back to the Internet.
  - **Virtual network gateway:** If the gateway is an ExpressRoute virtual network gateway, an Internet-connected device on-premises can network address translate and forward, or proxy the

traffic to the destination resource in the subnet, via ExpressRoute's [private peering](#).

If your virtual network is connected to an Azure VPN gateway, do not associate a route table to the [gateway subnet](#) that includes a route with a destination of 0.0.0.0/0. Doing so can prevent the gateway from functioning properly. For details, see the *Why are certain ports opened on my VPN gateway?* question in the [VPN Gateway FAQ](#).

See [DMZ between Azure and your on-premises datacenter](#) and [DMZ between Azure and the Internet](#) for implementation details when using virtual network gateways and virtual appliances between the Internet and Azure.

## Routing example

To illustrate the concepts in this article, the sections that follow describe:

- A scenario, with requirements
- The custom routes necessary to meet the requirements
- The route table that exists for one subnet that includes the default and custom routes necessary to meet the requirements

### NOTE

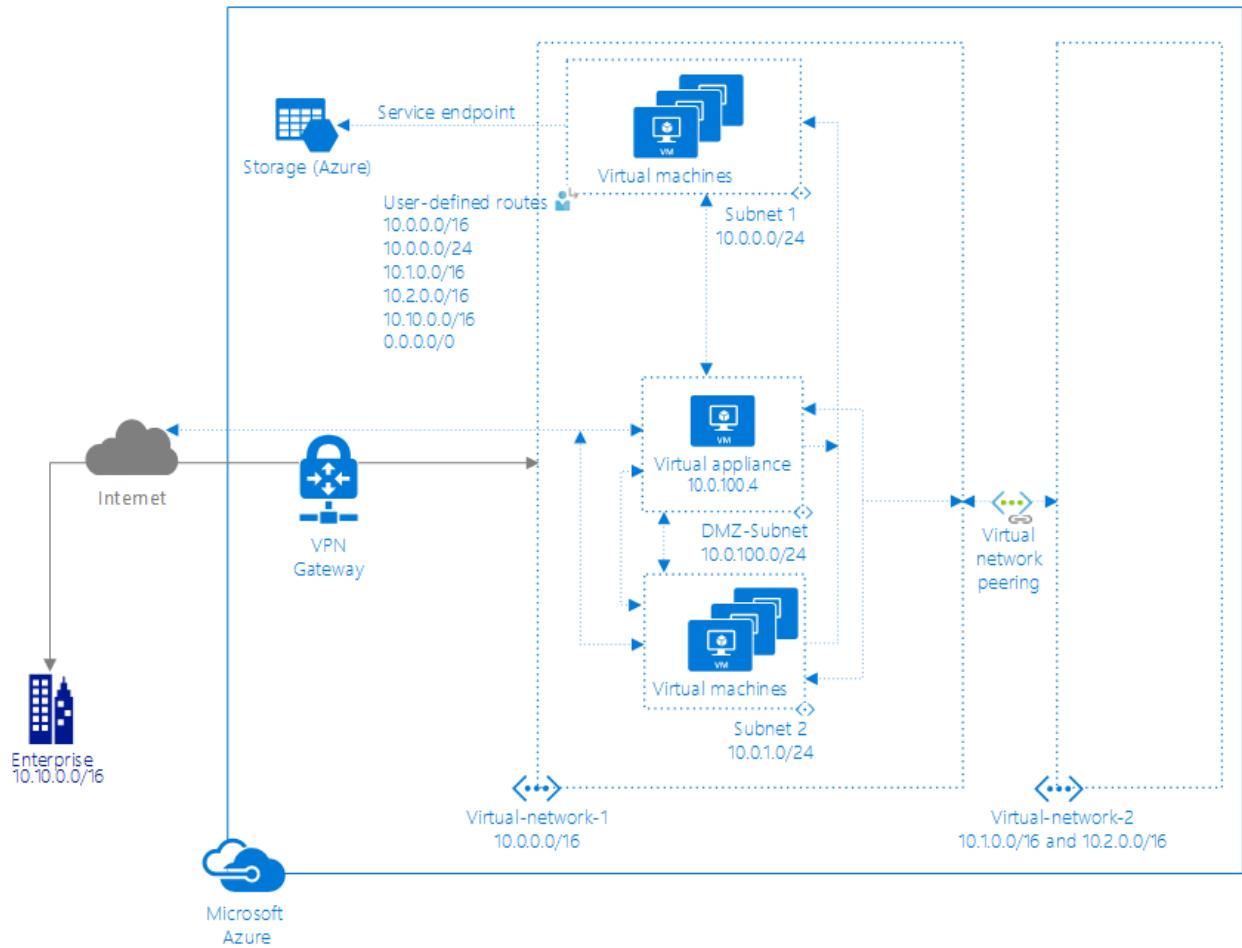
This example is not intended to be a recommended or best practice implementation. Rather, it is provided only to illustrate concepts in this article.

### Requirements

1. Implement two virtual networks in the same Azure region and enable resources to communicate between the virtual networks.
2. Enable an on-premises network to communicate securely with both virtual networks through a VPN tunnel over the Internet. *Alternatively, an ExpressRoute connection could be used, but in this example, a VPN connection is used.*
3. For one subnet in one virtual network:
  - Force all outbound traffic from the subnet, except to Azure Storage and within the subnet, to flow through a network virtual appliance, for inspection and logging.
  - Do not inspect traffic between private IP addresses within the subnet; allow traffic to flow directly between all resources.
  - Drop any outbound traffic destined for the other virtual network.
  - Enable outbound traffic to Azure storage to flow directly to storage, without forcing it through a network virtual appliance.
4. Allow all traffic between all other subnets and virtual networks.

### Implementation

The following picture shows an implementation through the Azure Resource Manager deployment model that meets the previous requirements:



Arrows show the flow of traffic.

## Route tables

### Subnet1

The route table for **Subnet1** in the picture contains the following routes:

ID	SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP IP ADDRESS	USER-DEFINED ROUTE NAME
1	Default	Invalid	10.0.0.0/16	Virtual network		
2	User	Active	10.0.0.0/16	Virtual appliance	10.0.100.4	Within-VNet1
3	User	Active	10.0.0.0/24	Virtual network		Within-Subnet1
4	Default	Invalid	10.1.0.0/16	VNet peering		
5	Default	Invalid	10.2.0.0/16	VNet peering		
6	User	Active	10.1.0.0/16	None		ToVNet2-1-Drop
7	User	Active	10.2.0.0/16	None		ToVNet2-2-Drop

ID	SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP IP ADDRESS	USER-DEFINED ROUTE NAME
8	Default	Invalid	10.10.0.0/16	Virtual network gateway	[X.X.X.X]	
9	User	Active	10.10.0.0/16	Virtual appliance	10.0.100.4	To-On-Prem
10	Default	Active	[X.X.X.X]	VirtualNetworkServiceEndpoint		
11	Default	Invalid	0.0.0.0/0	Internet		
12	User	Active	0.0.0.0/0	Virtual appliance	10.0.100.4	Default-NVA

An explanation of each route ID follows:

- Azure automatically added this route for all subnets within *Virtual-network-1*, because 10.0.0.0/16 is the only address range defined in the address space for the virtual network. If the user-defined route in route ID2 weren't created, traffic sent to any address between 10.0.0.1 and 10.0.255.254 would be routed within the virtual network, because the prefix is longer than 0.0.0.0/0, and not within the address prefixes of any of the other routes. Azure automatically changed the state from *Active* to *Invalid*, when ID2, a user-defined route, was added, since it has the same prefix as the default route, and user-defined routes override default routes. The state of this route is still *Active* for *Subnet2*, because the route table that user-defined route, ID2 is in, isn't associated to *Subnet2*.
- Azure added this route when a user-defined route for the 10.0.0.0/16 address prefix was associated to the *Subnet1* subnet in the *Virtual-network-1* virtual network. The user-defined route specifies 10.0.100.4 as the IP address of the virtual appliance, because the address is the private IP address assigned to the virtual appliance virtual machine. The route table this route exists in is not associated to *Subnet2*, so doesn't appear in the route table for *Subnet2*. This route overrides the default route for the 10.0.0.0/16 prefix (ID1), which automatically routed traffic addressed to 10.0.0.1 and 10.0.255.254 within the virtual network through the virtual network next hop type. This route exists to meet [requirement 3](#), to force all outbound traffic through a virtual appliance.
- Azure added this route when a user-defined route for the 10.0.0.0/24 address prefix was associated to the *Subnet1* subnet. Traffic destined for addresses between 10.0.0.1 and 10.0.0.254 remains within the subnet, rather than being routed to the virtual appliance specified in the previous rule (ID2), because it has a longer prefix than the ID2 route. This route was not associated to *Subnet2*, so the route does not appear in the route table for *Subnet2*. This route effectively overrides the ID2 route for traffic within *Subnet1*. This route exists to meet [requirement 3](#).
- Azure automatically added the routes in IDs 4 and 5 for all subnets within *Virtual-network-1*, when the virtual network was peered with *Virtual-network-2*. *Virtual-network-2* has two address ranges in its address space: 10.1.0.0/16 and 10.2.0.0/16, so Azure added a route for each range. If the user-defined routes in route IDs 6 and 7 weren't created, traffic sent to any address between 10.1.0.1-10.1.255.254 and 10.2.0.1-10.2.255.254 would be routed to the peered virtual network, because the prefix is longer than 0.0.0.0/0, and not within the address prefixes of any of the other routes. Azure automatically changed the state from *Active* to *Invalid*, when the routes in IDs 6 and 7 were added, since they have the same prefixes as the routes in IDs 4 and 5, and user-defined routes override default routes. The state of the routes in IDs 4 and 5 are still *Active* for *Subnet2*, because the route table that the user-defined routes in IDs 6 and 7 are in, isn't associated to *Subnet2*. A virtual network peering was created to meet [requirement 1](#).

5. Same explanation as ID4.
6. Azure added this route and the route in ID7, when user-defined routes for the 10.1.0.0/16 and 10.2.0.0/16 address prefixes were associated to the *Subnet1* subnet. Traffic destined for addresses between 10.1.0.1-10.1.255.254 and 10.2.0.1-10.2.255.254 is dropped by Azure, rather than being routed to the peered virtual network, because user-defined routes override default routes. The routes are not associated to *Subnet2*, so the routes do not appear in the route table for *Subnet2*. The routes override the ID4 and ID5 routes for traffic leaving *Subnet1*. The ID6 and ID7 routes exist to meet [requirement 3](#) to drop traffic destined to the other virtual network.
7. Same explanation as ID6.
8. Azure automatically added this route for all subnets within *Virtual-network-1* when a VPN type virtual network gateway was created within the virtual network. Azure added the public IP address of the virtual network gateway to the route table. Traffic sent to any address between 10.10.0.1 and 10.10.255.254 is routed to the virtual network gateway. The prefix is longer than 0.0.0.0/0 and not within the address prefixes of any of the other routes. A virtual network gateway was created to meet [requirement 2](#).
9. Azure added this route when a user-defined route for the 10.10.0.0/16 address prefix was added to the route table associated to *Subnet1*. This route overrides ID8. The route sends all traffic destined for the on-premises network to an NVA for inspection, rather than routing traffic directly on-premises. This route was created to meet [requirement 3](#).
10. Azure automatically added this route to the subnet when a service endpoint to an Azure service was enabled for the subnet. Azure routes traffic from the subnet to a public IP address of the service, over the Azure infrastructure network. The prefix is longer than 0.0.0.0/0 and not within the address prefixes of any of the other routes. A service endpoint was created to meet [requirement 3](#), to enable traffic destined for Azure Storage to flow directly to Azure Storage.
11. Azure automatically added this route to the route table of all subnets within *Virtual-network-1* and *Virtual-network-2*. The 0.0.0.0/0 address prefix is the shortest prefix. Any traffic sent to addresses within a longer address prefix are routed based on other routes. By default, Azure routes all traffic destined for addresses other than the addresses specified in one of the other routes to the Internet. Azure automatically changed the state from *Active* to *Invalid* for the *Subnet1* subnet when a user-defined route for the 0.0.0.0/0 address prefix (ID12) was associated to the subnet. The state of this route is still *Active* for all other subnets within both virtual networks, because the route isn't associated to any other subnets within any other virtual networks.
12. Azure added this route when a user-defined route for the 0.0.0.0/0 address prefix was associated to the *Subnet1* subnet. The user-defined route specifies 10.0.100.4 as the IP address of the virtual appliance. This route is not associated to *Subnet2*, so the route does not appear in the route table for *Subnet2*. All traffic for any address not included in the address prefixes of any of the other routes is sent to the virtual appliance. The addition of this route changed the state of the default route for the 0.0.0.0/0 address prefix (ID11) from *Active* to *Invalid* for *Subnet1*, because a user-defined route overrides a default route. This route exists to meet the third [requirement](#).

## **Subnet2**

The route table for *Subnet2* in the picture contains the following routes:

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP IP ADDRESS
Default	Active	10.0.0.0/16	Virtual network	
Default	Active	10.1.0.0/16	VNet peering	
Default	Active	10.2.0.0/16	VNet peering	
Default	Active	10.10.0.0/16	Virtual network gateway	[X.X.X.X]

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE	NEXT HOP IP ADDRESS
Default	Active	0.0.0.0/0	Internet	
Default	Active	10.0.0.0/8	None	
Default	Active	100.64.0.0/10	None	
Default	Active	192.168.0.0/16	None	

The route table for *Subnet2* contains all Azure-created default routes and the optional VNet peering and Virtual network gateway optional routes. Azure added the optional routes to all subnets in the virtual network when the gateway and peering were added to the virtual network. Azure removed the routes for the 10.0.0.0/8, 192.168.0.0/16, and 100.64.0.0/10 address prefixes from the *Subnet1* route table when the user-defined route for the 0.0.0.0/0 address prefix was added to *Subnet1*.

## Next steps

- [Create a user-defined route table with routes and a network virtual appliance](#)
- [Configure BGP for an Azure VPN Gateway](#)
- [Use BGP with ExpressRoute](#)
- [View all routes for a subnet.](#) A user-defined route table only shows you the user-defined routes, not the default, and BGP routes for a subnet. Viewing all routes shows you the default, BGP, and user-defined routes for the subnet a network interface is in.
- [Determine the next hop type](#) between a virtual machine and a destination IP address. The Azure Network Watcher next hop feature enables you to determine whether traffic is leaving a subnet and being routed to where you think it should be.

# Security groups

1/23/2020 • 16 minutes to read • [Edit Online](#)

You can filter network traffic to and from Azure resources in an Azure [virtual network](#) with a network security group. A network security group contains [security rules](#) that allow or deny inbound network traffic to, or outbound network traffic from, several types of Azure resources. To learn about which Azure resources can be deployed into a virtual network and have network security groups associated to them, see [Virtual network integration for Azure services](#). For each rule, you can specify source and destination, port, and protocol.

This article explains network security group concepts, to help you use them effectively. If you've never created a network security group, you can complete a quick [tutorial](#) to get some experience creating one. If you're familiar with network security groups and need to manage them, see [Manage a network security group](#). If you're having communication problems and need to troubleshoot network security groups, see [Diagnose a virtual machine network traffic filter problem](#). You can enable [network security group flow logs](#) to [analyze network traffic](#) to and from resources that have an associated network security group.

## Security rules

A network security group contains zero, or as many rules as desired, within Azure subscription [limits](#). Each rule specifies the following properties:

PROPERTY	EXPLANATION
Name	A unique name within the network security group.
Priority	A number between 100 and 4096. Rules are processed in priority order, with lower numbers processed before higher numbers, because lower numbers have higher priority. Once traffic matches a rule, processing stops. As a result, any rules that exist with lower priorities (higher numbers) that have the same attributes as rules with higher priorities are not processed.
Source or destination	Any, or an individual IP address, classless inter-domain routing (CIDR) block (10.0.0.0/24, for example), <a href="#">service tag</a> , or <a href="#">application security group</a> . If you specify an address for an Azure resource, specify the private IP address assigned to the resource. Network security groups are processed after Azure translates a public IP address to a private IP address for inbound traffic, and before Azure translates a private IP address to a public IP address for outbound traffic. Learn more about Azure <a href="#">IP addresses</a> . Specifying a range, a service tag, or application security group, enables you to create fewer security rules. The ability to specify multiple individual IP addresses and ranges (you cannot specify multiple service tags or application groups) in a rule is referred to as <a href="#">augmented security rules</a> . Augmented security rules can only be created in network security groups created through the Resource Manager deployment model. You cannot specify multiple IP addresses and IP address ranges in network security groups created through the classic deployment model. Learn more about <a href="#">Azure deployment models</a> .
Protocol	TCP, UDP, ICMP or Any.

PROPERTY	EXPLANATION
Direction	Whether the rule applies to inbound, or outbound traffic.
Port range	You can specify an individual or range of ports. For example, you could specify 80 or 10000-10005. Specifying ranges enables you to create fewer security rules. Augmented security rules can only be created in network security groups created through the Resource Manager deployment model. You cannot specify multiple ports or port ranges in the same security rule in network security groups created through the classic deployment model.
Action	Allow or deny

Network security group security rules are evaluated by priority using the 5-tuple information (source, source port, destination, destination port, and protocol) to allow or deny the traffic. A flow record is created for existing connections. Communication is allowed or denied based on the connection state of the flow record. The flow record allows a network security group to be stateful. If you specify an outbound security rule to any address over port 80, for example, it's not necessary to specify an inbound security rule for the response to the outbound traffic. You only need to specify an inbound security rule if communication is initiated externally. The opposite is also true. If inbound traffic is allowed over a port, it's not necessary to specify an outbound security rule to respond to traffic over the port. Existing connections may not be interrupted when you remove a security rule that enabled the flow. Traffic flows are interrupted when connections are stopped and no traffic is flowing in either direction, for at least a few minutes.

There are limits to the number of security rules you can create in a network security group. For details, see [Azure limits](#).

## Augmented security rules

Augmented security rules simplify security definition for virtual networks, allowing you to define larger and complex network security policies, with fewer rules. You can combine multiple ports and multiple explicit IP addresses and ranges into a single, easily understood security rule. Use augmented rules in the source, destination, and port fields of a rule. To simplify maintenance of your security rule definition, combine augmented security rules with [service tags](#) or [application security groups](#). There are limits to the number of addresses, ranges, and ports that you can specify in a rule. For details, see [Azure limits](#).

## Service tags

A service tag represents a group of IP address prefixes from a given Azure service. It helps to minimize complexity of frequent updates on network security rules.

For more information, see [Azure service tags](#).

## Default security rules

Azure creates the following default rules in each network security group that you create:

### Inbound

**AllowVNetInBound**

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65000	VirtualNetwork	0-65535	VirtualNetwork	0-65535	Any	Allow

#### AllowAzureLoadBalancerInBound

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65001	AzureLoadBalancer	0-65535	0.0.0.0/0	0-65535	Any	Allow

#### DenyAllInbound

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65500	0.0.0.0/0	0-65535	0.0.0.0/0	0-65535	Any	Deny

#### Outbound

##### AllowVnetOutBound

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65000	VirtualNetwork	0-65535	VirtualNetwork	0-65535	Any	Allow

##### AllowInternetOutBound

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65001	0.0.0.0/0	0-65535	Internet	0-65535	Any	Allow

##### DenyAllOutBound

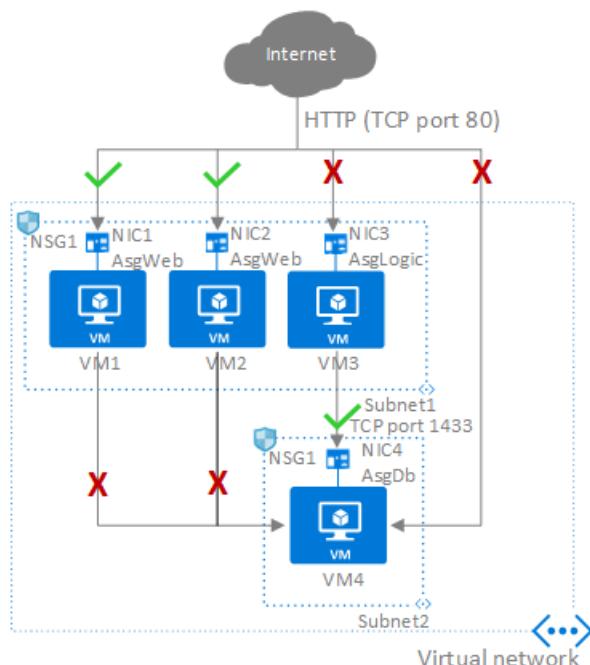
PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65500	0.0.0.0/0	0-65535	0.0.0.0/0	0-65535	Any	Deny

In the **Source** and **Destination** columns, *VirtualNetwork*, *AzureLoadBalancer*, and *Internet* are [service tags](#), rather than IP addresses. In the protocol column, **Any** encompasses TCP, UDP, and ICMP. When creating a rule, you can specify TCP, UDP, ICMP or Any. *0.0.0.0/0* in the **Source** and **Destination** columns represents all addresses. Clients like Azure portal, Azure CLI, or Powershell can use \* or any for this expression.

You cannot remove the default rules, but you can override them by creating rules with higher priorities.

## Application security groups

Application security groups enable you to configure network security as a natural extension of an application's structure, allowing you to group virtual machines and define network security policies based on those groups. You can reuse your security policy at scale without manual maintenance of explicit IP addresses. The platform handles the complexity of explicit IP addresses and multiple rule sets, allowing you to focus on your business logic. To better understand application security groups, consider the following example:



In the previous picture, *NIC1* and *NIC2* are members of the *AsgWeb* application security group. *NIC3* is a member of the *AsgLogic* application security group. *NIC4* is a member of the *AsgDb* application security group. Though each network interface in this example is a member of only one application security group, a network interface can be a member of multiple application security groups, up to the [Azure limits](#). None of the network interfaces have an associated network security group. *NSG1* is associated to both subnets and contains the following rules:

#### Allow-HTTP-Inbound-Internet

This rule is needed to allow traffic from the internet to the web servers. Because inbound traffic from the internet is denied by the [DenyAllInbound](#) default security rule, no additional rule is needed for the *AsgLogic* or *AsgDb* application security groups.

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
100	Internet	*	AsgWeb	80	TCP	Allow

#### Deny-Database-All

Because the [AllowVNetInBound](#) default security rule allows all communication between resources in the same virtual network, this rule is needed to deny traffic from all resources.

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
120	*	*	AsgDb	1433	Any	Deny

#### Allow-Database-BusinessLogic

This rule allows traffic from the *AsgLogic* application security group to the *AsgDb* application security group. The priority for this rule is higher than the priority for the *Deny-Database-All* rule. As a result, this rule is processed before the *Deny-Database-All* rule, so traffic from the *AsgLogic* application security group is allowed, whereas all other traffic is blocked.

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
110	AsgLogic	*	AsgDb	1433	TCP	Allow

The rules that specify an application security group as the source or destination are only applied to the network interfaces that are members of the application security group. If the network interface is not a member of an application security group, the rule is not applied to the network interface, even though the network security group is associated to the subnet.

Application security groups have the following constraints:

- There are limits to the number of application security groups you can have in a subscription, as well as other limits related to application security groups. For details, see [Azure limits](#).
- You can specify one application security group as the source and destination in a security rule. You cannot specify multiple application security groups in the source or destination.
- All network interfaces assigned to an application security group have to exist in the same virtual network that the first network interface assigned to the application security group is in. For example, if the first network interface assigned to an application security group named *AsgWeb* is in the virtual network named *VNet1*, then all subsequent network interfaces assigned to *ASGWeb* must exist in *VNet1*. You cannot add network interfaces from different virtual networks to the same application security group.
- If you specify an application security group as the source and destination in a security rule, the network interfaces in both application security groups must exist in the same virtual network. For example, if *AsgLogic* contained network interfaces from *VNet1*, and *AsgDb* contained network interfaces from *VNet2*, you could not assign *AsgLogic* as the source and *AsgDb* as the destination in a rule. All network interfaces for both the source and destination application security groups need to exist in the same virtual network.

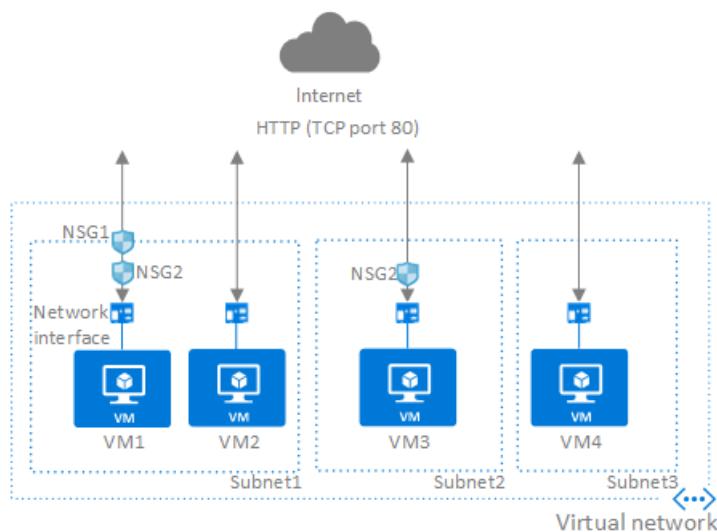
#### TIP

To minimize the number of security rules you need, and the need to change the rules, plan out the application security groups you need and create rules using service tags or application security groups, rather than individual IP addresses, or ranges of IP addresses, whenever possible.

## How traffic is evaluated

You can deploy resources from several Azure services into an Azure virtual network. For a complete list, see [Services that can be deployed into a virtual network](#). You can associate zero, or one, network security group to each virtual network [subnet](#) and [network interface](#) in a virtual machine. The same network security group can be associated to as many subnets and network interfaces as you choose.

The following picture illustrates different scenarios for how network security groups might be deployed to allow network traffic to and from the internet over TCP port 80:



Reference the previous picture, along with the following text, to understand how Azure processes inbound and outbound rules for network security groups:

### Inbound traffic

For inbound traffic, Azure processes the rules in a network security group associated to a subnet first, if there is one, and then the rules in a network security group associated to the network interface, if there is one.

- **VM1:** The security rules in *NSG1* are processed, since it is associated to *Subnet1* and *VM1* is in *Subnet1*. Unless you've created a rule that allows port 80 inbound, the traffic is denied by the [DenyAllInbound](#) default security rule, and never evaluated by *NSG2*, since *NSG2* is associated to the network interface. If *NSG1* has a security rule that allows port 80, the traffic is then processed by *NSG2*. To allow port 80 to the virtual machine, both *NSG1* and *NSG2* must have a rule that allows port 80 from the internet.
- **VM2:** The rules in *NSG1* are processed because *VM2* is also in *Subnet1*. Since *VM2* does not have a network security group associated to its network interface, it receives all traffic allowed through *NSG1* or is denied all traffic denied by *NSG1*. Traffic is either allowed or denied to all resources in the same subnet when a network security group is associated to a subnet.
- **VM3:** Since there is no network security group associated to *Subnet2*, traffic is allowed into the subnet and processed by *NSG2*, because *NSG2* is associated to the network interface attached to *VM3*.
- **VM4:** Traffic is allowed to *VM4*, because a network security group isn't associated to *Subnet3*, or the network interface in the virtual machine. All network traffic is allowed through a subnet and network interface if they don't have a network security group associated to them.

### Outbound traffic

For outbound traffic, Azure processes the rules in a network security group associated to a network interface first, if there is one, and then the rules in a network security group associated to the subnet, if there is one.

- **VM1:** The security rules in *NSG2* are processed. Unless you create a security rule that denies port 80 outbound to the internet, the traffic is allowed by the [AllowInternetOutbound](#) default security rule in both *NSG1* and *NSG2*. If *NSG2* has a security rule that denies port 80, the traffic is denied, and never evaluated by *NSG1*. To deny port 80 from the virtual machine, either, or both of the network security groups must have a rule that denies port 80 to the internet.
- **VM2:** All traffic is sent through the network interface to the subnet, since the network interface attached to *VM2* does not have a network security group associated to it. The rules in *NSG1* are processed.
- **VM3:** If *NSG2* has a security rule that denies port 80, the traffic is denied. If *NSG2* has a security rule that allows port 80, then port 80 is allowed outbound to the internet, since a network security group is not associated to *Subnet2*.
- **VM4:** All network traffic is allowed from *VM4*, because a network security group isn't associated to the network interface attached to the virtual machine, or to *Subnet3*.

## Intra-Subnet traffic

It's important to note that security rules in an NSG associated to a subnet can affect connectivity between VM's within it. For example, if a rule is added to *NSG1* which denies all inbound and outbound traffic, *VM1* and *VM2* will no longer be able to communicate with each other. Another rule would have to be added specifically to allow this.

You can easily view the aggregate rules applied to a network interface by viewing the [effective security rules](#) for a network interface. You can also use the [IP flow verify](#) capability in Azure Network Watcher to determine whether communication is allowed to or from a network interface. IP flow verify tells you whether communication is allowed or denied, and which network security rule allows or denies the traffic.

### NOTE

Network security groups are associated to subnets or to virtual machines and cloud services deployed in the classic deployment model, and to subnets or network interfaces in the Resource Manager deployment model. To learn more about Azure deployment models, see [Understand Azure deployment models](#).

### TIP

Unless you have a specific reason to, we recommended that you associate a network security group to a subnet, or a network interface, but not both. Since rules in a network security group associated to a subnet can conflict with rules in a network security group associated to a network interface, you can have unexpected communication problems that require troubleshooting.

## Azure platform considerations

- **Virtual IP of the host node:** Basic infrastructure services such as DHCP, DNS, IMDS, and health monitoring are provided through the virtualized host IP addresses 168.63.129.16 and 169.254.169.254. These IP addresses belong to Microsoft and are the only virtualized IP addresses used in all regions for this purpose.
- **Licensing (Key Management Service):** Windows images running in virtual machines must be licensed. To ensure licensing, a request is sent to the Key Management Service host servers that handle such queries. The request is made outbound through port 1688. For deployments using [default route 0.0.0.0/0](#) configuration, this platform rule will be disabled.
- **Virtual machines in load-balanced pools:** The source port and address range applied are from the originating computer, not the load balancer. The destination port and address range are for the destination computer, not the load balancer.
- **Azure service instances:** Instances of several Azure services, such as HDInsight, Application Service Environments, and Virtual Machine Scale Sets are deployed in virtual network subnets. For a complete list of services you can deploy into virtual networks, see [Virtual network for Azure services](#). Ensure you familiarize yourself with the port requirements for each service before applying a network security group to the subnet the resource is deployed in. If you deny ports required by the service, the service doesn't function properly.
- **Sending outbound email:** Microsoft recommends that you utilize authenticated SMTP relay services (typically connected via TCP port 587, but often others, as well) to send email from Azure Virtual Machines. SMTP relay services specialize in sender reputation, to minimize the possibility that third-party email providers reject messages. Such SMTP relay services include, but are not limited to, Exchange Online Protection and SendGrid. Use of SMTP relay services is in no way restricted in Azure, regardless of your subscription type.

If you created your Azure subscription prior to November 15, 2017, in addition to being able to use SMTP relay services, you can send email directly over TCP port 25. If you created your subscription after November 15, 2017, you may not be able to send email directly over port 25. The behavior of outbound communication over port 25 depends on the type of subscription you have, as follows:

- **Enterprise Agreement:** Outbound port 25 communication is allowed. You are able to send outbound email directly from virtual machines to external email providers, with no restrictions from the Azure platform.
- **Pay-as-you-go:** Outbound port 25 communication is blocked from all resources. If you need to send email from a virtual machine directly to external email providers (not using an authenticated SMTP relay), you can make a request to remove the restriction. Requests are reviewed and approved at Microsoft's discretion and are only granted after anti-fraud checks are performed. To make a request, open a support case with the issue type *Technical, Virtual Network Connectivity, Cannot send e-mail (SMTP/Port 25)*. In your support case, include details about why your subscription needs to send email directly to mail providers, instead of going through an authenticated SMTP relay. If your subscription is exempted, only virtual machines created after the exemption date are able to communicate outbound over port 25.
- **MSDN, Azure Pass, Azure in Open, Education, BizSpark, and Free trial:** Outbound port 25 communication is blocked from all resources. No requests to remove the restriction can be made, because requests are not granted. If you need to send email from your virtual machine, you have to use an SMTP relay service.
- **Cloud service provider:** Customers that are consuming Azure resources via a cloud service provider can create a support case with their cloud service provider, and request that the provider create an unblock case on their behalf, if a secure SMTP relay cannot be used.

If Azure allows you to send email over port 25, Microsoft cannot guarantee email providers will accept inbound email from your virtual machine. If a specific provider rejects mail from your virtual machine, work directly with the provider to resolve any message delivery or spam filtering issues, or use an authenticated SMTP relay service.

## Next steps

- Learn how to [Create a network security group](#).

# Virtual network service tags

2/21/2020 • 8 minutes to read • [Edit Online](#)

A service tag represents a group of IP address prefixes from a given Azure service. Microsoft manages the address prefixes encompassed by the service tag and automatically updates the service tag as addresses change, minimizing the complexity of frequent updates to network security rules.

You can use service tags to define network access controls on [network security groups](#) or [Azure Firewall](#). Use service tags in place of specific IP addresses when you create security rules. By specifying the service tag name (for example, **ApiManagement**) in the appropriate *source* or *destination* field of a rule, you can allow or deny the traffic for the corresponding service.

You can use service tags to achieve network isolation and protect your Azure resources from the general Internet while accessing Azure services that have public endpoints. Create inbound/outbound network security group rules to deny traffic to/from **Internet** and allow traffic to/from **AzureCloud** or other [available service tags](#) of specific Azure services.

## Available service tags

The following table includes all the service tags available for use in [network security group](#) rules.

The columns indicate whether the tag:

- Is suitable for rules that cover inbound or outbound traffic.
- Supports [regional](#) scope.
- Is usable in [Azure Firewall](#) rules.

By default, service tags reflect the ranges for the entire cloud. Some service tags also allow more granular control by restricting the corresponding IP ranges to a specified region. For example, the service tag **Storage** represents Azure Storage for the entire cloud, but **Storage.WestUS** narrows the range to only the storage IP address ranges from the WestUS region. The following table indicates whether each service tag supports such regional scope.

TAG	PURPOSE	CAN USE INBOUND OR OUTBOUND?	CAN BE REGIONAL?	CAN USE WITH AZURE FIREWALL?
<b>ApiManagement</b>	Management traffic for Azure API Management dedicated deployments.  <i>Note:</i> This tag represents the Azure API Management service endpoint for control plane per region. This enables customers to perform management operations on the APIs, Operations, Policies, NamedValues configured on the API Management service.	Inbound	Yes	Yes

TAG	PURPOSE	CAN USE INBOUND OR OUTBOUND?	CAN BE REGIONAL?	CAN USE WITH AZURE FIREWALL?
<b>ApplicationInsightsAvailability</b>	Application Insights Availability.	Inbound	No	No
<b>AppService</b>	Azure App Service. This tag is recommended for outbound security rules to web app front ends.	Outbound	Yes	Yes
<b>AppServiceManagement</b>	Management traffic for deployments dedicated to App Service Environment.	Both	No	Yes
<b>AzureActiveDirectory</b>	Azure Active Directory.	Outbound	No	Yes
<b>AzureActiveDirectoryDomainServices</b>	Management traffic for deployments dedicated to Azure Active Directory Domain Services.	Both	No	Yes
<b>AzureAdvancedThreatProtection</b>	Azure Advanced Threat Protection.	Outbound	No	No
<b>AzureBackup</b>	Azure Backup.  <i>Note:</i> This tag has a dependency on the <b>Storage</b> and <b>AzureActiveDirectory</b> tags.	Outbound	No	Yes
<b>AzureBotService</b>	Azure Bot Service.	Outbound	No	No
<b>AzureCloud</b>	All <a href="#">datacenter public IP addresses</a> .	Outbound	Yes	Yes
<b>AzureCognitiveSearch</b>	Azure Cognitive Search.  This tag or the IP addresses covered by this tag can be used to grant indexers secure access to data sources. Refer the <a href="#">indexer connection documentation</a> for more details.	Inbound	No	No

TAG	PURPOSE	CAN USE INBOUND OR OUTBOUND?	CAN BE REGIONAL?	CAN USE WITH AZURE FIREWALL?
<b>AzureConnectors</b>	Azure Logic Apps connectors for probe/back-end connections.	Inbound	Yes	Yes
<b>AzureContainerRegistry</b>	Azure Container Registry.	Outbound	Yes	Yes
<b>AzureCosmosDB</b>	Azure Cosmos DB.	Outbound	Yes	Yes
<b>AzureDatabricks</b>	Azure Databricks.	Both	No	No
<b>AzureDataExplorerManagement</b>	Azure Data Explorer Management.	Inbound	No	No
<b>AzureDataLake</b>	Azure Data Lake.	Outbound	No	Yes
<b>AzureEventGrid</b>	Azure Event Grid.  <i>Note:</i> This tag covers Azure Event Grid endpoints in US South Central, US East, US East 2, US West 2, and US Central only.	Both	No	No
<b>AzureFrontDoor</b>	Azure Front Door.	Both	No	No
<b>AzureInformationProtection</b>	Azure Information Protection.  <i>Note:</i> This tag has a dependency on the <b>AzureActiveDirectory</b> and <b>AzureFrontDoor.Frontend</b> tags. Please also whitelist following IPs (this dependency will be removed soon): 13.107.6.181 & 13.107.9.181.	Outbound	No	No
<b>AzureIoTHub</b>	Azure IoT Hub.	Outbound	No	No
<b>AzureKeyVault</b>	Azure Key Vault.  <i>Note:</i> This tag has a dependency on the <b>AzureActiveDirectory</b> tag.	Outbound	Yes	Yes

TAG	PURPOSE	CAN USE INBOUND OR OUTBOUND?	CAN BE REGIONAL?	CAN USE WITH AZURE FIREWALL?
<b>AzureLoadBalancer</b>	The Azure infrastructure load balancer. The tag translates to the <a href="#">virtual IP address of the host</a> (168.63.129.16) where the Azure health probes originate. This does not include traffic to your Azure Load Balancer resource. If you're not using Azure Load Balancer, you can override this rule.	Both	No	No
<b>AzureMachineLearning</b>	Azure Machine Learning.	Both	No	Yes
<b>AzureMonitor</b>	Log Analytics, Application Insights, AzMon, and custom metrics (GiG endpoints).  <i>Note:</i> For Log Analytics, this tag has a dependency on the <b>Storage</b> tag.	Outbound	No	Yes
<b>AzurePlatformDNS</b>	The basic infrastructure (default) DNS service.  You can use this tag to disable the default DNS. Be cautious when you use this tag. We recommend that you read <a href="#">Azure platform considerations</a> . We also recommend that you perform testing before you use this tag.	Outbound	No	No

TAG	PURPOSE	CAN USE INBOUND OR OUTBOUND?	CAN BE REGIONAL?	CAN USE WITH AZURE FIREWALL?
<b>AzurePlatformIMDS</b>	<p>Azure Instance Metadata Service (IMDS), which is a basic infrastructure service.</p> <p>You can use this tag to disable the default IMDS. Be cautious when you use this tag. We recommend that you read <a href="#">Azure platform considerations</a>. We also recommend that you perform testing before you use this tag.</p>	Outbound	No	No
<b>AzurePlatformLKM</b>	<p>Windows licensing or key management service.</p> <p>You can use this tag to disable the defaults for licensing. Be cautious when you use this tag. We recommend that you read <a href="#">Azure platform considerations</a>. We also recommend that you perform testing before you use this tag.</p>	Outbound	No	No
<b>AzureResourceManager</b>	Azure Resource Manager.	Outbound	No	No
<b>AzureSiteRecovery</b>	<p>Azure Site Recovery.</p> <p><i>Note:</i> This tag has a dependency on the <b>Storage</b>, <b>AzureActiveDirectory</b>, and <b>EventHub</b> tags.</p>	Outbound	No	No
<b>AzureTrafficManager</b>	<p>Azure Traffic Manager probe IP addresses.</p> <p>For more information on Traffic Manager probe IP addresses, see <a href="#">Azure Traffic Manager FAQ</a>.</p>	Inbound	No	Yes

TAG	PURPOSE	CAN USE INBOUND OR OUTBOUND?	CAN BE REGIONAL?	CAN USE WITH AZURE FIREWALL?
<b>BatchNodeManagement</b>	Management traffic for deployments dedicated to Azure Batch.	Both	No	Yes
<b>CognitiveServicesManagement</b>	The address ranges for traffic for Azure Cognitive Services.	Outbound	No	No
<b>Dynamics365ForMarketingEmail</b>	The address ranges for the marketing email service of Dynamics 365.	Outbound	Yes	No
<b>ElasticAFD</b>	Elastic Azure Front Door.	Both	No	No
<b>EventHub</b>	Azure Event Hubs.	Outbound	Yes	Yes
<b>GatewayManager</b>	Management traffic for deployments dedicated to Azure VPN Gateway and Application Gateway.	Inbound	No	No
<b>GuestAndHybridManagement</b>	Azure Automation and Guest Configuration.	Outbound	No	Yes
<b>HDInsight</b>	Azure HDInsight.	Inbound	Yes	No
<b>Internet</b>	The IP address space that's outside the virtual network and reachable by the public internet.  The address range includes the <a href="#">Azure-owned public IP address space</a> .	Both	No	No
<b>MicrosoftCloudAppSecurity</b>	Microsoft Cloud App Security.	Outbound	No	No
<b>MicrosoftContainerRegistry</b>	Container registry for Microsoft container images.  <i>Note:</i> Please also whitelist following IP (this dependency will be removed soon): 204.79.197.219.	Outbound	Yes	Yes

<b>TAG</b>	<b>PURPOSE</b>	<b>CAN USE INBOUND OR OUTBOUND?</b>	<b>CAN BE REGIONAL?</b>	<b>CAN USE WITH AZURE FIREWALL?</b>
<b>ServiceBus</b>	Azure Service Bus traffic that uses the Premium service tier.	Outbound	Yes	Yes
<b>ServiceFabric</b>	Azure Service Fabric.  <i>Note:</i> This tag represents the Service Fabric service endpoint for control plane per region. This enables customers to perform management operations for their Service Fabric clusters from their VNET (endpoint eg. <a href="https://westus.servicefabric.azure.com">https://westus.servicefabric.azure.com</a> )	Both	No	No
<b>Sql</b>	Azure SQL Database, Azure Database for MySQL, Azure Database for PostgreSQL, and Azure SQL Data Warehouse.  <i>Note:</i> This tag represents the service, but not specific instances of the service. For example, the tag represents the Azure SQL Database service, but not a specific SQL database or server.	Outbound	Yes	Yes
<b>SqlManagement</b>	Management traffic for SQL-dedicated deployments.	Both	No	Yes
<b>Storage</b>	Azure Storage.  <i>Note:</i> This tag represents the service, but not specific instances of the service. For example, the tag represents the Azure Storage service, but not a specific Azure Storage account.	Outbound	Yes	Yes

TAG	PURPOSE	CAN USE INBOUND OR OUTBOUND?	CAN BE REGIONAL?	CAN USE WITH AZURE FIREWALL?
<b>VirtualNetwork</b>	The virtual network address space (all IP address ranges defined for the virtual network), all connected on-premises address spaces, <a href="#">peered</a> virtual networks, virtual networks connected to a <a href="#">virtual network gateway</a> , the <a href="#">virtual IP address of the host</a> , and address prefixes used on <a href="#">user-defined routes</a> . This tag might also contain default routes.	Both	No	No

#### NOTE

In the classic deployment model (before Azure Resource Manager), a subset of the tags listed in the previous table are supported. These tags are spelled differently:

CLASSIC SPELLING	EQUIVALENT RESOURCE MANAGER TAG
AZURE_LOADBALANCER	AzureLoadBalancer
INTERNET	Internet
VIRTUAL_NETWORK	VirtualNetwork

#### NOTE

Service tags of Azure services denote the address prefixes from the specific cloud being used. For example, the underlying IP ranges that correspond to the [Sql](#) tag value on the Azure Public cloud will be different from the underlying ranges on the Azure China cloud.

#### NOTE

If you implement a [virtual network service endpoint](#) for a service, such as Azure Storage or Azure SQL Database, Azure adds a [route](#) to a virtual network subnet for the service. The address prefixes in the route are the same address prefixes, or CIDR ranges, as those of the corresponding service tag.

## Service tags on-premises

You can obtain the current service tag and range information to include as part of your on-premises firewall configurations. This information is the current point-in-time list of the IP ranges that correspond to each service tag. You can obtain the information programmatically or via a JSON file download, as described in the following sections.

## Use the Service Tag Discovery API (public preview)

You can programmatically retrieve the current list of service tags together with IP address range details:

- [REST](#)
- [Azure PowerShell](#)
- [Azure CLI](#)

### NOTE

While it's in public preview, the Discovery API might return information that's less current than information returned by the JSON downloads. (See the next section.)

## Discover service tags by using downloadable JSON files

You can download JSON files that contain the current list of service tags together with IP address range details. These lists are updated and published weekly. Locations for each cloud are:

- [Azure Public](#)
- [Azure US Government](#)
- [Azure China](#)
- [Azure Germany](#)

### NOTE

A subset of this information has been published in XML files for [Azure Public](#), [Azure China](#), and [Azure Germany](#). These XML downloads will be deprecated by June 30, 2020 and will no longer be available after that date. You should migrate to using the Discovery API or JSON file downloads as described in the previous sections.

## Tips

- You can detect updates from one publication to the next by noting increased *changeNumber* values in the JSON file. Each subsection (for example, **Storage.WestUS**) has its own *changeNumber* that's incremented as changes occur. The top level of the file's *changeNumber* is incremented when any of the subsections is changed.
- For examples of how to parse the service tag information (for example, get all address ranges for Storage in WestUS), see the [Service Tag Discovery API PowerShell](#) documentation.

## Next steps

- Learn how to [create a network security group](#).

# Virtual Network service endpoints

11/22/2019 • 11 minutes to read • [Edit Online](#)

Virtual Network (VNet) service endpoints extend your virtual network private address space. The endpoints also extend the identity of your VNet to the Azure services over a direct connection. Endpoints allow you to secure your critical Azure service resources to only your virtual networks. Traffic from your VNet to the Azure service always remains on the Microsoft Azure backbone network.

This feature is available for the following Azure services and regions. The *Microsoft.\** resource is in parenthesis. Enable this resource from the subnet side while configuring service endpoints for your service:

## Generally available

- **Azure Storage** (*Microsoft.Storage*): Generally available in all Azure regions.
- **Azure SQL Database** (*Microsoft.Sql*): Generally available in all Azure regions.
- **Azure SQL Data Warehouse** (*Microsoft.Sql*): Generally available in all Azure regions.
- **Azure Database for PostgreSQL server** (*Microsoft.Sql*): Generally available in Azure regions where database service is available.
- **Azure Database for MySQL server** (*Microsoft.Sql*): Generally available in Azure regions where database service is available.
- **Azure Database for MariaDB** (*Microsoft.Sql*): Generally available in Azure regions where database service is available.
- **Azure Cosmos DB** (*Microsoft.AzureCosmosDB*): Generally available in all Azure regions.
- **Azure Key Vault** (*Microsoft.KeyVault*): Generally available in all Azure regions.
- **Azure Service Bus** (*Microsoft.ServiceBus*): Generally available in all Azure regions.
- **Azure Event Hubs** (*Microsoft.EventHub*): Generally available in all Azure regions.
- **Azure Data Lake Store Gen 1** (*Microsoft.AzureActiveDirectory*): Generally available in all Azure regions where ADLS Gen1 is available.
- **Azure App Service**: Generally available in all Azure regions where App service is available.

## Public Preview

- **Azure Container Registry** (*Microsoft.ContainerRegistry*): Preview available in all Azure regions where Azure Container Registry is available.

For the most up-to-date notifications, check the [Azure Virtual Network updates](#) page.

## Key benefits

Service endpoints provide the following benefits:

- **Improved security for your Azure service resources:** VNet private address spaces can overlap. You can't use overlapping spaces to uniquely identify traffic that originates from your VNet. Service endpoints provide the ability to secure Azure service resources to your virtual network by extending VNet identity to the service. Once you enable service endpoints in your virtual network, you can add a virtual network rule to secure the Azure service resources to your virtual network. The rule addition provides improved security by fully removing public internet access to resources and allowing traffic only from your virtual network.
- **Optimal routing for Azure service traffic from your virtual network:** Today, any routes in your virtual network that force internet traffic to your on-premises and/or virtual appliances also force Azure

service traffic to take the same route as the internet traffic. Service endpoints provide optimal routing for Azure traffic.

Endpoints always take service traffic directly from your virtual network to the service on the Microsoft Azure backbone network. Keeping traffic on the Azure backbone network allows you to continue auditing and monitoring outbound Internet traffic from your virtual networks, through forced-tunneling, without impacting service traffic. For more information about user-defined routes and forced-tunneling, see [Azure virtual network traffic routing](#).

- **Simple to set up with less management overhead:** You no longer need reserved, public IP addresses in your virtual networks to secure Azure resources through IP firewall. There are no Network Address Translation (NAT) or gateway devices required to set up the service endpoints. You can configure service endpoints through a simple click on a subnet. There's no additional overhead to maintaining the endpoints.

## Limitations

- The feature is available only to virtual networks deployed through the Azure Resource Manager deployment model.
- Endpoints are enabled on subnets configured in Azure virtual networks. Endpoints can't be used for traffic from your premises to Azure services. For more information, see [Secure Azure service access from on-premises](#)
- For Azure SQL, a service endpoint applies only to Azure service traffic within a virtual network's region. For Azure Storage, endpoints also extend to include paired regions where you deploy the virtual network to support Read-Access Geo-Redundant Storage (RA-GRS) and Geo-Redundant Storage (GRS) traffic. For more information, see [Azure paired regions](#).
- For Azure Data Lake Storage (ADLS) Gen 1, the VNet Integration capability is only available for virtual networks within the same region. Also note that virtual network integration for ADLS Gen1 uses the virtual network service endpoint security between your virtual network and Azure Active Directory (Azure AD) to generate additional security claims in the access token. These claims are then used to authenticate your virtual network to your Data Lake Storage Gen1 account and allow access. The *Microsoft.AzureActiveDirectory* tag listed under services supporting service endpoints is used only for supporting service endpoints to ADLS Gen 1. Azure AD doesn't support service endpoints natively. For more information about Azure Data Lake Store Gen 1 VNet integration, see [Network security in Azure Data Lake Storage Gen1](#).

## Secure Azure services to virtual networks

- A virtual network service endpoint provides the identity of your virtual network to the Azure service. Once you enable service endpoints in your virtual network, you can add a virtual network rule to secure the Azure service resources to your virtual network.
- Today, Azure service traffic from a virtual network uses public IP addresses as source IP addresses. With service endpoints, service traffic switches to use virtual network private addresses as the source IP addresses when accessing the Azure service from a virtual network. This switch allows you to access the services without the need for reserved, public IP addresses used in IP firewalls.

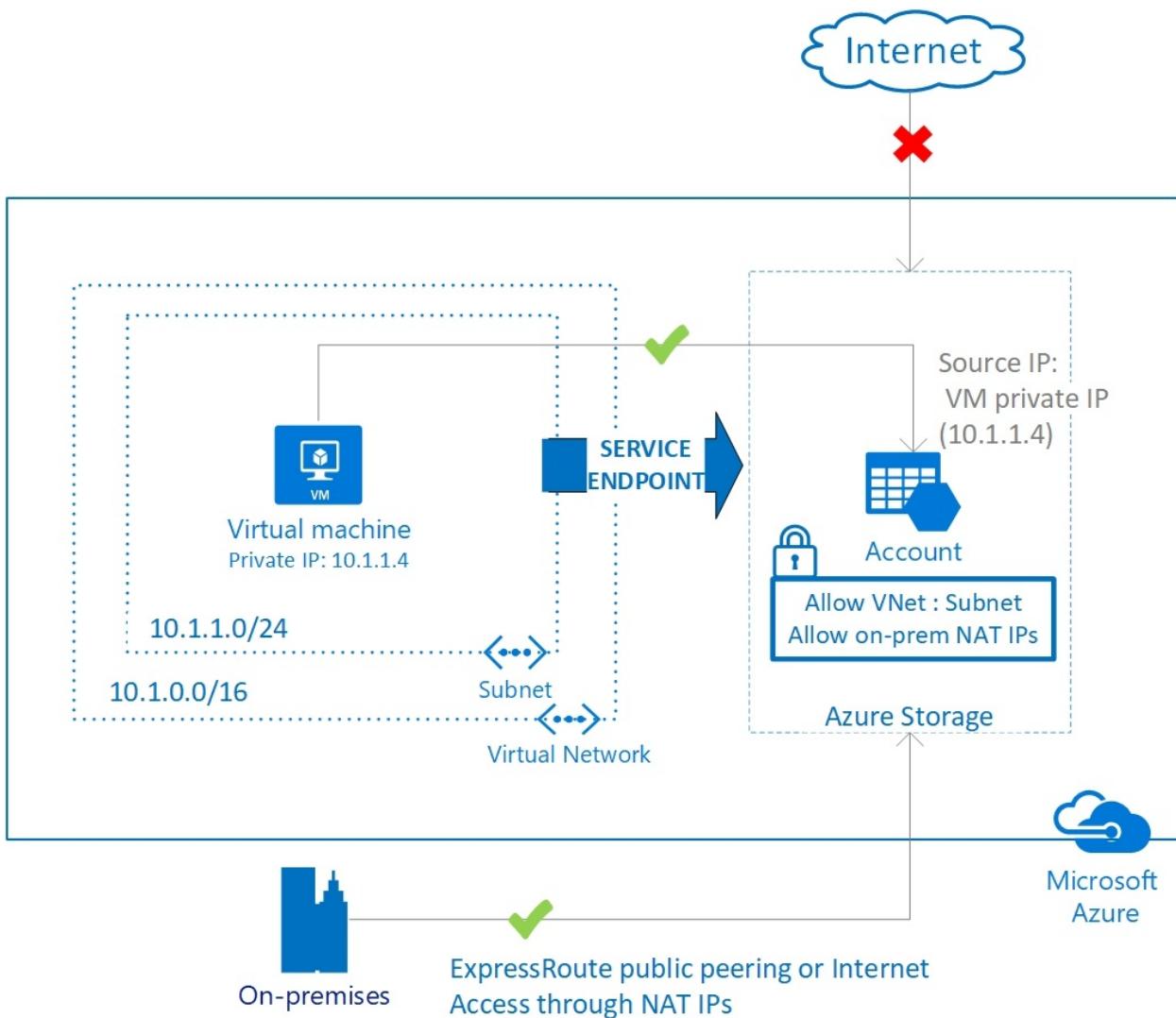
#### NOTE

With service endpoints, the source IP addresses of the virtual machines in the subnet for service traffic switches from using public IPv4 addresses to using private IPv4 addresses. Existing Azure service firewall rules using Azure public IP addresses will stop working with this switch. Please ensure Azure service firewall rules allow for this switch before setting up service endpoints. You may also experience temporary interruption to service traffic from this subnet while configuring service endpoints.

## Secure Azure service access from on-premises

By default, Azure service resources secured to virtual networks aren't reachable from on-premises networks. If you want to allow traffic from on-premises, you must also allow public (typically, NAT) IP addresses from your on-premises or ExpressRoute. You can add these IP addresses through the IP firewall configuration for Azure service resources.

ExpressRoute: If you're using [ExpressRoute](#) for public peering or Microsoft peering from your premises, you'll need to identify the NAT IP addresses that you're using. For public peering, each ExpressRoute circuit uses two NAT IP addresses, by default, applied to Azure service traffic when the traffic enters the Microsoft Azure network backbone. For Microsoft peering, the NAT IP addresses are either customer provided or provided by the service provider. To allow access to your service resources, you must allow these public IP addresses in the resource IP firewall setting. To find your public peering ExpressRoute circuit IP addresses, [open a support ticket with ExpressRoute](#) via the Azure portal. For more information about NAT for ExpressRoute public and Microsoft peering, see [ExpressRoute NAT requirements](#).



## Configuration

- Configure service endpoints on a subnet in a virtual network. Endpoints work with any type of compute instances running within that subnet.
- You can configure multiple service endpoints for all supported Azure services (Azure Storage or Azure SQL Database, for example) on a subnet.
- For Azure SQL Database, virtual networks must be in the same region as the Azure service resource. If using GRS and RA-GRS Azure Storage accounts, the primary account must be in the same region as the virtual network. For all other services, you can secure Azure service resources to virtual networks in any region.
- The virtual network where the endpoint is configured can be in the same or different subscription than the Azure service resource. For more information on permissions required for setting up endpoints and securing Azure services, see [Provisioning](#).
- For supported services, you can secure new or existing resources to virtual networks using service endpoints.

## Considerations

- After enabling a service endpoint, the source IP addresses of virtual machines in the subnet switch. The source IP addresses switch from using public IPv4 addresses to using their private IPv4 address when communicating with the service from that subnet. Any existing open TCP connections to the service are closed during this switch. Ensure that no critical tasks are running when enabling or disabling a service endpoint to a service for a subnet. Also, ensure that your applications can automatically connect to Azure services after the IP address switch.

The IP address switch only impacts service traffic from your virtual network. There's no impact to any other traffic addressed to or from the public IPv4 addresses assigned to your virtual machines. For Azure services, if you have existing firewall rules using Azure public IP addresses, these rules stop working with the switch to virtual network private addresses.

- With service endpoints, DNS entries for Azure services remain as-is today and continue to resolve to public IP addresses assigned to the Azure service.
- Network security groups (NSGs) with service endpoints:
  - By default, NSGs allow outbound internet traffic and also allow traffic from your VNet to Azure services. This traffic continues to work with service endpoints as is.
  - If you want to deny all outbound internet traffic and allow only traffic to specific Azure services, you can do so using [service tags](#) in your NSGs. You can specify supported Azure services as destination in your NSG rules and Azure also provides the maintenance of IP addresses underlying each tag. For more information, see [Azure Service tags for NSGs](#).

## Scenarios

- Peered, connected, or multiple virtual networks:** To secure Azure services to multiple subnets within a virtual network or across multiple virtual networks, you can enable service endpoints on each of the subnets independently, and secure Azure service resources to all of the subnets.
- Filtering outbound traffic from a virtual network to Azure services:** If you want to inspect or filter the traffic sent to an Azure service from a virtual network, you can deploy a network virtual appliance within the virtual network. You can then apply service endpoints to the subnet where the network virtual appliance is deployed, and secure Azure service resources only to this subnet. This scenario might be helpful if you want use network virtual appliance filtering to restrict Azure service access from your virtual network only to specific Azure resources. For more information, see [egress with network virtual appliances](#).
- Securing Azure resources to services deployed directly into virtual networks:** You can directly deploy various Azure services into specific subnets in a virtual network. You can secure Azure service resources to [managed service](#) subnets by setting up a service endpoint on the managed service subnet.
- Disk traffic from an Azure virtual machine:** Virtual Machine Disk traffic for managed and unmanaged disks isn't affected by service endpoints routing changes for Azure Storage. This traffic includes diskIO as well

as mount and unmount. You can limit REST access to page blobs to select networks through service endpoints and [Azure Storage network rules](#).

## Logging and troubleshooting

Once you configure service endpoints to a specific service, validate that the service endpoint route is in effect by:

- Validating the source IP address of any service request in the service diagnostics. All new requests with service endpoints show the source IP address for the request as the virtual network private IP address, assigned to the client making the request from your virtual network. Without the endpoint, the address is an Azure public IP address.
- Viewing the effective routes on any network interface in a subnet. The route to the service:
  - Shows a more specific default route to address prefix ranges of each service
  - Has a nextHopType of *VirtualNetworkServiceEndpoint*
  - Indicates that a more direct connection to the service is in effect compared to any forced-tunneling routes

### NOTE

Service endpoint routes override any BGP or UDR routes for the address prefix match of an Azure service. For more information, see [troubleshooting with effective routes](#).

## Provisioning

Service endpoints can be configured on virtual networks independently by a user with write access to a virtual network. To secure Azure service resources to a VNet, the user must have permission to *Microsoft.Network/virtualNetworks/subnets/joinViaServiceEndpoint/action* for the added subnets. The built-in service administrator roles include this permission by default. You can modify the permission by creating custom roles.

For more information about built-in roles, see [Built-in roles for Azure resources](#). For more information about assigning specific permissions to custom roles, see [Custom roles for Azure resources](#).

Virtual networks and Azure service resources can be in the same or different subscriptions. If the virtual network and Azure service resources are in different subscriptions, the resources must be under the same Active Directory (AD) tenant.

## Pricing and limits

There's no additional charge for using service endpoints. The current pricing model for Azure services (Azure Storage, Azure SQL Database, etc.) applies as-is today.

There's no limit on the total number of service endpoints in a virtual network.

Certain Azure services, such as Azure Storage Accounts, may enforce limits on the number of subnets used for securing the resource. Refer to the documentation for various services in the [Next steps](#) section for details.

## VNet service endpoint policies

VNet service endpoint policies allow you to filter virtual network traffic to Azure services. This filter allows only specific Azure service resources over service endpoints. Service endpoint policies provide granular access control for virtual network traffic to Azure services. For more information, see [Virtual Network Service Endpoint Policies](#).

## FAQs

For FAQs, see [Virtual Network Service Endpoint FAQs](#).

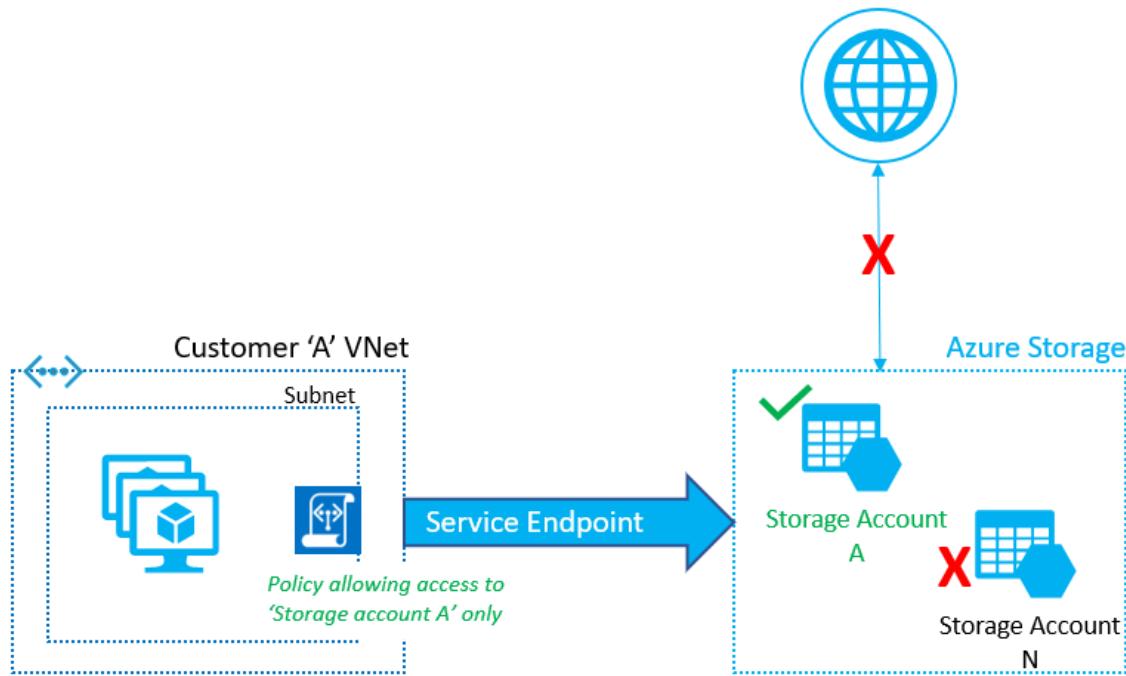
## Next steps

- [Configure virtual network service endpoints](#)
- [Secure an Azure Storage account to a virtual network](#)
- [Secure an Azure SQL Database to a virtual network](#)
- [Secure an Azure SQL Data Warehouse to a virtual network](#)
- [Azure service integration in virtual networks](#)
- [Virtual Network Service Endpoint Policies](#)
- [Azure Resource Manager template](#)

# Virtual network service endpoint policies for Azure Storage

2/26/2020 • 6 minutes to read • [Edit Online](#)

Virtual Network (VNet) service endpoint policies allow you to filter egress virtual network traffic to Azure Storage accounts over service endpoint, and allow data exfiltration to only specific Azure Storage accounts. Endpoint policies provide granular access control for virtual network traffic to Azure Storage when connecting over service endpoint.



This feature is generally available for **Azure Storage** in **all global Azure regions**.

## Key benefits

Virtual network service endpoint policies provide following benefits:

- **Improved security for your Virtual Network traffic to Azure Storage**

Azure service tags for network security groups allow you to restrict virtual network outbound traffic to specific Azure Storage regions. However, this allows traffic to any account within selected Azure Storage region.

Endpoint policies allow you to specify the Azure Storage accounts that are allowed virtual network outbound access and restricts access to all the other storage accounts. This gives much more granular security control for protecting data exfiltration from your virtual network.

- **Scalable, highly available policies to filter Azure service traffic**

Endpoint policies provide horizontally scalable, highly available solution to filter Azure service traffic from virtual networks, over service endpoints. No additional overhead is required to maintain central network appliances for this traffic in your virtual networks.

## JSON Object for Service Endpoint policies

Let's take a quick look at the Service Endpoint Policy object.

```

"serviceEndpointPolicyDefinitions": [
    {
        "description": null,
        "name": "MySEP-Definition",
        "resourceGroup": "MySEDeployment",
        "service": "Microsoft.Storage",
        "serviceResources": [
            "/subscriptions/subscriptionID/resourceGroups/MySEDeployment/providers/Microsoft.Storage/storageAccounts/mystgacc"
        ],
        "type": "Microsoft.Network/serviceEndpointPolicies/serviceEndpointPolicyDefinitions"
    }
]

```

## Configuration

- You can configure the endpoint policies to restrict virtual network traffic to specific Azure Storage accounts.
- Endpoint policy is configured on a subnet in a virtual network. Service endpoints for Azure Storage should be enabled on the subnet to apply the policy.
- Endpoint policy allows you to add specific Azure Storage accounts to allow list, using the resourceId format. You can restrict access to
  - all storage accounts in a subscription  
E.g. /subscriptions/subscriptionId
  - all storage accounts in a resource group  
E.g. subscriptions/subscriptionId/resourceGroups/resourceGroupName
  - an individual storage account by listing the corresponding Azure Resource Manager resourceId. This covers traffic to blobs, tables, queues, files and Azure Data Lake Storage Gen2.  
E.g. /subscriptions/subscriptionId/resourceGroups/resourceGroupName/providers/Microsoft.Storage/storageAccounts/storageAccountName
- By default, if no policies are attached to a subnet with endpoints, you can access all storage accounts in the service. Once a policy is configured on that subnet, only the resources specified in the policy can be accessed from compute instances in that subnet. Access to all other storage accounts will be denied.
- When applying Service Endpoint policies on a subnet, the Azure Storage *Service Endpoint* scope gets upgraded from regional to **global**. This means that all the traffic to Azure Storage is secured over service endpoint thereafter. The Service endpoint policies are also applicable globally, so any storage accounts, that are not explicitly allowed, will be denied access.
- You can apply multiple policies to a subnet. When multiple policies are associated to the subnet, virtual network traffic to resources specified across any of these policies will be allowed. Access to all other service resources, not specified in any of the policies, will be denied.

### NOTE

Service endpoint policies are **allow policies**, so apart from the specified resources, all other resources are restricted. Please ensure that all service resource dependencies for your applications are identified and listed in the policy.

- Only storage accounts using the Azure Resource Model can be specified in the endpoint policy. Your classic Azure Storage accounts will not support Azure Service Endpoint Policies.
- RA-GRS secondary access will be automatically allowed if the primary account is listed.
- Storage accounts can be in the same or a different subscription or Azure Active Directory tenant as the virtual network.

## Scenarios

- **Peered, connected or multiple virtual networks:** To filter traffic in peered virtual networks, endpoint policies should be applied individually to these virtual networks.
- **Filtering Internet traffic with Network Appliances or Azure Firewall:** Filter Azure service traffic with policies, over service endpoints, and filter rest of the Internet or Azure traffic via appliances or Azure Firewall.
- **Filtering traffic on Azure services deployed into Virtual Networks:** At this time, Azure Service Endpoint Policies are not supported for any managed Azure services that are deployed into your virtual network.
- **Filtering traffic to Azure services from on-premises:** Service endpoint policies only apply to the traffic from subnets associated to the policies. To allow access to specific Azure service resources from on-premises, traffic should be filtered using network virtual appliances or firewalls.

## Logging and troubleshooting

No centralized logging is available for service endpoint policies. For service diagnostic logs, see [Service endpoints logging](#).

### Troubleshooting scenarios

- Access denied to storage accounts that were working in preview (not in geo-paired region)
  - With Azure Storage upgrading to use Global Service Tags, the scope of Service Endpoint and thus Service Endpoint policies is now Global. So any traffic to Azure Storage is encrypted over Service Endpoints and only Storage accounts that are explicitly listed in policy are allowed access.
  - Explicitly allow list all the required Storage accounts to restore access.
  - Contact Azure support.
- Access is denied for accounts listed in the endpoint policies
  - Network security groups or firewall filtering could be blocking access
  - If removing/re-applying the policy results in connectivity loss:
    - Validate whether the Azure service is configured to allow access from the virtual network over endpoints, or that the default policy for the resource is set to *Allow All*.
    - Validate that the service diagnostics show the traffic over endpoints.
    - Check whether network security group flow logs show the access and that storage logs show the access, as expected, over service endpoints.
    - Contact Azure support.
- Access is denied for accounts not listed in the service endpoint policies
  - Validate whether Azure Storage is configured to allow access from the virtual network over endpoints, or whether the default policy for the resource is set to *Allow All*.
  - Ensure the accounts are not **classic storage accounts** with service endpoint policies on the subnet.
- A managed Azure Service stopped working after applying a Service Endpoint Policy over the subnet
  - Managed services are not supported with service endpoint policies at this time. *Watch this space for updates*.

## Provisioning

Service endpoint policies can be configured on subnets by a user with write access to a virtual network. Learn more about Azure [built-in roles](#) and assigning specific permissions to [custom roles](#).

Virtual networks and Azure Storage accounts can be in the same or different subscriptions, or Azure Active Directory tenants.

## Limitations

- You can only deploy service endpoint policies on virtual networks deployed through the Azure Resource Manager deployment model.
- Virtual networks must be in the same region as the service endpoint policy.
- You can only apply service endpoint policy on a subnet if service endpoints are configured for the Azure services listed in the policy.
- You can't use service endpoint policies for traffic from your on-premises network to Azure services.
- Azure managed services do not currently support Endpoint policies. This includes managed services deployed into the shared subnets (e.g. *Azure HDInsight*, *Azure Batch*, *Azure ADDS*, *Azure Application Gateway*, *Azure VPN gateway*, *Azure*

*Firewall) or into the dedicated subnets (e.g. Azure App Service Environment, Azure Redis Cache, Azure API Management, Azure SQL MI, classic managed services).*

#### **WARNING**

Azure services deployed into your virtual network, such as Azure HDInsight, access other Azure services, such as Azure Storage, for infrastructure requirements. Restricting endpoint policy to specific resources could break access to these infrastructure resources for the Azure services deployed in your virtual network.

- Classic storage accounts are not supported in endpoint policies. Policies will deny access to all classic storage accounts, by default. If your application needs access to Azure Resource Manager and classic storage accounts, endpoint policies should not be used for this traffic.

## Pricing and limits

There is no additional charge for using service endpoint policies. The current pricing model for Azure services (such as, Azure Storage) applies as is today, over service endpoints.

Following limits are enforced on service endpoint policies:

RESOURCE	DEFAULT LIMIT
ServiceEndpointPoliciesPerSubscription	500
ServiceEndpointPoliciesPerSubnet	100
ServiceResourcesPerServiceEndpointPolicyDefinition	200

## Next Steps

- Learn [how to configure virtual network service endpoint policies](#)
- Learn more about [Virtual network Service Endpoints](#)

# Virtual network TAP

12/23/2019 • 2 minutes to read • [Edit Online](#)

Azure virtual network TAP (Terminal Access Point) allows you to continuously stream your virtual machine network traffic to a network packet collector or analytics tool. The collector or analytics tool is provided by a [network virtual appliance](#) partner. For a list of partner solutions that are validated to work with virtual network TAP, see [partner solutions](#).

## IMPORTANT

Virtual network TAP is currently in preview in all the Azure regions. To use virtual network TAP, you must enroll in the preview by sending an email to [azurevnett@microsoft.com](mailto:azurevnett@microsoft.com) with your subscription ID. You will receive an email back once your subscription has been enrolled. You aren't able to use the capability until you receive a confirmation email. This preview is provided without a service level agreement and should not be used for production workloads. Certain features may not be supported, may have constrained capabilities, or may not be available in all Azure locations. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Virtual network TAP partner solutions

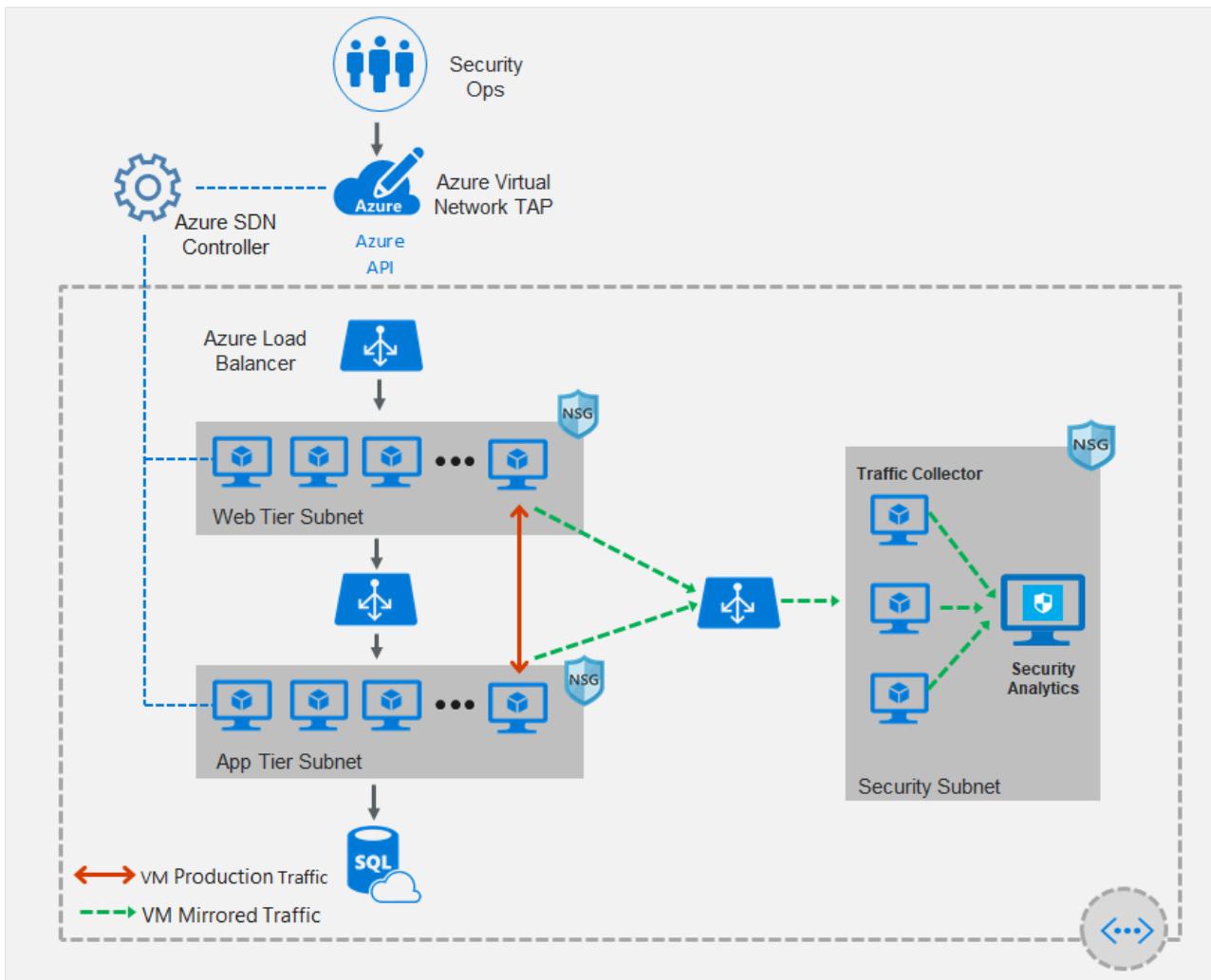
### Network packet brokers

- [Big Switch Big Monitoring Fabric](#)
- [Gigamon GigaSECURE](#)
- [Ixia CloudLens](#)
- [Nubeva Prisms](#)

### Security analytics, network/application performance management

- [Awake Security](#)
- [Cisco Stealthwatch Cloud](#)
- [Darktrace](#)
- [ExtraHop Reveal\(x\)](#)
- [Fidelis Cybersecurity](#)
- [Flowmon](#)
- [NetFort LANGuardian](#)
- [Netscout vSTREAM](#)
- [Riverbed SteelCentral AppResponse](#)
- [RSA NetWitness® Platform](#)
- [Vectra Cognito](#)

The following picture shows how virtual network TAP works. You can add a TAP configuration on a [network interface](#) that is attached to a virtual machine deployed in your virtual network. The destination is a virtual network IP address in the same virtual network as the monitored network interface or a [peered virtual](#) network. The collector solution for virtual network TAP can be deployed behind an Azure Internal Load balancer for high availability. To evaluate deployment options for individual solution, see [partner solutions](#).



## Prerequisites

Before you create a virtual network TAP, you must have received a confirmation mail that you are enrolled in the preview, and have one or more virtual machines created using [Azure Resource Manager](#) deployment model and a partner solution for aggregating the TAP traffic in the same azure region. If you don't have a partner solution in your virtual network, see [partner solutions](#) to deploy one. You can use the same virtual network TAP resource to aggregate traffic from multiple network interfaces in the same or different subscriptions. If the monitored network interfaces are in different subscriptions, the subscriptions must be associated to the same Azure Active Directory tenant. Additionally, the monitored network interfaces and the destination endpoint for aggregating the TAP traffic can be in peered virtual networks in the same region. If you are using this deployment model ensure that the [virtual network peering](#) is enabled before you configure virtual network TAP.

## Permissions

The accounts you use to apply TAP configuration on network interfaces must be assigned to the [network contributor](#) role or a [custom role](#) that is assigned the necessary actions from the following table:

ACTION	NAME
Microsoft.Network/virtualNetworkTaps/*	Required to create, update, read and delete a virtual network TAP resource
Microsoft.Network/networkInterfaces/read	Required to read the network interface resource on which the TAP will be configured

ACTION	NAME
Microsoft.Network/tapConfigurations/*	Required to create, update, read and delete the TAP configuration on a network interface

## Next steps

- Learn how to [Create a virtual network TAP](#).

# Subnet extension

11/4/2019 • 2 minutes to read • [Edit Online](#)

Workload migration to the public cloud requires careful planning and coordination. One of the key considerations can be the ability to retain your IP addresses. Which can be important especially if your applications have IP address dependency or you have compliance requirements to use specific IP addresses. Azure Virtual Network solves this problem for you by allowing you to create VNet and Subnets using an IP address range of your choice.

Migrations can get a bit challenging when the above requirement is coupled with an additional requirement to keep some applications on-premises. In such a situation, you'll have to split the applications between Azure and on-premises, without renumbering the IP addresses on either side. Additionally, you'll have to allow the applications to communicate as if they are in the same network.

One solution to the above problem is subnet extension. Extending a network allows applications to talk over the same broadcast domain when they exist at different physical locations, removing the need to rearchitect your network topology.

While extending your network isn't a good practice in general, below use cases can make it necessary.

- **Phased Migration:** The most common scenario is that you want to phase your migration. You want to bring a few applications first and over time migrate rest of the applications to Azure.
- **Latency:** Low latency requirements can be another reason for you to keep some applications on-premises to ensure that they're as close as possible to your datacenter.
- **Compliance:** Another use case is that you might have compliance requirements to keep some of your applications on-premises.

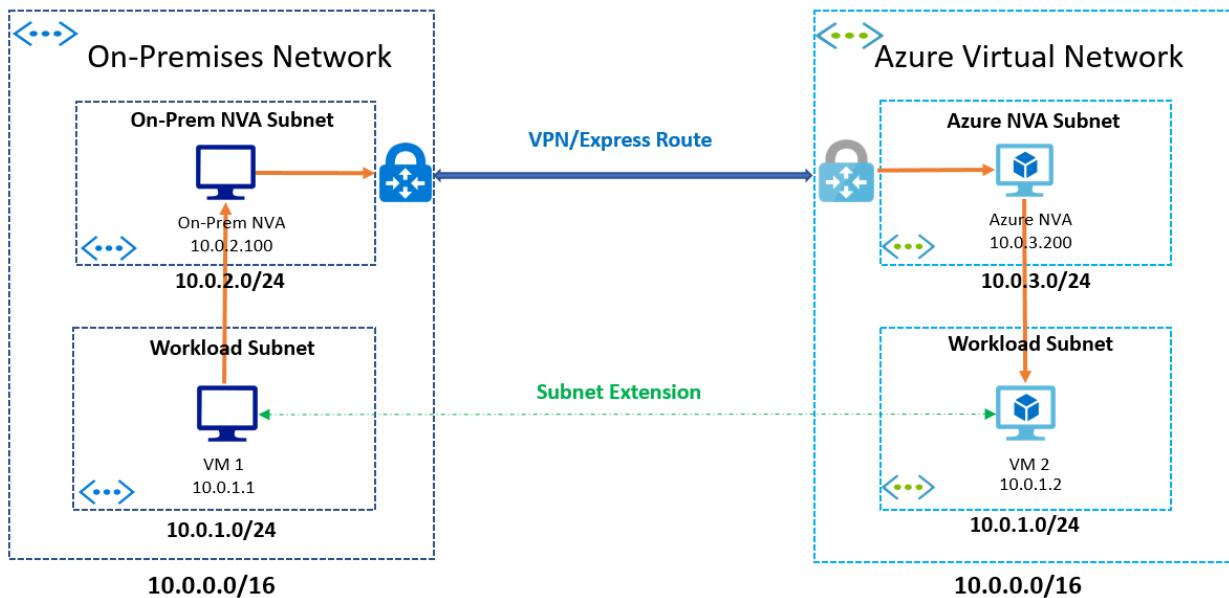
## NOTE

You should not extend your subnets unless it is necessary. In the cases where you do extend your subnets, you should try to make it an intermediate step. With time, you should try re-number applications in your on-premises network and migrate them to Azure.

In the next section, we'll discuss how you can extend your subnets into Azure.

## Extend your subnet to Azure

You can extend your on-premises subnets to Azure using a layer-3 overlay network based solution. Most solutions use an overlay technology such as VXLAN to extend the layer-2 network using an layer-3 overlay network. The diagram below shows a generalized solution. In this solution, the same subnet exists on both sides that is, Azure and on-premises.



The IP addresses from the subnet are assigned to VMs on Azure and on-premises. Both Azure and on-premises have an NVA inserted in their networks. When a VM in Azure tries to talk to a VM in on-premises network, the Azure NVA captures the packet, encapsulates it, and sends it over VPN/Express Route to the on-premises network. The on-premises NVA receives the packet, decapsulates it and forwards it to the intended recipient in its network. The return traffic uses a similar path and logic.

In the above example, the Azure NVA and the on-premises NVA communicate and learn about IP addresses behind each other. More complex networks can also have a mapping service, which maintains the mapping between the NVAs and the IP addresses behind them. When an NVA receives a packet, it queries the mapping service to find out the address of the NVA that has the destination IP address behind it.

In the next section, you'll find details on subnet extension solutions we've tested on Azure.

## Next steps

[Extend your subnet to Azure using vendor solutions.](#)

# What is subnet delegation?

11/20/2019 • 2 minutes to read • [Edit Online](#)

Subnet delegation enables you to designate a specific subnet for an Azure PaaS service of your choice that needs to be injected into your virtual network. Subnet delegation provides full control to the customer on managing the integration of Azure services into their virtual networks.

When you delegate a subnet to an Azure service, you allow that service to establish some basic network configuration rules for that subnet, which help the Azure service operate their instances in a stable manner. As a result, the Azure service may establish some pre or post deployment conditions, such as:

- deploy the service in a shared versus dedicated subnet.
- add to the service a set of Network Intent Policies post deployment that is required for the service to work properly.

## Advantages of subnet delegation

Delegating a subnet to specific services provides the following advantages:

- helps to designate a subnet for one or more Azure services and manage the instances in the subnet as per requirements. For example, the virtual network owner can define the following for a delegated subnet to better manage resources and access as follows:
  - network filtering traffic policies with network security groups.
  - routing policies with user-defined routes.
  - services integration with service endpoints configurations.
- helps injected services to better integrate with the virtual network by defining their pre-conditions of deployments in the form of Network Intent Policies. This ensures any actions that can affect functioning of the injected service can be blocked at PUT.

## Who can delegate?

Subnet delegation is an exercise that the virtual network owners need to perform to designate one of the subnets for a specific Azure Service. Azure Service in turn deploys the instances into this subnet for consumption by the customer workloads.

## Impact of subnet delegation on your subnet

Each Azure service defines their own deployment model, where they can define what properties they do or do not support in a delegated subnet for injection purposes, such as follows:

- shared subnet with other Azure Services or VM / virtual machine scale set in the same subnet, or it only supports a dedicated subnet with only instances of this service in it.
- supports NSG association with the delegated subnet.
- supports NSG associated with the delegated subnet can be also associated with any other subnet.
- allows route table association with the delegated subnet.
- allows the route table associated with the delegated subnet to be associated with any other subnet.
- dictates the minimum number of IP Addresses in the delegated subnet.
- dictates the IP Address space in the delegated subnet to be from Private IP Address space (10.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12).

- dictates that the custom DNS configuration has an Azure DNS entry.

Injected services can also add their own policies as follows:

- **Security policies:** Collection of security rules required for a given service to work.
- **Route policies:** Collection of routes required for a given service to work.

## What subnet delegation does not do

The Azure services being injected into a delegated subnet still have the basic set of properties that are available for non-delegated subnets, such as:

- Azure services can inject instances into customer subnets, but cannot impact the existing workloads.
- The policies or routes that these services apply are flexible and can be overridden by the customer.

## Next steps

- [Delegate a subnet](#)

# Plan virtual networks

2/12/2020 • 10 minutes to read • [Edit Online](#)

Creating a virtual network to experiment with is easy enough, but chances are, you will deploy multiple virtual networks over time to support the production needs of your organization. With some planning, you will be able to deploy virtual networks and connect the resources you need more effectively. The information in this article is most helpful if you're already familiar with virtual networks and have some experience working with them. If you are not familiar with virtual networks, it's recommended that you read [Virtual network overview](#).

## Naming

All Azure resources have a name. The name must be unique within a scope, that may vary for each resource type. For example, the name of a virtual network must be unique within a [resource group](#), but can be duplicated within a [subscription](#) or [Azure region](#). Defining a naming convention that you can use consistently when naming resources is helpful when managing several network resources over time. For suggestions, see [Naming conventions](#).

## Regions

All Azure resources are created in an Azure region and subscription. A resource can only be created in a virtual network that exists in the same region and subscription as the resource. You can however, connect virtual networks that exist in different subscriptions and regions. For more information, see [connectivity](#). When deciding which region(s) to deploy resources in, consider where consumers of the resources are physically located:

- Consumers of resources typically want the lowest network latency to their resources. To determine relative latencies between a specified location and Azure regions, see [View relative latencies](#).
- Do you have data residency, sovereignty, compliance, or resiliency requirements? If so, choosing the region that aligns to the requirements is critical. For more information, see [Azure geographies](#).
- Do you require resiliency across Azure Availability Zones within the same Azure region for the resources you deploy? You can deploy resources, such as virtual machines (VM) to different availability zones within the same virtual network. Not all Azure regions support availability zones however. To learn more about availability zones and the regions that support them, see [Availability zones](#).

## Subscriptions

You can deploy as many virtual networks as required within each subscription, up to the [limit](#). Some organizations have different subscriptions for different departments, for example. For more information and considerations around subscriptions, see [Subscription governance](#).

## Segmentation

You can create multiple virtual networks per subscription and per region. You can create multiple subnets within each virtual network. The considerations that follow help you determine how many virtual networks and subnets you require:

### Virtual networks

A virtual network is a virtual, isolated portion of the Azure public network. Each virtual network is dedicated to your subscription. Things to consider when deciding whether to create one virtual network, or multiple virtual networks in a subscription:

- Do any organizational security requirements exist for isolating traffic into separate virtual networks? You can

choose to connect virtual networks or not. If you connect virtual networks, you can implement a network virtual appliance, such as a firewall, to control the flow of traffic between the virtual networks. For more information, see [security](#) and [connectivity](#).

- Do any organizational requirements exist for isolating virtual networks into separate [subscriptions](#) or [regions](#)?
- A [network interface](#) enables a VM to communicate with other resources. Each network interface has one or more private IP addresses assigned to it. How many network interfaces and [private IP addresses](#) do you require in a virtual network? There are [limits](#) to the number of network interfaces and private IP addresses that you can have within a virtual network.
- Do you want to connect the virtual network to another virtual network or on-premises network? You may choose to connect some virtual networks to each other or on-premises networks, but not others. For more information, see [connectivity](#). Each virtual network that you connect to another virtual network, or on-premises network, must have a unique address space. Each virtual network has one or more public or private address ranges assigned to its address space. An address range is specified in classless internet domain routing (CIDR) format, such as 10.0.0.0/16. Learn more about [address ranges](#) for virtual networks.
- Do you have any organizational administration requirements for resources in different virtual networks? If so, you might separate resources into separate virtual network to simplify [permission assignment](#) to individuals in your organization or to assign different policies to different virtual networks.
- When you deploy some Azure service resources into a virtual network, they create their own virtual network. To determine whether an Azure service creates its own virtual network, see information for each [Azure service that can be deployed into a virtual network](#).

## Subnets

A virtual network can be segmented into one or more subnets up to the [limits](#). Things to consider when deciding whether to create one subnet, or multiple virtual networks in a subscription:

- Each subnet must have a unique address range, specified in CIDR format, within the address space of the virtual network. The address range cannot overlap with other subnets in the virtual network.
- If you plan to deploy some Azure service resources into a virtual network, they may require, or create, their own subnet, so there must be enough unallocated space for them to do so. To determine whether an Azure service creates its own subnet, see information for each [Azure service that can be deployed into a virtual network](#). For example, if you connect a virtual network to an on-premises network using an Azure VPN Gateway, the virtual network must have a dedicated subnet for the gateway. Learn more about [gateway subnets](#).
- Azure routes network traffic between all subnets in a virtual network, by default. You can override Azure's default routing to prevent Azure routing between subnets, or to route traffic between subnets through a network virtual appliance, for example. If you require that traffic between resources in the same virtual network flow through a network virtual appliance (NVA), deploy the resources to different subnets. Learn more in [security](#).
- You can limit access to Azure resources such as an Azure storage account or Azure SQL database, to specific subnets with a virtual network service endpoint. Further, you can deny access to the resources from the internet. You may create multiple subnets, and enable a service endpoint for some subnets, but not others. Learn more about [service endpoints](#), and the Azure resources you can enable them for.
- You can associate zero or one network security group to each subnet in a virtual network. You can associate the same, or a different, network security group to each subnet. Each network security group contains rules, which allow or deny traffic to and from sources and destinations. Learn more about [network security groups](#).

## Security

You can filter network traffic to and from resources in a virtual network using network security groups and network virtual appliances. You can control how Azure routes traffic from subnets. You can also limit who in your organization can work with resources in virtual networks.

### Traffic filtering

- You can filter network traffic between resources in a virtual network using a network security group, an NVA that filters network traffic, or both. To deploy an NVA, such as a firewall, to filter network traffic, see the [Azure Marketplace](#). When using an NVA, you also create custom routes to route traffic from subnets to the NVA. Learn more about [traffic routing](#).
- A network security group contains several default security rules that allow or deny traffic to or from resources. A network security group can be associated to a network interface, the subnet the network interface is in, or both. To simplify management of security rules, it's recommended that you associate a network security group to individual subnets, rather than individual network interfaces within the subnet, whenever possible.
- If different VMs within a subnet need different security rules applied to them, you can associate the network interface in the VM to one or more application security groups. A security rule can specify an application security group in its source, destination, or both. That rule then only applies to the network interfaces that are members of the application security group. Learn more about [network security groups](#) and [application security groups](#).
- Azure creates several default security rules within each network security group. One default rule allows all traffic to flow between all resources in a virtual network. To override this behavior, use network security groups, custom routing to route traffic to an NVA, or both. It's recommended that you familiarize yourself with all of Azure's [default security rules](#) and understand how network security group rules are applied to a resource.

You can view sample designs for implementing a perimeter network (also known as a DMZ) between Azure and the internet using an [NVA](#).

## Traffic routing

Azure creates several default routes for outbound traffic from a subnet. You can override Azure's default routing by creating a route table and associating it to a subnet. Common reasons for overriding Azure's default routing are:

- Because you want traffic between subnets to flow through an NVA. To learn more about how to [configure route tables to force traffic through an NVA](#).
- Because you want to force all internet-bound traffic through an NVA, or on-premises, through an Azure VPN gateway. Forcing internet traffic on-premises for inspection and logging is often referred to as forced tunneling. Learn more about how to configure [forced tunneling](#).

If you need to implement custom routing, it's recommended that you familiarize yourself with [routing in Azure](#).

## Connectivity

You can connect a virtual network to other virtual networks using virtual network peering, or to your on-premises network, using an Azure VPN gateway.

### Peering

When using [virtual network peering](#), the virtual networks can be in the same, or different, supported Azure regions. The virtual networks can be in the same or different Azure subscriptions (even subscriptions belonging to different Azure Active Directory tenants). Before creating a peering, it's recommended that you familiarize yourself with all of the peering [requirements and constraints](#). Bandwidth between resources in virtual networks peered in the same region is the same as if the resources were in the same virtual network.

### VPN gateway

You can use an Azure [VPN Gateway](#) to connect a virtual network to your on-premises network using a [site-to-site VPN](#), or using a dedicated connection with Azure [ExpressRoute](#).

You can combine peering and a VPN gateway to create [hub and spoke networks](#), where spoke virtual networks connect to a hub virtual network, and the hub connects to an on-premises network, for example.

### Name resolution

Resources in one virtual network cannot resolve the names of resources in a peered virtual network using Azure's

[built-in DNS](#). To resolve names in a peered virtual network, [deploy your own DNS server](#), or use Azure DNS [private domains](#). Resolving names between resources in a virtual network and on-premises networks also requires you to deploy your own DNS server.

## Permissions

Azure utilizes [role based access control](#) (RBAC) to resources. Permissions are assigned to a [scope](#) in the following hierarchy: management group, subscription, resource group, and individual resource. To learn more about the hierarchy, see [Organize your resources](#). To work with Azure virtual networks and all of their related capabilities such as peering, network security groups, service endpoints, and route tables, you can assign members of your organization to the built-in [Owner](#), [Contributor](#), or [Network contributor](#) roles, and then assign the role to the appropriate scope. If you want to assign specific permissions for a subset of virtual network capabilities, create a [custom role](#) and assign the specific permissions required for [virtual networks](#), [subnets and service endpoints](#), [network interfaces](#), [peering](#), [network and application security groups](#), or [route tables](#) to the role.

## Policy

Azure Policy enables you to create, assign, and manage policy definitions. Policy definitions enforce different rules over your resources, so the resources stay compliant with your organizational standards and service level agreements. Azure Policy runs an evaluation of your resources, scanning for resources that are not compliant with the policy definitions you have. For example, you can define and apply a policy that allows creation of virtual networks in only a specific resource group or region. Another policy can require that every subnet has a network security group associated to it. The policies are then evaluated when creating and updating resources.

Policies are applied to the following hierarchy: management group, subscription, and resource group. Learn more about [Azure policy](#) or deploy some virtual network [policy template](#) samples.

## Next steps

Learn about all tasks, settings, and options for a [virtual network](#), [subnet and service endpoint](#), [network interface](#), [peering](#), [network and application security group](#), or [route table](#).

# Name resolution for resources in Azure virtual networks

2/18/2020 • 13 minutes to read • [Edit Online](#)

Depending on how you use Azure to host IaaS, PaaS, and hybrid solutions, you might need to allow the virtual machines (VMs), and other resources deployed in a virtual network to communicate with each other. Although you can enable communication by using IP addresses, it is much simpler to use names that can be easily remembered, and do not change.

When resources deployed in virtual networks need to resolve domain names to internal IP addresses, they can use one of two methods:

- [Azure-provided name resolution](#)
- [Name resolution that uses your own DNS server](#) (which might forward queries to the Azure-provided DNS servers)

The type of name resolution you use depends on how your resources need to communicate with each other. The following table illustrates scenarios and corresponding name resolution solutions:

## NOTE

Depending on your scenario, you might want to use Azure DNS private zones. For more information, see [Using Azure DNS for private domains](#).

SCENARIO	SOLUTION	SUFFIX
Name resolution between VMs located in the same virtual network, or Azure Cloud Services role instances in the same cloud service.	<a href="#">Azure DNS private zones</a> or <a href="#">Azure-provided name resolution</a>	Hostname or FQDN
Name resolution between VMs in different virtual networks or role instances in different cloud services.	<a href="#">Azure DNS private zones</a> or, Customer-managed DNS servers forwarding queries between virtual networks for resolution by Azure (DNS proxy). See <a href="#">Name resolution using your own DNS server</a> .	FQDN only
Name resolution from an Azure App Service (Web App, Function, or Bot) using virtual network integration to role instances or VMs in the same virtual network.	Customer-managed DNS servers forwarding queries between virtual networks for resolution by Azure (DNS proxy). See <a href="#">Name resolution using your own DNS server</a> .	FQDN only
Name resolution from App Service Web Apps to VMs in the same virtual network.	Customer-managed DNS servers forwarding queries between virtual networks for resolution by Azure (DNS proxy). See <a href="#">Name resolution using your own DNS server</a> .	FQDN only

SCENARIO	SOLUTION	SUFFIX
Name resolution from App Service Web Apps in one virtual network to VMs in a different virtual network.	Customer-managed DNS servers forwarding queries between virtual networks for resolution by Azure (DNS proxy). See <a href="#">Name resolution using your own DNS server</a> .	FQDN only
Resolution of on-premises computer and service names from VMs or role instances in Azure.	Customer-managed DNS servers (on-premises domain controller, local read-only domain controller, or a DNS secondary synced using zone transfers, for example). See <a href="#">Name resolution using your own DNS server</a> .	FQDN only
Resolution of Azure hostnames from on-premises computers.	Forward queries to a customer-managed DNS proxy server in the corresponding virtual network, the proxy server forwards queries to Azure for resolution. See <a href="#">Name resolution using your own DNS server</a> .	FQDN only
Reverse DNS for internal IPs.	<a href="#">Name resolution using your own DNS server</a> .	Not applicable
Name resolution between VMs or role instances located in different cloud services, not in a virtual network.	Not applicable. Connectivity between VMs and role instances in different cloud services is not supported outside a virtual network.	Not applicable

## Azure-provided name resolution

Along with resolution of public DNS names, Azure provides internal name resolution for VMs and role instances that reside within the same virtual network or cloud service. VMs and instances in a cloud service share the same DNS suffix, so the host name alone is sufficient. But in virtual networks deployed using the classic deployment model, different cloud services have different DNS suffixes. In this situation, you need the FQDN to resolve names between different cloud services. In virtual networks deployed using the Azure Resource Manager deployment model, the DNS suffix is consistent across the virtual network, so the FQDN is not needed. DNS names can be assigned to both VMs and network interfaces. Although Azure-provided name resolution does not require any configuration, it is not the appropriate choice for all deployment scenarios, as detailed in the previous table.

### NOTE

When using cloud services web and worker roles, you can also access the internal IP addresses of role instances using the Azure Service Management REST API. For more information, see the [Service Management REST API Reference](#). The address is based on the role name and instance number.

## Features

Azure-provided name resolution includes the following features:

- Ease of use. No configuration is required.
- High availability. You don't need to create and manage clusters of your own DNS servers.
- You can use the service in conjunction with your own DNS servers, to resolve both on-premises and Azure host names.
- You can use name resolution between VMs and role instances within the same cloud service, without the need

for an FQDN.

- You can use name resolution between VMs in virtual networks that use the Azure Resource Manager deployment model, without need for an FQDN. Virtual networks in the classic deployment model require an FQDN when you are resolving names in different cloud services.
- You can use host names that best describe your deployments, rather than working with auto-generated names.

## Considerations

Points to consider when you are using Azure-provided name resolution:

- The Azure-created DNS suffix cannot be modified.
- You cannot manually register your own records.
- WINS and NetBIOS are not supported. You cannot see your VMs in Windows Explorer.
- Host names must be DNS-compatible. Names must use only 0-9, a-z, and '-', and cannot start or end with a '-'.
- DNS query traffic is throttled for each VM. Throttling shouldn't impact most applications. If request throttling is observed, ensure that client-side caching is enabled. For more information, see [DNS client configuration](#).
- Only VMs in the first 180 cloud services are registered for each virtual network in a classic deployment model. This limit does not apply to virtual networks in Azure Resource Manager.
- The Azure DNS IP address is 168.63.129.16. This is a static IP address and will not change.

## DNS client configuration

This section covers client-side caching and client-side retries.

### Client-side caching

Not every DNS query needs to be sent across the network. Client-side caching helps reduce latency and improve resilience to network blips, by resolving recurring DNS queries from a local cache. DNS records contain a time-to-live (TTL) mechanism, which allows the cache to store the record for as long as possible without impacting record freshness. Thus, client-side caching is suitable for most situations.

The default Windows DNS client has a DNS cache built-in. Some Linux distributions do not include caching by default. If you find that there isn't a local cache already, add a DNS cache to each Linux VM.

There are a number of different DNS caching packages available (such as dnsmasq). Here's how to install dnsmasq on the most common distributions:

- **Ubuntu (uses resolvconf):**
  - Install the dnsmasq package with `sudo apt-get install dnsmasq`.
- **SUSE (uses netconf):**
  - Install the dnsmasq package with `sudo zypper install dnsmasq`.
  - Enable the dnsmasq service with `systemctl enable dnsmasq.service`.
  - Start the dnsmasq service with `systemctl start dnsmasq.service`.
  - Edit **/etc/sysconfig/network/config**, and change **NETCONFIG\_DNS\_FORWARDER=""** to **dnsmasq**.
  - Update resolv.conf with `netconfig update`, to set the cache as the local DNS resolver.
- **CentOS (uses NetworkManager):**
  - Install the dnsmasq package with `sudo yum install dnsmasq`.
  - Enable the dnsmasq service with `systemctl enable dnsmasq.service`.
  - Start the dnsmasq service with `systemctl start dnsmasq.service`.
  - Add **prepend domain-name-servers 127.0.0.1;** to **/etc/dhclient-eth0.conf**.
  - Restart the network service with `service network restart`, to set the cache as the local DNS resolver.

## NOTE

The dnsmasq package is only one of many DNS caches available for Linux. Before using it, check its suitability for your particular needs, and check that no other cache is installed.

## Client-side retries

DNS is primarily a UDP protocol. Because the UDP protocol doesn't guarantee message delivery, retry logic is handled in the DNS protocol itself. Each DNS client (operating system) can exhibit different retry logic, depending on the creator's preference:

- Windows operating systems retry after one second, and then again after another two seconds, four seconds, and another four seconds.
- The default Linux setup retries after five seconds. We recommend changing the retry specifications to five times, at one-second intervals.

Check the current settings on a Linux VM with `cat /etc/resolv.conf`. Look at the *options* line, for example:

```
options timeout:1 attempts:5
```

The resolv.conf file is usually auto-generated, and should not be edited. The specific steps for adding the *options* line vary by distribution:

- **Ubuntu** (uses resolvconf):
  1. Add the *options* line to **/etc/resolvconf/resolv.conf.d/tail**.
  2. Run `resolvconf -u` to update.
- **SUSE** (uses netconfig):
  1. Add *timeout:1 attempts:5* to the **NETCONFIG\_DNS\_RESOLVER\_OPTIONS=""** parameter in **/etc/sysconfig/network/config**.
  2. Run `netconfig update` to update.
- **CentOS** (uses NetworkManager):
  1. Add `echo "options timeout:1 attempts:5"` to **/etc/NetworkManager/dispatcher.d/11-dhclient**.
  2. Update with `service network restart`.

## Name resolution that uses your own DNS server

This section covers VMs, role instances, and web apps.

### VMs and role instances

Your name resolution needs might go beyond the features provided by Azure. For example, you might need to use Microsoft Windows Server Active Directory domains, resolve DNS names between virtual networks. To cover these scenarios, Azure provides the ability for you to use your own DNS servers.

DNS servers within a virtual network can forward DNS queries to the recursive resolvers in Azure. This enables you to resolve host names within that virtual network. For example, a domain controller (DC) running in Azure can respond to DNS queries for its domains, and forward all other queries to Azure. Forwarding queries allows VMs to see both your on-premises resources (via the DC) and Azure-provided host names (via the forwarder). Access to the recursive resolvers in Azure is provided via the virtual IP 168.63.129.16.

DNS forwarding also enables DNS resolution between virtual networks, and allows your on-premises machines to resolve Azure-provided host names. In order to resolve a VM's host name, the DNS server VM must reside in the same virtual network, and be configured to forward host name queries to Azure. Because the DNS suffix is different in each virtual network, you can use conditional forwarding rules to send DNS queries to the correct

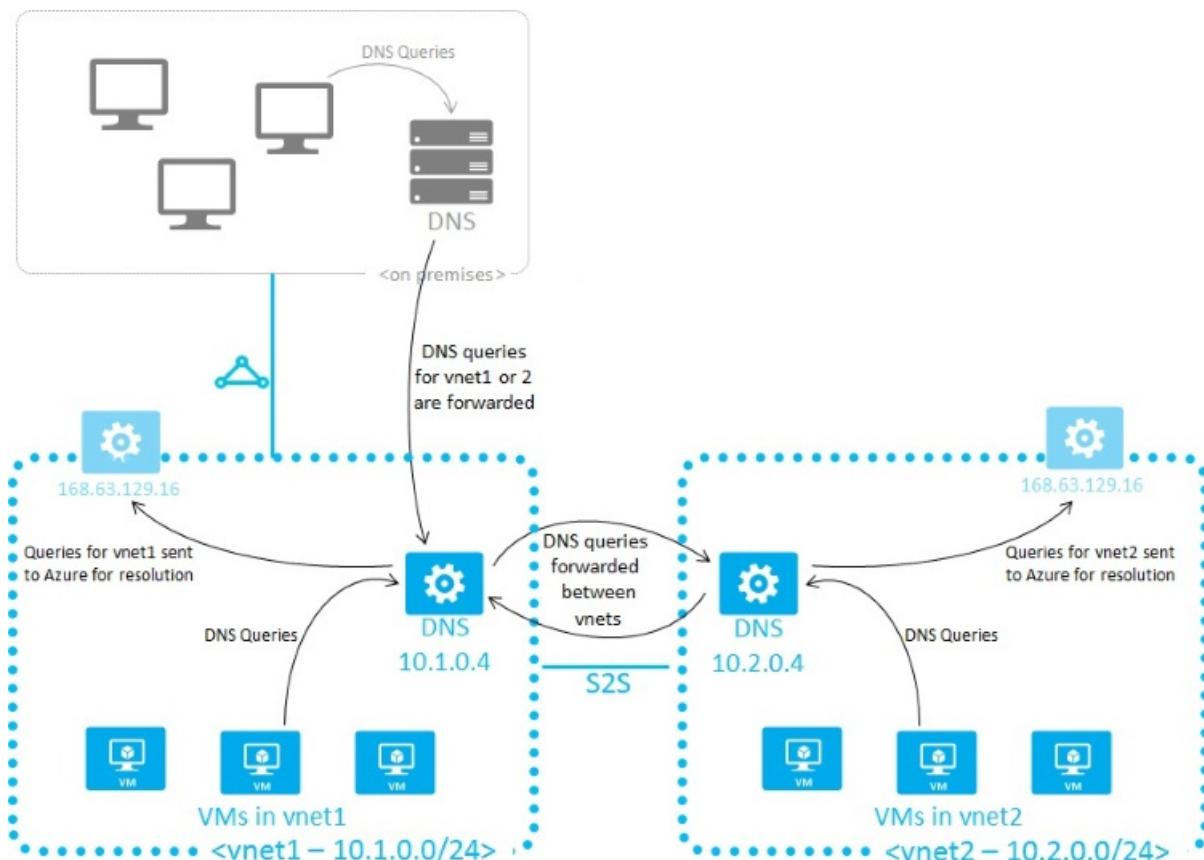
virtual network for resolution. The following image shows two virtual networks and an on-premises network doing DNS resolution between virtual networks, by using this method. An example DNS forwarder is available in the [Azure Quickstart Templates gallery](#) and [GitHub](#).

#### NOTE

A role instance can perform name resolution of VMs within the same virtual network. It does so by using the FQDN, which consists of the VM's host name and **internal.cloudapp.net** DNS suffix. However, in this case, name resolution is only successful if the role instance has the VM name defined in the [Role Schema \(.cscfg file\)](#).

```
<Role name=<role-name> vmName=<vm-name>>
```

Role instances that need to perform name resolution of VMs in another virtual network (FQDN by using the **internal.cloudapp.net** suffix) have to do so by using the method described in this section (custom DNS servers forwarding between the two virtual networks).



When you are using Azure-provided name resolution, Azure Dynamic Host Configuration Protocol (DHCP) provides an internal DNS suffix (**.internal.cloudapp.net**) to each VM. This suffix enables host name resolution because the host name records are in the **internal.cloudapp.net** zone. When you are using your own name resolution solution, this suffix is not supplied to VMs because it interferes with other DNS architectures (like domain-joined scenarios). Instead, Azure provides a non-functioning placeholder (**reddog.microsoft.com**).

If necessary, you can determine the internal DNS suffix by using PowerShell or the API:

- For virtual networks in Azure Resource Manager deployment models, the suffix is available via the [network interface REST API](#), the [Get-AzNetworkInterface](#) PowerShell cmdlet, and the [az network nic show](#) Azure CLI command.
- In classic deployment models, the suffix is available via the [Get Deployment API](#) call or the [Get-AzureVM - Debug](#) cmdlet.

If forwarding queries to Azure doesn't suit your needs, you should provide your own DNS solution. Your DNS solution needs to:

- Provide appropriate host name resolution, via [DDNS](#), for example. If you are using DDNS, you might need to disable DNS record scavenging. Azure DHCP leases are long, and scavenging might remove DNS records prematurely.
- Provide appropriate recursive resolution to allow resolution of external domain names.
- Be accessible (TCP and UDP on port 53) from the clients it serves, and be able to access the internet.
- Be secured against access from the internet, to mitigate threats posed by external agents.

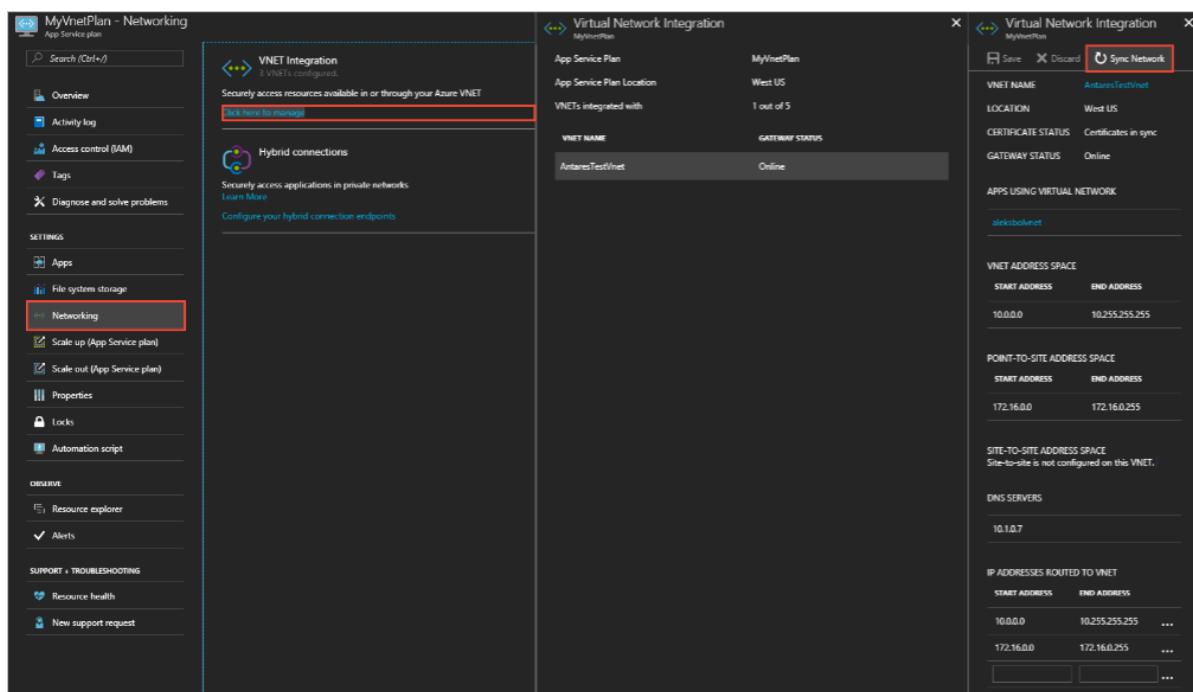
#### **NOTE**

For best performance, when you are using Azure VMs as DNS servers, IPv6 should be disabled. A [public IP address](#) should be assigned to each DNS server VM.

## Web apps

Suppose you need to perform name resolution from your web app built by using App Service, linked to a virtual network, to VMs in the same virtual network. In addition to setting up a custom DNS server that has a DNS forwarder that forwards queries to Azure (virtual IP 168.63.129.16), perform the following steps:

1. Enable virtual network integration for your web app, if not done already, as described in [Integrate your app with a virtual network](#).
2. In the Azure portal, for the App Service plan hosting the web app, select **Sync Network** under **Networking, Virtual Network Integration**.



If you need to perform name resolution from your web app built by using App Service, linked to a virtual network, to VMs in a different virtual network, you have to use custom DNS servers on both virtual networks, as follows:

- Set up a DNS server in your target virtual network, on a VM that can also forward queries to the recursive resolver in Azure (virtual IP 168.63.129.16). An example DNS forwarder is available in the [Azure Quickstart Templates gallery](#) and [GitHub](#).
- Set up a DNS forwarder in the source virtual network on a VM. Configure this DNS forwarder to forward queries to the DNS server in your target virtual network.
- Configure your source DNS server in your source virtual network's settings.
- Enable virtual network integration for your web app to link to the source virtual network, following the instructions in [Integrate your app with a virtual network](#).
- In the Azure portal, for the App Service plan hosting the web app, select **Sync Network** under **Networking**,

## **Virtual Network Integration.**

# Specify DNS servers

When you are using your own DNS servers, Azure provides the ability to specify multiple DNS servers per virtual network. You can also specify multiple DNS servers per network interface (for Azure Resource Manager), or per cloud service (for the classic deployment model). DNS servers specified for a network interface or cloud service get precedence over DNS servers specified for the virtual network.

### **NOTE**

Network connection properties, such as DNS server IPs, should not be edited directly within VMs. This is because they might get erased during service heal when the virtual network adaptor gets replaced. This applies to both Windows and Linux VMs.

When you are using the Azure Resource Manager deployment model, you can specify DNS servers for a virtual network and a network interface. For details, see [Manage a virtual network](#) and [Manage a network interface](#).

### **NOTE**

If you opt for custom DNS server for your virtual network, you must specify at least one DNS server IP address; otherwise, virtual network will ignore the configuration and use Azure-provided DNS instead.

When you are using the classic deployment model, you can specify DNS servers for the virtual network in the Azure portal or the [Network Configuration file](#). For cloud services, you can specify DNS servers via the [Service Configuration file](#) or by using PowerShell, with [New-AzureVM](#).

### **NOTE**

If you change the DNS settings for a virtual network or virtual machine that is already deployed, for the new DNS settings to take effect, you must perform a DHCP lease renewal on all affected VMs in the virtual network. For VMs running the Windows OS, you can do this by typing `ipconfig /renew` directly in the VM. The steps vary depending on the OS. See the relevant documentation for your OS type.

## Next steps

Azure Resource Manager deployment model:

- [Manage a virtual network](#)
- [Manage a network interface](#)

Classic deployment model:

- [Azure Service Configuration Schema](#)
- [Virtual Network Configuration Schema](#)
- [Configure a Virtual Network by using a network configuration file](#)

# Use dynamic DNS to register hostnames in your own DNS server

3/14/2019 • 3 minutes to read • [Edit Online](#)

Azure provides [name resolution](#) for virtual machines (VM) and role instances. When your name resolution needs exceed the capabilities provided by Azure's default DNS, you can provide your own DNS servers. Using your own DNS servers gives you the ability to tailor your DNS solution to suit your own specific needs. For example, you may need to access on-premises resources via your Active Directory domain controller.

When your custom DNS servers are hosted as Azure VMs, you can forward hostname queries for the same virtual network to Azure to resolve hostnames. If you do not wish to use this option, you can register your VM hostnames in your DNS server using dynamic DNS (DDNS). Azure doesn't have the credentials to directly create records in your DNS servers, so alternative arrangements are often needed. Some common scenarios, with alternatives follow:

## Windows clients

Non-domain-joined Windows clients attempt unsecured DDNS updates when they boot, or when their IP address changes. The DNS name is the hostname plus the primary DNS suffix. Azure leaves the primary DNS suffix blank, but you can set the suffix in the VM, via the [user interface](#) or [PowerShell](#).

Domain-joined Windows clients register their IP addresses with the domain controller by using secure DDNS. The domain-join process sets the primary DNS suffix on the client and creates and maintains the trust relationship.

## Linux clients

Linux clients generally don't register themselves with the DNS server on startup, they assume the DHCP server does it. Azure's DHCP servers do not have the credentials to register records in your DNS server. You can use a tool called `nsupdate`, which is included in the Bind package, to send DDNS updates. Because the DDNS protocol is standardized, you can use `nsupdate` even when you're not using Bind on the DNS server.

You can use the hooks that are provided by the DHCP client to create and maintain the hostname entry in the DNS server. During the DHCP cycle, the client executes the scripts in `/etc/dhcp/dhclient-exit-hooks.d/`. You can use the hooks to register the new IP address using `nsupdate`. For example:

```

#!/bin/sh
requireddomain=mydomain.local

# only execute on the primary nic
if [ "$interface" != "eth0" ]
then
    return
fi

# When you have a new IP, perform nsupdate
if [ "$reason" = BOUND ] || [ "$reason" = RENEW ] ||
[ "$reason" = REBIND ] || [ "$reason" = REBOOT ]
then
    host=`hostname`
    nsupdatecmds=/var/tmp/nsupdatecmds
    echo "update delete $host.$requireddomain a" > $nsupdatecmds
    echo "update add $host.$requireddomain 3600 a $new_ip_address" >> $nsupdatecmds
    echo "send" >> $nsupdatecmds

    nsupdate $nsupdatecmds
fi

```

You can also use the `nsupdate` command to perform secure DDNS updates. For example, when you're using a Bind DNS server, a public-private key pair is [generated](#). The DNS server is [configured](#) with the public part of the key, so that it can verify the signature on the request. To provide the key-pair to `nsupdate`, use the `-k` option, for the DDNS update request to be signed.

When you're using a Windows DNS server, you can use Kerberos authentication with the `-g` parameter in `nsupdate`, but it's not available in the Windows version of `nsupdate`. To use Kerberos, use `kinit` to load the credentials. For example, you can load credentials from a [keytab file](#), then `nsupdate -g` picks up the credentials, from the cache.

If needed, you can add a DNS search suffix to your VMs. The DNS suffix is specified in the `/etc/resolv.conf` file. Most Linux distros automatically manage the content of this file, so usually you can't edit it. However, you can override the suffix by using the DHCP client's `supersede` command. To override the suffix, add the following line to the `/etc/dhcp/dhclient.conf` file:

```
supersede domain-name <required-dns-suffix>;
```

# Optimize network throughput for Azure virtual machines

1/6/2020 • 3 minutes to read • [Edit Online](#)

Azure virtual machines (VM) have default network settings that can be further optimized for network throughput. This article describes how to optimize network throughput for Microsoft Azure Windows and Linux VMs, including major distributions such as Ubuntu, CentOS, and Red Hat.

## Windows VM

If your Windows VM supports [Accelerated Networking](#), enabling that feature would be the optimal configuration for throughput. For all other Windows VMs, using Receive Side Scaling (RSS) can reach higher maximal throughput than a VM without RSS. RSS may be disabled by default in a Windows VM. To determine whether RSS is enabled, and enable it if it's currently disabled, complete the following steps:

1. See if RSS is enabled for a network adapter with the `Get-NetAdapterRss` PowerShell command. In the following example output returned from the `Get-NetAdapterRss`, RSS is not enabled.

```
Name      : Ethernet
InterfaceDescription : Microsoft Hyper-V Network Adapter
Enabled    : False
```

2. To enable RSS, enter the following command:

```
Get-NetAdapter | % {Enable-NetAdapterRss -Name $_.Name}
```

The previous command does not have an output. The command changed NIC settings, causing temporary connectivity loss for about one minute. A Reconnecting dialog box appears during the connectivity loss. Connectivity is typically restored after the third attempt.

3. Confirm that RSS is enabled in the VM by entering the `Get-NetAdapterRss` command again. If successful, the following example output is returned:

```
Name      : Ethernet
InterfaceDescription : Microsoft Hyper-V Network Adapter
Enabled    : True
```

## Linux VM

RSS is always enabled by default in an Azure Linux VM. Linux kernels released since October 2017 include new network optimizations options that enable a Linux VM to achieve higher network throughput.

### Ubuntu for new deployments

The Ubuntu Azure kernel provides the best network performance on Azure and has been the default kernel since September 21, 2017. In order to get this kernel, first install the latest supported version of 16.04-LTS, as follows:

```
"Publisher": "Canonical",
"Offer": "UbuntuServer",
"Sku": "16.04-LTS",
"Version": "latest"
```

After the creation is complete, enter the following commands to get the latest updates. These steps also work for VMs currently running the Ubuntu Azure kernel.

```
#run as root or preface with sudo
apt-get -y update
apt-get -y upgrade
apt-get -y dist-upgrade
```

The following optional command set may be helpful for existing Ubuntu deployments that already have the Azure kernel but that have failed to further updates with errors.

```
#optional steps may be helpful in existing deployments with the Azure kernel
#run as root or preface with sudo
apt-get -f install
apt-get --fix-missing install
apt-get clean
apt-get -y update
apt-get -y upgrade
apt-get -y dist-upgrade
```

#### **Ubuntu Azure kernel upgrade for existing VMs**

Significant throughput performance can be achieved by upgrading to the Azure Linux kernel. To verify whether you have this kernel, check your kernel version.

```
#Azure kernel name ends with "-azure"
uname -r

#sample output on Azure kernel:
#4.13.0-1007-azure
```

If your VM does not have the Azure kernel, the version number usually begins with "4.4." If the VM does not have the Azure kernel, run the following commands as root:

```
#run as root or preface with sudo
apt-get update
apt-get upgrade -y
apt-get dist-upgrade -y
apt-get install "linux-azure"
reboot
```

#### **CentOS**

In order to get the latest optimizations, it is best to create a VM with the latest supported version by specifying the following parameters:

```
"Publisher": "OpenLogic",
"Offer": "CentOS",
"Sku": "7.4",
"Version": "latest"
```

New and existing VMs can benefit from installing the latest Linux Integration Services (LIS). The throughput

optimization is in LIS, starting from 4.2.2-2, although later versions contain further improvements. Enter the following commands to install the latest LIS:

```
sudo yum update  
sudo reboot  
sudo yum install microsoft-hyper-v
```

## Red Hat

In order to get the optimizations, it is best to create a VM with the latest supported version by specifying the following parameters:

```
"Publisher": "RedHat"  
"Offer": "RHEL"  
"Sku": "7-RAW"  
"Version": "latest"
```

New and existing VMs can benefit from installing the latest Linux Integration Services (LIS). The throughput optimization is in LIS, starting from 4.2. Enter the following commands to download and install LIS:

```
wget https://aka.ms/lis  
tar xvf lis  
cd LISISO  
sudo ./install.sh #or upgrade.sh if prior LIS was previously installed
```

Learn more about Linux Integration Services Version 4.2 for Hyper-V by viewing the [download page](#).

## Next steps

- See the optimized result with [Bandwidth/Throughput testing Azure VM](#) for your scenario.
- Read about how [bandwidth is allocated to virtual machines](#)
- Learn more with [Azure Virtual Network frequently asked questions \(FAQ\)](#)

# Viewing and modifying hostnames

9/17/2019 • 2 minutes to read • [Edit Online](#)

To allow your role instances to be referenced by host name, you must set the value for the host name in the service configuration file for each role. You do that by adding the desired host name to the **vmName** attribute of the **Role** element. The value of the **vmName** attribute is used as a base for the host name of each role instance. For example, if **vmName** is *webrole* and there are three instances of that role, the host names of the instances will be *webrole0*, *webrole1*, and *webrole2*. You do not need to specify a host name for virtual machines in the configuration file, because the host name for a virtual machine is populated based on the virtual machine name. For more information about configuring a Microsoft Azure service, see [Azure Service Configuration Schema \(.cscfg File\)](#)

## Viewing hostnames

You can view the host names of virtual machines and role instances in a cloud service by using any of the tools below.

### Service configuration file

You can download the service configuration file for a deployed service from the **Configure** blade of the service in the Azure portal. You can then look for the **vmName** attribute for the **Role name** element to see the host name. Keep in mind that this host name is used as a base for the host name of each role instance. For example, if **vmName** is *webrole* and there are three instances of that role, the host names of the instances will be *webrole0*, *webrole1*, and *webrole2*.

### Remote Desktop

After you enable Remote Desktop (Windows), Windows PowerShell remoting (Windows), or SSH (Linux and Windows) connections to your virtual machines or role instances, you can view the host name from an active Remote Desktop connection in various ways:

- Type hostname at the command prompt or SSH terminal.
- Type ipconfig /all at the command prompt (Windows only).
- View the computer name in the system settings (Windows only).

### Azure Service Management REST API

From a REST client, follow these instructions:

1. Ensure that you have a client certificate to connect to the Azure portal. To obtain a client certificate, follow the steps presented in [How to: Download and Import Publish Settings and Subscription Information](#).
2. Set a header entry named x-ms-version with a value of 2013-11-01.
3. Send a request in the following format: `https://management.core.windows.net/<subscription-id>/services/hostedservices/<service-name>?embed-detail=true`
4. Look for the **HostName** element for each **RoleInstance** element.

#### WARNING

You can also view the internal domain suffix for your cloud service from the REST call response by checking the **InternalDnsSuffix** element, or by running ipconfig /all from a command prompt in a Remote Desktop session (Windows), or by running cat /etc/resolv.conf from an SSH terminal (Linux).

## Modifying a hostname

You can modify the host name for any virtual machine or role instance by uploading a modified service configuration file, or by renaming the computer from a Remote Desktop session.

### Next steps

[Name Resolution \(DNS\)](#)

[Azure Service Configuration Schema \(.cscfg\)](#)

[Azure Virtual Network Configuration Schema](#)

[Specify DNS settings using network configuration files](#)

# Diagnostic logging for a network security group

1/8/2020 • 7 minutes to read • [Edit Online](#)

A network security group (NSG) includes rules that allow or deny traffic to a virtual network subnet, network interface, or both. When you enable diagnostic logging for an NSG, you can log the following categories of information:

- **Event:** Entries are logged for which NSG rules are applied to VMs, based on MAC address.
- **Rule counter:** Contains entries for how many times each NSG rule is applied to deny or allow traffic. The status for these rules is collected every 60 seconds.

Diagnostic logs are only available for NSGs deployed through the Azure Resource Manager deployment model. You cannot enable diagnostic logging for NSGs deployed through the classic deployment model. For a better understanding of the two models, see [Understanding Azure deployment models](#).

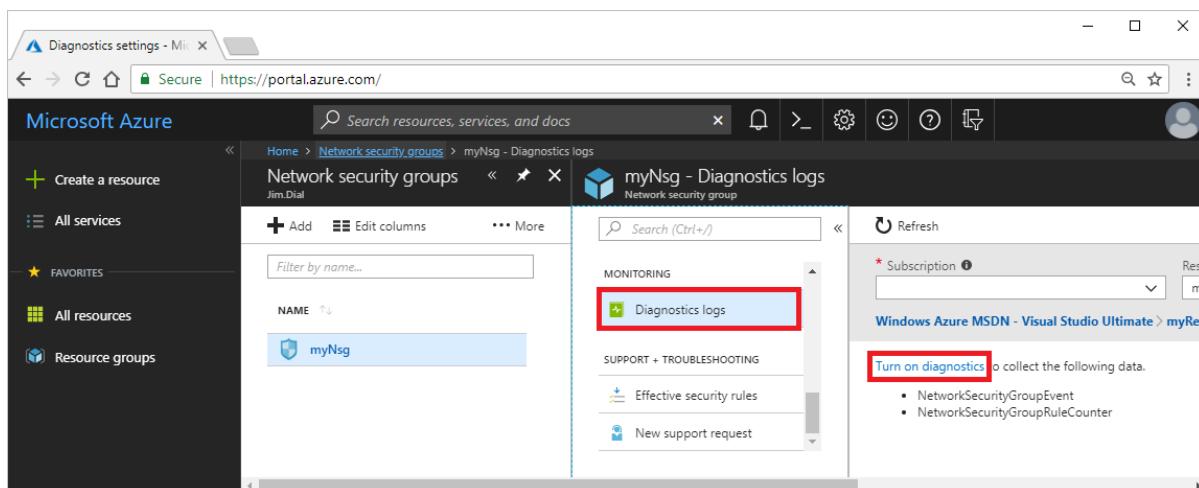
Diagnostic logging is enabled separately for *each* NSG you want to collect diagnostic data for. If you're interested in operational, or activity, logs instead, see Azure [activity logging](#).

## Enable logging

You can use the [Azure Portal](#), [PowerShell](#), or the [Azure CLI](#) to enable diagnostic logging.

### Azure Portal

1. Sign in to the [portal](#).
2. Select **All services**, then type *network security groups*. When **Network security groups** appear in the search results, select it.
3. Select the NSG you want to enable logging for.
4. Under **MONITORING**, select **Diagnostics logs**, and then select **Turn on diagnostics**, as shown in the following picture:



5. Under **Diagnostics settings**, enter, or select the following information, and then select **Save**:

SETTING	VALUE
Name	A name of your choosing. For example: <i>myNsgDiagnostics</i>

SETTING	VALUE
<b>Archive to a storage account, Stream to an event hub, and Send to Log Analytics</b>	You can select as many destinations as you choose. To learn more about each, see <a href="#">Log destinations</a> .
LOG	Select either, or both log categories. To learn more about the data logged for each category, see <a href="#">Log categories</a> .

- View and analyze logs. For more information, see [View and analyze logs](#).

## PowerShell

### 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

You can run the commands that follow in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell. It has common Azure tools preinstalled and configured to use with your account. If you run PowerShell from your computer, you need the Azure PowerShell module, version 1.0.0 or later. Run `Get-Module -ListAvailable Az` on your computer, to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to sign in to Azure with an account that has the [necessary permissions](#).

To enable diagnostic logging, you need the Id of an existing NSG. If you don't have an existing NSG, you can create one with [New-AzNetworkSecurityGroup](#).

Retrieve the network security group that you want to enable diagnostic logging for with [Get-AzNetworkSecurityGroup](#). For example, to retrieve an NSG named *myNsg* that exists in a resource group named *myResourceGroup*, enter the following command:

```
$Nsg=Get-AzNetworkSecurityGroup ` 
-Name myNsg ` 
-ResourceGroupName myResourceGroup
```

You can write diagnostic logs to three destination types. For more information, see [Log destinations](#). In this article, logs are sent to the *Log Analytics* destination, as an example. Retrieve an existing Log Analytics workspace with [Get-AzOperationalInsightsWorkspace](#). For example, to retrieve an existing workspace named *myWorkspace* in a resource group named *myWorkspaces*, enter the following command:

```
$Oms=Get-AzOperationalInsightsWorkspace ` 
-ResourceGroupName myWorkspaces ` 
-Name myWorkspace
```

If you don't have an existing workspace, you can create one with [New-AzOperationalInsightsWorkspace](#).

There are two categories of logging you can enable logs for. For more information, see [Log categories](#). Enable diagnostic logging for the NSG with [Set-AzDiagnosticSetting](#). The following example logs both event and counter category data to the workspace for an NSG, using the IDs for the NSG and workspace you retrieved previously:

```
Set-AzDiagnosticSetting ` 
-ResourceId $Nsg.Id ` 
-WorkspaceId $Oms.ResourceId ` 
-Enabled $true
```

If you only want to log data for one category or the other, rather than both, add the `-categories` option to the previous command, followed by *NetworkSecurityGroupEvent* or *NetworkSecurityGroupRuleCounter*. If you want to log to a different [destination](#) than a Log Analytics workspace, use the appropriate parameters for an Azure [Storage account](#) or [Event Hub](#).

View and analyze logs. For more information, see [View and analyze logs](#).

## Azure CLI

You can run the commands that follow in the [Azure Cloud Shell](#), or by running the Azure CLI from your computer. The Azure Cloud Shell is a free interactive shell. It has common Azure tools preinstalled and configured to use with your account. If you run the CLI from your computer, you need version 2.0.38 or later. Run `az --version` on your computer, to find the installed version. If you need to upgrade, see [Install Azure CLI](#). If you are running the CLI locally, you also need to run `az login` to sign in to Azure with an account that has the [necessary permissions](#).

To enable diagnostic logging, you need the Id of an existing NSG. If you don't have an existing NSG, you can create one with `az network nsg create`.

Retrieve the network security group that you want to enable diagnostic logging for with `az network nsg show`. For example, to retrieve an NSG named *myNsg* that exists in a resource group named *myResourceGroup*, enter the following command:

```
nsgId=$(az network nsg show \
--name myNsg \
--resource-group myResourceGroup \
--query id \
--output tsv)
```

You can write diagnostic logs to three destination types. For more information, see [Log destinations](#). In this article, logs are sent to the *Log Analytics* destination, as an example. For more information, see [Log categories](#).

Enable diagnostic logging for the NSG with `az monitor diagnostic-settings create`. The following example logs both event and counter category data to an existing workspace named *myWorkspace*, which exists in a resource group named *myWorkspaces*, and the ID of the NSG you retrieved previously:

```
az monitor diagnostic-settings create \
--name myNsgDiagnostics \
--resource $nsgId \
--logs '[ { "category": "NetworkSecurityGroupEvent", "enabled": true, "retentionPolicy": { "days": 30, "enabled": true } }, { "category": "NetworkSecurityGroupRuleCounter", "enabled": true, "retentionPolicy": { "days": 30, "enabled": true } } ]' \
--workspace myWorkspace \
--resource-group myWorkspaces
```

If you don't have an existing workspace, you can create one using the [Azure portal](#) or [PowerShell](#). There are two categories of logging you can enable logs for.

If you only want to log data for one category or the other, remove the category you don't want to log data for in the previous command. If you want to log to a different [destination](#) than a Log Analytics workspace, use the appropriate parameters for an Azure [Storage account](#) or [Event Hub](#).

View and analyze logs. For more information, see [View and analyze logs](#).

# Log destinations

Diagnostics data can be:

- [Written to an Azure Storage account](#), for auditing or manual inspection. You can specify the retention time (in days) using resource diagnostic settings.
- [Streamed to an Event hub](#) for ingestion by a third-party service, or custom analytics solution, such as PowerBI.
- [Written to Azure Monitor logs](#).

# Log categories

JSON-formatted data is written for the following log categories:

## Event

The event log contains information about which NSG rules are applied to VMs, based on MAC address. The following data is logged for each event. In the following example, the data is logged for a virtual machine with the IP address 192.168.1.4 and a MAC address of 00-0D-3A-92-6A-7C:

```
{  
  "time": "[DATE-TIME]",  
  "systemId": "[ID]",  
  "category": "NetworkSecurityGroupEvent",  
  "resourceId": "/SUBSCRIPTIONS/[SUBSCRIPTION-ID]/RESOURCEGROUPS/[RESOURCE-GROUP-  
NAME]/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/[NSG-NAME]",  
  "operationName": "NetworkSecurityGroupEvents",  
  "properties": {  
    "vnetResourceGuid": "[ID]",  
    "subnetPrefix": "192.168.1.0/24",  
    "macAddress": "00-0D-3A-92-6A-7C",  
    "primaryIPv4Address": "192.168.1.4",  
    "ruleName": "[SECURITY-RULE-NAME]",  
    "direction": "[DIRECTION-SPECIFIED-IN-RULE]",  
    "priority": "[PRIORITY-SPECIFIED-IN-RULE]",  
    "type": "[ALLOW-OR-DENY-AS-SPECIFIED-IN-RULE]",  
    "conditions": {  
      "protocols": "[PROTOCOLS-SPECIFIED-IN-RULE]",  
      "destinationPortRange": "[PORT-RANGE-SPECIFIED-IN-RULE]",  
      "sourcePortRange": "[PORT-RANGE-SPECIFIED-IN-RULE]",  
      "sourceIP": "[SOURCE-IP-OR-RANGE-SPECIFIED-IN-RULE]",  
      "destinationIP": "[DESTINATION-IP-OR-RANGE-SPECIFIED-IN-RULE]"  
    }  
  }  
}
```

## Rule counter

The rule counter log contains information about each rule applied to resources. The following example data is logged each time a rule is applied. In the following example, the data is logged for a virtual machine with the IP address 192.168.1.4 and a MAC address of 00-0D-3A-92-6A-7C:

```
{
  "time": "[DATE-TIME]",
  "systemId": "[ID]",
  "category": "NetworkSecurityGroupRuleCounter",
  "resourceId": "/SUBSCRIPTIONS/[SUBSCRIPTION ID]/RESOURCEGROUPS/[RESOURCE-GROUP-NAME]/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/[NSG-NAME]",
  "operationName": "NetworkSecurityGroupCounters",
  "properties": {
    "vnetResourceGuid": "[ID]",
    "subnetPrefix": "192.168.1.0/24",
    "macAddress": "00-0D-3A-92-6A-7C",
    "primaryIPv4Address": "192.168.1.4",
    "ruleName": "[SECURITY-RULE-NAME]",
    "direction": "[DIRECTION-SPECIFIED-IN-RULE]",
    "type": "[ALLOW-OR-DENY-AS-SPECIFIED-IN-RULE]",
    "matchedConnections": 125
  }
}
```

#### NOTE

The source IP address for the communication is not logged. You can enable [NSG flow logging](#) for an NSG however, which logs all of the rule counter information, as well as the source IP address that initiated the communication. NSG flow log data is written to an Azure Storage account. You can analyze the data with the [traffic analytics](#) capability of Azure Network Watcher.

## View and analyze logs

To learn how to view diagnostic log data, see [Azure Diagnostic Logs overview](#). If you send diagnostics data to:

- **Azure Monitor logs:** You can use the [network security group analytics](#) solution for enhanced insights. The solution provides visualizations for NSG rules that allow or deny traffic, per MAC address, of the network interface in a virtual machine.
- **Azure Storage account:** Data is written to a PT1H.json file. You can find the:
  - Event log in the following path:  
`insights-logs-networksecuritygroupevent/resourceId=/SUBSCRIPTIONS/[ID]/RESOURCEGROUPS/[RESOURCE-GROUP-NAME-FOR-NSG]/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/[NSG NAME]/y=[YEAR]/m=[MONTH/d=[DAY]/h=[HOUR]/m=[MINUTE]`
  - Rule counter log in the following path:  
`insights-logs-networksecuritygrouprulecounter/resourceId=/SUBSCRIPTIONS/[ID]/RESOURCEGROUPS/[RESOURCE-GROUP-NAME-FOR-NSG]/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/[NSG NAME]/y=[YEAR]/m=[MONTH/d=[DAY]/h=[HOUR]/m=[MINUTE]`

## Next steps

- Learn more about [Activity logging](#), previously known as audit or operational logs. Activity logging is enabled by default for NSGs created through either Azure deployment model. To determine which operations were completed on NSGs in the activity log, look for entries that contain the following resource types:
  - Microsoft.ClassicNetwork/networkSecurityGroups
  - Microsoft.ClassicNetwork/networkSecurityGroups/securityRules
  - Microsoft.Network/networkSecurityGroups
  - Microsoft.Network/networkSecurityGroups/securityRules
- To learn how to log diagnostic information, to include the source IP address for each flow, see [NSG flow logging](#).

# Quickstart: Create a NAT gateway using the Azure portal

2/24/2020 • 5 minutes to read • [Edit Online](#)

This quickstart shows you how to use Azure Virtual Network NAT service. You'll create a NAT gateway to provide outbound connectivity for a virtual machine in Azure.

## NOTE

Azure Virtual Network NAT is available as public preview at this time and available in a limited set of [regions](#). This preview is provided without a service level agreement and isn't recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Sign in to Azure

Sign in to the [Azure portal](#).

### Create a virtual network

Before you deploy a VM and can use your NAT gateway, we need to create the resource group and virtual network.

1. On the upper-left side of the screen, select **Create a resource** > **Networking** > **Virtual network**, or search for **Virtual network** in the Marketplace search.
2. In **Create virtual network**, enter or select this information:

SETTING	VALUE
Name	Enter <b>myVNet</b> .
Address space	Enter <b>192.168.0.0/16</b> .
Subscription	Select your subscription.
Resource group	Select create new - <b>myResourceGroupNAT</b> .
Location	Select <b>East US 2</b> .
Subnet - Name	Enter <b>mySubnet</b> .
Subnet - Address range	Enter <b>192.168.0.0/24</b> .

3. Leave the rest of the defaults and select **Create**.

### Create a VM to use the NAT gateway

We'll now create a VM to use the NAT service. This VM has a public IP to use as an instance-level Public IP to allow you to access the VM. NAT service is flow direction aware and will replace the default Internet destination in your subnet. The VM's public IP address won't be used for outbound connections.

1. On the upper-left side of the portal, select **Create a resource** > **Compute** > **Ubuntu Server 18.04 LTS**,

or search for **Ubuntu Server 18.04 LTS** in the Marketplace search.

2. In **Create a virtual machine**, type or select the following values in the **Basics** tab:

- **Subscription > Resource Group**: Select **myResourceGroupNAT**.
- **Instance Details > Virtual machine name**: Type **myVM**.
- **Instance Details > Region** > select **East US 2**.
- **Administrator account > Authentication type**: Select **Password**.
- **Administrator account** > Enter the **Username**, **Password**, and **Confirm password** information.
- **Inbound port rules > Public inbound ports**: Select **Allow selected ports**.
- **Inbound port rules > Select inbound ports**: Select **SSH (22)**
- Select the **Networking** tab, or select **Next: Disks**, then **Next: Networking**.

3. In the **Networking** tab make sure the following are selected:

- **Virtual network**: **myVnet**
- **Subnet**: **mySubnet**
- **Public IP** > Select **Create new**. In the **Create public IP address** window, type **myPublicIPVM** in the **Name** field, and choose **Standard** for the **SKU**. Click **OK**.
- **NIC network security group**: Select **Basic**.
- **Public inbound ports**: Select **Allow selected ports**.
- **Select inbound ports**: Confirm **SSH** is selected.

4. In the **Management** tab, under **Monitoring**, set **Boot diagnostics** to **Off**.

5. Select **Review + create**.

6. Review the settings and click **Create**.

## Create the NAT gateway

You can use one or more public IP address resources, public IP prefixes, or both. We'll add a public IP resource, public IP prefix, and a NAT gateway resource.

This section details how you can create and configure the following components of the NAT service using the NAT gateway resource:

- A public IP pool and public IP prefix to use for outbound flows translated by the NAT gateway resource.
- Change the idle timeout from the default of 4 minutes to 10 minutes.

### Create a public IP address

1. On the upper-left side of the portal, select **Create a resource > Networking > Public IP address**, or search for **Public IP address** in the Marketplace search.

2. In **Create public IP address**, enter or select this information:

SETTING	VALUE
IP Version	Select <b>IPv4</b> .
SKU	Select <b>Standard</b> .
Name	Enter <b>myPublicIP</b> .
Subscription	Select your subscription.

SETTING	VALUE
Resource group	Select <b>myResourceGroupNAT</b> .
Location	Select <b>East US 2</b> .

3. Leave the rest of the defaults and select **Create**.

#### Create a public IP prefix

1. On the upper-left side of the portal, select **Create a resource > Networking > Public IP prefix**, or search for **Public IP prefix** in the Marketplace search.

2. In **Create a public IP prefix**, type or select the following values in the **Basics** tab:

- **Subscription > Resource Group**: Select **myResourceGroupNAT**.
- **Instance details > Name**: Type **myPublicIPprefix**.
- **Instance details > Region**: Select **East US 2**.
- **Instance details > Prefix size**: Select **/31 (2 addresses)**

3. Leave the rest the defaults and select **Review + create**.

4. Review the settings, and then select **Create**.

#### Create a NAT gateway resource

1. On the upper-left side of the portal, select **Create a resource > Networking > NAT gateway**, or search for **NAT gateway** in the Marketplace search.

2. In **Create network address translation (NAT) gateway**, type or select the following values in the **Basics** tab:

- **Subscription > Resource Group**: Select **myResourceGroupNAT**.
- **Instance details > NAT gateway name**: Type **myNATgateway**.
- **Instance details > Region**: Select **East US 2**.
- **Instance details > Idle timeout (minutes)**: Type **10**.
- Select the **Public IP** tab, or select **Next: Public IP**.

3. In the **Public IP** tab, type or select the following values:

- **Public IP addresses**: Select **myPublicIP**.
- **Public IP Prefixes**: Select **myPublicIPprefix**.
- Select the **Subnet** tab, or select **Next: Subnet**.

4. In the **Subnet** tab, type or select the following values:

- **Virtual Network**: Select **myResourceGroupNAT > myVnet**.
- **Subnet name**: Select the box next to **mySubnet**.

5. Select **Review + create**.

6. Review the settings, and then select **Create**.

## Discover the IP address of the VM

1. On the left side of the portal, select **Resource groups**.

2. Select **myResourceGroupNAT**.

3. Select **myVM**.

4. In **Overview**, copy the **Public IP address** value, and paste into notepad so you can use it to access the VM.

### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it to access the VM.

## Sign in to VM

Open an [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh <username>@<ip-address-destination>
```

You're now ready to use the NAT service.

## Clean up resources

When no longer needed, delete the resource group, NAT gateway, and all related resources. Select the resource group **myResourceGroupNAT** that contains the NAT gateway, and then select **Delete**.

## Next steps

In this tutorial, you created a NAT gateway and a VM to use it.

Review metrics in Azure Monitor to see your NAT service operating. Diagnose issues such as resource exhaustion of available SNAT ports. Resource exhaustion of SNAT ports is addressed by adding additional public IP address resources or public IP prefix resources or both.

- Learn about [Azure Virtual Network NAT](#)
- Learn about [NAT gateway resource](#).
- Quickstart for deploying [NAT gateway resource using Azure CLI](#).
- Quickstart for deploying [NAT gateway resource using Azure PowerShell](#).
- Quickstart for deploying [NAT gateway resource using Azure portal](#).
- [Provide feedback on the Public Preview](#).

# Quickstart: Create a NAT gateway using Azure PowerShell

2/27/2020 • 8 minutes to read • [Edit Online](#)

This quickstart shows you how to use Azure Virtual Network NAT service. You'll create a NAT gateway to provide outbound connectivity for a virtual machine in Azure.

## NOTE

Azure Virtual Network NAT is available as public preview at this time and available in a limited set of [regions](#). This preview is provided without a service level agreement and isn't recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

You can complete this tutorial using Azure Cloud Shell or run the commands locally. If you haven't used Azure Cloud Shell, [sign in now](#).

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

## Create a resource group

Create a resource group with [New-AzResourceGroup](#). An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named **myResourceGroupNAT** in the **eastus2** location:

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
  
New-AzResourceGroup -Name $rsg -Location $loc
```

## Create the NAT gateway

Public IP options for NAT gateway are:

- **Public IP addresses**
- **Public IP prefixes**

Both can be used with NAT gateway.

We'll add a public IP address and a public IP prefix to this scenario to demonstrate.

### Create a public IP address

To access the Internet, you need one or more public IP addresses for the NAT gateway. Use [New-AzPublicIpAddress](#) to create a public IP address resource named **myPublicIP** in **myResourceGroupNAT**. The result of this command will be stored in a variable **\$publicIP** for later use.

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
$sku = 'Standard'  
$pbnm = 'myPublicIP'  
  
$publicIP =  
New-AzPublicIpAddress -Name $pbnm -ResourceGroupName $rsg -AllocationMethod Static -Location $loc -Sku $sku
```

### Create a public IP prefix

Use [New-AzPublicIpPrefix](#) to create a public IP prefix resource named **myPublicIPprefix** in **myResourceGroupNAT**. The result of this command will be stored in a variable named **\$publicIPPrefix** for later use.

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
$pxnm = 'myPublicIPprefix'  
  
$publicIPPrefix =  
New-AzPublicIpPrefix -Name $pxnm -ResourceGroupName $rsg -Location $loc -PrefixLength 31
```

## Create a NAT gateway resource

This section details how you can create and configure the following components of the NAT service using the NAT gateway resource:

- A public IP pool and public IP prefix to use for outbound flows translated by the NAT gateway resource.
- Change the idle timeout from the default of 4 minutes to 10 minutes.

Create a global Azure NAT gateway with [New-AzNatGateway](#). The result of this command will create a gateway resource named **myNATgateway** that uses the public IP address **myPublicIP** and the public IP prefix **myPublicIPPrefix**. The idle timeout is set to 10 minutes. The result of this command will be stored in a variable named **\$natGateway** for later use.

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
$sku = 'Standard'  
$gnm = 'myNATgateway'  
  
$natGateway =  
New-AzNatGateway -Name $gnm -ResourceGroupName $rsg -PublicIpAddress $publicIP -PublicIpPrefix $publicIPPrefix  
-Location $loc -Sku $sku -IdleTimeoutInMinutes 10
```

At this point, the NAT gateway is functional and all that is missing is to configure which subnets of a virtual network should use it.

## Configure virtual network

Create the virtual network and associate the subnet to the gateway.

Create a virtual network named **myVnet** with a subnet named **mySubnet** using [New-AzVirtualNetworkSubnetConfig](#) in the **myResourceGroup** using [New-AzVirtualNetwork](#). The IP address space for the virtual network is **192.168.0.0/16**. The subnet within the virtual network is **192.168.0.0/24**. The result of the commands will be stored in variables named **\$subnet** and **\$vnet** for later use.

```
$sbnm = 'mySubnet'  
$vnmm = 'myVnet'  
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
$pfxsub = '192.168.0.0/24'  
$pfxvn = '192.168.0.0/16'  
  
$subnet =  
New-AzVirtualNetworkSubnetConfig -Name $sbnm -AddressPrefix $pfxsub -NatGateway $natGateway  
  
$vnet =  
New-AzVirtualNetwork -Name $vnmm -ResourceGroupName $rsg -Location $loc -AddressPrefix $pfxvn -Subnet $subnet
```

All outbound traffic to Internet destinations is now using the NAT service. It isn't necessary to configure a UDR.

## Create a VM to use the NAT service

We'll now create a VM to use the NAT service. This VM has a public IP to use as an instance-level Public IP to allow you to access the VM. NAT service is flow direction aware and will replace the default Internet destination in your subnet. The VM's public IP address won't be used for outbound connections.

### Create public IP for source VM

We create a public IP to be used to access the VM. Use [New-AzPublicIpAddress](#) to create a public IP address resource named **myPublicIPVM** in **myResourceGroupNAT**. The result of this command will be stored in a

variable named **\$publicIpVM** for later use.

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
$ipnm = 'myPublicIPVM'  
$sku = 'Standard'  
  
$publicIpVM =  
New-AzPublicIpAddress -Name $ipnm -ResourceGroupName $rsg -AllocationMethod Static -Location $loc -Sku $sku
```

## Create an NSG and expose SSH endpoint for VM

Standard public IP addresses are 'secure by default', we need to create an NSG to allow inbound access for ssh.

Use [New-AzNetworkSecurityGroup](#) to create an NSG resource named **myNSG**. Use [New-AzNetworkSecurityRuleConfig](#) to create an NSG rule for SSH access named **ssh** in **myResourceGroupNAT**. The result of this command will be stored in a variable named **\$nsg** for later use.

```
$rnm = 'ssh'  
$rdesc = 'SSH access'  
$acc = 'Allow'  
$pro = 'Tcp'  
$dir = 'Inbound'  
$pri = '100'  
$prt = '22'  
$rsg = 'myResourceGroupNAT'  
$rnm = 'myNSG'  
$loc = 'eastus2'  
  
$sshrule =  
New-AzNetworkSecurityRuleConfig -Name $rnm -Description $rdesc -Access $acc -Protocol $pro -Direction $dir -Priority $pri -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * -DestinationPortRange $prt  
  
$nsg =  
New-AzNetworkSecurityGroup -ResourceGroupName $rsg -Name $rnm -Location $loc -SecurityRules $sshrule
```

## Create NIC for VM

Create a network interface with [New-AzNetworkInterface](#) named **myNic**. This command associates the Public IP address and the network security group. The result of this command will be stored in a variable named **\$nic** for later use.

```
$rsg = 'myResourceGroupNAT'  
$nnm = 'myNic'  
$loc = 'eastus2'  
  
$nic =  
New-AzNetworkInterface -ResourceGroupName $rsg -Name $nnm -NetworkSecurityGroupID $nsg.Id -PublicIPAddressID $publicIPVM.Id -SubnetID $vnet.Subnets[0].Id -Location $loc
```

## Create VM

### Create SSH key pair

You need an SSH key pair to complete this quickstart. If you already have an SSH key pair, you can skip this step.

Use ssh-keygen to create an SSH key pair.

```
ssh-keygen -t rsa -b 2048
```

For more detailed information on how to create SSH key pairs, including the use of PuTTY, see [How to use SSH keys with Windows](#).

If you create the SSH key pair using the Cloud Shell, the key pair is stored in a container image. This [storage account is automatically created](#). Don't delete the storage account, or the file share within, until after you've retrieved your keys.

#### Create VM Configuration

To create a VM in PowerShell, you create a configuration that has settings for the image to use, size, and authentication options. The configuration is used to build the VM.

Define the SSH credentials, OS information, and VM size. In this example, the SSH key is stored in `~/.ssh/id_rsa.pub`.

```
#Define a credential object

$securePassword =
ConvertTo-SecureString ' ' -AsPlainText -Force

$cred =
New-Object System.Management.Automation.PSCredential ("azureuser", $securePassword)

# Create a virtual machine configuration

$vnm = 'myVM'
$vsz = 'Standard_D1'
$pub = 'Canonical'
$off = 'UbuntuServer'
$sku = '18.04-LTS'
$ver = 'latest'

$vmConfig =
New-AzVMConfig -VMName $vnm -VMSize $vsz

Set-AzVMOperatingSystem -VM $vmConfig -Linux -ComputerName $vnm -Credential $cred -
DisablePasswordAuthentication

Set-AzVMSourceImage -VM $vmConfig -PublisherName $pub -Offer $off -Skus $sku -Version $ver

Add-AzVMNetworkInterface -VM $vmConfig -Id $nic.Id

# Configure the SSH key

$sshPublicKey = cat ~/.ssh/id_rsa.pub

Add-AzVMSshPublicKey -VM $vmConfig -KeyData $sshPublicKey -Path "/home/azureuser/.ssh/authorized_keys"
```

Combine the configuration definitions to create a VM named **myVM** with [New-AzVM](#) in **myResourceGroupNAT**.

```
$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'

New-AzVM -ResourceGroupName $rsg -Location $loc -VM $vmConfig
```

Wait for the VM to prepare to deploy then continue with the rest of the steps.

## Discover the IP address of the VM

First we need to discover the IP address of the VM you've created. To get the public IP address of the VM, use [Get-AzPublicIpAddress](#).

```
$rsg = 'myResourceGroupNAT'  
$nmm = 'myPublicIPVM'  
  
Get-AzPublicIpAddress -ResourceGroupName $rsg -Name $nmm | select IPAddress
```

#### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it to access the VM.

### Sign in to VM

The SSH credentials should be stored in your Cloud Shell from the previous operation. Open an [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh azureuser@<ip-address-destination>
```

You're now ready to use the NAT service.

### Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) command to remove the resource group and all resources contained within.

```
Remove-AzResourceGroup -Name myResourceGroupNAT
```

### Next steps

In this tutorial, you created a NAT gateway and a VM to use it.

Review metrics in Azure Monitor to see your NAT service operating. Diagnose issues such as resource exhaustion of available SNAT ports. Resource exhaustion of SNAT ports is addressed by adding additional public IP address resources or public IP prefix resources or both.

- Learn about [Azure Virtual Network NAT](#)
- Learn about [NAT gateway resource](#).
- Quickstart for deploying [NAT gateway resource using Azure CLI](#).
- Quickstart for deploying [NAT gateway resource using Azure PowerShell](#).
- Quickstart for deploying [NAT gateway resource using Azure portal](#).
- [Provide feedback on the Public Preview](#).

# Quickstart: Create a NAT gateway using Azure CLI

2/24/2020 • 7 minutes to read • [Edit Online](#)

This quickstart shows you how to use Azure Virtual Network NAT service. You'll create a NAT gateway to provide outbound connectivity for a virtual machine in Azure.

## NOTE

Azure Virtual Network NAT is available as public preview at this time and available in a limited set of [regions](#). This preview is provided without a service level agreement and isn't recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

You can complete this tutorial using Azure Cloud Shell or run the respective commands locally. If you have never used Azure Cloud Shell, [sign in now](#) to go through the initial setup. If you choose to run these commands locally, you need to install CLI. This tutorial requires that you're running a version of the Azure CLI version 2.0.71 or later. To find the version, run `az --version`. If you need to install or upgrade, see [Install Azure CLI](#).

## Create a resource group

Create a resource group with [az group create](#). An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named **myResourceGroupNAT** in the **eastus2** location:

```
az group create \
--name myResourceGroupNAT \
--location eastus2
```

## Create the NAT Gateway

### Create a public IP address

To access the public Internet, you need one or more public IP addresses for the NAT gateway. Use [az network public-ip create](#) to create a public IP address resource named **myPublicIP** in **myResourceGroupNAT**.

```
az network public-ip create \
--resource-group myResourceGroupNAT \
--name myPublicIP \
--sku standard
```

### Create a public IP prefix

You can use one or more public IP address resources, public IP prefixes, or both with NAT gateway. We'll add a public IP prefix resource to this scenario to demonstrate. Use [az network public-ip prefix create](#) to create a public IP prefix resource named **myPublicIPprefix** in **myResourceGroupNAT**.

```
az network public-ip prefix create \
--resource-group myResourceGroupNAT \
--name myPublicIPprefix \
--length 31
```

### Create a NAT gateway resource

This section details how you can create and configure the following components of the NAT service using the NAT gateway resource:

- A public IP pool and public IP prefix to use for outbound flows translated by the NAT gateway resource.
- Change the idle timeout from the default of 4 minutes to 10 minutes.

Create a global Azure NAT gateway with [az network nat gateway create](#) named **myNATgateway**. The command uses both the public IP address **myPublicIP** and the public IP prefix **myPublicIPprefix**. The command changes the idle timeout to **10** minutes.

```
az network nat gateway create \
--resource-group myResourceGroupNAT \
--name myNATgateway \
--public-ip-addresses myPublicIP \
--public-ip-prefixes myPublicIPprefix \
--idle-timeout 10
```

At this point, the NAT gateway is functional and all that is missing is to configure which subnets of a virtual network should use it.

## Configure virtual network

Before you deploy a VM and can use your NAT gateway, we need to create the virtual network.

Create a virtual network named **myVnet** with a subnet named **mySubnet** in the **myResourceGroupNAT** using [az network vnet create](#). The IP address space for the virtual network is **192.168.0.0/16**. The subnet within the virtual network is **192.168.0.0/24**.

```
az network vnet create \
--resource-group myResourceGroupNAT \
--location eastus2 \
--name myVnet \
--address-prefix 192.168.0.0/16 \
--subnet-name mySubnet \
--subnet-prefix 192.168.0.0/24
```

### Configure NAT service for source subnet

We'll configure the source subnet **mySubnet** in virtual network **myVnet** to use a specific NAT gateway resource **myNATgateway** with [az network vnet subnet update](#). This command will activate the NAT service on the specified subnet.

```
az network vnet subnet update \
--resource-group myResourceGroupNAT \
--vnet-name myVnet \
--name mySubnet \
--nat-gateway myNATgateway
```

All outbound traffic to Internet destinations is now using the NAT gateway. It's not necessary to configure a UDR.

## Create a VM to use the NAT service

We'll now create a VM to use the NAT service. This VM has a public IP to use as an instance-level Public IP to allow you to access the VM. NAT service is flow direction aware and will replace the default Internet destination in your subnet. The VM's public IP address won't be used for outbound connections.

### Create public IP for source VM

We create a public IP to be used to access the VM. Use [az network public-ip create](#) to create a public IP address resource named **myPublicIPVM** in **myResourceGroupNAT**.

```
az network public-ip create \
--resource-group myResourceGroupNAT \
--name myPublicIPVM \
--sku standard
```

### Create an NSG for VM

Because Standard Public IP addresses are 'secure by default', we need to create an NSG to allow inbound access for ssh access. Use [az network nsg create](#) to create an NSG resource named **myNSG** in **myResourceGroupNAT**.

```
az network nsg create \
--resource-group myResourceGroupNAT \
--name myNSG
```

### Expose SSH endpoint on source VM

We create a rule in the NSG for SSH access to the source vm. Use [az network nsg rule create](#) to create an NSG rule named **ssh** in the NSG named **myNSG** in **myResourceGroupNAT**.

```
az network nsg rule create \
--resource-group myResourceGroupNAT \
--nsg-name myNSG \
--priority 100 \
--name ssh \
--description "SSH access" \
--access allow \
--protocol tcp \
--direction inbound \
--destination-port-ranges 22
```

## Create NIC for VM

Create a network interface with [az network nic create](#) and associate with the Public IP address and the network security group.

```
az network nic create \
--resource-group myResourceGroupNAT \
--name myNic \
--vnet-name myVnet \
--subnet mySubnet \
--public-ip-address myPublicIPVM \
--network-security-group myNSG
```

## Create VM

Create the virtual machine with [az vm create](#). We generate ssh keys for this VM and store the private key to use later.

```
az vm create \
--resource-group myResourceGroupNAT \
--name myVM \
--nics myNic \
--image UbuntuLTS \
--generate-ssh-keys
```

Wait for the VM to deploy then continue with the rest of the steps.

## Discover the IP address of the VM

First we need to discover the IP address of the VM you've created. To retrieve the public IP address of the VM, use [az network public-ip show](#).

```
az network public-ip show \
--resource-group myResourceGroupNAT \
--name myPublicIPVM \
--query [ipAddress] \
--output tsv
```

### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it to access the VM.

## Sign in to VM

The SSH credentials should be stored in your Cloud Shell from the previous operation. Open an [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh <ip-address-destination>
```

You're now ready to use the NAT service.

## Clean up resources

When no longer needed, you can use the [az group delete](#) command to remove the resource group and all resources contained within.

```
az group delete \
--name myResourceGroupNAT
```

## Next steps

In this tutorial, you created a NAT gateway and a VM to use it.

Review metrics in Azure Monitor to see your NAT service operating. Diagnose issues such as resource exhaustion of available SNAT ports. Resource exhaustion of SNAT ports is addressed by adding additional public IP address resources or public IP prefix resources or both.

- Learn about [Azure Virtual Network NAT](#)
- Learn about [NAT gateway resource](#).
- Quickstart for deploying [NAT gateway resource using Azure CLI](#).
- Quickstart for deploying [NAT gateway resource using Azure PowerShell](#).
- Quickstart for deploying [NAT gateway resource using Azure portal](#).
- [Provide feedback on the Public Preview](#).

# Tutorial: Create a NAT Gateway using the Azure portal and test the NAT service

2/24/2020 • 10 minutes to read • [Edit Online](#)

In this tutorial, you'll create a NAT gateway to provide outbound connectivity for virtual machines in Azure. To test the NAT gateway, you deploy a source and destination virtual machine. You'll test the NAT gateway by making outbound connections to a public IP address from the source to the destination virtual machine. This tutorial deploys source and destination in two different virtual networks in the same resource group for simplicity only.

## NOTE

Azure Virtual Network NAT is available as public preview at this time and available in a limited set of [regions](#). This preview is provided without a service level agreement and isn't recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Sign in to Azure

Sign in to the [Azure portal](#).

## Prepare the source for outbound traffic

We'll guide you through configuration of a full test environment and the execution of the tests itself in the next steps. We'll start with the source, which will use the NAT gateway resource we create in later steps.

### Create a virtual network

Before you deploy a VM and can use your NAT gateway, we need to create the resource group and virtual network.

1. On the upper-left side of the screen, select **Create a resource > Networking > Virtual network**, or search for **Virtual Network** in the Marketplace search.
2. In **Create virtual network**, enter or select this information:

SETTING	VALUE
Name	Enter <b>myVNetsource</b> .
Address space	Enter <b>192.168.0.0/16</b> .
Subscription	Select your subscription.
Resource group	Select create new - <b>myResourceGroupNAT</b> .
Location	Select <b>East US 2</b> .
Subnet - Name	Enter <b>mySubnetsource</b> .
Subnet - Address range	Enter <b>192.168.0.0/24</b> .

3. Leave the rest of the defaults and select **Create**.

## Create source virtual machine

We'll now create a VM to use the NAT service. This VM has a public IP to use as an instance-level Public IP to allow you to access the VM. NAT service is flow direction aware and will replace the default Internet destination in your subnet. The VM's public IP address won't be used for outbound connections.

To test the NAT gateway, we'll assign a public IP address resource as an instance-level Public IP to access this VM from the outside. This address is only used to access it for the test. We'll demonstrate how the NAT service takes precedence over other outbound options.

You could also create this VM without a public IP and create another VM to use as a jumpbox without a public IP as an exercise.

1. On the upper-left side of the portal, select **Create a resource > Compute > Ubuntu Server 18.04 LTS**, or search for **Ubuntu Server 18.04 LTS** in the Marketplace search.
2. In **Create a virtual machine**, enter or select the following values in the **Basics** tab:
  - **Subscription > Resource Group:** Select **myResourceGroupNAT**.
  - **Instance Details > Virtual machine name:** enter **myVMsource**.
  - **Instance Details > Region:** select **East US 2**.
  - **Administrator account > Authentication enter:** Select **Password**.
  - **Administrator account > Enter the Username, Password, and Confirm password information.**
  - **Inbound port rules > Public inbound ports:** Select **Allow selected ports**.
  - **Inbound port rules > Select inbound ports:** Select **SSH (22)**
  - Select the **Networking** tab, or select **Next: Disks**, then **Next: Networking**.
3. In the **Networking** tab make sure the following are selected:
  - **Virtual network:** **myVnetsource**
  - **Subnet:** **mySubnetsource**
  - **Public IP:** > Select **Create new**. In the **Create public IP address** window, enter **myPublicIPsourceVM** in the **Name** field. Select **Standard** for the **SKU**. Leave the rest at the defaults and click **OK**.
  - **NIC network security group:** Select **Basic**.
  - **Public inbound ports:** Select **Allow selected ports**.
  - **Select inbound ports:** Confirm **SSH** is selected.
4. In the **Management** tab, under **Monitoring**, set **Boot diagnostics** to **Off**.
5. Select **Review + create**.
6. Review the settings and click **Create**.

## Create the NAT Gateway

You can use one or more public IP address resources, public IP prefixes, or both with NAT gateway. We'll add a public IP resource, public IP prefix, and a NAT gateway resource.

This section details how you can create and configure the following components of the NAT service using the NAT gateway resource:

- A public IP pool and public IP prefix to use for outbound flows translated by the NAT gateway resource.
- Change the idle timeout from the default of 4 minutes to 10 minutes.

### Create a public IP address

1. On the upper-left side of the portal, select **Create a resource > Networking > Public IP address**, or search for **Public IP address** in the Marketplace search.

2. In **Create public IP address**, enter or select this information:

SETTING	VALUE
IP Version	Select <b>IPv4</b> .
SKU	Select <b>Standard</b> .
Name	Enter <b>myPublicIPsource</b> .
Subscription	Select your subscription.
Resource group	Select <b>myResourceGroupNAT</b> .
Location	Select <b>East US 2</b> .

3. Leave the rest of the defaults and select **Create**.

#### Create a public IP prefix

1. On the upper-left side of the portal, select **Create a resource > Networking > Public IP prefix**, or search for **Public IP prefix** in the Marketplace search.
2. In **Create a public IP prefix**, enter or select the following values in the **Basics** tab:
  - **Subscription > Resource Group**: Select **myResourceGroupNAT**>
  - **Instance details > Name**: enter **myPublicIPprefixsource**.
  - **Instance details > Region**: Select **East US 2**.
  - **Instance details > Prefix size**: Select **/31 (2 addresses)**
3. Leave the rest the defaults and select **Review + create**.
4. Review the settings, and then select **Create**.

#### Create a NAT gateway resource

1. On the upper-left side of the portal, select **Create a resource > Networking > NAT gateway**, or search for **NAT gateway** in the Marketplace search.
2. In **Create network address translation (NAT) gateway**, enter or select the following values in the **Basics** tab:
  - **Subscription > Resource Group**: Select **myResourceGroupNAT**.
  - **Instance details > NAT gateway name**: enter **myNATgateway**.
  - **Instance details > Region**: Select **East US 2**.
  - **Instance details > Idle timeout (minutes)**: enter **10**.
  - Select the **Public IP** tab, or select **Next: Public IP**.
3. In the **Public IP** tab, enter or select the following values:
  - **Public IP addresses**: Select **myPublicIPsource**.
  - **Public IP Prefixes**: Select **myPublicIPprefixsource**.
  - Select the **Subnet** tab, or select **Next: Subnet**.
4. In the **Subnet** tab, enter or select the following values:
  - **Virtual Network**: Select **myResourceGroupNAT > myVnetsource**.
  - **Subnet name**: Select the box next to **mySubnetsource**.
5. Select **Review + create**.

6. Review the settings, and then select **Create**.

All outbound traffic to Internet destinations is now using the NAT service. It isn't necessary to configure a UDR.

## Prepare destination for outbound traffic

We'll now create a destination for the outbound traffic translated by the NAT service to allow you to test it.

### Configure virtual network for destination

Before you deploy a VM for the destination, we need to create a virtual network where the destination virtual machine can reside. The following are the same steps as for the source VM with some small changes to expose the destination endpoint.

1. On the upper-left side of the screen, select **Create a resource > Networking > Virtual network**.

2. In **Create virtual network**, enter or select this information:

SETTING	VALUE
Name	Enter <b>myVNetdestination</b> .
Address space	Enter <b>192.168.0.0/16</b> .
Subscription	Select your subscription.
Resource group	Select create new - <b>myResourceGroupNAT</b> .
Location	Select <b>East US 2</b> .
Subnet - Name	Enter <b>mySubnetdestination</b> .
Subnet - Address range	Enter <b>192.168.0.0/24</b> .

### Create destination virtual machine

1. On the upper-left side of the portal, select **Create a resource > Compute > Ubuntu Server 18.04 LTS**, or search for **Ubuntu Server 18.04 LTS** in the Marketplace search.

2. In **Create a virtual machine**, enter or select the following values in the **Basics** tab:

- **Subscription > Resource Group**: Select **myResourceGroupNAT**.
- **Instance Details > Virtual machine name**: enter **myVMdestination**.
- **Instance Details > Region** > select **East US 2**.
- **Administrator account > Authentication enter**: Select **Password**.
- **Administrator account > Enter the Username, Password, and Confirm password** information.
- **Inbound port rules > Public inbound ports**: Select **Allow selected ports**.
- **Inbound port rules > Select inbound ports**: Select **SSH (22)** and **HTTP (80)**.
- Select the **Networking** tab, or select **Next: Disks**, then **Next: Networking**.

3. In the **Networking** tab make sure the following are selected:

- **Virtual network**: **myVnetdestination**
- **Subnet**: **mySubnetdestination**
- **Public IP** > Select **Create new**. In the **Create public IP address** window, enter **myPublicIPdestinationVM** in the **Name** field. Select **Standard** for **SKU**. Leave the rest at the defaults and click **OK**.

- **NIC network security group:** Select **Basic**.
  - **Public inbound ports:** Select **Allow selected ports**.
  - **Select inbound ports:** Confirm **SSH** and **HTTP** is selected.
4. In the **Management** tab, under **Monitoring**, set **Boot diagnostics** to **Off**.
  5. Select **Review + create**.
  6. Review the settings, and then select **Create**.

## Prepare a web server and test payload on destination VM

First we need to discover the IP address of the destination VM.

1. On the left side of the portal, select **Resource groups**.
2. Select **myResourceGroupNAT**.
3. Select **myVMdestination**.
4. In **Overview**, copy the **Public IP address** value, and paste into notepad so you can use it to access the VM.

### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it in subsequent steps. Indicate this is the destination virtual machine.

### Sign in to destination VM

Open an [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh <username>@<ip-address-destination>
```

Copy and paste the following commands once you've logged in.

```
sudo apt-get -y update && \
sudo apt-get -y upgrade && \
sudo apt-get -y dist-upgrade && \
sudo apt-get -y autoremove && \
sudo apt-get -y autoclean && \
sudo apt-get -y install nginx && \
sudo ln -sf /dev/null /var/log/nginx/access.log && \
sudo touch /var/www/html/index.html && \
sudo rm /var/www/html/index.nginx-debian.html && \
sudo dd if=/dev/zero of=/var/www/html/100k bs=1024 count=100
```

These commands will update your virtual machine, install nginx, and create a 100-KBytes file. This file will be retrieved from the source VM using the NAT service.

Close the SSH session with the destination VM.

## Prepare test on source VM

First we need to discover the IP address of the source VM.

1. On the left side of the portal, select **Resource groups**.
2. Select **myResourceGroupNAT**.
3. Select **myVMsource**.

4. In **Overview**, copy the **Public IP address** value, and paste into notepad so you can use it to access the VM.

#### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it in subsequent steps. Indicate this is the source virtual machine.

## Log into source VM

Open a new tab for [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh <username>@<ip-address-source>
```

Copy and paste the following commands to prepare for testing the NAT service.

```
sudo apt-get -y update && \
sudo apt-get -y upgrade && \
sudo apt-get -y dist-upgrade && \
sudo apt-get -y autoremove && \
sudo apt-get -y autoclean && \
sudo apt-get install -y nload golang && \
echo 'export GOPATH=${HOME}/go' >> .bashrc && \
echo 'export PATH=${PATH}: ${GOPATH}/bin' >> .bashrc && \
. ~/.bashrc &&
go get -u github.com/rakyll/hey
```

This command will update your virtual machine, install go, install [hey](#) from GitHub, and update your shell environment.

You're now ready to test the NAT service.

## Validate NAT service

While logged into the source VM, you can use **curl** and **hey** to generate requests to the destination IP address.

Use curl to retrieve the 100-KBytes file. Replace **<ip-address-destination>** in the example below with the destination IP address you have previously copied. The **--output** parameter indicates that the retrieved file will be discarded.

```
curl http://<ip-address-destination>/100k --output /dev/null
```

You can also generate a series of requests using **hey**. Again, replace **<ip-address-destination>** with the destination IP address you have previously copied.

```
hey -n 100 -c 10 -t 30 --disable-keepalive http://<ip-address-destination>/100k
```

This command will generate 100 requests, 10 concurrently, with a timeout of 30 seconds, and without reusing the TCP connection. Each request will retrieve 100 Kbytes. At the end of the run, **hey** will report some statistics about how well the NAT service did.

## Clean up resources

When no longer needed, delete the resource group, NAT gateway, and all related resources. Select the resource

group **myResourceGroupNAT** that contains the NAT gateway, and then select **Delete**.

## Next steps

In this tutorial, you created a NAT gateway, created a source and destination VM, and then tested the NAT gateway.

Review metrics in Azure Monitor to see your NAT service operating. Diagnose issues such as resource exhaustion of available SNAT ports. Resource exhaustion of SNAT ports is easily addressed by adding additional public IP address resources or public IP prefix resources or both.

- Learn about [Virtual Network NAT](#)
- Learn about [NAT gateway resource](#).
- Quickstart for deploying NAT gateway resource using [Azure CLI](#).
- Quickstart for deploying NAT gateway resource using [Azure PowerShell](#).
- Quickstart for deploying NAT gateway resource using [Azure portal](#).
- [Provide feedback on the Public Preview](#).

# Tutorial: Create a NAT gateway using Azure PowerShell and test the NAT service

2/27/2020 • 14 minutes to read • [Edit Online](#)

In this tutorial, you'll create a NAT gateway to provide outbound connectivity for virtual machines in Azure. To test the NAT gateway, you deploy a source and destination virtual machine. You'll test the NAT gateway by making outbound connections to a public IP address. These connections will come from the source to the destination virtual machine. This tutorial deploys source and destination in two different virtual networks in the same resource group for simplicity.

## NOTE

Azure Virtual Network NAT is available as public preview at this time and available in a limited set of [regions](#). This preview is provided without a service level agreement and isn't recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

You can complete this tutorial using Azure Cloud Shell or run the respective commands locally. If you have never used Azure Cloud Shell, you should [sign in now](#).

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

## Create a resource group

Create a resource group with [az group create](#). An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named **myResourceGroupNAT** in the **eastus2** location:

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
  
New-AzResourceGroup -Name $rsg -Location $loc
```

## Create the NAT gateway

### Create a public IP address

To access the Internet, you need one or more public IP addresses for the NAT gateway. Use [New-AzPublicIpAddress](#) to create a public IP address resource named **myPublicIPsource** in **myResourceGroupNAT**. The result of this command will be stored in a variable named **\$publicIPsource** for later use.

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
$pips = 'myPublicIPsource'  
$alm = 'Static'  
$sku = 'Standard'  
  
$publicIPsource =  
New-AzPublicIpAddress -Name $pips -ResourceGroupName $rsg -AllocationMethod $alm -Location $loc -Sku $sku
```

### Create a public IP prefix

Use [New-AzPublicIpPrefix](#) to create a public IP prefix resource named **myPublicIPprefixsource** in **myResourceGroupNAT**. The result of this command will be stored in a variable named **\$publicIPPrefixsource** for later use.

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
$prips = 'mypublicIPprefixsource'  
  
$publicIPPrefixsource =  
New-AzPublicIpPrefix -Name $prips -ResourceGroupName $rsg -Location $loc -PrefixLength 31
```

### Create a NAT gateway resource

This section details how you can create and configure the following components of the NAT service using the NAT gateway resource:

- A public IP pool and public IP prefix to use for outbound flows translated by the NAT gateway resource.
- Change the idle timeout from the default of 4 minutes to 10 minutes.

Create a global Azure NAT gateway with [New-AzNatGateway](#). The result of this command will create a gateway resource named **myNATgateway** that uses the public IP address **myPublicIPsource** and the public IP prefix **myPublicIPprefixsource**. The idle timeout is set to 10 minutes. The result of this command will be stored in a variable named **\$natGateway** for later use.

```
$rsg = 'myResourceGroupNAT'  
$loc = 'eastus2'  
$sku = 'Standard'  
$nnm = 'myNATgateway'  
  
$natGateway =  
New-AzNatGateway -Name $nnm -ResourceGroupName $rsg -PublicIpAddress $publicIPsource -PublicIpPrefix  
$publicIPPrefixsource -Location $loc -Sku $sku -IdleTimeoutInMinutes 10
```

At this point, the NAT gateway is functional and all that is missing is to configure which subnets of a virtual network should use it.

## Prepare the source for outbound traffic

We'll guide you through setup of a full test environment. You'll set up a test using open-source tools to verify the NAT gateway. We'll start with the source, which will use the NAT gateway we created previously.

### Configure virtual network for source

Create the virtual network and associate the subnet to the gateway.

Create a virtual network named **myVnetsource** with a subnet named **mySubnetsource** using [New-AzVirtualNetworkSubnetConfig](#) in the **myResourceGroupNAT** using [New-AzVirtualNetwork](#). The IP address space for the virtual network is **192.168.0.0/16**. The subnet within the virtual network is **192.168.0.0/24**. The result of the commands will be stored in variables named **\$subnetsource** and **\$vnetsource** for later use.

```
$rsg = 'myResourceGroupNAT'  
$sbn = 'mySubnetsource'  
$spfx = '192.168.0.0/24'  
$vnm = 'myVnetsource'  
$vpfx = '192.168.0.0/16'  
$loc = 'eastus2'  
  
$subnetsource =  
New-AzVirtualNetworkSubnetConfig -Name $sbn -AddressPrefix $spfx -NatGateway $natGateway  
  
$vnetsource =  
New-AzVirtualNetwork -Name $vnm -ResourceGroupName $rsg -Location $loc -AddressPrefix $vpfx -Subnet $subnet
```

All outbound traffic to Internet destinations is now using the NAT service. It isn't necessary to configure a UDR.

Before we can test the NAT gateway, we need to create a source VM. We'll assign a public IP address resource as an instance-level Public IP to access this VM from the outside. This address is only used to access it for the test. We'll demonstrate how the NAT service takes precedence over other outbound options.

You could also create this VM without a public IP and create another VM to use as a jumpbox without a public IP as an exercise.

### Create public IP for source VM

We create a public IP to be used to access the VM. Use [New-AzPublicIpAddress](#) to create a public IP address resource named **myPublicIPVM** in **myResourceGroupNAT**. The result of this command will be stored in a variable named **\$publicIPsourceVM** for later use.

```

$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'
$sku = 'Standard'
$pipvm = 'myPublicIpsoourceVM'
$alm = 'Static'

$publicIpsoourceVM =
New-AzPublicIpAddress -Name $pipvm -ResourceGroupName $rsg -AllocationMethod $alm -Location $loc -sku $sku

```

## Create an NSG and expose SSH endpoint for VM

Because Standard Public IP addresses are 'secure by default', we create an NSG to allow inbound access for ssh. NAT service is flow direction aware. This NSG won't be used for outbound once NAT gateway is configured on the same subnet. Use [New-AzNetworkSecurityGroup](#) to create an NSG resource named **myNSGsource**. Use [New-AzNetworkSecurityRuleConfig](#) to create an NSG rule for SSH access named **ssh** in **myResourceGroupNAT**. The result of this command will be stored in variable named **\$nsgsource** for later use.

```

$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'
$rnm = 'ssh'
$rdsc = 'SSH access'
$acc = 'Allow'
$prt = 'Tcp'
$dir = 'Inbound'
$nsnm = 'myNSGsource'

$sshrule =
New-AzNetworkSecurityRuleConfig -Name $rnm -Description $rdsc -Access $acc -Protocol $prt -Direction $dir -
Priority 100 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * -DestinationPortRange 22

$nsgsource =
New-AzNetworkSecurityGroup -ResourceGroupName $rsg -Name $nsnm -Location $loc -SecurityRules $sshrule

```

## Create NIC for source VM

Create a network interface with [New-AzNetworkInterface](#) named **myNicsource**. This command will associate the Public IP address and the network security group. The result of this command will be stored in a variable named **\$nicsource** for later use.

```

$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'
$nin = 'myNicsource'

$nicsource =
New-AzNetworkInterface -ResourceGroupName $rsg -Name $nin -NetworkSecurityGroupID $nsgsource.Id -
PublicIPAddressID $publicIPVMsource.Id -SubnetID $vnetsource.Subnets[0].Id -Location $loc

```

## Create a source VM

### Create SSH key pair

You need an SSH key pair to complete this quickstart. If you already have an SSH key pair, you can skip this step.

Use ssh-keygen to create an SSH key pair.

```
ssh-keygen -t rsa -b 2048
```

For more detailed information on how to create SSH key pairs, including the use of PuTTy, see [How to use SSH keys with Windows](#).

If you create the SSH key pair using the Cloud Shell, the key pair is stored in a container image. This [storage account is automatically created](#). Don't delete the storage account, or the file share within, until after you've retrieved your keys.

#### Create VM Configuration

To create a VM in PowerShell, you create a configuration that has settings like the image to use, size, and authentication options. Then the configuration is used to build the VM.

Define the SSH credentials, OS information, and VM size. In this example, the SSH key is stored in `~/ssh/id_rsa.pub`.

```
# Define a credential object

$securePassword =
ConvertTo-SecureString ' ' -AsPlainText -Force
$cred =
New-Object System.Management.Automation.PSCredential ("azureuser", $securePassword)

# Create a virtual machine configuration
$vnn = 'myVMsource'
$vms = 'Standard_D1'
$pub = 'Canonical'
$off = 'UbuntuServer'
$skus = '18.04-LTS'
$ver = 'latest'

$vmConfigsource =
New-AzVMConfig -VMName $vnn -VMSize $vms

Set-AzVMOperatingSystem -VM $vmConfigsource -Linux -ComputerName $vnn -Credential $cred -
DisablePasswordAuthentication

Set-AzVMSourceImage -VM $vmConfigsource -PublisherName $pub -Offer $off -Skus $skus -Version $ver

Add-AzVMNetworkInterface -VM $vmConfigsource -Id $nicsource.Id

# Configure the SSH key

$sshPublicKey = cat ~/ssh/id_rsa.pub

Add-AzVMSshPublicKey -VM $vmConfigsource -KeyData $sshPublicKey -Path "/home/azureuser/.ssh/authorized_keys"
```

Combine the configuration definitions to create a VM named **myVMsource** with [New-AzVM](#) in **myResourceGroupNAT**.

```
$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'

New-AzVM -ResourceGroupName $rsg -Location $loc -VM $vmConfigsource
```

While the command will return immediately, it may take a few minutes for the VM to get deployed.

## Prepare destination for outbound traffic

We'll now create a destination for the outbound traffic translated by the NAT service to allow you to test it.

#### Configure virtual network for destination

We need to create a virtual network where the destination virtual machine will be. These commands are the same steps as for the source VM. Small changes have been added to expose the destination endpoint.

Create a virtual network named **myVnetdestination** with a subnet named **mySubnetdestination** using [New-AzVirtualNetworkSubnetConfig](#) in the **myResourceGroupNAT** using [New-AzVirtualNetwork](#). The IP address space for the virtual network is **192.168.0.0/16**. The subnet within the virtual network is **192.168.0.0/24**. The result of the commands will be stored in variables named **\$subnetdestination** and **\$vnetdestination** for later use.

```
$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'
$sbdn = 'mySubnetdestination'
$spfx = '192.168.0.0/24'
$vdn = 'myVnetdestination'
$vpfx = '192.168.0.0/16'

$subnetdestination =
New-AzVirtualNetworkSubnetConfig -Name $sbdn -AddressPrefix $spfx

$vnetdestination =
New-AzVirtualNetwork -Name $vdn -ResourceGroupName $rsg -Location $loc -AddressPrefix $vpfx -Subnet
$subnetdestination
```

### Create public IP for destination VM

We create a public IP to be used to access the source VM. Use [New-AzPublicIpAddress](#) to create a public IP address resource named **myPublicIPdestinationVM** in **myResourceGroupNAT**. The result of this command will be stored in a variable named **\$publicIpdestinationVM** for later use.

```
$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'
$sku = 'Standard'
$all = 'Static'
$pipd = 'myPublicIPdestinationVM'

$publicIpdestinationVM =
New-AzPublicIpAddress -Name $pipd -ResourceGroupName $rsg -AllocationMethod $all -Location $loc -Sku $sku
```

### Create an NSG and expose SSH and HTTP endpoint for VM

Standard Public IP addresses are 'secure by default', we create an NSG to allow inbound access for ssh. Use [New-AzNetworkSecurityGroup](#) to create an NSG resource named **myNSGdestination**. Use [New-AzNetworkSecurityRuleConfig](#) to create an NSG rule for SSH access named **ssh**. Use [New-AzNetworkSecurityRuleConfig](#) to create an NSG rule for HTTP access named **http**. Both rules will be created in **myResourceGroupNAT**. The result of this command will be stored in a variable named **\$nsgdestination** for later use.

```

$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'
$snm = 'ssh'
$sdsc = 'SSH access'
$acc = 'Allow'
$prt = 'Tcp'
$dir = 'Inbound'
$hnm = 'http'
$hdsc = 'HTTP access'
$nsnm = 'myNSGdestination'

$sshrule =
New-AzNetworkSecurityRuleConfig -Name $snm -Description $sdsc -Access $acc -Protocol $prt -Direction $dir -
Priority 100 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * -DestinationPortRange 22

$httprule =
New-AzNetworkSecurityRuleConfig -Name $hnm -Description $hdsc -Access $acc -Protocol $prt -Direction $dir -
Priority 101 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix * -DestinationPortRange 80

$nsgdestination =
New-AzNetworkSecurityGroup -ResourceGroupName $rsg -Name $nsnm -Location $loc -SecurityRules $sshrule,$httprule

```

## Create NIC for destination VM

Create a network interface with [New-AzNetworkInterface](#) named **myNicdestination**. This command will associate with the Public IP address and the network security group. The result of this command will be stored in a variable named **\$nicdestination** for later use.

```

$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'
$nnm = 'myNicdestination'

$nicdestination =
New-AzNetworkInterface -ResourceGroupName $rsg -Name $nnm -NetworkSecurityGroupID $nsgdestination.Id -
PublicIPAddressID $publicIPdestinationVM.Id -SubnetID $vnetdestination.Subnets[0].Id -Location $loc

```

## Create a destination VM

### Create VM Configuration

To create a VM in PowerShell, you create a configuration that has settings like the image to use, size, and authentication options. Then the configuration is used to build the VM.

Define the SSH credentials, OS information, and VM size. In this example, the SSH key is stored in `~/ssh/id_rsa.pub`.

```

# Define a credential object

$securePassword =
ConvertTo-SecureString ' ' -AsPlainText -Force
$cred =
New-Object System.Management.Automation.PSCredential ("azureuser", $securePassword)

# Create a virtual machine configuration

$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'
$vmd = 'myVMdestination'
$vms = 'Standard_D1'
$pub = 'Canonical'
$off = 'UbuntuServer'
$skus = '18.04-LTS'
$ver = 'latest'

$vmConfigdestination = New-AzVMConfig -VMName $vmd -VMSize $vms

Set-AzVMOperatingSystem -VM $vmConfigdestination -Linux -ComputerName $vmd -Credential $cred -
DisablePasswordAuthentication

Set-AzVMSourceImage -VM $vmConfigdestination -PublisherName $pub -Offer $off -Skus $skus -Version $ver

Add-AzVMNetworkInterface -VM $vmConfigdestination -Id $nicdestination.Id

# Configure the SSH key

$sshPublicKey = cat ~/.ssh/id_rsa.pub

Add-AzVMSshPublicKey -VM $vmConfigdestination -KeyData $sshPublicKey -Path
"/home/azureuser/.ssh/authorized_keys"

```

Combine the configuration definitions to create a VM named **myVMdestination** with [New-AzVM](#) in **myResourceGroupNAT**.

```

$rsg = 'myResourceGroupNAT'
$loc = 'eastus2'

New-AzVM -ResourceGroupName $rsg -Location $loc -VM $vmConfigdestination

```

While the command will return immediately, it may take a few minutes for the VM to get deployed.

## Prepare a web server and test payload on destination VM

First we need to discover the IP address of the destination VM. To get the public IP address of the VM, use [Get-AzPublicIpAddress](#).

```

$rsg = 'myResourceGroupNAT'
$pipn = 'myPublicIPdestinationVM'

Get-AzPublicIpAddress -ResourceGroupName $rsg -Name $pipn | select IPAddress

```

### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it in subsequent steps. Indicate this is the destination virtual machine.

## Sign in to destination VM

The SSH credentials should be stored in your Cloud Shell from the previous operation. Open an [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh azureuser@<ip-address-destination>
```

Copy and paste the following commands once you've logged in.

```
sudo apt-get -y update && \
sudo apt-get -y upgrade && \
sudo apt-get -y dist-upgrade && \
sudo apt-get -y autoremove && \
sudo apt-get -y autoclean && \
sudo apt-get -y install nginx && \
sudo ln -sf /dev/null /var/log/nginx/access.log && \
sudo touch /var/www/html/index.html && \
sudo rm /var/www/html/index.nginx-debian.html && \
sudo dd if=/dev/zero of=/var/www/html/100k bs=1024 count=100
```

These commands will update your virtual machine, install nginx, and create a 100-KBytes file. This file will be retrieved from the source VM using the NAT service.

Close the SSH session with the destination VM.

## Prepare test on source VM

First we need to discover the IP address of the source VM. To get the public IP address of the VM, use [Get-AzPublicIpAddress](#).

```
$rsg = 'myResourceGroupNAT'
$pipn = 'myPublicIPsourceVM'

Get-AzPublicIpAddress -ResourceGroupName $rsg -Name $pipn | select IPAddress
```

### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it in subsequent steps. Indicate this is the source virtual machine.

## Log into source VM

Again, the SSH credentials are stored in Cloud Shell. Open a new tab for [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh azureuser@<ip-address-source>
```

Copy and paste the following commands to prepare for testing the NAT service.

```
sudo apt-get -y update && \
sudo apt-get -y upgrade && \
sudo apt-get -y dist-upgrade && \
sudo apt-get -y autoremove && \
sudo apt-get -y autoclean && \
sudo apt-get install -y nload golang && \
echo 'export GOPATH=${HOME}/go' >> .bashrc && \
echo 'export PATH=$PATH:${GOPATH}/bin' >> .bashrc && \
. ~/.bashrc &&
go get -u github.com/rakyll/hey
```

These commands will update your virtual machine, install go, install [hey](#) from GitHub, and update your shell environment.

You're now ready to test NAT service.

## Validate NAT service

While logged into the source VM, you can use **curl** and **hey** to generate requests to the destination IP address.

Use curl to retrieve the 100-KBytes file. Replace **<ip-address-destination>** in the example below with the destination IP address you have previously copied. The **--output** parameter indicates that the retrieved file will be discarded.

```
curl http://<ip-address-destination>/100k --output /dev/null
```

You can also generate a series of requests using **hey**. Again, replace **<ip-address-destination>** with the destination IP address you have previously copied.

```
hey -n 100 -c 10 -t 30 --disable-keepalive http://<ip-address-destination>/100k
```

This command will generate 100 requests, 10 concurrently, with a timeout of 30 seconds. The TCP connection won't be reused. Each request will retrieve 100 Kbytes. At the end of the run, **hey** will report some statistics about how well the NAT service did.

## Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) command to remove the resource group and all resources contained within.

```
Remove-AzResourceGroup -Name myResourceGroupNAT
```

## Next steps

In this tutorial, you created a NAT gateway, created a source and destination VM, and then tested the NAT gateway.

Review metrics in Azure Monitor to see your NAT service operating. Diagnose issues such as resource exhaustion of available SNAT ports. Resource exhaustion of SNAT ports is easily addressed by adding additional public IP address resources or public IP prefix resources or both.

- Learn about [Virtual Network NAT](#)
- Learn about [NAT gateway resource](#).
- Quickstart for deploying [NAT gateway resource using Azure CLI](#).

- Quickstart for deploying NAT gateway resource using Azure PowerShell.
- Quickstart for deploying NAT gateway resource using Azure portal.
- [Provide feedback on the Public Preview.](#)

# Tutorial: Create a NAT gateway using Azure CLI and test the NAT service

2/27/2020 • 12 minutes to read • [Edit Online](#)

In this tutorial, you'll create a NAT gateway to provide outbound connectivity for virtual machines in Azure. To test the NAT gateway, you deploy a source and destination virtual machine. You'll test the NAT gateway by making outbound connections to a public IP address. These connections will come from the source to the destination virtual machine. This tutorial deploys source and destination in two different virtual networks in the same resource group for simplicity only.

## NOTE

Azure Virtual Network NAT is available as public preview at this time and available in a limited set of [regions](#). This preview is provided without a service level agreement and isn't recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

You can complete this tutorial using Azure Cloud Shell or run the respective commands locally. If you haven't used Azure Cloud Shell, you should [sign in now](#).

If you choose to run these commands locally, you need to install CLI. This tutorial requires that you're running a

version of the Azure CLI version 2.0.71 or later. To find the version, run `az --version`. If you need to install or upgrade, see [Install Azure CLI](#).

## Create a resource group

Create a resource group with [az group create](#). An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named **myResourceGroupNAT** in the **eastus2** location:

```
az group create \
--name myResourceGroupNAT \
--location eastus2
```

## Create the NAT Gateway

### Create a public IP address

To access the public Internet, you need one or more public IP addresses for the NAT gateway. Use [az network public-ip create](#) to create a public IP address resource named **myPublicIPsource** in **myResourceGroupNAT**.

```
az network public-ip create \
--resource-group myResourceGroupNAT \
--name myPublicIPsource \
--sku standard
```

### Create a public IP prefix

You can use one or more public IP address resources, public IP prefixes or both with NAT gateway. We'll add a public IP prefix resource to this scenario to demonstrate. Use [az network public-ip prefix create](#) to create a public IP prefix resource named **myPublicIPprefixsource** in **myResourceGroupNAT**.

```
az network public-ip prefix create \
--resource-group myResourceGroupNAT \
--name myPublicIPprefixsource \
--length 31
```

### Create a NAT gateway resource

This section details how you can create and configure the following components of the NAT service using the NAT gateway resource:

- A public IP pool and public IP prefix to use for outbound flows translated by the NAT gateway resource.
- Change the idle timeout from the default of 4 minutes to 10 minutes.

Create a global Azure NAT gateway with [az network nat gateway create](#) named **myNATgateway**. The command uses both the public IP address **myPublicIP** and the public IP prefix **myPublicIPprefix**. The command also changes the idle timeout to 10 minutes.

```
az network nat gateway create \
--resource-group myResourceGroupNAT \
--name myNATgateway \
--public-ip-addresses myPublicIPsource \
--public-ip-prefixes myPublicIPprefixsource \
--idle-timeout 10
```

At this point, the NAT gateway is functional and all that is missing is to configure which subnets of a virtual

network should use it.

## Prepare the source for outbound traffic

We'll guide you through setup of a full test environment. You'll set up a test using open-source tools to verify the NAT gateway. We'll start with the source, which will use the NAT gateway we created previously.

### Configure virtual network for source

Before you deploy a VM and can test your NAT gateway, we need to create the virtual network.

Create a virtual network named **myVnetsource** with a subnet named **mySubnetsource** in the **myResourceGroupNAT** using [az network Microsoft Azure Virtual Network create](#). The IP address space for the virtual network is **192.168.0.0/16**. The subnet within the virtual network is **192.168.0.0/24**.

```
az network vnet create \
--resource-group myResourceGroupNAT \
--location eastus2 \
--name myVnetsource \
--address-prefix 192.168.0.0/16 \
--subnet-name mySubnetsource \
--subnet-prefix 192.168.0.0/24
```

### Configure NAT service for source subnet

Configure the source subnet **mySubnetsource** in virtual network **myVnetsource** to use a specific NAT gateway resource **myNATgateway** with [az network Microsoft Azure Virtual Network subnet update](#). This command will activate the NAT service on the specified subnet.

```
az network vnet subnet update \
--resource-group myResourceGroupNAT \
--vnet-name myVnetsource \
--name mySubnetsource \
--nat-gateway myNATgateway
```

All outbound traffic to Internet destinations is now using the NAT service. It's not necessary to configure a UDR.

Before we can test the NAT gateway, we need to create a source VM. We'll assign a public IP address resource as an instance-level public IP to access this VM from the outside. This address is only used to access it for the test. We'll demonstrate how the NAT service takes precedence over other outbound options.

You could also create this VM without a public IP and create another VM to use as a jumpbox without a public IP as an exercise.

### Create public IP for source VM

We create a public IP to be used to access the source VM. Use [az network public-ip create](#) to create a public IP address resource named **myPublicIPsourceVM** in **myResourceGroupNAT**.

```
az network public-ip create \
--resource-group myResourceGroupNAT \
--name myPublicIPsourceVM \
--sku standard
```

### Create an NSG for source VM

Because Standard public IP addresses are 'secure by default', we need to create an NSG to allow inbound access for ssh access. Azure NAT service is flow direction aware. This NSG won't be used for outbound once the NAT gateway is configured on the same subnet. Use [az network nsg create](#) to create an NSG resource named

## myNSGsource in myResourceGroupNAT.

```
az network nsg create \
--resource-group myResourceGroupNAT \
--name myNSGsource
```

## Expose SSH endpoint on source VM

We create a rule in the NSG for SSH access to the source vm. Use [az network nsg rule create](#) to create an NSG rule named **ssh**. This rule will be created in the NSG named **myNSGsource** in the resource group **myResourceGroupNAT**.

```
az network nsg rule create \
--resource-group myResourceGroupNAT \
--nsg-name myNSGsource \
--priority 100 \
--name ssh \
--description "SSH access" \
--access allow \
--protocol tcp \
--direction inbound \
--destination-port-ranges 22
```

## Create NIC for source VM

Create a network interface with [az network nic create](#) and associate with the public IP address and the network security group.

```
az network nic create \
--resource-group myResourceGroupNAT \
--name myNicsource \
--vnet-name myVnetsource \
--subnet mySubnetsource \
--public-ip-address myPublicIPSourceVM \
--network-security-group myNSGsource
```

## Create a source VM

Create the virtual machine with [az vm create](#). We generate ssh keys for this VM and store the private key to use later.

```
az vm create \
--resource-group myResourceGroupNAT \
--name myVMsource \
--nics myNicsource \
--image UbuntuLTS \
--generate-ssh-keys \
--no-wait
```

While the command will return immediately, it may take a few minutes for the VM to get deployed.

## Prepare destination for outbound traffic

We'll now create a destination for the outbound traffic translated by the NAT service to allow you to test it.

### Configure virtual network for destination

We need to create a virtual network where the destination virtual machine will be. These commands are the same steps as for the source VM with small changes to expose the destination endpoint.

Create a virtual network named **myVnetdestination** with a subnet named **mySubnetdestination** in the **myResourceGroupNAT** using [az network Microsoft Azure Virtual Network create](#). The IP address space for the virtual network is **192.168.0.0/16**. The subnet within the virtual network is **192.168.0.0/24**.

```
az network vnet create \
--resource-group myResourceGroupNAT \
--location westus \
--name myVnetdestination \
--address-prefix 192.168.0.0/16 \
--subnet-name mySubnetdestination \
--subnet-prefix 192.168.0.0/24
```

### Create public IP for destination VM

We create a public IP to be used to access the source VM. Use [az network public-ip create](#) to create a public IP address resource named **myPublicIPdestinationVM** in **myResourceGroupNAT**.

```
az network public-ip create \
--resource-group myResourceGroupNAT \
--name myPublicIPdestinationVM \
--sku standard \
--location westus
```

### Create an NSG for destination VM

Standard Public IP addresses are 'secure by default', you'll need to create an NSG to allow inbound access for ssh. The Azure NAT service is flow direction aware. This NSG won't be used for outbound once the NAT gateway is configured on the same subnet. Use [az network nsg create](#) to create an NSG resource named **myNSGdestination** in **myResourceGroupNAT**.

```
az network nsg create \
--resource-group myResourceGroupNAT \
--name myNSGdestination \
--location westus
```

### Expose SSH endpoint on destination VM

We create a rule in the NSG for SSH access to the destination vm. Use [az network nsg rule create](#) to create an NSG rule named **ssh**. This rule will be created in the NSG named **myNSGdestination** in the resource group **myResourceGroupNAT**.

```
az network nsg rule create \
--resource-group myResourceGroupNAT \
--nsg-name myNSGdestination \
--priority 100 \
--name ssh \
--description "SSH access" \
--access allow \
--protocol tcp \
--direction inbound \
--destination-port-ranges 22
```

### Expose HTTP endpoint on destination VM

We create a rule in the NSG for HTTP access to the destination vm. Use [az network nsg rule create](#) to create an NSG rule named **http** in the NSG named **myNSGdestination** in **myResourceGroupNAT**.

```
az network nsg rule create \
--resource-group myResourceGroupNAT \
--nsg-name myNSGdestination \
--priority 101 \
--name http \
--description "HTTP access" \
--access allow \
--protocol tcp \
--direction inbound \
--destination-port-ranges 80
```

## Create NIC for destination VM

Create a network interface with [az network nic create](#) and associate with the public IP address **myPublicIPdestinationVM** and the network security group **myNSGdestination**.

```
az network nic create \
--resource-group myResourceGroupNAT \
--name myNicdestination \
--vnet-name myVnetdestination \
--subnet mySubnetdestination \
--public-ip-address myPublicIPdestinationVM \
--network-security-group myNSGdestination \
--location westus
```

## Create a destination VM

Create the virtual machine with [az vm create](#). We generate ssh keys for this VM and store the private key to use later.

```
az vm create \
--resource-group myResourceGroupNAT \
--name myVMdestination \
--nics myNicdestination \
--image UbuntuLTS \
--generate-ssh-keys \
--no-wait \
--location westus
```

While the command will return immediately, it may take a few minutes for the VM to get deployed.

## Prepare a web server and test payload on destination VM

First we need to discover the IP address of the destination VM. To get the public IP address of the destination VM, use [az network public-ip show](#).

```
az network public-ip show \
--resource-group myResourceGroupNAT \
--name myPublicIPdestinationVM \
--query [ipAddress] \
--output tsv
```

### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it in subsequent steps. Indicate this is the destination virtual machine.

## Sign in to destination VM

The SSH credentials should be stored in your Cloud Shell from the previous operation. Open an [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh <ip-address-destination>
```

Copy and paste the following commands once you've signed in.

```
sudo apt-get -y update && \
sudo apt-get -y upgrade && \
sudo apt-get -y dist-upgrade && \
sudo apt-get -y autoremove && \
sudo apt-get -y autoclean && \
sudo apt-get -y install nginx && \
sudo ln -sf /dev/null /var/log/nginx/access.log && \
sudo touch /var/www/html/index.html && \
sudo rm /var/www/html/index.nginx-debian.html && \
sudo dd if=/dev/zero of=/var/www/html/100k bs=1024 count=100
```

These commands will update your virtual machine, install nginx, and create a 100-KBytes file. This file will be retrieved from the source VM using the NAT service.

Close the SSH session with the destination VM.

## Prepare test on source VM

First we need to discover the IP address of the source VM. To get the public IP address of the source VM, use [az network public-ip show](#).

```
az network public-ip show \
--resource-group myResourceGroupNAT \
--name myPublicIPsourceVM \
--query [ipAddress] \
--output tsv
```

### IMPORTANT

Copy the public IP address, and then paste it into a notepad so you can use it in subsequent steps. Indicate this is the source virtual machine.

## Sign in to source VM

Again, the SSH credentials are stored in Cloud Shell. Open a new tab for [Azure Cloud Shell](#) in your browser. Use the IP address retrieved in the previous step to SSH to the virtual machine.

```
ssh <ip-address-source>
```

Copy and paste the following commands to prepare for testing the NAT service.

```
sudo apt-get -y update && \
sudo apt-get -y upgrade && \
sudo apt-get -y dist-upgrade && \
sudo apt-get -y autoremove && \
sudo apt-get autoclean && \
sudo apt-get install -y nload golang && \
echo 'export GOPATH=${HOME}/go' >> .bashrc && \
echo 'export PATH=$PATH:${GOPATH}/bin' >> .bashrc && \
. ~/.bashrc &&
go get -u github.com/rakyll/hey
```

This command will update your virtual machine, install go, install [hey](#) from GitHub, and update your shell environment.

You're now ready to test the NAT service.

## Validate NAT service

While logged into the source VM, you can use **curl** and **hey** to generate requests to the destination IP address.

Use curl to retrieve the 100-KBytes file. Replace **<ip-address-destination>** in the example below with the destination IP address you have previously copied. The **--output** parameter indicates that the retrieved file will be discarded.

```
curl http://<ip-address-destination>/100k --output /dev/null
```

You can also generate a series of requests using **hey**. Again, replace **<ip-address-destination>** with the destination IP address you have previously copied.

```
hey -n 100 -c 10 -t 30 --disable-keepalive http://<ip-address-destination>/100k
```

This command will generate 100 requests, 10 concurrently, with a timeout of 30 seconds. The TCP connection won't be reused. Each request will retrieve 100 Kbytes. At the end of the run, **hey** will report some statistics about how well the NAT service did.

## Clean up resources

When no longer needed, you can use the [az group delete](#) command to remove the resource group and all resources contained within.

```
az group delete --name myResourceGroupNAT
```

## Next steps

In this tutorial, you created a NAT gateway, created a source and destination VM, and then tested the NAT gateway.

Review metrics in Azure Monitor to see your NAT service operating. Diagnose issues such as resource exhaustion of available SNAT ports. Resource exhaustion of SNAT ports is easily addressed by adding additional public IP address resources or public IP prefix resources or both.

- Learn about [Virtual Network NAT](#)
- Learn about [NAT gateway resource](#).

- Quickstart for deploying [NAT gateway resource using Azure CLI](#).
- Quickstart for deploying [NAT gateway resource using Azure PowerShell](#).
- Quickstart for deploying [NAT gateway resource using Azure portal](#).
- [Provide feedback on the Public Preview](#).

# Create, change, or delete a network interface

1/23/2020 • 21 minutes to read • [Edit Online](#)

Learn how to create, change settings for, and delete a network interface. A network interface enables an Azure Virtual Machine to communicate with internet, Azure, and on-premises resources. When creating a virtual machine using the Azure portal, the portal creates one network interface with default settings for you. You may instead choose to create network interfaces with custom settings and add one or more network interfaces to a virtual machine when you create it. You may also want to change default network interface settings for an existing network interface. This article explains how to create a network interface with custom settings, change existing settings, such as network filter (network security group) assignment, subnet assignment, DNS server settings, and IP forwarding, and delete a network interface.

If you need to add, change, or remove IP addresses for a network interface, see [Manage IP addresses](#). If you need to add network interfaces to, or remove network interfaces from virtual machines, see [Add or remove network interfaces](#).

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.28 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

The account you log into, or connect to Azure with, must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Permissions](#).

## Create a network interface

When creating a virtual machine using the Azure portal, the portal creates a network interface with default settings for you. If you'd rather specify all your network interface settings, you can create a network interface with

custom settings and attach the network interface to a virtual machine when creating the virtual machine (using PowerShell or the Azure CLI). You can also create a network interface and add it to an existing virtual machine (using PowerShell or the Azure CLI). To learn how to create a virtual machine with an existing network interface or to add to, or remove network interfaces from existing virtual machines, see [Add or remove network interfaces](#). Before creating a network interface, you must have an existing [virtual network](#) in the same location and subscription you create a network interface in.

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*.

When **network interfaces** appear in the search results, select it.

2. Select **+ Add** under **Network interfaces**.

3. Enter, or select values for the following settings, then select **Create**:

SETTING	REQUIRED?	DETAILS
Name	Yes	The name must be unique within the resource group you select. Over time, you'll likely have several network interfaces in your Azure subscription. For suggestions when creating a naming convention to make managing several network interfaces easier, see <a href="#">Naming conventions</a> . The name cannot be changed after the network interface is created.
Virtual network	Yes	Select the virtual network for the network interface. You can only assign a network interface to a virtual network that exists in the same subscription and location as the network interface. Once a network interface is created, you cannot change the virtual network it is assigned to. The virtual machine you add the network interface to must also exist in the same location and subscription as the network interface.
Subnet	Yes	Select a subnet within the virtual network you selected. You can change the subnet the network interface is assigned to after it's created.

SETTING	REQUIRED?	DETAILS
Private IP address assignment	Yes	<p>In this setting, you're choosing the assignment method for the IPv4 address. Choose from the following assignment methods: <b>Dynamic</b>: When selecting this option, Azure automatically assigns the next available address from the address space of the subnet you selected. <b>Static</b>: When selecting this option, you must manually assign an available IP address from within the address space of the subnet you selected. Static and dynamic addresses do not change until you change them or the network interface is deleted. You can change the assignment method after the network interface is created. The Azure DHCP server assigns this address to the network interface within the operating system of the virtual machine.</p>
Network security group	No	<p>Leave set to <b>None</b>, select an existing <a href="#">network security group</a>, or <a href="#">create a network security group</a>. Network security groups enable you to filter network traffic in and out of a network interface. You can apply zero or one network security group to a network interface. Zero or one network security group can also be applied to the subnet the network interface is assigned to. When a network security group is applied to a network interface and the subnet the network interface is assigned to, sometimes unexpected results occur. To troubleshoot network security groups applied to network interfaces and subnets, see <a href="#">Troubleshoot network security groups</a>.</p>
Subscription	Yes	Select one of your Azure <a href="#">subscriptions</a> . The virtual machine you attach a network interface to and the virtual network you connect it to must exist in the same subscription.

Setting	Required?	Details
Private IP address (IPv6)	No	If you select this checkbox, an IPv6 address is assigned to the network interface, in addition to the IPv4 address assigned to the network interface. See the IPv6 section of this article for important information about use of IPv6 with network interfaces. You cannot select an assignment method for the IPv6 address. If you choose to assign an IPv6 address, it is assigned with the dynamic method.
IPv6 name (only appears when the <b>Private IP address (IPv6)</b> checkbox is checked)	Yes, if the <b>Private IP address (IPv6)</b> checkbox is checked.	This name is assigned to a secondary IP configuration for the network interface. To learn more about IP configurations, see <a href="#">View network interface settings</a> .
Resource group	Yes	Select an existing <a href="#">resource group</a> or create one. A network interface can exist in the same, or different resource group, than the virtual machine you attach it to, or the virtual network you connect it to.
Location	Yes	The virtual machine you attach a network interface to and the virtual network you connect it to must exist in the same <a href="#">location</a> , also referred to as a region.

The portal doesn't provide the option to assign a public IP address to the network interface when you create it, though the portal does create a public IP address and assign it to a network interface when you create a virtual machine using the portal. To learn how to add a public IP address to the network interface after creating it, see [Manage IP addresses](#). If you want to create a network interface with a public IP address, you must use the CLI or PowerShell to create the network interface.

The portal doesn't provide the option to assign the network interface to application security groups when creating a network interface, but the Azure CLI and PowerShell do. You can assign an existing network interface to an application security group using the portal however, as long as the network interface is attached to a virtual machine. To learn how to assign a network interface to an application security group, see [Add to or remove from application security groups](#).

#### NOTE

Azure assigns a MAC address to the network interface only after the network interface is attached to a virtual machine and the virtual machine is started the first time. You cannot specify the MAC address that Azure assigns to the network interface. The MAC address remains assigned to the network interface until the network interface is deleted or the private IP address assigned to the primary IP configuration of the primary network interface is changed. To learn more about IP addresses and IP configurations, see [Manage IP addresses](#)

#### Commands

TOOL	COMMAND
CLI	<code>az network nic create</code>
PowerShell	<code>New-AzNetworkInterface</code>

## View network interface settings

You can view and change most settings for a network interface after it's created. The portal does not display the DNS suffix or application security group membership for the network interface. You can use the PowerShell or Azure CLI [commands](#) to view the DNS suffix and application security group membership.

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appear in the search results, select it.
2. Select the network interface you want to view or change settings for from the list.
3. The following items are listed for the network interface you selected:
  - **Overview:** Provides information about the network interface, such as the IP addresses assigned to it, the virtual network/subnet the network interface is assigned to, and the virtual machine the network interface is attached to (if it's attached to one). The following picture shows the overview settings for a network interface named **mywebserver256**:

Setting	Value
Resource group ( <a href="#">change</a> )	Private IP address MyRG
Location	10.0.0.4
Subscription name ( <a href="#">change</a> )	Virtual network/subnet <a href="#">MyVNet/Front-end</a>
Subscription ID	Public IP address <a href="#">13.64.66.250 (MyWebServer-ip)</a>
Tags	Network security group <a href="#">MyWebServer-nsg</a>
SETTINGS	Attached to <a href="#">MyWebServer</a>

You can move a network interface to a different resource group or subscription by selecting ([change](#)) next to the **Resource group** or **Subscription name**. If you move the network interface, you must move all resources related to the network interface with it. If the network interface is attached to a virtual machine, for example, you must also move the virtual machine, and other virtual machine-related resources. To move a network interface, see [Move resource to a new resource group or subscription](#). The article lists prerequisites, and how to move resources using the Azure portal, PowerShell, and the Azure CLI.

- **IP configurations:** Public and private IPv4 and IPv6 addresses assigned to IP configurations are listed here. If an IPv6 address is assigned to an IP configuration, the address is not displayed. To learn more about IP configurations and how to add and remove IP addresses, see [Configure IP addresses for an Azure network interface](#). IP forwarding and subnet assignment are also configured in this section. To learn more about these settings, see [Enable or disable IP forwarding](#) and [Change subnet assignment](#).
- **DNS servers:** You can specify which DNS server a network interface is assigned by the Azure DHCP servers. The network interface can inherit the setting from the virtual network the network interface is assigned to, or have a custom setting that overrides the setting for the virtual network it's assigned to. To modify what's displayed, see [Change DNS servers](#).
- **Network security group (NSG):** Displays which NSG is associated to the network interface (if

any). An NSG contains inbound and outbound rules to filter network traffic for the network interface. If an NSG is associated to the network interface, the name of the associated NSG is displayed. To modify what's displayed, see [Associate or dissociate a network security group](#).

- **Properties:** Displays key settings about the network interface, including its MAC address (blank if the network interface isn't attached to a virtual machine), and the subscription it exists in.
- **Effective security rules:** Security rules are listed if the network interface is attached to a running virtual machine, and an NSG is associated to the network interface, the subnet it's assigned to, or both. To learn more about what's displayed, see [View effective security rules](#). To learn more about NSGs, see [Network security groups](#).
- **Effective routes:** Routes are listed if the network interface is attached to a running virtual machine. The routes are a combination of the Azure default routes, any user-defined routes, and any BGP routes that may exist for the subnet the network interface is assigned to. To learn more about what's displayed, see [View effective routes](#). To learn more about Azure default routes and user-defined routes, see [Routing overview](#). Common Azure Resource Manager settings: To learn more about common Azure Resource Manager settings, see [Activity log](#), [Access control \(IAM\)](#), [Tags](#), [Locks](#), and [Automation script](#).

## Commands

If an IPv6 address is assigned to a network interface, the PowerShell output returns the fact that the address is assigned, but it doesn't return the assigned address. Similarly, the CLI returns the fact that the address is assigned, but returns *null* in its output for the address.

TOOL	COMMAND
CLI	<code>az network nic list</code> to view network interfaces in the subscription; <code>az network nic show</code> to view settings for a network interface
PowerShell	<code>Get-AzNetworkInterface</code> to view network interfaces in the subscription or view settings for a network interface

## Change DNS servers

The DNS server is assigned by the Azure DHCP server to the network interface within the virtual machine operating system. The DNS server assigned is whatever the DNS server setting is for a network interface. To learn more about name resolution settings for a network interface, see [Name resolution for virtual machines](#). The network interface can inherit the settings from the virtual network, or use its own unique settings that override the setting for the virtual network.

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appear in the search results, select it.
2. Select the network interface that you want to change a DNS server for from the list.
3. Select **DNS servers** under **SETTINGS**.
4. Select either:
  - **Inherit from virtual network:** Choose this option to inherit the DNS server setting defined for the virtual network the network interface is assigned to. At the virtual network level, either a custom DNS server or the Azure-provided DNS server is defined. The Azure-provided DNS server can resolve hostnames for resources assigned to the same virtual network. FQDN must be used to resolve for resources assigned to different virtual networks.
  - **Custom:** You can configure your own DNS server to resolve names across multiple virtual networks. Enter the IP address of the server you want to use as a DNS server. The DNS server address you specify

is assigned only to this network interface and overrides any DNS setting for the virtual network the network interface is assigned to.

**NOTE**

If the VM uses a NIC that's part of an availability set, all the DNS servers that are specified for each of the VMs from all NICs that are part of the availability set will be inherited.

5. Select **Save**.

## Commands

TOOL	COMMAND
CLI	<a href="#">az network nic update</a>
PowerShell	<a href="#">Set-AzNetworkInterface</a>

## Enable or disable IP forwarding

IP forwarding enables the virtual machine a network interface is attached to:

- Receive network traffic not destined for one of the IP addresses assigned to any of the IP configurations assigned to the network interface.
- Send network traffic with a different source IP address than the one assigned to one of a network interface's IP configurations.

The setting must be enabled for every network interface that is attached to the virtual machine that receives traffic that the virtual machine needs to forward. A virtual machine can forward traffic whether it has multiple network interfaces or a single network interface attached to it. While IP forwarding is an Azure setting, the virtual machine must also run an application able to forward the traffic, such as firewall, WAN optimization, and load balancing applications. When a virtual machine is running network applications, the virtual machine is often referred to as a network virtual appliance. You can view a list of ready to deploy network virtual appliances in the [Azure Marketplace](#). IP forwarding is typically used with user-defined routes. To learn more about user-defined routes, see [User-defined routes](#).

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appear in the search results, select it.
2. Select the network interface that you want to enable or disable IP forwarding for.
3. Select **IP configurations** in the **SETTINGS** section.
4. Select **Enabled** or **Disabled** (default setting) to change the setting.
5. Select **Save**.

## Commands

TOOL	COMMAND
CLI	<a href="#">az network nic update</a>
PowerShell	<a href="#">Set-AzNetworkInterface</a>

## Change subnet assignment

You can change the subnet, but not the virtual network, that a network interface is assigned to.

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appear in the search results, select it.
2. Select the network interface that you want to change subnet assignment for.
3. Select **IP configurations** under **SETTINGS**. If any private IP addresses for any IP configurations listed have **(Static)** next to them, you must change the IP address assignment method to dynamic by completing the steps that follow. All private IP addresses must be assigned with the dynamic assignment method to change the subnet assignment for the network interface. If the addresses are assigned with the dynamic method, continue to step five. If any IPv4 addresses are assigned with the static assignment method, complete the following steps to change the assignment method to dynamic:
  - Select the IP configuration you want to change the IPv4 address assignment method for from the list of IP configurations.
  - Select **Dynamic** for the private IP address **Assignment** method. You cannot assign an IPv6 address with the static assignment method.
  - Select **Save**.
4. Select the subnet you want to move the network interface to from the **Subnet** drop-down list.
5. Select **Save**. New dynamic addresses are assigned from the subnet address range for the new subnet. After assigning the network interface to a new subnet, you can assign a static IPv4 address from the new subnet address range if you choose. To learn more about adding, changing, and removing IP addresses for a network interface, see [Manage IP addresses](#).

## Commands

TOOL	COMMAND
CLI	<code>az network nic ip-config update</code>
PowerShell	<code>Set-AzNetworkInterfaceIpConfig</code>

## Add to or remove from application security groups

You can only add a network interface to, or remove a network interface from an application security group using the portal if the network interface is attached to a virtual machine. You can use PowerShell or the Azure CLI to add a network interface to, or remove a network interface from an application security group, whether the network interface is attached to a virtual machine or not. Learn more about [Application security groups](#) and how to [create an application security group](#).

1. In the *Search resources, services, and docs* box at the top of the portal, begin typing the name of a virtual machine that has a network interface that you want to add to, or remove from, an application security group. When the name of your VM appears in the search results, select it.
2. Under **SETTINGS**, select **Networking**. Select **Application Security Groups** then **Configure the application security group**. Select the application security groups that you want to add the network interface to, or unselect the application security groups that you want to remove the network interface from, and then select **Save**. Only network interfaces that exist in the same virtual network can be added to the same application security group. The application security group must exist in the same location as the network interface.

## Commands

TOOL	COMMAND
CLI	<a href="#">az network nic update</a>
PowerShell	<a href="#">Set-AzNetworkInterface</a>

## Associate or dissociate a network security group

1. In the search box at the top of the portal, enter *network interfaces* in the search box. When **network interfaces** appear in the search results, select it.
2. Select the network interface in the list that you want to associate a network security group to, or dissociate a network security group from.
3. Select **Network security group** under **SETTINGS**.
4. Select **Edit**.
5. Select **Network security group** and then select the network security group you want to associate to the network interface, or select **None**, to dissociate a network security group.
6. Select **Save**.

### Commands

- Azure CLI: [az network nic update](#)
- PowerShell: [Set-AzNetworkInterface](#)

## Delete a network interface

You can delete a network interface as long as it's not attached to a virtual machine. If a network interface is attached to a virtual machine, you must first place the virtual machine in the stopped (deallocated) state, then detach the network interface from the virtual machine. To detach a network interface from a virtual machine, complete the steps in [Detach a network interface from a virtual machine](#). You cannot detach a network interface from a virtual machine if it's the only network interface attached to the virtual machine however. A virtual machine must always have at least one network interface attached to it. Deleting a virtual machine detaches all network interfaces attached to it, but does not delete the network interfaces.

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appear in the search results, select it.
2. Select the network interface in the list that you want to delete.
3. Under **Overview** Select **Delete**.
4. Select **Yes** to confirm deletion of the network interface.

When you delete a network interface, any MAC or IP addresses assigned to it are released.

### Commands

TOOL	COMMAND
CLI	<a href="#">az network nic delete</a>
PowerShell	<a href="#">Remove-AzNetworkInterface</a>

## Resolve connectivity issues

If you are unable to communicate to or from a virtual machine, network security group security rules or routes effective for a network interface, may be causing the problem. You have the following options to help resolve the

issue:

## View effective security rules

The effective security rules for each network interface attached to a virtual machine are a combination of the rules you've created in a network security group and [default security rules](#). Understanding the effective security rules for a network interface may help you determine why you're unable to communicate to or from a virtual machine. You can view the effective rules for any network interface that is attached to a running virtual machine.

1. In the search box at the top of the portal, enter the name of a virtual machine you want to view effective security rules for. If you don't know the name of a virtual machine, enter *virtual machines* in the search box. When **Virtual machines** appear in the search results, select it, and then select a virtual machine from the list.
2. Select **Networking** under **SETTINGS**.
3. Select the name of a network interface.
4. Select **Effective security rules** under **SUPPORT + TROUBLESHOOTING**.
5. Review the list of effective security rules to determine if the correct rules exist for your required inbound and outbound communication. Learn more about what you see in the list in [Network security group overview](#).

The IP flow verify feature of Azure Network Watcher can also help you determine if security rules are preventing communication between a virtual machine and an endpoint. To learn more, see [IP flow verify](#).

## Commands

- Azure CLI: [az network nic list-effective-nsg](#)
- PowerShell: [Get-AzEffectiveNetworkSecurityGroup](#)

## View effective routes

The effective routes for the network interfaces attached to a virtual machine are a combination of default routes, any routes you've created, and any routes propagated from on-premises networks via BGP through an Azure virtual network gateway. Understanding the effective routes for a network interface may help you determine why you're unable to communicate to or from a virtual machine. You can view the effective routes for any network interface that is attached to a running virtual machine.

1. In the search box at the top of the portal, enter the name of a virtual machine you want to view effective security rules for. If you don't know the name of a virtual machine, enter *virtual machines* in the search box. When **Virtual machines** appear in the search results, select it, and then select a virtual machine from the list.
2. Select **Networking** under **SETTINGS**.
3. Select the name of a network interface.
4. Select **Effective routes** under **SUPPORT + TROUBLESHOOTING**.
5. Review the list of effective routes to determine if the correct routes exist for your required inbound and outbound communication. Learn more about what you see in the list in [Routing overview](#).

The next hop feature of Azure Network Watcher can also help you determine if routes are preventing communication between a virtual machine and an endpoint. To learn more, see [Next hop](#).

## Commands

- Azure CLI: [az network nic show-effective-route-table](#)
- PowerShell: [Get-AzEffectiveRouteTable](#)

## Permissions

To perform tasks on network interfaces, your account must be assigned to the [network contributor](#) role or to a [custom](#) role that is assigned the appropriate permissions listed in the following table:

ACTION	NAME
Microsoft.Network/networkInterfaces/read	Get network interface
Microsoft.Network/networkInterfaces/write	Create or update network interface
Microsoft.Network/networkInterfaces/join/action	Attach a network interface to a virtual machine
Microsoft.Network/networkInterfaces/delete	Delete network interface
Microsoft.Network/networkInterfaces/joinViaPrivateIp/action	Join a resource to a network interface via a servi...
Microsoft.Network/networkInterfaces/effectiveRouteTable/acti on	Get network interface effective route table
Microsoft.Network/networkInterfaces/effectiveNetworkSecurit yGroups/action	Get network interface effective security groups
Microsoft.Network/networkInterfaces/loadBalancers/read	Get network interface load balancers
Microsoft.Network/networkInterfaces/serviceAssociations/read	Get service association
Microsoft.Network/networkInterfaces/serviceAssociations/writ e	Create or update a service association
Microsoft.Network/networkInterfaces/serviceAssociations/delete	Delete service association
Microsoft.Network/networkInterfaces/serviceAssociations/vali date/action	Validate service association
Microsoft.Network/networkInterfaces/ipconfigurations/read	Get network interface IP configuration

## Next steps

- Create a VM with multiple NICs using the [Azure CLI](#) or [PowerShell](#)
- Create a single NIC VM with multiple IPv4 addresses using the [Azure CLI](#) or [PowerShell](#)
- Create a single NIC VM with a private IPv6 address (behind an Azure Load Balancer) using the [Azure CLI](#), [PowerShell](#), or [Azure Resource Manager template](#)
- Create a network interface using [PowerShell](#) or [Azure CLI](#) sample scripts, or using Azure [Resource Manager template](#)
- Create and apply [Azure policy](#) for virtual networks

# Create, change, or delete a virtual network

1/14/2020 • 13 minutes to read • [Edit Online](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Learn how to create and delete a virtual network and change settings, like DNS servers and IP address spaces, for an existing virtual network. If you're new to virtual networks, you can learn more about them in the [Virtual network overview](#) or by completing a [tutorial](#). A virtual network contains subnets. To learn how to create, change, and delete subnets, see [Manage subnets](#).

## Before you begin

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.31 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.
- The account you log into, or connect to Azure with, must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Permissions](#).

## Create a virtual network

1. Select + **Create a resource > Networking > Virtual network**.
2. Enter or select values for the following settings, then select **Create**:

- **Name:** The name must be unique in the [resource group](#) that you select to create the virtual network in. You cannot change the name after the virtual network is created. You can create multiple virtual networks over time. For naming suggestions, see [Naming conventions](#). Following a naming convention can help make it easier to manage multiple virtual networks.
- **Address space:** The address space for a virtual network is composed of one or more non-overlapping address ranges that are specified in CIDR notation. The address range you define can be public or private (RFC 1918). Whether you define the address range as public or private, the

address range is reachable only from within the virtual network, from interconnected virtual networks, and from any on-premises networks that you have connected to the virtual network. You cannot add the following address ranges:

- 224.0.0.0/4 (Multicast)
- 255.255.255.255/32 (Broadcast)
- 127.0.0.0/8 (Loopback)
- 169.254.0.0/16 (Link-local)
- 168.63.129.16/32 (Internal DNS, DHCP, and Azure Load Balancer [health probe](#))

Although you can define only one address range when you create the virtual network in the portal, you can add more address ranges to the address space after the virtual network is created. To learn how to add an address range to an existing virtual network, see [Add or remove an address range](#).

#### WARNING

If a virtual network has address ranges that overlap with another virtual network or on-premises network, the two networks cannot be connected. Before you define an address range, consider whether you might want to connect the virtual network to other virtual networks or on-premises networks in the future.

- **Subnet name:** The subnet name must be unique within the virtual network. You cannot change the subnet name after the subnet is created. The portal requires that you define one subnet when you create a virtual network, even though a virtual network isn't required to have any subnets. In the portal, you can define only one subnet when you create a virtual network. You can add more subnets to the virtual network later, after the virtual network is created. To add a subnet to a virtual network, see [Manage subnets](#). You can create a virtual network that has multiple subnets by using Azure CLI or PowerShell.

#### TIP

Sometimes, administrators create different subnets to filter or control traffic routing between the subnets. Before you define subnets, consider how you might want to filter and route traffic between your subnets. To learn more about filtering traffic between subnets, see [Network security groups](#). Azure automatically routes traffic between subnets, but you can override Azure default routes. To learn more about Azure's default subnet traffic routing, see [Routing overview](#).

- **Subnet address range:** The range must be within the address space you entered for the virtual network. The smallest range you can specify is /29, which provides eight IP addresses for the subnet. Azure reserves the first and last address in each subnet for protocol conformance. Three additional addresses are reserved for Azure service usage. As a result, a virtual network with a subnet address range of /29 has only three usable IP addresses. If you plan to connect a virtual network to a VPN gateway, you must create a gateway subnet. Learn more about [specific address range considerations for gateway subnets](#). You can change the address range after the subnet is created, under specific conditions. To learn how to change a subnet address range, see [Manage subnets](#).
- **Subscription:** Select a [subscription](#). You cannot use the same virtual network in more than one Azure subscription. However, you can connect a virtual network in one subscription to virtual networks in other subscriptions with [virtual network peering](#). Any Azure resource that you connect to the virtual network must be in the same subscription as the virtual network.
- **Resource group:** Select an existing [resource group](#) or create a new one. An Azure resource that you connect to the virtual network can be in the same resource group as the virtual network or in a different resource group.

- **Location:** Select an Azure [location](#), also known as a region. A virtual network can be in only one Azure location. However, you can connect a virtual network in one location to a virtual network in another location by using a VPN gateway. Any Azure resource that you connect to the virtual network must be in the same location as the virtual network.

## Commands

- Azure CLI: [az network vnet create](#)
- PowerShell: [New-AzVirtualNetwork](#)

## View virtual networks and settings

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it.
2. From the list of virtual networks, select the virtual network that you want to view settings for.
3. The following settings are listed for the virtual network you selected:
  - **Overview:** Provides information about the virtual network, including address space and DNS servers. The following screenshot shows the overview settings for a virtual network named **MyVNet**:

DEVICE	TYPE	IP ADDRESS	SUBNET
myvm2951	Network interface	10.0.0.4	MySubnet1
myvml607	Network interface	10.0.0.5	MySubnet1
MyLB1	Load balancer	10.0.0.6	MySubnet1
myvm3276	Network interface	10.0.1.4	MySubnet2

You can move a virtual network to a different subscription or resource group by selecting **Change** next to **Resource group** or **Subscription name**. To learn how to move a virtual network, see [Move resources to a different resource group or subscription](#). The article lists prerequisites, and how to move resources by using the Azure portal, PowerShell, and Azure CLI. All resources that are connected to the virtual network must move with the virtual network.

- **Address space:** The address spaces that are assigned to the virtual network are listed. To learn how to add and remove an address range to the address space, complete the steps in [Add or remove an address range](#).

- **Connected devices:** Any resources that are connected to the virtual network are listed. In the preceding screenshot, three network interfaces and one load balancer are connected to the virtual network. Any new resources that you create and connect to the virtual network are listed. If you delete a resource that was connected to the virtual network, it no longer appear in the list.
- **Subnets:** A list of subnets that exist within the virtual network is shown. To learn how to add and remove a subnet, see [Manage subnets](#).
- **DNS servers:** You can specify whether the Azure internal DNS server or a custom DNS server provides name resolution for devices that are connected to the virtual network. When you create a virtual network by using the Azure portal, Azure's DNS servers are used for name resolution within a virtual network, by default. To modify the DNS servers, complete the steps in [Change DNS servers](#) in this article.
- **Peerings:** If there are existing peerings in the subscription, they are listed here. You can view settings for existing peerings, or create, change, or delete peerings. To learn more about peerings, see [Virtual network peering](#).
- **Properties:** Displays settings about the virtual network, including the virtual network's resource ID and the Azure subscription it is in.
- **Diagram:** The diagram provides a visual representation of all devices that are connected to the virtual network. The diagram has some key information about the devices. To manage a device in this view, in the diagram, select the device.
- **Common Azure settings:** To learn more about common Azure settings, see the following information:
  - [Activity log](#)
  - [Access control \(IAM\)](#)
  - [Tags](#)
  - [Locks](#)
  - [Automation script](#)

## Commands

- Azure CLI: [az network vnet show](#)
- PowerShell: [Get-AzVirtualNetwork](#)

## Add or remove an address range

You can add and remove address ranges for a virtual network. An address range must be specified in CIDR notation, and cannot overlap with other address ranges within the same virtual network. The address ranges you define can be public or private (RFC 1918). Whether you define the address range as public or private, the address range is reachable only from within the virtual network, from interconnected virtual networks, and from any on-premises networks that you have connected to the virtual network.

You can decrease the address range for a virtual network as long as it still includes the ranges of any associated subnets. Additionally, you can extend the address range, for example, changing a /16 to /8.

You cannot add the following address ranges:

- 224.0.0.0/4 (Multicast)
- 255.255.255.255/32 (Broadcast)
- 127.0.0.0/8 (Loopback)
- 169.254.0.0/16 (Link-local)

- 168.63.129.16/32 (Internal DNS, DHCP, and Azure Load Balancer [health probe](#))

To add or remove an address range:

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it.
2. From the list of virtual networks, select the virtual network for which you want to add or remove an address range.
3. Select **Address space**, under **SETTINGS**.
4. Complete one of the following options:
  - **Add an address range:** Enter the new address range. The address range cannot overlap with an existing address range that is defined for the virtual network.
  - **Remove an address range:** On the right of the address range you want to remove, select ..., then select **Remove**. If a subnet exists in the address range, you cannot remove the address range. To remove an address range, you must first delete any subnets (and any resources in the subnets) that exist in the address range.
5. Select **Save**.

## Commands

- Azure CLI: [az network vnet update](#)
- PowerShell: [Set-AzVirtualNetwork](#)

## Change DNS servers

All VMs that are connected to the virtual network register with the DNS servers that you specify for the virtual network. They also use the specified DNS server for name resolution. Each network interface (NIC) in a VM can have its own DNS server settings. If a NIC has its own DNS server settings, they override the DNS server settings for the virtual network. To learn more about NIC DNS settings, see [Network interface tasks and settings](#). To learn more about name resolution for VMs and role instances in Azure Cloud Services, see [Name resolution for VMs and role instances](#). To add, change, or remove a DNS server:

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it.
2. From the list of virtual networks, select the virtual network for which you want to change DNS servers for.
3. Select **DNS servers**, under **SETTINGS**.
4. Select one of the following options:
  - **Default (Azure-provided):** All resource names and private IP addresses are automatically registered to the Azure DNS servers. You can resolve names between any resources that are connected to the same virtual network. You cannot use this option to resolve names across virtual networks. To resolve names across virtual networks, you must use a custom DNS server.
  - **Custom:** You can add one or more servers, up to the Azure limit for a virtual network. To learn more about DNS server limits, see [Azure limits](#). You have the following options:
    - **Add an address:** Adds the server to your virtual network DNS servers list. This option also registers the DNS server with Azure. If you've already registered a DNS server with Azure, you can select that DNS server in the list.
    - **Remove an address:** Next to the server that you want to remove, select ..., then **Remove**. Deleting the server removes the server only from this virtual network list. The DNS server remains registered in Azure for your other virtual networks to use.
    - **Reorder DNS server addresses:** It's important to verify that you list your DNS servers in the correct order for your environment. DNS server lists are used in the order that they are specified. They do not work as a round-robin setup. If the first DNS server in the list can be reached, the client uses that DNS

server, regardless of whether the DNS server is functioning properly. Remove all the DNS servers that are listed, and then add them back in the order that you want.

- **Change an address:** Highlight the DNS server in the list, and then enter the new address.
5. Select **Save**.
  6. Restart the VMs that are connected to the virtual network, so they are assigned the new DNS server settings. VMs continue to use their current DNS settings until they are restarted.

## Commands

- Azure CLI: [az network vnet update](#)
- PowerShell: [Set-AzVirtualNetwork](#)

## Delete a virtual network

You can delete a virtual network only if there are no resources connected to it. If there are resources connected to any subnet within the virtual network, you must first delete the resources that are connected to all subnets within the virtual network. The steps you take to delete a resource vary depending on the resource. To learn how to delete resources that are connected to subnets, read the documentation for each resource type you want to delete. To delete a virtual network:

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it.
2. From the list of virtual networks, select the virtual network you want to delete.
3. Confirm that there are no devices connected to the virtual network by selecting **Connected devices**, under **SETTINGS**. If there are connected devices, you must delete them before you can delete the virtual network. If there are no connected devices, select **Overview**.
4. Select **Delete**.
5. To confirm the deletion of the virtual network, select **Yes**.

## Commands

- Azure CLI: [azurerm network vnet delete](#)
- PowerShell: [Remove-AzVirtualNetwork](#)

## Permissions

To perform tasks on virtual networks, your account must be assigned to the [network contributor](#) role or to a [custom](#) role that is assigned the appropriate actions listed in the following table:

ACTION	NAME
Microsoft.Network/virtualNetworks/read	Read a virtual Network
Microsoft.Network/virtualNetworks/write	Create or update a virtual network
Microsoft.Network/virtualNetworks/delete	Delete a virtual network

## Next steps

- Create a virtual network using [PowerShell](#) or [Azure CLI](#) sample scripts, or using Azure [Resource Manager templates](#)
- Create and apply [Azure policy](#) for virtual networks

# Add, change, or delete a virtual network subnet

2/6/2020 • 8 minutes to read • [Edit Online](#)

Learn how to add, change, or delete a virtual network subnet. All Azure resources deployed into a virtual network are deployed into a subnet within a virtual network. If you're new to virtual networks, you can learn more about them in the [Virtual network overview](#) or by completing a [tutorial](#). To create, change, or delete a virtual network, see [Manage a virtual network](#).

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.31 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

The account you log into, or connect to Azure with, must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Permissions](#).

## Add a subnet

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it.
2. From the list of virtual networks, select the virtual network you want to add a subnet to.
3. Under **SETTINGS**, select **Subnets**.
4. Select **+Subnet**.
5. Enter values for the following parameters:
  - **Name:** The name must be unique within the virtual network. For maximum compatibility with

other Azure services, we recommend using a letter as the first character of the name. For example, Azure Application Gateway won't deploy into a subnet that has a name that starts with a number.

- **Address range:** The range must be unique within the address space for the virtual network. The range cannot overlap with other subnet address ranges within the virtual network. The address space must be specified by using Classless Inter-Domain Routing (CIDR) notation. For example, in a virtual network with address space 10.0.0.0/16, you might define a subnet address space of 10.0.0.0/24. The smallest range you can specify is /29, which provides eight IP addresses for the subnet. Azure reserves the first and last address in each subnet for protocol conformance. Three additional addresses are reserved for Azure service usage. As a result, defining a subnet with a /29 address range results in three usable IP addresses in the subnet. If you plan to connect a virtual network to a VPN gateway, you must create a gateway subnet. Learn more about [specific address range considerations for gateway subnets](#). You can change the address range after the subnet is added, under specific conditions. To learn how to change a subnet address range, see [Change subnet settings](#).
- **Network security group:** You can associate zero, or one existing network security group to a subnet to filter inbound and outbound network traffic for the subnet. The network security group must exist in the same subscription and location as the virtual network. Learn more about [network security groups](#) and [how to create a network security group](#).
- **Route table:** You can associate zero or one existing route table to a subnet to control network traffic routing to other networks. The route table must exist in the same subscription and location as the virtual network. Learn more about [Azure routing](#) and [how to create a route table](#)
- **Service endpoints:** A subnet can have zero or multiple service endpoints enabled for it. To enable a service endpoint for a service, select the service or services that you want to enable service endpoints for from the **Services** list. The location is configured automatically for an endpoint. By default, service endpoints are configured for the virtual network's region. For Azure Storage, to support regional failover scenarios, endpoints are automatically configured to [Azure paired regions](#).

To remove a service endpoint, unselect the service you want to remove the service endpoint for. To learn more about service endpoints, and the services they can be enabled for, see [Virtual network service endpoints overview](#). Once you enable a service endpoint for a service, you must also enable network access for the subnet for a resource created with the service. For example, if you enable the service endpoint for *Microsoft.Storage*, you must also enable network access to all Azure Storage accounts you want to grant network access to. For details about how to enable network access to subnets that a service endpoint is enabled for, see the documentation for the individual service you enabled the service endpoint for.

To validate that a service endpoint is enabled for a subnet, view the [effective routes](#) for any network interface in the subnet. When an endpoint is configured, you see a *default* route with the address prefixes of the service, and a nextHopType of **VirtualNetworkServiceEndpoint**. To learn more about routing, see [Routing overview](#).

- **Subnet delegation:** A subnet can have zero to multiple delegations enabled for it. Subnet delegation gives explicit permissions to the service to create service-specific resources in the subnet using a unique identifier when deploying the service. To delegate for a service, select the service you want to delegate to from the **Services** list.

6. To add the subnet to the virtual network that you selected, select **OK**.

## Commands

- Azure CLI: [az network vnet subnet create](#)
- PowerShell: [Add-AzVirtualNetworkSubnetConfig](#)

## Change subnet settings

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it.
2. From the list of virtual networks, select the virtual network that contains the subnet you want to change settings for.
3. Under **SETTINGS**, select **Subnets**.
4. In the list of subnets, select the subnet you want to change settings for. You can change the following settings:
  - **Address range:** If no resources are deployed within the subnet, you can change the address range. If any resources exist in the subnet, you must either move the resources to another subnet, or delete them from the subnet first. The steps you take to move or delete a resource vary depending on the resource. To learn how to move or delete resources that are in subnets, read the documentation for each resource type that you want to move or delete. See the constraints for **Address range** in step 5 of [Add a subnet](#).
  - **Users:** You can control access to the subnet by using built-in roles or your own custom roles. To learn more about assigning roles and users to access the subnet, see [Use role assignment to manage access to your Azure resources](#).
  - **Network security group** and **Route table:** See step 5 of [Add a subnet](#).
  - **Service endpoints:** See service endpoints in step 5 of [Add a subnet](#). When enabling a service endpoint for an existing subnet, ensure that no critical tasks are running on any resource in the subnet. Service endpoints switch routes on every network interface in the subnet from using the default route with the *0.0.0.0/0* address prefix and next hop type of *Internet*, to using a new route with the address prefixes of the service, and a next hop type of *VirtualNetworkServiceEndpoint*. During the switch, any open TCP connections may be terminated. The service endpoint is not enabled until traffic flows to the service for all network interfaces are updated with the new route. To learn more about routing, see [Routing overview](#).
  - **Subnet delegation:** See service endpoints in step 5 of [Add a subnet](#). Subnet delegation can be modified to zero or multiple delegations enabled for it. If a resource for a service is already deployed in the subnet, subnet delegation cannot be added or removed until the all resources for the service are removed. To delegate for a different service, select the service you want to delegate to from the **Services** list.
5. Select **Save**.

### Commands

- Azure CLI: [az network vnet subnet update](#)
- PowerShell: [Set-AzVirtualNetworkSubnetConfig](#)

## Delete a subnet

You can delete a subnet only if there are no resources in the subnet. If there are resources in the subnet, you must delete the resources that are in the subnet before you can delete the subnet. The steps you take to delete a resource vary depending on the resource. To learn how to delete resources that are in subnets, read the documentation for each resource type that you want to delete.

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it.
2. From the list of virtual networks, select the virtual network that contains the subnet you want to delete.
3. Under **SETTINGS**, select **Subnets**.
4. In the list of subnets, select ..., on the right, for the subnet you want to delete

5. Select **Delete**, and then select **Yes**.

## Commands

- Azure CLI: [az network vnet subnet delete](#)
- PowerShell: [Remove-AzVirtualNetworkSubnetConfig](#)

## Permissions

To perform tasks on subnets, your account must be assigned to the [network contributor](#) role or to a [custom](#) role that is assigned the appropriate actions listed in the following table:

ACTION	NAME
Microsoft.Network/virtualNetworks/subnets/read	Read a virtual network subnet
Microsoft.Network/virtualNetworks/subnets/write	Create or update a virtual network subnet
Microsoft.Network/virtualNetworks/subnets/delete	Delete a virtual network subnet
Microsoft.Network/virtualNetworks/subnets/join/action	Join a virtual network
Microsoft.Network/virtualNetworks/subnets/joinViaServiceEndpoint/action	Enable a service endpoint for a subnet
Microsoft.Network/virtualNetworks/subnets/virtualMachines/read	Get the virtual machines in a subnet

## Next steps

- Create a virtual network and subnets using [PowerShell](#) or [Azure CLI](#) sample scripts, or using [Azure Resource Manager templates](#)
- Create and apply [Azure policy](#) for virtual networks

# Add or remove a subnet delegation

2/13/2020 • 6 minutes to read • [Edit Online](#)

Subnet delegation gives explicit permissions to the service to create service-specific resources in the subnet using a unique identifier when deploying the service. This article describes how to add or remove a delegated subnet for an Azure service.

## Portal

### Sign in to Azure

Sign in to the Azure portal at <https://portal.azure.com>.

### Create the virtual network

In this section, you create a virtual network and the subnet that you'll later delegate to an Azure service.

1. On the upper-left side of the screen, select **Create a resource** > **Networking** > **Virtual network**.
2. In **Create virtual network**, enter or select this information:

SETTING	VALUE
Name	Enter <i>MyVirtualNetwork</i> .
Address space	Enter <i>10.0.0.0/16</i> .
Subscription	Select your subscription.
Resource group	Select <b>Create new</b> , enter <i>myResourceGroup</i> , then select <b>OK</b> .
Location	Select <b>EastUS</b> .
Subnet - Name	Enter <i>mySubnet</i> .
Subnet - Address range	Enter <i>10.0.0.0/24</i> .

3. Leave the rest as default, and then select **Create**.

### Permissions

If you didn't create the subnet you would like to delegate to an Azure service, you need the following permission:

`Microsoft.Network/virtualNetworks/subnets/write`

The built-in [Network Contributor](#) role also contains the necessary permissions.

### Delegate a subnet to an Azure service

In this section, you delegate the subnet that you created in the preceding section to an Azure service.

1. In the portal's search bar, enter *myVirtualNetwork*. When **myVirtualNetwork** appears in the search results, select it.
2. In the search results, select *myVirtualNetwork*.

3. Select **Subnets**, under **SETTINGS**, and then select **mySubnet**.
4. On the *mySubnet* page, for the **Subnet delegation** list, select from the services listed under **Delegate subnet to a service** (for example, **Microsoft.DBforPostgreSQL/serversv2**).

#### Remove subnet delegation from an Azure service

1. In the portal's search bar, enter *myVirtualNetwork*. When **myVirtualNetwork** appears in the search results, select it.
2. In the search results, select *myVirtualNetwork*.
3. Select **Subnets**, under **SETTINGS**, and then select **mySubnet**.
4. In *mySubnet* page, for the **Subnet delegation** list, select **None** from the services listed under **Delegate subnet to a service**.

## Azure CLI

### Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you decide to install and use Azure CLI locally instead, this article requires you to use Azure CLI version 2.0.28 or later. To find your installed version, run `az --version`. See [Install Azure CLI](#) for install or upgrade info.

#### Create a resource group

Create a resource group with [az group create](#). An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named **myResourceGroup** in the **eastus** location:

```
az group create \
--name myResourceGroup \
--location eastus
```

## Create a virtual network

Create a virtual network named **myVnet** with a subnet named **mySubnet** in the **myResourceGroup** using [az network vnet create](#).

```
az network vnet create \
--resource-group myResourceGroup \
--location eastus \
--name myVnet \
--address-prefix 10.0.0.0/16 \
--subnet-name mySubnet \
--subnet-prefix 10.0.0.0/24
```

## Permissions

If you didn't create the subnet you would like to delegate to an Azure service, you need the following permission:

```
Microsoft.Network/virtualNetworks/subnets/write
```

The built-in [Network Contributor](#) role also contains the necessary permissions.

## Delegate a subnet to an Azure service

In this section, you delegate the subnet that you created in the preceding section to an Azure service.

Use [az network vnet subnet update](#) to update the subnet named **mySubnet** with a delegation to an Azure service.

In this example **Microsoft.DBforPostgreSQL/serversv2** is used for the example delegation:

```
az network vnet subnet update \
--resource-group myResourceGroup \
--name mySubnet \
--vnet-name myVnet \
--delegations Microsoft.DBforPostgreSQL/serversv2
```

To verify the delegation was applied, use [az network vnet subnet show](#). Verify the service is delegated to the subnet under the property **serviceName**:

```
az network vnet subnet show \
--resource-group myResourceGroup \
--name mySubnet \
--vnet-name myVnet \
--query delegations
```

```
[  
  [  
    {  
      "actions": [  
        "Microsoft.Network/virtualNetworks/subnets/join/action"  
      ],  
      "etag": "W/\"8a8bf16a-38cf-409f-9434-fe3b5ab9ae54\"",  
      "id": "/subscriptions/3bf09329-ca61-4fee-88cb-  
7e30b9ee305b/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetworks/myVnet/subnets/mySubne  
t/delegations/0",  
      "name": "0",  
      "provisioningState": "Succeeded",  
      "resourceGroup": "myResourceGroup",  
      "serviceName": "Microsoft.DBforPostgreSQL/serversv2",  
      "type": "Microsoft.Network/virtualNetworks/subnets/delegations"  
    }  
  ]
```

## Remove subnet delegation from an Azure service

Use [az network vnet subnet update](#) to remove the delegation from the subnet named **mySubnet**:

```
az network vnet subnet update \  
--resource-group myResourceGroup \  
--name mySubnet \  
--vnet-name myVnet \  
--remove delegations
```

To verify the delegation was removed, use [az network vnet subnet show](#). Verify the service is removed from the subnet under the property **serviceName**:

```
az network vnet subnet show \  
--resource-group myResourceGroup \  
--name mySubnet \  
--vnet-name myVnet \  
--query delegations
```

Output from command is a null bracket:

```
[]
```

## Azure PowerShell

### 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

## Connect to Azure

```
Connect-AzAccount
```

## Create a resource group

Create a resource group with [New-AzResourceGroup](#). An Azure resource group is a logical container into which

Azure resources are deployed and managed.

The following example creates a resource group named *myResourceGroup* in the *eastus* location:

```
New-AzResourceGroup -Name myResourceGroup -Location eastus
```

## Create virtual network

Create a virtual network named **myVnet** with a subnet named **mySubnet** using [New-AzVirtualNetworkSubnetConfig](#) in the **myResourceGroup** using [New-AzVirtualNetwork](#). The IP address space for the virtual network is **10.0.0.0/16**. The subnet within the virtual network is **10.0.0.0/24**.

```
$subnet = New-AzVirtualNetworkSubnetConfig -Name mySubnet -AddressPrefix "10.0.0.0/24"

New-AzVirtualNetwork -Name myVnet -ResourceGroupName myResourceGroup -Location eastus -AddressPrefix
"10.0.0.0/16" -Subnet $subnet
```

## Permissions

If you didn't create the subnet you would like to delegate to an Azure service, you need the following permission:

```
Microsoft.Network/virtualNetworks/subnets/write
```

The built-in [Network Contributor](#) role also contains the necessary permissions.

## Delegate a subnet to an Azure service

In this section, you delegate the subnet that you created in the preceding section to an Azure service.

Use [Add-AzDelegation](#) to update the subnet named **mySubnet** with a delegation named **myDelegation** to an Azure service. In this example **Microsoft.DBforPostgreSQL/serversv2** is used for the example delegation:

```
$vnet = Get-AzVirtualNetwork -Name "myVNet" -ResourceGroupName "myResourceGroup"
$subnet = Get-AzVirtualNetworkSubnetConfig -Name "mySubnet" -VirtualNetwork $vnet
$subnet = Add-AzDelegation -Name "myDelegation" -ServiceName "Microsoft.DBforPostgreSQL/serversv2" -Subnet
$subnet
Set-AzVirtualNetwork -VirtualNetwork $vnet
```

Use [Get-AzDelegation](#) to verify the delegation:

```
$subnet = Get-AzVirtualNetwork -Name "myVnet" -ResourceGroupName "myResourceGroup" | Get-
AzVirtualNetworkSubnetConfig -Name "mySubnet"
Get-AzDelegation -Name "myDelegation" -Subnet $subnet

ProvisioningState : Succeeded
ServiceName      : Microsoft.DBforPostgreSQL/serversv2
Actions          : {Microsoft.Network/virtualNetworks/subnets/join/action}
Name             : myDelegation
Etag             : W/"9cba4b0e-2ceb-444b-b553-454f8da07d8a"
Id               : /subscriptions/3bf09329-ca61-4fee-88cb-
7e30b9ee305b/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetworks/myVnet/subnets/mySubne
t/delegations/myDelegation
```

## Remove subnet delegation from an Azure service

Use [Remove-AzDelegation](#) to remove the delegation from the subnet named **mySubnet**:

```
$vnet = Get-AzVirtualNetwork -Name "myVnet" -ResourceGroupName "myResourceGroup"
$subnet = Get-AzVirtualNetworkSubnetConfig -Name "mySubnet" -VirtualNetwork $vnet
$subnet = Remove-AzDelegation -Name "myDelegation" -Subnet $subnet
Set-AzVirtualNetwork -VirtualNetwork $vnet
```

Use [Get-AzDelegation](#) to verify the delegation was removed:

```
$subnet = Get-AzVirtualNetwork -Name "myVnet" -ResourceGroupName "myResourceGroup" | Get-
AzVirtualNetworkSubnetConfig -Name "mySubnet"
Get-AzDelegation -Name "myDelegation" -Subnet $subnet

Get-AzDelegation: Sequence contains no matching element
```

## Next steps

- Learn how to [manage subnets in Azure](#).

# Connect virtual networks with virtual network peering using PowerShell

2/13/2020 • 6 minutes to read • [Edit Online](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

You can connect virtual networks to each other with virtual network peering. Once virtual networks are peered, resources in both virtual networks are able to communicate with each other, with the same latency and bandwidth as if the resources were in the same virtual network. In this article, you learn how to:

- Create two virtual networks
- Connect two virtual networks with a virtual network peering
- Deploy a virtual machine (VM) into each virtual network
- Communicate between VMs

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.

#### 4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

## Create virtual networks

Before creating a virtual network, you have to create a resource group for the virtual network, and all other resources created in this article. Create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location.

```
New-AzResourceGroup -ResourceGroupName myResourceGroup -Location EastUS
```

Create a virtual network with [New-AzVirtualNetwork](#). The following example creates a virtual network named *myVirtualNetwork1* with the address prefix *10.0.0.0/16*.

```
$virtualNetwork1 = New-AzVirtualNetwork ` 
    -ResourceGroupName myResourceGroup ` 
    -Location EastUS ` 
    -Name myVirtualNetwork1 ` 
    -AddressPrefix 10.0.0.0/16
```

Create a subnet configuration with [New-AzVirtualNetworkSubnetConfig](#). The following example creates a subnet configuration with a *10.0.0.0/24* address prefix:

```
$subnetConfig = Add-AzVirtualNetworkSubnetConfig ` 
    -Name Subnet1 ` 
    -AddressPrefix 10.0.0.0/24 ` 
    -VirtualNetwork $virtualNetwork1
```

Write the subnet configuration to the virtual network with [Set-AzVirtualNetwork](#), which creates the subnet:

```
$virtualNetwork1 | Set-AzVirtualNetwork
```

Create a virtual network with a *10.1.0.0/16* address prefix and one subnet:

```
# Create the virtual network.
$virtualNetwork2 = New-AzVirtualNetwork ` 
    -ResourceGroupName myResourceGroup ` 
    -Location EastUS ` 
    -Name myVirtualNetwork2 ` 
    -AddressPrefix 10.1.0.0/16

# Create the subnet configuration.
$subnetConfig = Add-AzVirtualNetworkSubnetConfig ` 
    -Name Subnet1 ` 
    -AddressPrefix 10.1.0.0/24 ` 
    -VirtualNetwork $virtualNetwork2

# Write the subnet configuration to the virtual network.
$virtualNetwork2 | Set-AzVirtualNetwork
```

## Peer virtual networks

Create a peering with [Add-AzVirtualNetworkPeering](#). The following example peers *myVirtualNetwork1* to *myVirtualNetwork2*.

```
Add-AzVirtualNetworkPeering `  
-Name myVirtualNetwork1-myVirtualNetwork2 `  
-VirtualNetwork $virtualNetwork1 `  
-RemoteVirtualNetworkId $virtualNetwork2.Id
```

In the output returned after the previous command executes, you see that the **PeeringState** is *Initiated*. The peering remains in the *Initiated* state until you create the peering from *myVirtualNetwork2* to *myVirtualNetwork1*. Create a peering from *myVirtualNetwork2* to *myVirtualNetwork1*.

```
Add-AzVirtualNetworkPeering `  
-Name myVirtualNetwork2-myVirtualNetwork1 `  
-VirtualNetwork $virtualNetwork2 `  
-RemoteVirtualNetworkId $virtualNetwork1.Id
```

In the output returned after the previous command executes, you see that the **PeeringState** is *Connected*. Azure also changed the peering state of the *myVirtualNetwork1-myVirtualNetwork2* peering to *Connected*. Confirm that the peering state for the *myVirtualNetwork1-myVirtualNetwork2* peering changed to *Connected* with [Get-AzVirtualNetworkPeering](#).

```
Get-AzVirtualNetworkPeering `  
-ResourceGroupName myResourceGroup `  
-VirtualNetworkName myVirtualNetwork1 `  
| Select PeeringState
```

Resources in one virtual network cannot communicate with resources in the other virtual network until the **PeeringState** for the peerings in both virtual networks is *Connected*.

## Create virtual machines

Create a VM in each virtual network so that you can communicate between them in a later step.

### Create the first VM

Create a VM with [New-AzVm](#). The following example creates a VM named *myVm1* in the *myVirtualNetwork1* virtual network. The `-AsJob` option creates the VM in the background, so you can continue to the next step. When prompted, enter the user name and password you want to log in to the VM with.

```
New-AzVm `  
-ResourceGroupName "myResourceGroup" `  
-Location "East US" `  
-VirtualNetworkName "myVirtualNetwork1" `  
-SubnetName "Subnet1" `  
-ImageName "Win2016Datacenter" `  
-Name "myVm1" `  
-AsJob
```

### Create the second VM

```
New-AzVm ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "East US" ` 
-VirtualNetworkName "myVirtualNetwork2" ` 
-SubnetName "Subnet1" ` 
-ImageName "Win2016Datacenter" ` 
-Name "myVm2"
```

The VM takes a few minutes to create. Do not continue with later steps until Azure creates the VM and returns output to PowerShell.

## Communicate between VMs

You can connect to a VM's public IP address from the internet. Use [Get-AzPublicIpAddress](#) to return the public IP address of a VM. The following example returns the public IP address of the *myVm1* VM:

```
Get-AzPublicIpAddress ` 
-Name myVm1 ` 
-ResourceGroupName myResourceGroup | Select IpAddress
```

Use the following command to create a remote desktop session with the *myVm1* VM from your local computer. Replace `<publicIpAddress>` with the IP address returned from the previous command.

```
mstsc /v:<publicIpAddress>
```

A Remote Desktop Protocol (.rdp) file is created, downloaded to your computer, and opened. Enter the user name and password (you may need to select **More choices**, then **Use a different account**, to specify the credentials you entered when you created the VM), and then click **OK**. You may receive a certificate warning during the sign-in process. Click **Yes** or **Continue** to proceed with the connection.

On the *myVm1* VM, enable the Internet Control Message Protocol (ICMP) through the Windows firewall so you can ping this VM from *myVm2* in a later step, using PowerShell:

```
New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4
```

Though ping is used to communicate between VMs in this article, allowing ICMP through the Windows Firewall for production deployments is not recommended.

To connect to the *myVm2* VM, enter the following command from a command prompt on the *myVm1* VM:

```
mstsc /v:10.1.0.4
```

Since you enabled ping on *myVm1*, you can now ping it by IP address from a command prompt on the *myVm2* VM:

```
ping 10.0.0.4
```

You receive four replies. Disconnect your RDP sessions to both *myVm1* and *myVm2*.

## Clean up resources

When no longer needed, use [Remove-AzResourcegroup](#) to remove the resource group and all of the resources it

contains.

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

## Next steps

In this article, you learned how to connect two networks in the same Azure region, with virtual network peering. You can also peer virtual networks in different [supported regions](#) and in [different Azure subscriptions](#), as well as create [hub and spoke network designs](#) with peering. To learn more about virtual network peering, see [Virtual network peering overview](#) and [Manage virtual network peerings](#).

You can [connect your own computer to a virtual network](#) through a VPN, and interact with resources in a virtual network, or in peered virtual networks. For reusable scripts to complete many of the tasks covered in the virtual network articles, see [script samples](#).

# Connect virtual networks with virtual network peering using the Azure CLI

2/13/2020 • 5 minutes to read • [Edit Online](#)

You can connect virtual networks to each other with virtual network peering. Once virtual networks are peered, resources in both virtual networks are able to communicate with each other, with the same latency and bandwidth as if the resources were in the same virtual network. In this article, you learn how to:

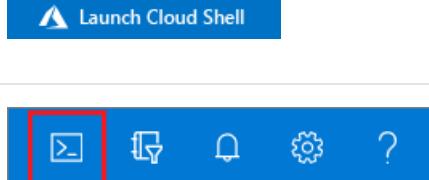
- Create two virtual networks
- Connect two virtual networks with a virtual network peering
- Deploy a virtual machine (VM) into each virtual network
- Communicate between VMs

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this article requires that you are running the Azure CLI version 2.0.28 or later. To find the version, run `az --version`. If you need to install or upgrade, see [Install Azure CLI](#).

## Create virtual networks

Before creating a virtual network, you have to create a resource group for the virtual network, and all other resources created in this article. Create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location.

```
az group create --name myResourceGroup --location eastus
```

Create a virtual network with [az network vnet create](#). The following example creates a virtual network named *myVirtualNetwork1* with the address prefix *10.0.0.0/16*.

```
az network vnet create \
--name myVirtualNetwork1 \
--resource-group myResourceGroup \
--address-prefixes 10.0.0.0/16 \
--subnet-name Subnet1 \
--subnet-prefix 10.0.0.0/24
```

Create a virtual network named *myVirtualNetwork2* with the address prefix *10.1.0.0/16*:

```
az network vnet create \
--name myVirtualNetwork2 \
--resource-group myResourceGroup \
--address-prefixes 10.1.0.0/16 \
--subnet-name Subnet1 \
--subnet-prefix 10.1.0.0/24
```

## Peer virtual networks

Peerings are established between virtual network IDs, so you must first get the ID of each virtual network with [az network vnet show](#) and store the ID in a variable.

```
# Get the id for myVirtualNetwork1.
vNet1Id=$(az network vnet show \
--resource-group myResourceGroup \
--name myVirtualNetwork1 \
--query id --out tsv)

# Get the id for myVirtualNetwork2.
vNet2Id=$(az network vnet show \
--resource-group myResourceGroup \
--name myVirtualNetwork2 \
--query id \
--out tsv)
```

Create a peering from *myVirtualNetwork1* to *myVirtualNetwork2* with [az network vnet peering create](#). If the `--allow-vnet-access` parameter is not specified, a peering is established, but no communication can flow through it.

```
az network vnet peering create \
--name myVirtualNetwork1-myVirtualNetwork2 \
--resource-group myResourceGroup \
--vnet-name myVirtualNetwork1 \
--remote-vnet $vNet2Id \
--allow-vnet-access
```

In the output returned after the previous command executes, you see that the **peeringState** is *Initiated*. The peering remains in the *Initiated* state until you create the peering from *myVirtualNetwork2* to

*myVirtualNetwork1*. Create a peering from *myVirtualNetwork2* to *myVirtualNetwork1*.

```
az network vnet peering create \
--name myVirtualNetwork2-myVirtualNetwork1 \
--resource-group myResourceGroup \
--vnet-name myVirtualNetwork2 \
--remote-vnet $vNet1Id \
--allow-vnet-access
```

In the output returned after the previous command executes, you see that the **peeringState** is *Connected*. Azure also changed the peering state of the *myVirtualNetwork1-myVirtualNetwork2* peering to *Connected*. Confirm that the peering state for the *myVirtualNetwork1-myVirtualNetwork2* peering changed to *Connected* with [az network vnet peering show](#).

```
az network vnet peering show \
--name myVirtualNetwork1-myVirtualNetwork2 \
--resource-group myResourceGroup \
--vnet-name myVirtualNetwork1 \
--query peeringState
```

Resources in one virtual network cannot communicate with resources in the other virtual network until the **peeringState** for the peerings in both virtual networks is *Connected*.

## Create virtual machines

Create a VM in each virtual network so that you can communicate between them in a later step.

### Create the first VM

Create a VM with [az vm create](#). The following example creates a VM named *myVm1* in the *myVirtualNetwork1* virtual network. If SSH keys do not already exist in a default key location, the command creates them. To use a specific set of keys, use the `--ssh-key-value` option. The `--no-wait` option creates the VM in the background, so you can continue to the next step.

```
az vm create \
--resource-group myResourceGroup \
--name myVm1 \
--image UbuntuLTS \
--vnet-name myVirtualNetwork1 \
--subnet Subnet1 \
--generate-ssh-keys \
--no-wait
```

### Create the second VM

Create a VM in the *myVirtualNetwork2* virtual network.

```
az vm create \
--resource-group myResourceGroup \
--name myVm2 \
--image UbuntuLTS \
--vnet-name myVirtualNetwork2 \
--subnet Subnet1 \
--generate-ssh-keys
```

The VM takes a few minutes to create. After the VM is created, the Azure CLI shows information similar to the following example:

```
{  
    "fqdns": "",  
    "id": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVm2",  
    "location": "eastus",  
    "macAddress": "00-0D-3A-23-9A-49",  
    "powerState": "VM running",  
    "privateIpAddress": "10.1.0.4",  
    "publicIpAddress": "13.90.242.231",  
    "resourceGroup": "myResourceGroup"  
}
```

Take note of the **publicIpAddress**. This address is used to access the VM from the internet in a later step.

## Communicate between VMs

Use the following command to create an SSH session with the *myVm2* VM. Replace <publicIpAddress> with the public IP address of your VM. In the previous example, the public IP address is 13.90.242.231.

```
ssh <publicIpAddress>
```

Ping the VM in *myVirtualNetwork1*.

```
ping 10.0.0.4 -c 4
```

You receive four replies.

Close the SSH session to the *myVm2* VM.

## Clean up resources

When no longer needed, use [az group delete](#) to remove the resource group and all of the resources it contains.

```
az group delete --name myResourceGroup --yes
```

## Next steps

In this article, you learned how to connect two networks in the same Azure region, with virtual network peering. You can also peer virtual networks in different [supported regions](#) and in [different Azure subscriptions](#), as well as create [hub and spoke network designs](#) with peering. To learn more about virtual network peering, see [Virtual network peering overview](#) and [Manage virtual network peerings](#).

You can [connect your own computer to a virtual network](#) through a VPN, and interact with resources in a virtual network, or in peered virtual networks. For reusable scripts to complete many of the tasks covered in the virtual network articles, see [script samples](#).

# Create a virtual network peering - Resource Manager, different subscriptions

2/13/2020 • 16 minutes to read • [Edit Online](#)

In this tutorial, you learn to create a virtual network peering between virtual networks created through Resource Manager. The virtual networks exist in different subscriptions. Peering two virtual networks enables resources in different virtual networks to communicate with each other with the same bandwidth and latency as though the resources were in the same virtual network. Learn more about [Virtual network peering](#).

The steps to create a virtual network peering are different, depending on whether the virtual networks are in the same, or different, subscriptions, and which [Azure deployment model](#) the virtual networks are created through. Learn how to create a virtual network peering in other scenarios by selecting the scenario from the following table:

AZURE DEPLOYMENT MODEL	AZURE SUBSCRIPTION
Both Resource Manager	Same
One Resource Manager, one classic	Same
One Resource Manager, one classic	Different

A virtual network peering cannot be created between two virtual networks deployed through the classic deployment model. If you need to connect virtual networks that were both created through the classic deployment model, you can use an Azure [VPN Gateway](#) to connect the virtual networks.

This tutorial peers virtual networks in the same region. You can also peer virtual networks in different [supported regions](#). It's recommended that you familiarize yourself with the [peering requirements and constraints](#) before peering virtual networks.

You can use the [Azure portal](#), the Azure [command-line interface](#) (CLI), Azure [PowerShell](#), or an [Azure Resource Manager template](#) to create a virtual network peering. Select any of the previous tool links to go directly to the steps for creating a virtual network peering using your tool of choice.

If the virtual networks are in different subscriptions, and the subscriptions are associated with different Azure Active Directory tenants, complete the following steps before continuing:

1. Add the user from each Active Directory tenant as a [guest user](#) in the opposite Azure Active Directory tenant.
2. Each user must accept the guest user invitation from the opposite Azure Active Directory tenant.

## Create peering - Azure portal

The following steps use different accounts for each subscription. If you're using an account that has permissions to both subscriptions, you can use the same account for all steps, skip the steps for logging out of the portal, and skip the steps for assigning another user permissions to the virtual networks.

1. Log in to the [Azure portal](#) as *UserA*. The account you log in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
2. Select + **Create a resource**, select **Networking**, and then select **Virtual network**.
3. Select or enter the following example values for the following settings, then select **Create**:

- **Name:** *myVnetA*
  - **Address space:** *10.0.0.0/16*
  - **Subnet name:** *default*
  - **Subnet address range:** *10.0.0.0/24*
  - **Subscription:** Select subscription A.
  - **Resource group:** Select **Create new** and enter *myResourceGroupA*
  - **Location:** *East US*
4. In the **Search resources** box at the top of the portal, type *myVnetA*. Select **myVnetA** when it appears in the search results.
5. Select **Access control (IAM)** from the vertical list of options on the left side.
6. Under **myVnetA - Access control (IAM)**, select **+ Add role assignment**.
7. Select **Network contributor** in the **Role** box.
8. In the **Select** box, select *UserB*, or type UserB's email address to search for it.
9. Select **Save**.
10. Under **myVnetA - Access control (IAM)**, select **Properties** from the vertical list of options on the left side. Copy the **RESOURCE ID**, which is used in a later step. The resource ID is similar to the following example:
- ```
/subscriptions/<Subscription  
Id>/resourceGroups/myResourceGroupA/providers/Microsoft.Network/virtualNetworks/myVnetA
```
11. Log out of the portal as UserA, then log in as UserB.
12. Complete steps 2-3, entering or selecting the following values in step 3:
- **Name:** *myVnetB*
  - **Address space:** *10.1.0.0/16*
  - **Subnet name:** *default*
  - **Subnet address range:** *10.1.0.0/24*
  - **Subscription:** Select subscription B.
  - **Resource group:** Select **Create new** and enter *myResourceGroupB*
  - **Location:** *East US*
13. In the **Search resources** box at the top of the portal, type *myVnetB*. Select **myVnetB** when it appears in the search results.
14. Under **myVnetB**, select **Properties** from the vertical list of options on the left side. Copy the **RESOURCE ID**, which is used in a later step. The resource ID is similar to the following example:
- ```
/subscriptions/<Subscription  
Id>/resourceGroups/myResourceGroupB/providers/Microsoft.ClassicNetwork/virtualNetworks/myVnetB
```
15. Select **Access control (IAM)** under **myVnetB**, and then complete steps 5-10 for myVnetB, entering **UserA** in step 8.
16. Log out of the portal as UserB and log in as UserA.
17. In the **Search resources** box at the top of the portal, type *myVnetA*. Select **myVnetA** when it appears in the search results.
18. Select **myVnetA**.
19. Under **SETTINGS**, select **Peerings**.

20. Under **myVnetA - Peerings**, select + **Add**

21. Under **Add peering**, enter, or select, the following options, then select **OK**:

- **Name:** *myVnetAToMyVnetB*
- **Virtual network deployment model:** Select **Resource Manager**.
- **I know my resource ID:** Check this box.
- **Resource ID:** Enter the resource ID from step 14.
- **Allow virtual network access:** Ensure that **Enabled** is selected. No other settings are used in this tutorial. To learn about all peering settings, read [Manage virtual network peerings](#).

22. The peering you created appears a short wait after selecting **OK** in the previous step. **Initiated** is listed in the **PEERING STATUS** column for the **myVnetAToMyVnetB** peering you created. You've peered myVnetA to myVnetB, but now you must peer myVnetB to myVnetA. The peering must be created in both directions to enable resources in the virtual networks to communicate with each other.

23. Log out of the portal as UserA and log in as UserB.

24. Complete steps 17-21 again for myVnetB. In step 21, name the peering *myVnetBToMyVnetA*, select *myVnetA* for **Virtual network**, and enter the ID from step 10 in the **Resource ID** box.

25. A few seconds after selecting **OK** to create the peering for myVnetB, the **myVnetBToMyVnetA** peering you just created is listed with **Connected** in the **PEERING STATUS** column.

26. Log out of the portal as UserB and log in as UserA.

27. Complete steps 17-19 again. The **PEERING STATUS** for the **myVnetAToVNetB** peering is now also **Connected**. The peering is successfully established after you see **Connected** in the **PEERING STATUS** column for both virtual networks in the peering. Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the resources in the virtual networks are not able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).

28. **Optional:** Though creating virtual machines is not covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.

29. **Optional:** To delete the resources that you create in this tutorial, complete the steps in the [Delete resources](#) section of this article.

## Create peering - Azure CLI

This tutorial uses different accounts for each subscription. If you're using an account that has permissions to both subscriptions, you can use the same account for all steps, skip the steps for logging out of Azure, and remove the lines of script that create user role assignments. Replace UserA@azure.com and UserB@azure.com in all of the following scripts with the usernames you're using for UserA and UserB.

The following scripts:

- Requires the Azure CLI version 2.0.4 or later. To find the version, run `az --version`. If you need to upgrade, see [Install Azure CLI](#).
- Works in a Bash shell. For options on running Azure CLI scripts on Windows client, see [Install the Azure CLI on Windows](#).

Instead of installing the CLI and its dependencies, you can use the Azure Cloud Shell. The Azure Cloud Shell is a free Bash shell that you can run directly within the Azure portal. It has the Azure CLI preinstalled and configured to use with your account. Select the **Try it** button in the script that follows, which invokes a Cloud Shell that you can log in to your Azure account with.

1. Open a CLI session and log in to Azure as UserA using the `az login` command. The account you log in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
2. Copy the following script to a text editor on your PC, replace `<SubscriptionA-Id>` with the ID of SubscriptionA, then copy the modified script, paste it in your CLI session, and press `Enter`. If you don't know your subscription Id, enter the `az account show` command. The value for `id` in the output is your subscription Id.

```
# Create a resource group.
az group create \
--name myResourceGroupA \
--location eastus

# Create virtual network A.
az network vnet create \
--name myVnetA \
--resource-group myResourceGroupA \
--location eastus \
--address-prefix 10.0.0.0/16

# Assign UserB permissions to virtual network A.
az role assignment create \
--assignee UserB@azure.com \
--role "Network Contributor" \
--scope /subscriptions/<SubscriptionA-Id>/resourceGroups/myResourceGroupA/providers/Microsoft.Network/VirtualNetworks/myVnetA
```

3. Log out of Azure as UserA using the `az logout` command, then log in to Azure as UserB. The account you log in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
4. Create myVnetB. Copy the script contents in step 2 to a text editor on your PC. Replace `<SubscriptionA-Id>` with the ID of SubscriptionB. Change 10.0.0.0/16 to 10.1.0.0/16, change all As to B, and all Bs to A. Copy the modified script, paste it in to your CLI session, and press `Enter`.
5. Log out of Azure as UserB and log in to Azure as UserA.
6. Create a virtual network peering from myVnetA to myVnetB. Copy the following script contents to a text editor on your PC. Replace `<SubscriptionB-Id>` with the ID of SubscriptionB. To execute the script, copy the modified script, paste it into your CLI session, and press Enter.

```
# Get the id for myVnetA.
vnetAId=$(az network vnet show \
--resource-group myResourceGroupA \
--name myVnetA \
--query id --out tsv)

# Peer myVNetA to myVnetB.
az network vnet peering create \
--name myVnetAToMyVnetB \
--resource-group myResourceGroupA \
--vnet-name myVnetA \
--remote-vnet-id /subscriptions/<SubscriptionB-Id>/resourceGroups/myResourceGroupB/providers/Microsoft.Network/VirtualNetworks/myVnetB \
--allow-vnet-access
```

7. View the peering state of myVnetA.

```
az network vnet peering list \
--resource-group myResourceGroupA \
--vnet-name myVnetA \
--output table
```

The state is **Initiated**. It changes to **Connected** once you create the peering to myVnetA from myVnetB.

8. Log out UserA from Azure and log in to Azure as UserB.
9. Create the peering from myVnetB to myVnetA. Copy the script contents in step 6 to a text editor on your PC. Replace `<SubscriptionB-Id>` with the ID for SubscriptionA and change all As to B and all Bs to A. Once you've made the changes, copy the modified script, paste it into your CLI session, and press `Enter`.
10. View the peering state of myVnetB. Copy the script contents in step 7 to a text editor on your PC. Change A to B for the resource group and virtual network names, copy the script, paste the modified script in to your CLI session, and then press `Enter`. The peering state is **Connected**. The peering state of myVnetA changes to **Connected** after you've created the peering from myVnetB to myVnetA. You can log UserA back in to Azure and complete step 7 again to verify the peering state of myVnetA.

#### NOTE

The peering is not established until the peering state is **Connected** for both virtual networks.

11. **Optional:** Though creating virtual machines is not covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
12. **Optional:** To delete the resources that you create in this tutorial, complete the steps in [Delete resources](#) in this article.

Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the resources in the virtual networks are not able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).

## Create peering - PowerShell

#### 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

This tutorial uses different accounts for each subscription. If you're using an account that has permissions to both subscriptions, you can use the same account for all steps, skip the steps for logging out of Azure, and remove the lines of script that create user role assignments. Replace UserA@azure.com and UserB@azure.com in all of the following scripts with the usernames you're using for UserA and UserB.

1. Confirm that you have Azure PowerShell version 1.0.0 or higher. You can do this by running the `Get-Module -Name Az`. We recommend installing the latest version of the PowerShell [Az module](#). If you're new to Azure PowerShell, see [Azure PowerShell overview](#).
2. Start a PowerShell session.

3. In PowerShell, log in to Azure as UserA by entering the `Connect-AzAccount` command. The account you log in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).

4. Create a resource group and virtual network A. Copy the following script to a text editor on your PC. Replace `<SubscriptionA-Id>` with the ID of SubscriptionA. If you don't know your subscription Id, enter the `Get-AzSubscription` command to view it. The value for **Id** in the returned output is your subscription ID. To execute the script, copy the modified script, paste it in to PowerShell, and then press `Enter`.

```
# Create a resource group.  
New-AzResourceGroup `  
    -Name MyResourceGroupA `  
    -Location eastus  
  
# Create virtual network A.  
$vNetA = New-AzVirtualNetwork `  
    -ResourceGroupName MyResourceGroupA `  
    -Name 'myVnetA' `  
    -AddressPrefix '10.0.0.0/16' `  
    -Location eastus  
  
# Assign UserB permissions to myVnetA.  
New-AzRoleAssignment `  
    -SignInName UserB@azure.com `  
    -RoleDefinitionName "Network Contributor" `  
    -Scope /subscriptions/<SubscriptionA-  
Id>/resourceGroups/myResourceGroupA/providers/Microsoft.Network/VirtualNetworks/myVnetA
```

5. Log out UserA from Azure and log in UserB. The account you log in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).

6. Copy the script contents in step 4 to a text editor on your PC. Replace `<SubscriptionA-Id>` with the ID for subscription B. Change 10.0.0.0/16 to 10.1.0.0/16. Change all As to B and all Bs to A. To execute the script, copy the modified script, paste into PowerShell, and then press `Enter`.

7. Log out UserB from Azure and log in UserA.

8. Create the peering from myVnetA to myVnetB. Copy the following script to a text editor on your PC. Replace `<SubscriptionB-Id>` with the ID of subscription B. To execute the script, copy the modified script, paste in to PowerShell, and then press `Enter`.

```
# Peer myVnetA to myVnetB.  
$vNetA=Get-AzVirtualNetwork -Name myVnetA -ResourceGroupName myResourceGroupA  
Add-AzVirtualNetworkPeering `  
    -Name 'myVnetAToMyVnetB' `  
    -VirtualNetwork $vNetA `  
    -RemoteVirtualNetworkId "/subscriptions/<SubscriptionB-  
Id>/resourceGroups/myResourceGroupB/providers/Microsoft.Network/virtualNetworks/myVnetB"
```

9. View the peering state of myVnetA.

```
Get-AzVirtualNetworkPeering `  
    -ResourceGroupName myResourceGroupA `  
    -VirtualNetworkName myVnetA `  
    | Format-Table VirtualNetworkName, PeeringState
```

The state is **Initiated**. It changes to **Connected** once you set up the peering to myVnetA from myVnetB.

10. Log out UserA from Azure and log in UserB.
11. Create the peering from myVnetB to myVnetA. Copy the script contents in step 8 to a text editor on your PC. Replace `<SubscriptionB-Id>` with the ID of subscription A and change all As to B and all Bs to A. To execute the script, copy the modified script, paste it in to PowerShell, and then press `Enter`.
12. View the peering state of myVnetB. Copy the script contents in step 9 to a text editor on your PC. Change A to B for the resource group and virtual network names. To execute the script, paste the modified script into PowerShell, and then press `Enter`. The state is **Connected**. The peering state of **myVnetA** changes to **Connected** after you've created the peering from **myVnetB** to **myVnetA**. You can log UserA back in to Azure and complete step 9 again to verify the peering state of myVnetA.

**NOTE**

The peering is not established until the peering state is **Connected** for both virtual networks.

Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the resources in the virtual networks are not able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).

13. **Optional:** Though creating virtual machines is not covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
14. **Optional:** To delete the resources that you create in this tutorial, complete the steps in [Delete resources](#) in this article.

## Create peering - Resource Manager template

1. To create a virtual network and assign the appropriate [permissions](#), complete the steps in the [Portal](#), [Azure CLI](#), or [PowerShell](#) sections of this article.
2. Save the text that follows to a file on your local computer. Replace `<subscription ID>` with UserA's subscription ID. You might save the file as vnetpeeringA.json, for example.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "variables": {},
    "resources": [
        {
            "apiVersion": "2016-06-01",
            "type": "Microsoft.Network/virtualNetworks/virtualNetworkPeerings",
            "name": "myVnetA/myVnetAToMyVnetB",
            "location": "[resourceGroup().location]",
            "properties": {
                "allowVirtualNetworkAccess": true,
                "allowForwardedTraffic": false,
                "allowGatewayTransit": false,
                "useRemoteGateways": false,
                "remoteVirtualNetwork": {
                    "id": "/subscriptions/<subscription
ID>/resourceGroups/PeeringTest/providers/Microsoft.Network/virtualNetworks/myVnetB"
                }
            }
        }
    ]
}
```

3. Log in to Azure as UserA and deploy the template using the [portal](#), [PowerShell](#), or the [Azure CLI](#). Specify the file name you saved the example json text in step 2 to.
4. Copy the example json from step 2 to a file on your computer and make changes to the lines that begin with:
  - **name:** Change *myVnetA/myVnetAToMyVnetB* to *myVnetB/myVnetBToMyVnetA*.
  - **id:** Replace `<subscription ID>` with UserB's subscription ID and change *myVnetB* to *myVnetA*.
5. Complete step 3 again, logged in to Azure as UserB.
6. **Optional:** Though creating virtual machines is not covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
7. **Optional:** To delete the resources that you create in this tutorial, complete the steps in the [Delete resources](#) section of this article, using either the Azure portal, PowerShell, or the Azure CLI.

## Delete resources

When you've finished this tutorial, you might want to delete the resources you created in the tutorial, so you don't incur usage charges. Deleting a resource group also deletes all resources that are in the resource group.

### Azure portal

1. Log in to the Azure portal as UserA.
2. In the portal search box, enter **myResourceGroupA**. In the search results, select **myResourceGroupA**.
3. Select **Delete**.
4. To confirm the deletion, in the **TYPE THE RESOURCE GROUP NAME** box, enter **myResourceGroupA**, and then select **Delete**.
5. Log out of the portal as UserA and log in as UserB.
6. Complete steps 2-4 for myResourceGroupB.

### Azure CLI

1. Log in to Azure as UserA and execute the following command:

```
az group delete --name myResourceGroupA --yes
```

2. Log out of Azure as UserA and log in as UserB.

3. Execute the following command:

```
az group delete --name myResourceGroupB --yes
```

## PowerShell

1. Log in to Azure as UserA and execute the following command:

```
Remove-AzResourceGroup -Name myResourceGroupA -force
```

2. Log out of Azure as UserA and log in as UserB.

3. Execute the following command:

```
Remove-AzResourceGroup -Name myResourceGroupB -force
```

## Next steps

- Thoroughly familiarize yourself with important [virtual network peering constraints and behaviors](#) before creating a virtual network peering for production use.
- Learn about all [virtual network peering settings](#).
- Learn how to [create a hub and spoke network topology](#) with virtual network peering.

# Create a virtual network peering - different deployment models, same subscription

2/4/2020 • 10 minutes to read • [Edit Online](#)

In this tutorial, you learn to create a virtual network peering between virtual networks created through different deployment models. Both virtual networks exist in the same subscription. Peering two virtual networks enables resources in different virtual networks to communicate with each other with the same bandwidth and latency as though the resources were in the same virtual network. Learn more about [Virtual network peering](#).

The steps to create a virtual network peering are different, depending on whether the virtual networks are in the same, or different, subscriptions, and which [Azure deployment model](#) the virtual networks are created through. Learn how to create a virtual network peering in other scenarios by clicking the scenario from the following table:

AZURE DEPLOYMENT MODEL	AZURE SUBSCRIPTION
Both Resource Manager	Same
Both Resource Manager	Different
One Resource Manager, one classic	Different

A virtual network peering can't be created between two virtual networks deployed through the classic deployment model. If you need to connect virtual networks that were both created through the classic deployment model, you can use an Azure [VPN Gateway](#) to connect the virtual networks.

This tutorial peers virtual networks in the same region. You can also peer virtual networks in different [supported regions](#). It's recommended that you familiarize yourself with the [peering requirements and constraints](#) before peering virtual networks.

You can use the Azure portal, the Azure [command-line interface](#) (CLI), Azure [PowerShell](#), or an Azure Resource Manager template to create a virtual network peering. Click any of the previous tool links to go directly to the steps for creating a virtual network peering using your tool of choice.

## Create peering - Azure portal

1. Sign in to the [Azure portal](#). The account you sign in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
2. Click **+ New**, click **Networking**, then click **Virtual network**.
3. In the **Create virtual network** blade, enter, or select values for the following settings, then click **Create**:
  - **Name:** *myVnet1*
  - **Address space:** *10.0.0.0/16*
  - **Subnet name:** *default*
  - **Subnet address range:** *10.0.0.0/24*
  - **Subscription:** Select your subscription
  - **Resource group:** Select **Create new** and enter *myResourceGroup*
  - **Location:** *East US*
4. Click **+ New**. In the **Search the Marketplace** box, type *Virtual network*. Click **Virtual network** when it

appears in the search results.

5. In the **Virtual network** blade, select **Classic** in the **Select a deployment model** box, and then click **Create**.
6. In the **Create virtual network** blade, enter, or select values for the following settings, then click **Create**:
  - **Name:** *myVnet2*
  - **Address space:** *10.1.0.0/16*
  - **Subnet name:** *default*
  - **Subnet address range:** *10.1.0.0/24*
  - **Subscription:** Select your subscription
  - **Resource group:** Select **Use existing** and select *myResourceGroup*
  - **Location:** *East US*
7. In the **Search resources** box at the top of the portal, type *myResourceGroup*. Click **myResourceGroup** when it appears in the search results. A blade appears for the **myresourcegroup** resource group. The resource group holds the two virtual networks created in previous steps.
8. Click **myVNet1**.
9. In the **myVnet1** blade that appears, click **Peerings** from the vertical list of options on the left side of the blade.
10. In the **myVnet1 - Peerings** blade that appeared, click **+ Add**
11. In the **Add peering** blade that appears, enter, or select the following options, then click **OK**:
  - **Name:** *myVnet1ToMyVnet2*
  - **Virtual network deployment model:** Select **Classic**.
  - **Subscription:** Select your subscription
  - **Virtual network:** Click **Choose a virtual network**, then click **myVnet2**.
  - **Allow virtual network access:** Ensure that **Enabled** is selected. No other settings are used in this tutorial. To learn about all peering settings, read [Manage virtual network peerings](#).
12. After clicking **OK** in the previous step, the **Add peering** blade closes and you see the **myVnet1 - Peerings** blade again. After a few seconds, the peering you created appears in the blade. **Connected** is listed in the **PEERING STATUS** column for the **myVnet1ToMyVnet2** peering you created.

The peering is now established. Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the resources in the virtual networks aren't able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).
13. **Optional:** Though creating virtual machines isn't covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
14. **Optional:** To delete the resources that you create in this tutorial, complete the steps in the [Delete resources](#) section of this article.

## Create peering - Azure CLI

Complete the following steps using the Azure classic CLI and the Azure CLI. You can complete the steps from the Azure Cloud Shell, by just selecting the **Try it** button in any of the following steps, or by installing the [classic CLI](#) and [CLI](#) and running the commands on your local computer.

1. If using the Cloud Shell, skip to step 2, because the Cloud Shell automatically signs you in to Azure. Open a

command session and sign in to Azure using the `azure login` command.

2. Run the CLI in Service Management mode by entering the `azure config mode asm` command.

3. Enter the following command to create the virtual network (classic):

```
azure network vnet create --vnet myVnet2 --address-space 10.1.0.0 --cidr 16 --location "East US"
```

4. Execute the following bash CLI script using the CLI, not the classic CLI. For options on running bash CLI scripts on Windows computer, see [Install the Azure CLI on Windows](#).

```
#!/bin/bash

# Create a resource group.
az group create \
--name myResourceGroup \
--location eastus

# Create the virtual network (Resource Manager).
az network vnet create \
--name myVnet1 \
--resource-group myResourceGroup \
--location eastus \
--address-prefix 10.0.0.0/16
```

5. Create a virtual network peering between the two virtual networks created through the different deployment models using the CLI. Copy the following script to a text editor on your PC. Replace `<subscription id>` with your subscription ID. If you don't know your subscription ID, enter the `az account show` command. The value for `id` in the output is your subscription ID. Paste the modified script in to your CLI session, and then press `Enter`.

```
# Get the ID for VNet1.
vnet1Id=$(az network vnet show \
--resource-group myResourceGroup \
--name myVnet1 \
--query id --out tsv)

# Peer VNet1 to VNet2.
az network vnet peering create \
--name myVnet1ToMyVnet2 \
--resource-group myResourceGroup \
--vnet-name myVnet1 \
--remote-vnet-id /subscriptions/<subscription id>/resourceGroups/Default-
Networking/providers/Microsoft.ClassicNetwork/virtualNetworks/myVnet2 \
--allow-vnet-access
```

6. After the script executes, review the peering for the virtual network (Resource Manager). Copy the following command, paste it in your CLI session, and then press `Enter`:

```
az network vnet peering list \
--resource-group myResourceGroup \
--vnet-name myVnet1 \
--output table
```

The output shows **Connected** in the **PeeringState** column.

Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the

resources in the virtual networks aren't able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).

7. **Optional:** Though creating virtual machines isn't covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
8. **Optional:** To delete the resources that you create in this tutorial, complete the steps in [Delete resources](#) in this article.

## Create peering - PowerShell

1. Install the latest version of the PowerShell [Azure](#) and [Az](#) modules. If you're new to Azure PowerShell, see [Azure PowerShell overview](#).
2. Start a PowerShell session.
3. In PowerShell, sign in to Azure by entering the `Add-AzureAccount` command. The account you sign in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
4. To create a virtual network (classic) with PowerShell, you must create a new, or modify an existing, network configuration file. Learn how to [export, update, and import network configuration files](#). The file should include the following **VirtualNetworkSite** element for the virtual network used in this tutorial:

```
<VirtualNetworkSite name="myVnet2" Location="East US">
  <AddressSpace>
    <AddressPrefix>10.1.0.0/16</AddressPrefix>
  </AddressSpace>
  <Subnets>
    <Subnet name="default">
      <AddressPrefix>10.1.0.0/24</AddressPrefix>
    </Subnet>
  </Subnets>
</VirtualNetworkSite>
```

### WARNING

Importing a changed network configuration file can cause changes to existing virtual networks (classic) in your subscription. Ensure you only add the previous virtual network and that you don't change or remove any existing virtual networks from your subscription.

5. Sign in to Azure to create the virtual network (Resource Manager) by entering the `Connect-AzAccount` command. The account you sign in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
6. Create a resource group and a virtual network (Resource Manager). Copy the script, paste it into PowerShell, and then press `Enter`.

```

# Create a resource group.
New-AzResourceGroup -Name myResourceGroup -Location eastus

# Create the virtual network (Resource Manager).
$vnet1 = New-AzVirtualNetwork ` 
-ResourceGroupName myResourceGroup ` 
-Name 'myVnet1' ` 
-AddressPrefix '10.0.0.0/16' ` 
-Location eastus

```

7. Create a virtual network peering between the two virtual networks created through the different deployment models. Copy the following script to a text editor on your PC. Replace <subscription id> with your subscription ID. If you don't know your subscription ID, enter the `Get-AzSubscription` command to view it. The value for **Id** in the returned output is your subscription ID. To execute the script, copy the modified script from your text editor, then right-click in your PowerShell session, and then press `Enter`.

```

# Peer VNet1 to VNet2.
Add-AzVirtualNetworkPeering ` 
-Name myVnet1ToMyVnet2 ` 
-VirtualNetwork $vnet1 ` 
-RemoteVirtualNetworkId /subscriptions/<subscription Id>/resourceGroups/Default-` 
Networking/providers/Microsoft.ClassicNetwork/virtualNetworks/myVnet2

```

8. After the script executes, review the peering for the virtual network (Resource Manager). Copy the following command, paste it in your PowerShell session, and then press `Enter`:

```

Get-AzVirtualNetworkPeering ` 
-ResourceGroupName myResourceGroup ` 
-VirtualNetworkName myVnet1 ` 
| Format-Table VirtualNetworkName, PeeringState

```

The output shows **Connected** in the **PeeringState** column.

Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the resources in the virtual networks aren't able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).

9. **Optional:** Though creating virtual machines isn't covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
10. **Optional:** To delete the resources that you create in this tutorial, complete the steps in [Delete resources](#) in this article.

## Delete resources

When you've finished this tutorial, you might want to delete the resources you created in the tutorial, so you don't incur usage charges. Deleting a resource group also deletes all resources that are in the resource group.

### Azure portal

1. In the portal search box, enter **myResourceGroup**. In the search results, click **myResourceGroup**.
2. On the **myResourceGroup** blade, click the **Delete** icon.
3. To confirm the deletion, in the **TYPE THE RESOURCE GROUP NAME** box, enter **myResourceGroup**, and then click **Delete**.

## Azure CLI

1. Use the Azure CLI to delete the virtual network (Resource Manager) with the following command:

```
az group delete --name myResourceGroup --yes
```

2. Use the classic CLI to delete the virtual network (classic) with the following commands:

```
azure config mode asm  
azure network vnet delete --vnet myVnet2 --quiet
```

## PowerShell

1. Enter the following command to delete the virtual network (Resource Manager):

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

2. To delete the virtual network (classic) with PowerShell, you must modify an existing network configuration file. Learn how to [export, update, and import network configuration files](#). Remove the following VirtualNetworkSite element for the virtual network used in this tutorial:

```
<VirtualNetworkSite name="myVnet2" Location="East US">  
  <AddressSpace>  
    <AddressPrefix>10.1.0.0/16</AddressPrefix>  
  </AddressSpace>  
  <Subnets>  
    <Subnet name="default">  
      <AddressPrefix>10.1.0.0/24</AddressPrefix>  
    </Subnet>  
  </Subnets>  
</VirtualNetworkSite>
```

### WARNING

Importing a changed network configuration file can cause changes to existing virtual networks (classic) in your subscription. Ensure you only remove the previous virtual network and that you don't change or remove any other existing virtual networks from your subscription.

## Next steps

- Thoroughly familiarize yourself with important [virtual network peering constraints and behaviors](#) before creating a virtual network peering for production use.
- Learn about all [virtual network peering settings](#).
- Learn how to [create a hub and spoke network topology](#) with virtual network peering.

# Create a virtual network peering - different deployment models and subscriptions

2/4/2020 • 14 minutes to read • [Edit Online](#)

In this tutorial, you learn to create a virtual network peering between virtual networks created through different deployment models. The virtual networks exist in different subscriptions. Peering two virtual networks enables resources in different virtual networks to communicate with each other with the same bandwidth and latency as though the resources were in the same virtual network. Learn more about [Virtual network peering](#).

The steps to create a virtual network peering are different, depending on whether the virtual networks are in the same, or different, subscriptions, and which [Azure deployment model](#) the virtual networks are created through. Learn how to create a virtual network peering in other scenarios by clicking the scenario from the following table:

AZURE DEPLOYMENT MODEL	AZURE SUBSCRIPTION
Both Resource Manager	Same
Both Resource Manager	Different
One Resource Manager, one classic	Same

A virtual network peering cannot be created between two virtual networks deployed through the classic deployment model. This tutorial uses virtual networks that exist in the same region. This tutorial peers virtual networks in the same region. You can also peer virtual networks in different [supported regions](#). It's recommended that you familiarize yourself with the [peering requirements and constraints](#) before peering virtual networks.

When creating a virtual network peering between virtual networks that exist in different subscriptions, the subscriptions must both be associated to the same Azure Active Directory tenant. If you don't already have an Azure Active Directory tenant, you can quickly [create one](#). You can connect virtual networks in different subscriptions and different Azure Active Directory tenants using an Azure [VPN Gateway](#).

You can use the [Azure portal](#), the Azure [command-line interface](#) (CLI), or Azure [PowerShell](#) to create a virtual network peering. Click any of the previous tool links to go directly to the steps for creating a virtual network peering using your tool of choice.

## Create peering - Azure portal

This tutorial uses different accounts for each subscription. If you're using an account that has permissions to both subscriptions, you can use the same account for all steps, skip the steps for logging out of the portal, and skip the steps for assigning another user permissions to the virtual networks.

1. Log in to the [Azure portal](#) as UserA. The account you log in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
2. Click **+ New**, click **Networking**, then click **Virtual network**.
3. In the **Create virtual network** blade, enter, or select values for the following settings, then click **Create**:
  - **Name:** *myVnetA*
  - **Address space:** *10.0.0.0/16*
  - **Subnet name:** *default*

- **Subnet address range:** 10.0.0.0/24
  - **Subscription:** Select subscription A.
  - **Resource group:** Select **Create new** and enter *myResourceGroupA*
  - **Location:** East US
- In the **Search resources** box at the top of the portal, type *myVnetA*. Click **myVnetA** when it appears in the search results. A blade appears for the **myVnetA** virtual network.
  - In the **myVnetA** blade that appears, click **Access control (IAM)** from the vertical list of options on the left side of the blade.
  - In the **myVnetA - Access control (IAM)** blade that appears, click **+ Add role assignment**.
  - In the **Add role assignment** blade that appears, select **Network contributor** in the **Role** box.
  - In the **Select** box, select UserB, or type UserB's email address to search for it. The list of users shown is from the same Azure Active Directory tenant as the virtual network you're setting up the peering for. Click UserB when it appears in the list.
  - Click **Save**.
  - Log out of the portal as UserA, then log in as UserB.
  - Click **+ New**, type *Virtual network* in the **Search the Marketplace** box, then click **Virtual network** in the search results.
  - In the **Virtual Network** blade that appears, select **Classic** in the **Select a deployment model** box, then click **Create**.
  - In the Create virtual network (classic) box that appears, enter the following values:
    - **Name:** *myVnetB*
    - **Address space:** 10.1.0.0/16
    - **Subnet name:** *default*
    - **Subnet address range:** 10.1.0.0/24
    - **Subscription:** Select subscription B.
    - **Resource group:** Select **Create new** and enter *myResourceGroupB*
    - **Location:** East US
  - In the **Search resources** box at the top of the portal, type *myVnetB*. Click **myVnetB** when it appears in the search results. A blade appears for the **myVnetB** virtual network.
  - In the **myVnetB** blade that appears, click **Properties** from the vertical list of options on the left side of the blade. Copy the **RESOURCE ID**, which is used in a later step. The resource ID is similar to the following example:

```
/subscriptions/<Subscription ID>/resourceGroups/myResourceGroupB/providers/Microsoft.ClassicNetwork/virtualNetworks/myVnetB
```
  - Complete steps 5-9 for myVnetB, entering **UserA** in step 8.
  - Log out of the portal as UserB and log in as UserA.
  - In the **Search resources** box at the top of the portal, type *myVnetA*. Click **myVnetA** when it appears in the search results. A blade appears for the **myVnet** virtual network.
  - Click **myVnetA**.
  - In the **myVnetA** blade that appears, click **Peerings** from the vertical list of options on the left side of the blade.

21. In the **myVnetA - Peerings** blade that appeared, click **+ Add**
22. In the **Add peering** blade that appears, enter, or select the following options, then click **OK**:
  - **Name:** *myVnetAToMyVnetB*
  - **Virtual network deployment model:** Select **Classic**.
  - **I know my resource ID:** Check this box.
  - **Resource ID:** Enter the resource ID of myVnetB from step 15.
  - **Allow virtual network access:** Ensure that **Enabled** is selected. No other settings are used in this tutorial. To learn about all peering settings, read [Manage virtual network peerings](#).
23. After clicking **OK** in the previous step, the **Add peering** blade closes and you see the **myVnetA - Peerings** blade again. After a few seconds, the peering you created appears in the blade. **Connected** is listed in the **PEERING STATUS** column for the **myVnetAToMyVnetB** peering you created. The peering is now established. There is no need to peer the virtual network (classic) to the virtual network (Resource Manager).
 

Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the resources in the virtual networks are not able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).
24. **Optional:** Though creating virtual machines is not covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
25. **Optional:** To delete the resources that you create in this tutorial, complete the steps in the [Delete resources](#) section of this article.

## Create peering - Azure CLI

This tutorial uses different accounts for each subscription. If you're using an account that has permissions to both subscriptions, you can use the same account for all steps, skip the steps for logging out of Azure, and remove the lines of script that create user role assignments. Replace UserA@azure.com and UserB@azure.com in all of the following scripts with the usernames you're using for UserA and UserB. Complete the following steps using the Azure classic CLI and the Azure CLI. You can complete the steps from the Azure Cloud Shell, by just selecting the **Try it** button in any of the following steps, or by installing the [classic CLI](#) and [CLI](#) and running the commands on your local computer.

1. If using the Cloud Shell, skip to step 2, because the Cloud Shell automatically signs you in to Azure. Open a command session and sign in to Azure using the `azure login` command.
2. Run the classic CLI in Service Management mode by entering the `azure config mode asm` command.
3. Enter the following classic CLI command to create the virtual network (classic):

```
azure network vnet create --vnet myVnetB --address-space 10.1.0.0 --cidr 16 --location "East US"
```

4. The remaining steps must be completed using a bash shell with the Azure CLI (not the classic CLI).
5. Copy the following script to a text editor on your PC. Replace `<SubscriptionB-Id>` with your subscription ID. If you don't know your subscription Id, enter the `az account show` command. The value for **id** in the output is your subscription Id. Copy the modified script, paste it in to your CLI session, and then press `Enter`.

```
az role assignment create \
--assignee UserA@azure.com \
--role "Classic Network Contributor" \
--scope /subscriptions/<SubscriptionB-Id>/resourceGroups/Default-
Networking/providers/Microsoft.ClassicNetwork/virtualNetworks/myVnetB
```

When you created the virtual network (classic) in step 4, Azure created the virtual network in the *Default-Networking* resource group.

6. Log UserB out of Azure and log in as UserA in the CLI.
7. Create a resource group and a virtual network (Resource Manager). Copy the following script, paste it in to your CLI session, and then press **Enter**.

```
#!/bin/bash

# Variables for common values used throughout the script.
rgName="myResourceGroupA"
location="eastus"

# Create a resource group.
az group create \
--name $rgName \
--location $location

# Create virtual network A (Resource Manager).
az network vnet create \
--name myVnetA \
--resource-group $rgName \
--location $location \
--address-prefix 10.0.0.0/16

# Get the id for myVnetA.
vNetAId=$(az network vnet show \
--resource-group $rgName \
--name myVnetA \
--query id --out tsv)

# Assign UserB permissions to myVnetA.
az role assignment create \
--assignee UserB@azure.com \
--role "Network Contributor" \
--scope $vNetAId
```

8. Create a virtual network peering between the two virtual networks created through the different deployment models. Copy the following script to a text editor on your PC. Replace `<SubscriptionB-id>` with your subscription Id. If you don't know your subscription Id, enter the `az account show` command. The value for **id** in the output is your subscription Id. Azure created the virtual network (classic) you created in step 4 in a resource group named *Default-Networking*. Paste the modified script in your CLI session, and then press **Enter**.

```
# Peer VNet1 to VNet2.
az network vnet peering create \
--name myVnetAToMyVnetB \
--resource-group $rgName \
--vnet-name myVnetA \
--remote-vnet-id /subscriptions/<SubscriptionB-id>/resourceGroups/Default-
Networking/providers/Microsoft.ClassicNetwork/virtualNetworks/myVnetB \
--allow-vnet-access
```

9. After the script executes, review the peering for the virtual network (Resource Manager). Copy the following script, and then paste it in your CLI session:

```
az network vnet peering list \
--resource-group $rgName \
--vnet-name myVnetA \
--output table
```

The output shows **Connected** in the **PeeringState** column.

Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the resources in the virtual networks are not able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).

10. **Optional:** Though creating virtual machines is not covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
11. **Optional:** To delete the resources that you create in this tutorial, complete the steps in [Delete resources](#) in this article.

## Create peering - PowerShell

This tutorial uses different accounts for each subscription. If you're using an account that has permissions to both subscriptions, you can use the same account for all steps, skip the steps for logging out of Azure, and remove the lines of script that create user role assignments. Replace UserA@azure.com and UserB@azure.com in all of the following scripts with the usernames you're using for UserA and UserB.

1. Install the latest version of the PowerShell [Azure](#) and [Az](#) modules. If you're new to Azure PowerShell, see [Azure PowerShell overview](#).
2. Start a PowerShell session.
3. In PowerShell, log in to UserB's subscription as UserB by entering the `Add-AzureAccount` command. The account you log in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
4. To create a virtual network (classic) with PowerShell, you must create a new, or modify an existing, network configuration file. Learn how to [export, update, and import network configuration files](#). The file should include the following **VirtualNetworkSite** element for the virtual network used in this tutorial:

```
<VirtualNetworkSite name="myVnetB" Location="East US">
<AddressSpace>
<AddressPrefix>10.1.0.0/16</AddressPrefix>
</AddressSpace>
<Subnets>
<Subnet name="default">
<AddressPrefix>10.1.0.0/24</AddressPrefix>
</Subnet>
</Subnets>
</VirtualNetworkSite>
```

**WARNING**

Importing a changed network configuration file can cause changes to existing virtual networks (classic) in your subscription. Ensure you only add the previous virtual network and that you don't change or remove any existing virtual networks from your subscription.

5. Log in to UserB's subscription as UserB to use Resource Manager commands by entering the `Connect-AzAccount` command.
6. Assign UserA permissions to virtual network B. Copy the following script to a text editor on your PC and replace `<SubscriptionB-id>` with the ID of subscription B. If you don't know the subscription Id, enter the `Get-AzSubscription` command to view it. The value for **Id** in the returned output is your subscription ID. Azure created the virtual network (classic) you created in step 4 in a resource group named *Default-Networking*. To execute the script, copy the modified script, paste it in to PowerShell, and then press `Enter`.

```
New-AzRoleAssignment ` 
    -SignInName UserA@azure.com ` 
    -RoleDefinitionName "Classic Network Contributor" ` 
    -Scope /subscriptions/<SubscriptionB-id>/resourceGroups/Default- 
        Networking/providers/Microsoft.ClassicNetwork/virtualNetworks/myVnetB
```

7. Log out of Azure as UserB and log in to UserA's subscription as UserA by entering the `Connect-AzAccount` command. The account you log in with must have the necessary permissions to create a virtual network peering. For a list of permissions, see [Virtual network peering permissions](#).
8. Create the virtual network (Resource Manager) by copying the following script, pasting it in to PowerShell, and then pressing `Enter`:

```
# Variables for common values
$rgName='MyResourceGroupA'
.setLocation='eastus'

# Create a resource group.
New-AzResourceGroup ` 
    -Name $rgName ` 
    -Location $location

# Create virtual network A.
$vnetA = New-AzVirtualNetwork ` 
    -ResourceGroupName $rgName ` 
    -Name 'myVnetA' ` 
    -AddressPrefix '10.0.0.0/16' ` 
    -Location $location
```

9. Assign UserB permissions to myVnetA. Copy the following script to a text editor on your PC and replace `<SubscriptionA-Id>` with the ID of subscription A. If you don't know the subscription Id, enter the `Get-AzSubscription` command to view it. The value for **Id** in the returned output is your subscription ID. Paste the modified version of the script in PowerShell, and then press `Enter` to execute it.

```
New-AzRoleAssignment ` 
    -SignInName UserB@azure.com ` 
    -RoleDefinitionName "Network Contributor" ` 
    -Scope /subscriptions/<SubscriptionA- 
        Id>/resourceGroups/myResourceGroupA/providers/Microsoft.Network/VirtualNetworks/myVnetA
```

10. Copy the following script to a text editor on your PC, and replace `<SubscriptionB-id>` with the ID of

subscription B. To peer myVnetA to myVNetB, copy the modified script, paste it in to PowerShell, and then press **Enter**.

```
Add-AzVirtualNetworkPeering ` 
-Name 'myVnetAToMyVnetB' ` 
-VirtualNetwork $vnetA ` 
-RemoteVirtualNetworkId /subscriptions/<SubscriptionB-id>/resourceGroups/Default- 
Networking/providers/Microsoft.ClassicNetwork/virtualNetworks/myVnetB
```

11. View the peering state of myVnetA by copying the following script, pasting it into PowerShell, and pressing **Enter**.

```
Get-AzVirtualNetworkPeering ` 
-ResourceGroupName $rgName ` 
-VirtualNetworkName myVnetA ` 
| Format-Table VirtualNetworkName, PeeringState
```

The state is **Connected**. It changes to **Connected** once you set up the peering to myVnetA from myVnetB.

Any Azure resources you create in either virtual network are now able to communicate with each other through their IP addresses. If you're using default Azure name resolution for the virtual networks, the resources in the virtual networks are not able to resolve names across the virtual networks. If you want to resolve names across virtual networks in a peering, you must create your own DNS server. Learn how to set up [Name resolution using your own DNS server](#).

12. **Optional:** Though creating virtual machines is not covered in this tutorial, you can create a virtual machine in each virtual network and connect from one virtual machine to the other, to validate connectivity.
13. **Optional:** To delete the resources that you create in this tutorial, complete the steps in [Delete resources](#) in this article.

## Delete resources

When you've finished this tutorial, you might want to delete the resources you created in the tutorial, so you don't incur usage charges. Deleting a resource group also deletes all resources that are in the resource group.

### Azure portal

1. In the portal search box, enter **myResourceGroupA**. In the search results, click **myResourceGroupA**.
2. On the **myResourceGroupA** blade, click the **Delete** icon.
3. To confirm the deletion, in the **TYPE THE RESOURCE GROUP NAME** box, enter **myResourceGroupA**, and then click **Delete**.
4. In the **Search resources** box at the top of the portal, type *myVnetB*. Click **myVnetB** when it appears in the search results. A blade appears for the **myVnetB** virtual network.
5. In the **myVnetB** blade, click **Delete**.
6. To confirm the deletion, click **Yes** in the **Delete virtual network** box.

### Azure CLI

1. Log in to Azure using the CLI to delete the virtual network (Resource Manager) with the following command:

```
az group delete --name myResourceGroupA --yes
```

2. Sign in to Azure using the classic CLI to delete the virtual network (classic) with the following commands:

```
azure config mode asm  
  
azure network vnet delete --vnet myVnetB --quiet
```

## PowerShell

- At the PowerShell command prompt, enter the following command to delete the virtual network (Resource Manager):

```
Remove-AzResourceGroup -Name myResourceGroupA -Force
```

- To delete the virtual network (classic) with PowerShell, you must modify an existing network configuration file. Learn how to [export, update, and import network configuration files](#). Remove the following VirtualNetworkSite element for the virtual network used in this tutorial:

```
<VirtualNetworkSite name="myVnetB" Location="East US">  
  <AddressSpace>  
    <AddressPrefix>10.1.0.0/16</AddressPrefix>  
  </AddressSpace>  
  <Subnets>  
    <Subnet name="default">  
      <AddressPrefix>10.1.0.0/24</AddressPrefix>  
    </Subnet>  
  </Subnets>  
</VirtualNetworkSite>
```

### WARNING

Importing a changed network configuration file can cause changes to existing virtual networks (classic) in your subscription. Ensure you only remove the previous virtual network and that you don't change or remove any other existing virtual networks from your subscription.

## Next steps

- Thoroughly familiarize yourself with important [virtual network peering constraints and behaviors](#) before creating a virtual network peering for production use.
- Learn about all [virtual network peering settings](#).
- Learn how to [create a hub and spoke network topology](#) with virtual network peering.

# Create, change, or delete a virtual network peering

1/14/2020 • 16 minutes to read • [Edit Online](#)

Learn how to create, change, or delete a virtual network peering. Virtual network peering enables you to connect virtual networks in the same region and across regions (also known as Global VNet Peering) through the Azure backbone network. Once peered, the virtual networks are still managed as separate resources. If you're new to virtual network peering, you can learn more about it in the [virtual network peering overview](#) or by completing a [tutorial](#).

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with an account that has the [necessary permissions](#) to work with peerings.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` with an account that has the [necessary permissions](#) to work with peering, to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.31 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` with an account that has the [necessary permissions](#) to work with peering, to create a connection with Azure.

The account you log into, or connect to Azure with, must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Permissions](#).

## Create a peering

Before creating a peering, familiarize yourself with the requirements and constraints and [necessary permissions](#).

1. In the search box at the top of the Azure portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it. Do not select **Virtual networks (classic)** if it appears in the list, as you cannot create a peering from a virtual network deployed through the classic deployment model.
2. Select the virtual network in the list that you want to create a peering for.

3. Under **SETTINGS**, select **Peerings**.
4. Select **+ Add**.
5. Enter or select values for the following settings:
  - **Name:** The name for the peering must be unique within the virtual network.
  - **Virtual network deployment model:** Select which deployment model the virtual network you want to peer with was deployed through.
  - **I know my resource ID:** If you have read access to the virtual network you want to peer with, leave this checkbox unchecked. If you don't have read access to the virtual network or subscription you want to peer with, check this box. Enter the full resource ID of the virtual network you want to peer with in the **Resource ID** box that appeared when you checked the box. The resource ID you enter must be for a virtual network that exists in the same, or [supported different Azure region](#) as this virtual network. The full resource ID looks similar to

```
/subscriptions/<Id>/resourceGroups/<resource-group-name>/providers/Microsoft.Network/virtualNetworks/<virtual-network-name>
```

. You can get the resource ID for a virtual network by viewing the properties for a virtual network. To learn how to view the properties for a virtual network, see [Manage virtual networks](#). If the subscription is associated to a different Azure Active Directory tenant than the subscription with the virtual network you're creating the peering from, first add a user from each tenant as a [guest user](#) in the opposite tenant.
  - **Subscription:** Select the [subscription](#) of the virtual network you want to peer with. One or more subscriptions are listed, depending on how many subscriptions your account has read access to. If you checked the **Resource ID** checkbox, this setting isn't available.
  - **Virtual network:** Select the virtual network you want to peer with. You can select a virtual network created through either Azure deployment model. If you want to select a virtual network in a different region, you must select a virtual network in a [supported region](#). You must have read access to the virtual network for it to be visible in the list. If a virtual network is listed, but grayed out, it may be because the address space for the virtual network overlaps with the address space for this virtual network. If virtual network address spaces overlap, they cannot be peered. If you checked the **Resource ID** checkbox, this setting isn't available.
  - **Allow virtual network access:** Select **Enabled** (default) if you want to enable communication between the two virtual networks. Enabling communication between virtual networks allows resources connected to either virtual network to communicate with each other with the same bandwidth and latency as if they were connected to the same virtual network. All communication between resources in the two virtual networks is over the Azure private network. The **VirtualNetwork** service tag for network security groups encompasses the virtual network and peered virtual network. To learn more about network security group service tags, see [Network security groups overview](#). Select **Disabled** if you don't want traffic to flow to the peered virtual network. You might select **Disabled** if you've peered a virtual network with another virtual network, but occasionally want to disable traffic flow between the two virtual networks. You may find enabling/disabling is more convenient than deleting and re-creating peerings. When this setting is disabled, traffic doesn't flow between the peered virtual networks.
  - **Allow forwarded traffic:** Check this box to allow traffic *forwarded* by a network virtual appliance in a virtual network (that didn't originate from the virtual network) to flow to this virtual network through a peering. For example, consider three virtual networks named Spoke1, Spoke2, and Hub. A peering exists between each spoke virtual network and the Hub virtual network, but peerings don't exist between the spoke virtual networks. A network virtual appliance is deployed in the Hub virtual network, and user-defined routes are applied to each spoke virtual network that route traffic

between the subnets through the network virtual appliance. If this checkbox is not checked for the peering between each spoke virtual network and the hub virtual network, traffic doesn't flow between the spoke virtual networks because the hub is not forwarding the traffic between the virtual networks. While enabling this capability allows the forwarded traffic through the peering, it does not create any user-defined routes or network virtual appliances. User-defined routes and network virtual appliances are created separately. Learn about [user-defined routes](#). You don't need to check this setting if traffic is forwarded between virtual networks through an Azure VPN Gateway.

- **Allow gateway transit:** Check this box if you have a virtual network gateway attached to this virtual network and want to allow traffic from the peered virtual network to flow through the gateway. For example, this virtual network may be attached to an on-premises network through a virtual network gateway. The gateway can be an ExpressRoute or VPN gateway. Checking this box allows traffic from the peered virtual network to flow through the gateway attached to this virtual network to the on-premises network. If you check this box, the peered virtual network cannot have a gateway configured. The peered virtual network must have the **Use remote gateways** checkbox checked when setting up the peering from the other virtual network to this virtual network. If you leave this box unchecked (default), traffic from the peered virtual network still flows to this virtual network, but cannot flow through a virtual network gateway attached to this virtual network. If the peering is between a virtual network (Resource Manager) and a virtual network (classic), the gateway must be in the virtual network (Resource Manager).

In addition to forwarding traffic to an on-premises network, a VPN gateway can forward network traffic between virtual networks that are peered with the virtual network the gateway is in, without the virtual networks needing to be peered with each other. Using a VPN gateway to forward traffic is useful when you want to use a VPN gateway in a hub (see the hub and spoke example described for **Allow forwarded traffic**) virtual network to route traffic between spoke virtual networks that aren't peered with each other. To learn more about allowing use of a gateway for transit, see [Configure a VPN gateway for transit in a virtual network peering](#). This scenario requires implementing user-defined routes that specify the virtual network gateway as the next hop type. Learn about [user-defined routes](#). You can only specify a VPN gateway as a next hop type in a user-defined route, you cannot specify an ExpressRoute gateway as the next hop type in a user-defined route.

- **Use remote gateways:** Check this box to allow traffic from this virtual network to flow through a virtual network gateway attached to the virtual network you're peering with. For example, the virtual network you're peering with has a VPN gateway attached that enables communication to an on-premises network. Checking this box allows traffic from this virtual network to flow through the VPN gateway attached to the peered virtual network. If you check this box, the peered virtual network must have a virtual network gateway attached to it and must have the **Allow gateway transit** checkbox checked. If you leave this box unchecked (default), traffic from the peered virtual network can still flow to this virtual network, but cannot flow through a virtual network gateway attached to this virtual network. Only one peering for this virtual network can have this setting enabled.

You cannot use remote gateways if you already have a gateway configured in your virtual network. To learn more about using a gateway for transit, see [Configure a VPN gateway for transit in a virtual network peering](#)

## 6. Select **OK** to add the peering to the virtual network you selected.

For step-by-step instructions for implementing peering between virtual networks in different subscriptions and deployment models, see [next steps](#).

### Commands

- **Azure CLI:** `az network vnet peering create`
- **PowerShell:** `Add-AzVirtualNetworkPeering`

# View or change peering settings

Before changing a peering, familiarize yourself with the requirements and constraints and [necessary permissions](#).

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it. Do not select **Virtual networks (classic)** if it appears in the list, as you cannot create a peering from a virtual network deployed through the classic deployment model.
2. Select the virtual network in the list that you want to change peering settings for.
3. Under **SETTINGS**, select **Peerings**.
4. Select the peering you want to view or change settings for.
5. Change the appropriate setting. Read about the options for each setting in [step 5](#) of Create a peering.
6. Select **Save**.

## Commands

- **Azure CLI:** [az network vnet peering list](#) to list peerings for a virtual network, [az network vnet peering show](#) to show settings for a specific peering, and [az network vnet peering update](#) to change peering settings.|
- **PowerShell:** [Get-AzVirtualNetworkPeering](#) to retrieve view peering settings and [Set-AzVirtualNetworkPeering](#) to change settings.

# Delete a peering

Before deleting a peering, ensure your account has the [necessary permissions](#).

When a peering is deleted, traffic from a virtual network no longer flows to the peered virtual network. When virtual networks deployed through Resource Manager are peered, each virtual network has a peering to the other virtual network. Though deleting the peering from one virtual network disables the communication between the virtual networks, it does not delete the peering from the other virtual network. The peering status for the peering that exists in the other virtual network is **Disconnected**. You cannot recreate the peering until you re-create the peering in the first virtual network and the peering status for both virtual networks changes to *Connected*.

If you want virtual networks to communicate sometimes, but not always, rather than deleting a peering, you can set the **Allow virtual network access** setting to **Disabled** instead. To learn how, read step 6 of the Create a peering section of this article. You may find disabling and enabling network access easier than deleting and recreating peerings.

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it. Do not select **Virtual networks (classic)** if it appears in the list, as you cannot create a peering from a virtual network deployed through the classic deployment model.
2. Select the virtual network in the list that you want to delete a peering for.
3. Under **SETTINGS**, select **Peerings**.
4. On the right side of the peering you want to delete, select ..., select **Delete**, then select **Yes** to delete the peering from the first virtual network.
5. Complete the previous steps to delete the peering from the other virtual network in the peering.

## Commands

- **Azure CLI:** [az network vnet peering delete](#)
- **PowerShell:** [Remove-AzVirtualNetworkPeering](#)

# Requirements and constraints

- You can peer virtual networks in the same region, or different regions. Peering virtual networks in different regions is also referred to as *Global VNet Peering*.

- When creating a global peering, the peered virtual networks can exist in any Azure public cloud region or China cloud regions or Government cloud regions. You cannot peer across clouds. For example, a VNet in Azure public cloud cannot be peered to a VNet in Azure China cloud.
- Resources in one virtual network cannot communicate with the front-end IP address of a Basic internal load balancer in a globally peered virtual network. Support for Basic Load Balancer only exists within the same region. Support for Standard Load Balancer exists for both, VNet Peering and Global VNet Peering. Services that use a Basic load balancer which will not work over Global VNet Peering are documented [here](#).
- You can use remote gateways or allow gateway transit in globally peered virtual networks and locally peered virtual networks.
- The virtual networks can be in the same, or different subscriptions. When you peer virtual networks in different subscriptions, both subscriptions can be associated to the same or different Azure Active Directory tenant. If you don't already have an AD tenant, you can [create one](#). Support for peering across virtual networks from subscriptions associated to different Azure Active Directory tenants is not available in Portal. You can use CLI, PowerShell, or Templates.
- The virtual networks you peer must have non-overlapping IP address spaces.
- You can't add address ranges to, or delete address ranges from a virtual network's address space once a virtual network is peered with another virtual network. To add or remove address ranges, delete the peering, add or remove the address ranges, then re-create the peering. To add address ranges to, or remove address ranges from virtual networks, see [Manage virtual networks](#).
- You can peer two virtual networks deployed through Resource Manager or a virtual network deployed through Resource Manager with a virtual network deployed through the classic deployment model. You cannot peer two virtual networks created through the classic deployment model. If you're not familiar with Azure deployment models, read the [Understand Azure deployment models](#) article. You can use a [VPN Gateway](#) to connect two virtual networks created through the classic deployment model.
- When peering two virtual networks created through Resource Manager, a peering must be configured for each virtual network in the peering. You see one of the following types for peering status:
  - Initiated*: When you create the peering to the second virtual network from the first virtual network, the peering status is *Initiated*.
  - Connected*: When you create the peering from the second virtual network to the first virtual network, its peering status is *Connected*. If you view the peering status for the first virtual network, you see its status changed from *Initiated* to *Connected*. The peering is not successfully established until the peering status for both virtual network peerings is *Connected*.
- When peering a virtual network created through Resource Manager with a virtual network created through the classic deployment model, you only configure a peering for the virtual network deployed through Resource Manager. You cannot configure peering for a virtual network (classic), or between two virtual networks deployed through the classic deployment model. When you create the peering from the virtual network (Resource Manager) to the virtual network (Classic), the peering status is *Updating*, then shortly changes to *Connected*.
- A peering is established between two virtual networks. Peerings are not transitive. If you create peerings between:
  - VirtualNetwork1 & VirtualNetwork2
  - VirtualNetwork2 & VirtualNetwork3

There is no peering between VirtualNetwork1 and VirtualNetwork3 through VirtualNetwork2. If you want to create a virtual network peering between VirtualNetwork1 and VirtualNetwork3, you have to create a peering between VirtualNetwork1 and VirtualNetwork3.

- You can't resolve names in peered virtual networks using default Azure name resolution. To resolve names in other virtual networks, you must use [Azure DNS for private domains](#) or a custom DNS server. To learn how to set up your own DNS server, see [Name resolution using your own DNS server](#).
- Resources in peered virtual networks in the same region can communicate with each other with the same bandwidth and latency as if they were in the same virtual network. Each virtual machine size has its own maximum network bandwidth however. To learn more about maximum network bandwidth for different virtual machine sizes, see [Windows](#) or [Linux](#) virtual machine sizes.
- A virtual network can be peered to another virtual network, and also be connected to another virtual network with an Azure virtual network gateway. When virtual networks are connected through both peering and a gateway, traffic between the virtual networks flows through the peering configuration, rather than the gateway.
- Point-to-Site VPN clients must be downloaded again after virtual network peering has been successfully configured to ensure the new routes are downloaded to the client.
- There is a nominal charge for ingress and egress traffic that utilizes a virtual network peering. For more information, see the [pricing page](#).

## Permissions

The accounts you use to work with virtual network peering must be assigned to the following roles:

- [Network Contributor](#): For a virtual network deployed through Resource Manager.
- [Classic Network Contributor](#): For a virtual network deployed through the classic deployment model.

If your account is not assigned to one of the previous roles, it must be assigned to a [custom role](#) that is assigned the necessary actions from the following table:

ACTION	NAME
Microsoft.Network/virtualNetworks/virtualNetworkPeerings/write	Required to create a peering from virtual network A to virtual network B. Virtual network A must be a virtual network (Resource Manager)
Microsoft.Network/virtualNetworks/peer/action	Required to create a peering from virtual network B (Resource Manager) to virtual network A
Microsoft.ClassicNetwork/virtualNetworks/peer/action	Required to create a peering from virtual network B (classic) to virtual network A
Microsoft.Network/virtualNetworks/virtualNetworkPeerings/read	Read a virtual network peering
Microsoft.Network/virtualNetworks/virtualNetworkPeerings/delete	Delete a virtual network peering

## Next steps

- A virtual network peering is created between virtual networks created through the same, or different deployment models that exist in the same, or different subscriptions. Complete a tutorial for one of the following scenarios:

AZURE DEPLOYMENT MODEL	SUBSCRIPTION
Both Resource Manager	Same
	Different
One Resource Manager, one classic	Same
	Different

- Learn how to create a [hub and spoke network topology](#)
- Create a virtual network peering using [PowerShell](#) or Azure CLI sample scripts, or using Azure [Resource Manager templates](#)
- Create and apply [Azure policy](#) for virtual networks

# Configure a VNet-to-VNet VPN gateway connection using PowerShell

1/10/2020 • 18 minutes to read • [Edit Online](#)

This article helps you connect virtual networks by using the VNet-to-VNet connection type. The virtual networks can be in the same or different regions, and from the same or different subscriptions. When connecting VNets from different subscriptions, the subscriptions do not need to be associated with the same Active Directory tenant.

The steps in this article apply to the Resource Manager deployment model and use PowerShell. You can also create this configuration using a different deployment tool or deployment model by selecting a different option from the following list:

## About connecting VNets

There are multiple ways to connect VNets. The sections below describe different ways to connect virtual networks.

### VNet-to-VNet

Configuring a VNet-to-VNet connection is a good way to easily connect VNets. Connecting a virtual network to another virtual network using the VNet-to-VNet connection type (VNet2VNet) is similar to creating a Site-to-Site IPsec connection to an on-premises location. Both connectivity types use a VPN gateway to provide a secure tunnel using IPsec/IKE, and both function the same way when communicating. The difference between the connection types is the way the local network gateway is configured. When you create a VNet-to-VNet connection, you do not see the local network gateway address space. It is automatically created and populated. If you update the address space for one VNet, the other VNet automatically knows to route to the updated address space. Creating a VNet-to-VNet connection is typically faster and easier than creating a Site-to-Site connection between VNets.

### Site-to-Site (IPsec)

If you are working with a complicated network configuration, you may prefer to connect your VNets using the [Site-to-Site](#) steps, instead the VNet-to-VNet steps. When you use the Site-to-Site steps, you create and configure the local network gateways manually. The local network gateway for each VNet treats the other VNet as a local site. This lets you specify additional address space for the local network gateway in order to route traffic. If the address space for a VNet changes, you need to update the corresponding local network gateway to reflect the change. It does not automatically update.

### VNet peering

You may want to consider connecting your VNets using VNet Peering. VNet peering does not use a VPN gateway and has different constraints. Additionally, [VNet peering pricing](#) is calculated differently than [VNet-to-VNet VPN Gateway pricing](#). For more information, see [VNet peering](#).

## Why create a VNet-to-VNet connection?

You may want to connect virtual networks using a VNet-to-VNet connection for the following reasons:

- **Cross region geo-redundancy and geo-presence**

- You can set up your own geo-replication or synchronization with secure connectivity without going over Internet-facing endpoints.
- With Azure Traffic Manager and Load Balancer, you can set up highly available workload with geo-redundancy across multiple Azure regions. One important example is to set up SQL Always On with

Availability Groups spreading across multiple Azure regions.

- **Regional multi-tier applications with isolation or administrative boundary**

- Within the same region, you can set up multi-tier applications with multiple virtual networks connected together due to isolation or administrative requirements.

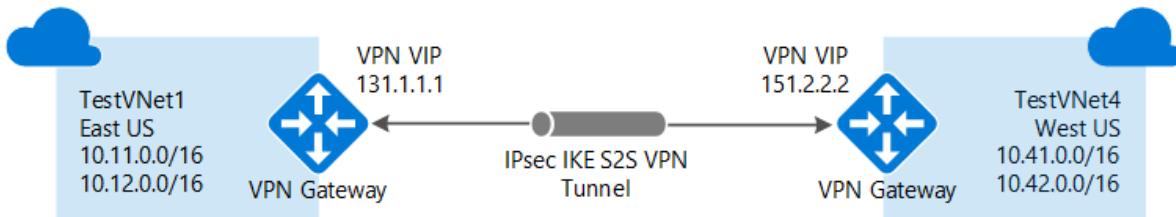
VNet-to-VNet communication can be combined with multi-site configurations. This lets you establish network topologies that combine cross-premises connectivity with inter-virtual network connectivity.

## Which VNet-to-VNet steps should I use?

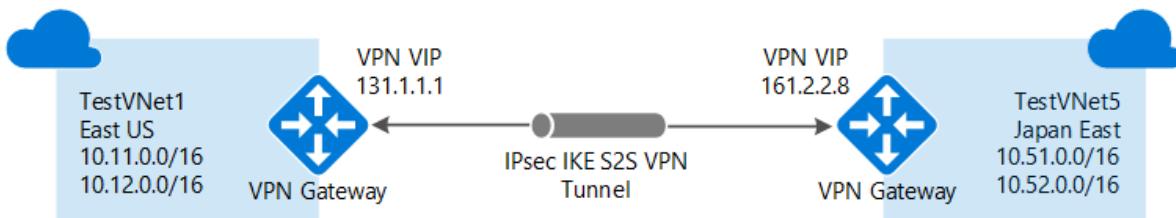
In this article, you see two different sets of steps. One set of steps for [VNets that reside in the same subscription](#) and one for [VNets that reside in different subscriptions](#). The key difference between the sets is that you must use separate PowerShell sessions when configuring the connections for VNets that reside in different subscriptions.

For this exercise, you can combine configurations, or just choose the one that you want to work with. All of the configurations use the VNet-to-VNet connection type. Network traffic flows between the VNets that are directly connected to each other. In this exercise, traffic from TestVNet4 does not route to TestVNet5.

- [VNets that reside in the same subscription](#): The steps for this configuration use TestVNet1 and TestVNet4.



- [VNets that reside in different subscriptions](#): The steps for this configuration use TestVNet1 and TestVNet5.



## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

- 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](#).

### 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](#).

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.

```
Connect-AzAccount
```

Check the subscriptions for the account.

```
Get-AzSubscription
```

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

```
Select-AzSubscription -SubscriptionName nameofsubscription
```

2. 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.

```
$RG1 = "TestRG1"
$Location1 = "East US"
$VNetName1 = "TestVNet1"
$FESubName1 = "FrontEnd"
$BESubName1 = "Backend"
$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"
```

3. Create a resource group.

```
New-AzResourceGroup -Name $RG1 -Location $Location1
```

4. 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. For this reason, it is not assigned via variable below.

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.

```
$fesub1 = New-AzVirtualNetworkSubnetConfig -Name $FESubName1 -AddressPrefix $FESubPrefix1
$besub1 = New-AzVirtualNetworkSubnetConfig -Name $BESubName1 -AddressPrefix $BESubPrefix1
$gwsb1 = New-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -AddressPrefix $GWSubPrefix1
```

5. Create TestVNet1.

```
New-AzVirtualNetwork -Name $VNetName1 -ResourceGroupName $RG1 ` 
-Location $Location1 -AddressPrefix $VNetPrefix11,$VNetPrefix12 -Subnet $fesub1,$besub1,$gwsb1
```

6. 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.

```
$gwpip1 = New-AzPublicIpAddress -Name $GWIPName1 -ResourceGroupName $RG1 ` 
-Location $Location1 -AllocationMethod Dynamic
```

7. 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.

```
$vnet1 = Get-AzVirtualNetwork -Name $VNetName1 -ResourceGroupName $RG1  
$subnet1 = Get-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet1  
$gwipconf1 = New-AzVirtualNetworkGatewayIpConfig -Name $GWIPconfName1 `  
-Subnet $subnet1 -PublicIpAddress $gwipip1
```

8. 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.

```
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.

```
$RG4 = "TestRG4"  
$Location4 = "West US"  
$VnetName4 = "TestVNet4"  
$FESubName4 = "FrontEnd"  
$BESubName4 = "Backend"  
$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"
```

2. Create a resource group.

```
New-AzResourceGroup -Name $RG4 -Location $Location4
```

3. Create the subnet configurations for TestVNet4.

```
$fesub4 = New-AzVirtualNetworkSubnetConfig -Name $FESubName4 -AddressPrefix $FESubPrefix4  
$besub4 = New-AzVirtualNetworkSubnetConfig -Name $BESubName4 -AddressPrefix $BESubPrefix4  
$gwsub4 = New-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -AddressPrefix $GWSubPrefix4
```

4. Create TestVNet4.

```
New-AzVirtualNetwork -Name $VnetName4 -ResourceGroupName $RG4 `  
-Location $Location4 -AddressPrefix $VnetPrefix41,$VnetPrefix42 -Subnet $fesub4,$besub4,$gwsb4
```

5. Request a public IP address.

```
$gwpip4 = New-AzPublicIpAddress -Name $GWIPName4 -ResourceGroupName $RG4 `  
-Location $Location4 -AllocationMethod Dynamic
```

6. Create the gateway configuration.

```
$vnet4 = Get-AzVirtualNetwork -Name $VnetName4 -ResourceGroupName $RG4  
$subnet4 = Get-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet4  
$gwpip4 = New-AzVirtualNetworkGatewayIpConfig -Name $GWIPconfName4 -Subnet $subnet4 -PublicIpAddress  
$gwpip4
```

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

```
New-AzVirtualNetworkGateway -Name $GWName4 -ResourceGroupName $RG4 `  
-Location $Location4 -IpConfigurations $gwpip4 -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.

```
$vnet1gw = Get-AzVirtualNetworkGateway -Name $GWName1 -ResourceGroupName $RG1  
$vnet4gw = Get-AzVirtualNetworkGateway -Name $GWName4 -ResourceGroupName $RG4
```

2. 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.

```
New-AzVirtualNetworkGatewayConnection -Name $Connection14 -ResourceGroupName $RG1 `  
-VirtualNetworkGateway1 $vnet1gw -VirtualNetworkGateway2 $vnet4gw -Location $Location1 `  
-ConnectionType Vnet2Vnet -SharedKey 'AzureA1b2C3'
```

3. 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.

```
New-AzVirtualNetworkGatewayConnection -Name $Connection41 -ResourceGroupName $RG4 `  
-VirtualNetworkGateway1 $vnet4gw -VirtualNetworkGateway2 $vnet1gw -Location $Location4 `  
-ConnectionType Vnet2Vnet -SharedKey 'AzureA1b2C3'
```

4. Verify your connection. See the section [How to verify your connection](#).

## How to connect VNets that are in different subscriptions

In this scenario, you connect TestVNet1 and TestVNet5. TestVNet1 and TestVNet5 reside in different subscriptions. The subscriptions do not need to be associated with the same Active Directory tenant.

The difference between these steps and the previous set is that some of the configuration steps need to be performed in a separate PowerShell session in the context of the second subscription. Especially when the two subscriptions belong to different organizations.

Due to changing subscription context in this exercise, you may find it easier to use PowerShell locally on your computer, rather than using the Azure Cloud Shell, when you get to Step 8.

### **Step 5 - Create and configure TestVNet1**

You must complete [Step 1](#) and [Step 2](#) from the previous section to create and configure TestVNet1 and the VPN Gateway for TestVNet1. For this configuration, you are not required to create TestVNet4 from the previous section, although if you do create it, it will not conflict with these steps. Once you complete Step 1 and Step 2, continue with Step 6 to create TestVNet5.

### **Step 6 - Verify the IP address ranges**

It is important to make sure that the IP address space of the new virtual network, TestVNet5, does not overlap with any of your VNet ranges or local network gateway ranges. In this example, the virtual networks may belong to different organizations. For this exercise, you can use the following values for the TestVNet5:

#### **Values for TestVNet5:**

- VNet Name: TestVNet5
- Resource Group: TestRG5
- Location: Japan East
- TestVNet5: 10.51.0.0/16 & 10.52.0.0/16
- FrontEnd: 10.51.0.0/24
- BackEnd: 10.52.0.0/24
- GatewaySubnet: 10.52.255.0.0/27
- GatewayName: VNet5GW
- Public IP: VNet5GWIP
- VPNTYPE: RouteBased
- Connection: VNet5toVNet1
- ConnectionType: VNet2VNet

### **Step 7 - Create and configure TestVNet5**

This step must be done in the context of the new subscription. This part may be performed by the administrator in a different organization that owns the subscription.

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

```
$Sub5 = "Replace_With_the_New_Subscription_Name"
$RG5 = "TestRG5"
$Location5 = "Japan East"
$VnetName5 = "TestVNet5"
$FESubName5 = "FrontEnd"
$BESubName5 = "Backend"
$GWSubName5 = "GatewaySubnet"
$VnetPrefix51 = "10.51.0.0/16"
$VnetPrefix52 = "10.52.0.0/16"
$FESubPrefix5 = "10.51.0.0/24"
$BESubPrefix5 = "10.52.0.0/24"
$GWSubPrefix5 = "10.52.255.0/27"
$GWName5 = "VNet5GW"
$GWIPName5 = "VNet5GWIP"
$GWIPconfName5 = "gwipconf5"
$Connection51 = "VNet5toVNet1"
```

2. Connect to subscription 5. Open your PowerShell console and connect to your account. Use the following sample to help you connect:

```
Connect-AzAccount
```

Check the subscriptions for the account.

```
Get-AzSubscription
```

Specify the subscription that you want to use.

```
Select-AzSubscription -SubscriptionName $Sub5
```

3. Create a new resource group.

```
New-AzResourceGroup -Name $RG5 -Location $Location5
```

4. Create the subnet configurations for TestVNet5.

```
$fesub5 = New-AzVirtualNetworkSubnetConfig -Name $FESubName5 -AddressPrefix $FESubPrefix5
$besub5 = New-AzVirtualNetworkSubnetConfig -Name $BESubName5 -AddressPrefix $BESubPrefix5
$gwsub5 = New-AzVirtualNetworkSubnetConfig -Name $GWSubName5 -AddressPrefix $GWSubPrefix5
```

5. Create TestVNet5.

```
New-AzVirtualNetwork -Name $VnetName5 -ResourceGroupName $RG5 -Location $Location5 ` 
-AddressPrefix $VnetPrefix51,$VnetPrefix52 -Subnet $fesub5,$besub5,$gwsub5
```

6. Request a public IP address.

```
$gwpip5 = New-AzPublicIpAddress -Name $GWIPName5 -ResourceGroupName $RG5 ` 
-Location $Location5 -AllocationMethod Dynamic
```

7. Create the gateway configuration.

```
$vnet5 = Get-AzVirtualNetwork -Name $VnetName5 -ResourceGroupName $RG5
$subnet5 = Get-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet5
$gwipconf5 = New-AzVirtualNetworkGatewayIpConfig -Name $GWIPconfName5 -Subnet $subnet5 -PublicIpAddress
$gwip5
```

## 8. Create the TestVNet5 gateway.

```
New-AzVirtualNetworkGateway -Name $GWName5 -ResourceGroupName $RG5 -Location $Location5 ` -IpConfigurations $gwipconf5 -GatewayType Vpn -VpnType RouteBased -GatewaySku VpnGw1
```

## Step 8 - Create the connections

In this example, because the gateways are in the different subscriptions, we've split this step into two PowerShell sessions marked as [Subscription 1] and [Subscription 5].

1. **[Subscription 1]** Get the virtual network gateway for Subscription 1. Sign in and connect to Subscription 1 before running the following example:

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

Copy the output of the following elements and send these to the administrator of Subscription 5 via email or another method.

```
$vnet1gw.Name  
$vnet1gw.Id
```

These two elements will have values similar to the following example output:

```
PS D:\> $vnet1gw.Name  
VNet1GW  
PS D:\> $vnet1gw.Id  
/subscriptions/b636ca99-6f88-4df4-a7c3-  
2f8dc4545509/resourceGroupsTestRG1/providers/Microsoft.Network/virtualNetworkGateways/VNet1GW
```

2. **[Subscription 5]** Get the virtual network gateway for Subscription 5. Sign in and connect to Subscription 5 before running the following example:

```
$vnet5gw = Get-AzVirtualNetworkGateway -Name $GWName5 -ResourceGroupName $RG5
```

Copy the output of the following elements and send these to the administrator of Subscription 1 via email or another method.

```
$vnet5gw.Name  
$vnet5gw.Id
```

These two elements will have values similar to the following example output:

```
PS C:\> $vnet5gw.Name  
VNet5GW  
PS C:\> $vnet5gw.Id  
/subscriptions/66c8e4f1-ecd6-47ed-9de7-  
7e530de23994/resourceGroups/TestRG5/providers/Microsoft.Network/virtualNetworkGateways/VNet5GW
```

3. **[Subscription 1]** Create the TestVNet1 to TestVNet5 connection. In this step, you create the connection from TestVNet1 to TestVNet5. The difference here is that \$vnet5gw cannot be obtained directly because it is in a different subscription. You will need to create a new PowerShell object with the values communicated from Subscription 1 in the steps above. Use the example below. Replace the Name, Id, and shared key with your own values. The important thing is that the shared key must match for both connections. Creating a connection can take a short while to complete.

Connect to Subscription 1 before running the following example:

```
$vnet5gw = New-Object -TypeName Microsoft.Azure.Commands.Network.Models.PSVirtualNetworkGateway
$vnet5gw.Name = "VNet5GW"
$vnet5gw.Id = "/subscriptions/66c8e4f1-ecd6-47ed-9de7-
7e530de23994/resourceGroups/TestRG5/providers/Microsoft.Network/virtualNetworkGateways/VNet5GW"
$Connection15 = "VNet1toVNet5"
New-AzVirtualNetworkGatewayConnection -Name $Connection15 -ResourceGroupName $RG1 -
VirtualNetworkGateway1 $vnet1gw -VirtualNetworkGateway2 $vnet5gw -Location $Location1 -ConnectionType
Vnet2Vnet -SharedKey 'AzureA1b2C3'
```

4. **[Subscription 5]** Create the TestVNet5 to TestVNet1 connection. This step is similar to the one above, except you are creating the connection from TestVNet5 to TestVNet1. The same process of creating a PowerShell object based on the values obtained from Subscription 1 applies here as well. In this step, be sure that the shared keys match.

Connect to Subscription 5 before running the following example:

```
$vnet1gw = New-Object -TypeName Microsoft.Azure.Commands.Network.Models.PSVirtualNetworkGateway
$vnet1gw.Name = "VNet1GW"
$vnet1gw.Id = "/subscriptions/b636ca99-6f88-4df4-a7c3-
2f8dc4545509/resourceGroups/TestRG1/providers/Microsoft.Network/virtualNetworkGateways/VNet1GW "
$Connection51 = "VNet5toVNet1"
New-AzVirtualNetworkGatewayConnection -Name $Connection51 -ResourceGroupName $RG5 -
VirtualNetworkGateway1 $vnet5gw -VirtualNetworkGateway2 $vnet1gw -Location $Location5 -ConnectionType
Vnet2Vnet -SharedKey 'AzureA1b2C3'
```

## 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 Virtual Network gateway(VPN, Express Route gateway) to stop functioning as expected. For more information about network security groups, see [What is a network security group?](#)

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.

```
Get-AzVirtualNetworkGatewayConnection -Name VNet1toSite1 -ResourceGroupName TestRG1
```

2. 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.

```
"connectionStatus": "Connected",
"ingressBytesTransferred": 33509044,
"egressBytesTransferred": 4142431
```

## VNet-to-VNet FAQ

The VNet-to-VNet FAQ applies to VPN gateway connections. For information about VNet peering, see [Virtual network peering](#).

### **Does Azure charge for traffic between VNets?**

VNet-to-VNet traffic within the same region is free for both directions when you use a VPN gateway connection. Cross-region VNet-to-VNet egress traffic is charged with the outbound inter-VNet data transfer rates based on the source regions. For more information, see [VPN Gateway pricing page](#). If you're connecting your VNets by using VNet peering instead of a VPN gateway, see [Virtual network pricing](#).

### **Does VNet-to-VNet traffic travel across the internet?**

No. VNet-to-VNet traffic travels across the Microsoft Azure backbone, not the internet.

### **Can I establish a VNet-to-VNet connection across Azure Active Directory (AAD) tenants?**

Yes, VNet-to-VNet connections that use Azure VPN gateways work across AAD tenants.

### **Is VNet-to-VNet traffic secure?**

Yes, it's protected by IPsec/IKE encryption.

### **Do I need a VPN device to connect VNets together?**

No. Connecting multiple Azure virtual networks together doesn't require a VPN device unless cross-premises connectivity is required.

### **Do my VNets need to be in the same region?**

No. The virtual networks can be in the same or different Azure regions (locations).

### **If the VNets aren't in the same subscription, do the subscriptions need to be associated with the same Active Directory tenant?**

No.

### **Can I use VNet-to-VNet to connect virtual networks in separate Azure instances?**

No. VNet-to-VNet supports connecting virtual networks within the same Azure instance. For example, you can't create a connection between global Azure and Chinese/German/US government Azure instances. Consider using a Site-to-Site VPN connection for these scenarios.

### **Can I use VNet-to-VNet along with multi-site connections?**

Yes. Virtual network connectivity can be used simultaneously with multi-site VPNs.

### **How many on-premises sites and virtual networks can one virtual network connect to?**

See the [Gateway requirements](#) table.

### **Can I use VNet-to-VNet to connect VMs or cloud services outside of a VNet?**

No. VNet-to-VNet supports connecting virtual networks. It doesn't support connecting virtual machines or cloud services that aren't in a virtual network.

### **Can a cloud service or a load-balancing endpoint span VNets?**

No. A cloud service or a load-balancing endpoint can't span across virtual networks, even if they're connected together.

### **Can I use a PolicyBased VPN type for VNet-to-VNet or Multi-Site connections?**

No. VNet-to-VNet and Multi-Site connections require Azure VPN gateways with RouteBased (previously called dynamic routing) VPN types.

**Can I connect a VNet with a RouteBased VPN Type to another VNet with a PolicyBased VPN type?**

No, both virtual networks MUST use route-based (previously called dynamic routing) VPNs.

**Do VPN tunnels share bandwidth?**

Yes. All VPN tunnels of the virtual network share the available bandwidth on the Azure VPN gateway and the same VPN gateway uptime SLA in Azure.

**Are redundant tunnels supported?**

Redundant tunnels between a pair of virtual networks are supported when one virtual network gateway is configured as active-active.

**Can I have overlapping address spaces for VNet-to-VNet configurations?**

No. You can't have overlapping IP address ranges.

**Can there be overlapping address spaces among connected virtual networks and on-premises local sites?**

No. You can't have overlapping IP address ranges.

## Next steps

- Once your connection is complete, you can add virtual machines to your virtual networks. See the [Virtual Machines documentation](#) for more information.
- For information about BGP, see the [BGP Overview](#) and [How to configure BGP](#).

# Connect virtual networks from different deployment models using the portal

2/11/2020 • 21 minutes to read • [Edit Online](#)

This article shows you how to connect classic VNets to Resource Manager VNets to allow the resources located in the separate deployment models to communicate with each other. The steps in this article primarily use the Azure portal, but you can also create this configuration using the PowerShell by selecting the article from this list.

Connecting a classic VNet to a Resource Manager VNet is similar to connecting a VNet to an on-premises site location. Both connectivity types use a VPN gateway to provide a secure tunnel using IPsec/IKE. You can create a connection between VNets that are in different subscriptions and in different regions. You can also connect VNets that already have connections to on-premises networks, as long as the gateway that they have been configured with is dynamic or route-based. For more information about VNet-to-VNet connections, see the [VNet-to-VNet FAQ](#) at the end of this article.

If you do not already have a virtual network gateway and do not want to create one, you may want to instead consider connecting your VNets using VNet Peering. VNet peering does not use a VPN gateway. For more information, see [VNet peering](#).

## Before you begin

- These steps assume that both VNets have already been created. If you are using this article as an exercise and don't have VNets, there are links in the steps to help you create them.
- Verify that the address ranges for the VNets do not overlap with each other, or overlap with any of the ranges for other connections that the gateways may be connected to.
- Install the latest PowerShell cmdlets for both Resource Manager and Service Management (classic). In this article, we use both the Azure portal and PowerShell. PowerShell is required to create the connection from the classic VNet to the Resource Manager VNet. For more information, see [How to install and configure Azure PowerShell](#).

## Example settings

You can use these values to create a test environment, or refer to them to better understand the examples in this article.

### Classic VNet

VNet name = ClassicVNet  
Address space = 10.0.0.0/24  
Subnet name = Subnet-1  
Subnet address range = 10.0.0.0/27  
Subscription = the subscription you want to use  
Resource Group = ClassicRG  
Location = West US  
GatewaySubnet = 10.0.0.32/28  
Local site = RMVNetLocal

### Resource Manager VNet

VNet name = RMVNet  
Address space = 192.168.0.0/16  
Resource Group = RG1

Location = East US  
 Subnet name = Subnet-1  
 Address range = 192.168.1.0/24  
 GatewaySubnet = 192.168.0.0/26  
 Virtual network gateway name = RMGateway  
 Gateway type = VPN  
 VPN type = Route-based  
 SKU = VpnGw1  
 Location = East US  
 Virtual network = RMVNet  
 (associate the VPN gateway to this VNet) First IP configuration = rmgwpip  
 (gateway public IP address) Local network gateway = ClassicVNetLocal  
 Connection name = RMtoClassic

### Connection overview

For this configuration, you create a VPN gateway connection over an IPsec/IKE VPN tunnel between the virtual networks. Make sure that none of your VNet ranges overlap with each other, or with any of the local networks that they connect to.

The following table shows an example of how the example VNets and local sites are defined:

VIRTUAL NETWORK	ADDRESS SPACE	REGION	CONNECTS TO LOCAL NETWORK SITE
ClassicVNet	(10.0.0.0/24)	West US	RMVNetLocal (192.168.0.0/16)
RMVNet	(192.168.0.0/16)	East US	ClassicVNetLocal (10.0.0.0/24)

## Section 1 - Configure the classic VNet settings

In this section, you create the classic VNet, the local network (local site), and the virtual network gateway. Screenshots are provided as examples. Be sure to replace the values with your own, or use the [Example](#) values.

### 1. Create a classic VNet

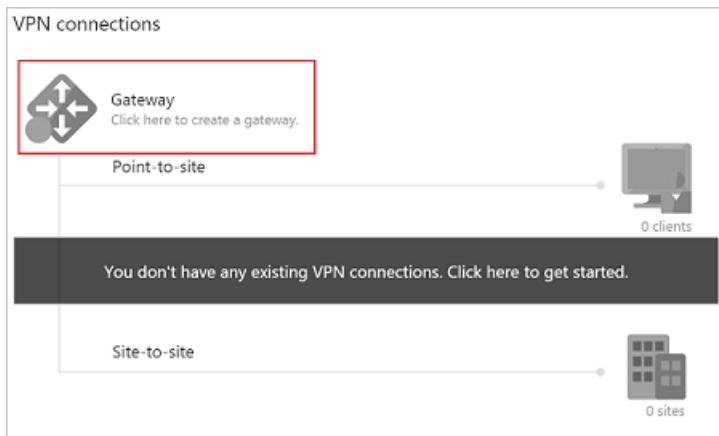
If you don't have a classic VNet and are running these steps as an exercise, you can create a VNet by using [this article](#) and the [Example](#) settings values from above.

If you already have a VNet with a VPN gateway, verify that the gateway is Dynamic. If it's Static, you must first delete the VPN gateway before you proceed to [Configure the local site](#).

1. Open the [Azure portal](#) and sign in with your Azure account.
2. Click + **Create a resource** to open the 'New' page.
3. In the 'Search the marketplace' field, type 'Virtual Network'. If you instead, select Networking -> Virtual Network, you will not get the option to create a classic VNet.
4. Locate 'Virtual Network' from the returned list and click it to open the Virtual Network page.
5. On the virtual network page, select 'Classic' to create a classic VNet. If you take the default here, you will wind up with a Resource Manager VNet instead.

### 2. Configure the local site

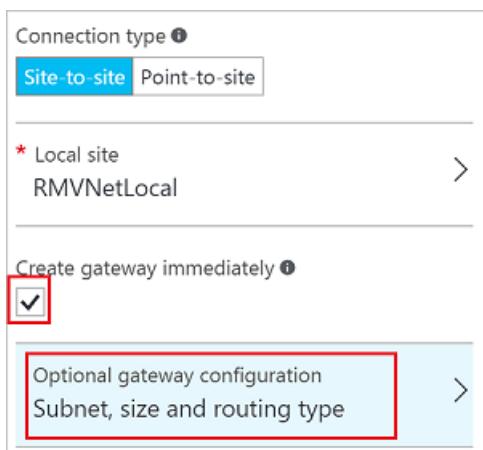
1. Navigate to **All resources** and locate the **ClassicVNet** in the list.
2. Click **Gateway** in the **Settings** section of the menu, and then click on the banner to create a gateway.



3. On the **New VPN Connection** page, for **Connection type**, select **Site-to-site**.
4. For **Local site**, click **Configure required settings**. This opens the **Local site** page.
5. On the **Local site** page, create a name to refer to the Resource Manager VNet. For example, 'RMVNetLocal'.
6. If the VPN gateway for the Resource Manager VNet already has a Public IP address, use the value for the **VPN gateway IP address** field. If you are doing these steps as an exercise, or don't yet have a virtual network gateway for your Resource Manager VNet, you can make up a placeholder IP address. Make sure that the placeholder IP address uses a valid format. Later, you replace the placeholder IP address with the Public IP address of the Resource Manager virtual network gateway.
7. For **Client Address Space**, use the **values** for the virtual network IP address spaces for the Resource Manager VNet. This setting is used to specify the address spaces to route to the Resource Manager virtual network. In the example, we use 192.168.0.0/16, the address range for the RMVNet.
8. Click **OK** to save the values and return to the **New VPN Connection** page.

### 3. Create the virtual network gateway

1. On the **New VPN Connection** page, select the **Create gateway immediately** checkbox.
2. Click **Optional gateway configuration** to open the **Gateway configuration** page.



3. Click **Subnet - Configure required settings** to open the **Add subnet** page. The **Name** is already configured with the required value: **GatewaySubnet**.
4. The **Address range** refers to the range for the gateway subnet. Although you can create a gateway subnet with a /29 address range (3 addresses), we recommend creating a gateway subnet that contains more IP addresses. This will accommodate future configurations that may require more available IP addresses. If possible, use /27 or /28. If you are using these steps as an exercise, you can refer to the **Example values**. For this example, we use '10.0.0.32/28'. Click **OK** to create the gateway subnet.
5. On the **Gateway configuration** page, **Size** refers to the gateway SKU. Select the gateway SKU for your VPN gateway.
6. Verify the **Routing Type is Dynamic**, then click **OK** to return to the **New VPN Connection** page.

7. On the **New VPN Connection** page, click **OK** to begin creating your VPN gateway. Creating a VPN gateway can take up to 45 minutes to complete.

#### 4. Copy the virtual network gateway Public IP address

After the virtual network gateway has been created, you can view the gateway IP address.

1. Navigate to your classic VNet, and click **Overview**.
2. Click **VPN connections** to open the VPN connections page. On the VPN connections page, you can view the Public IP address. This is the Public IP address assigned to your virtual network gateway. Make a note of the IP address. You use it in later steps when you work with your Resource Manager local network gateway configuration settings.
3. You can view the status of your gateway connections. Notice the local network site you created is listed as 'Connecting'. The status will change after you have created your connections. You can close this page when you are finished viewing the status.

## Section 2 - Configure the Resource Manager VNet settings

In this section, you create the virtual network gateway and the local network gateway for your Resource Manager VNet. Screenshots are provided as examples. Be sure to replace the values with your own, or use the [Example](#) values.

### 1. Create a virtual network

#### Example values:

- VNet name = RMVNet
- Address space = 192.168.0.0/16
- Resource Group = RG1
- Location = East US
- Subnet name = Subnet-1
- Address range = 192.168.1.0/24

If you don't have a Resource Manager VNet and are running these steps as an exercise, create a virtual network with the steps in [Create a virtual network](#), using the example values.

### 2. Create a virtual network gateway

In this step, you create the virtual network gateway for your VNet. Creating a gateway can often take 45 minutes or more, depending on the selected gateway SKU.

The virtual network gateway uses specific subnet called the gateway subnet. The gateway subnet is part of the virtual network IP address range that you specify when configuring your virtual network. It contains the IP addresses that the virtual network gateway resources and services use.

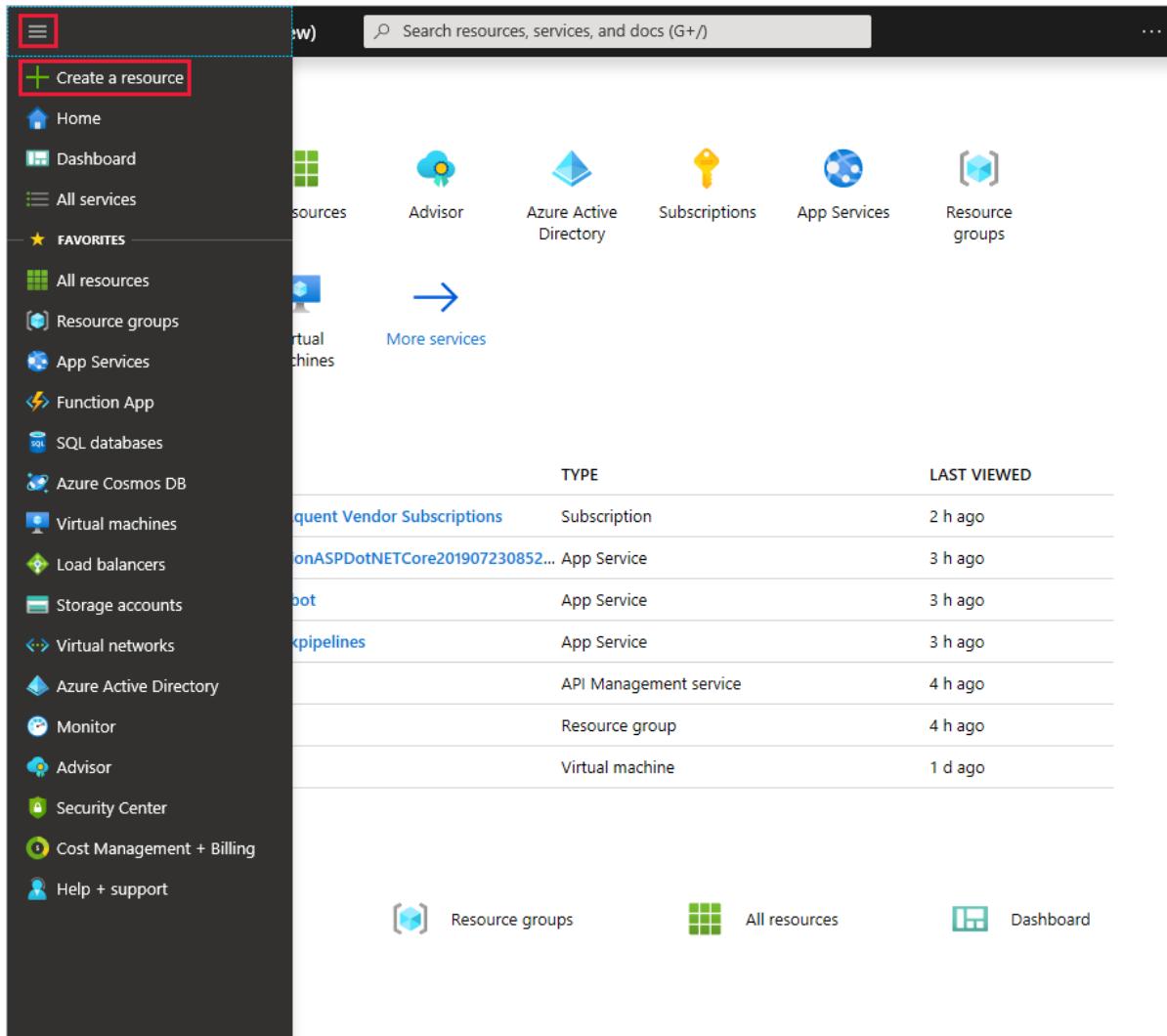
When you create the gateway subnet, you specify the number of IP addresses that the subnet contains. The number of IP addresses needed depends on the VPN gateway configuration that you want to create. Some configurations require more IP addresses than others. We recommend that you create a gateway subnet that uses a /27 or /28.

If you see an error that specifies that the address space overlaps with a subnet, or that the subnet is not contained within the address space for your virtual network, check your VNet address range. You may not have enough IP addresses available in the address range you created for your virtual network. For example, if your default subnet encompasses the entire address range, there are no IP addresses left to create additional subnets. You can either adjust your subnets within the existing address space to free up IP addresses, or specify an additional address range and create the gateway subnet there.

#### Example values:

- Virtual network gateway name = RMGateway
- Gateway type = VPN
- VPN type = Route-based
- SKU = VpnGw1
- Location = East US
- Virtual network = RMVNet
- GatewaySubnet = 192.168.0.0/26
- First IP configuration = rmgwpip

1. From the [Azure portal](#) menu, select **Create a resource**.



The screenshot shows the Azure portal interface. On the left, a dark sidebar lists various services like Home, Dashboard, and Resource groups. A red box highlights the '+ Create a resource' button at the top of this sidebar. The main area has a light gray background. At the top right is a search bar with placeholder text 'Search resources, services, and docs (G+/)'. Below the search bar are several quick access icons: Advisor, Azure Active Directory, Subscriptions, App Services, and Resource groups. A large blue arrow points from the sidebar towards a 'More services' link. To the right of the arrow is a table titled 'LAST VIEWED' showing recent resources. The table has columns for 'TYPE' and 'LAST VIEWED'. The resources listed are: 'Recent Vendor Subscriptions' (Subscription, 2 h ago), 'onASPDotNETCore201907230852...' (App Service, 3 h ago), 'bot' (App Service, 3 h ago), 'pipelines' (App Service, 3 h ago), 'Azure Active Directory' (API Management service, 4 h ago), 'Monitor' (Resource group, 4 h ago), and 'Virtual machine' (Virtual machine, 1 d ago). At the bottom of the main area are three navigation links: 'Resource groups', 'All resources', and 'Dashboard'.

2. In the **Search the Marketplace** field, type 'Virtual Network Gateway'. Locate **Virtual network gateway** in the search return and click the entry. On the **Virtual network gateway** page, click **Create**. This opens the **Create virtual network gateway** page.

## Create virtual network gateway

Azure has provided a planning and design guide to help you configure the various VPN gateway options. [Learn more.](#)

### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*

Resource group ⓘ

TestRG1 (derived from virtual network's resource group)

### Instance details

Name \*

 VNet1GW ✓

Region \*

 (US) East US ✓

Gateway type \* ⓘ

VPN  ExpressRoute

VPN type \* ⓘ

Route-based  Policy-based

SKU \* ⓘ

 VpnGw1 ✓

Generation ⓘ

 Generation1 ✓

Virtual network \* ⓘ

 VNet1 ✓

[Create virtual network](#)

i Only virtual networks in the currently selected subscription and region are listed.

Gateway subnet address range \* ⓘ

 10.11.255.0/27 ✓

10.11.255.0 - 10.11.255.31 (32 addresses)

### Public IP address

Public IP address \* ⓘ

Create new  Use existing

Public IP address name \*

 VNet1GWpip ✓

Public IP address SKU

Basic

Assignment

Dynamic  Static

Enable active-active mode \* ⓘ

Enabled  Disabled

Configure BGP ASN \* ⓘ

Enabled  Disabled

3. On the **Create virtual network gateway** page, fill in the values for your virtual network gateway.

### Project details

- **Subscription:** Select the subscription you want to use from the dropdown.
- **Resource Group:** This setting is autofilled when you select your virtual network on this page.

### Instance details

- **Name:** Name your gateway. Naming your gateway not the same as naming a gateway subnet. It's the name of the gateway object you are creating.
- **Region:** Select the region in which you want to create this resource. The region for the gateway must be the same as the virtual network.
- **Gateway type:** Select **VPN**. VPN gateways use the virtual network gateway type **VPN**.

- **VPN type:** Select the VPN type that is specified for your configuration. Most configurations require a Route-based VPN type.

- **SKU:** Select the gateway SKU from the dropdown. The SKUs listed in the dropdown depend on the VPN type you select. For more information about gateway SKUs, see [Gateway SKUs](#).

**Virtual network:** Choose the virtual network to which you want to add this gateway.

**Gateway subnet address range:** This field only appears if your VNet doesn't have a gateway subnet. If possible, make the range /27 or larger (/26,/25 etc.). We don't recommend creating a range any smaller than /28. If you already have a gateway subnet, you can view GatewaySubnet details by navigating to your virtual network. Click **Subnets** to view the range. If you want to change the range, you can delete and recreate the GatewaySubnet.

**Public IP address:** This setting specifies the public IP address object that gets associated to the VPN gateway. The public IP address is dynamically assigned to this object when the VPN gateway is created. The only time the Public IP address changes is when the gateway is deleted and re-created. It doesn't change across resizing, resetting, or other internal maintenance/upgrades of your VPN gateway.

- **Public IP address:** Leave **Create new** selected.

- **Public IP address name:** In the text box, type a name for your public IP address instance.

- **Assignment:** VPN gateway supports only Dynamic.

**Active-Active mode:** Only select **Enable active-active mode** if you are creating an active-active gateway configuration. Otherwise, leave this setting unselected.

Leave **Configure BGP ASN** deselected, unless your configuration specifically requires this setting. If you do require this setting, the default ASN is 65515, although this can be changed.

4. Click **Review + create** to run validation. Once validation passes, click **Create** to deploy the VPN gateway. A gateway can take up to 45 minutes to fully create and deploy. You can see the deployment status on the Overview page for your gateway.

After the gateway is created, you can view the IP address that has been assigned to it by looking at the virtual network in the portal. The gateway appears as a connected device.

#### 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 Virtual Network gateway(VPN, Express Route gateway) to stop functioning as expected. For more information about network security groups, see [What is a network security group?](#)

### 3. Create a local network gateway

**Example values:** Local network gateway = ClassicVNetLocal

VIRTUAL NETWORK	ADDRESS SPACE	REGION	CONNECTS TO LOCAL NETWORK SITE	GATEWAY PUBLIC IP ADDRESS
ClassicVNet	(10.0.0.0/24)	West US	RMVNetLocal (192.168.0.0/16)	The Public IP address that is assigned to the ClassicVNet gateway
RMVNet	(192.168.0.0/16)	East US	ClassicVNetLocal (10.0.0.0/24)	The Public IP address that is assigned to the RMVNet gateway.

The local network gateway specifies the address range and the Public IP address associated with your classic VNet

and its virtual network gateway. If you are doing these steps as an exercise, refer to the Example values.

1. In the portal, click **+Create a resource**.
2. In the search box, type **Local network gateway**, then press **Enter** to search. This will return a list of results. Click **Local network gateway**, then click the **Create** button to open the **Create local network gateway** page.

The screenshot shows the 'Create local network gateway' configuration page. It includes fields for Name, IP address, Address space, BGP settings, Subscription, Resource group, Location, and a 'Create' button.

**Name:** A text input field containing a placeholder value, marked with a green checkmark.

**IP address:** A text input field containing a placeholder value, marked with a green checkmark.

**Address space:** A dropdown menu labeled '...' with an 'Add additional address range' button below it.

**BGP settings:** A checkbox labeled 'Configure BGP settings'.

**Subscription:** A dropdown menu.

**Resource group:** A dropdown menu with a 'Create new' link below it.

**Location:** A dropdown menu.

**Create button:** A blue button at the bottom left.

3. On the **Create local network gateway page**, specify the values for your local network gateway.

- **Name:** Specify a name for your local network gateway object.
- **IP address:** This is the public IP address of the VPN device that you want Azure to connect to. Specify a valid public IP address. If you don't have the IP address right now, you can use the values shown in the example, but you'll need to go back and replace your placeholder IP address with the public IP address of your VPN device. Otherwise, Azure will not be able to connect.
- **Address Space** refers to the address ranges for the network that this local network represents. You can

add multiple address space ranges. Make sure that the ranges you specify here do not overlap with ranges of other networks that you want to connect to. Azure will route the address range that you specify to the on-premises VPN device IP address. *Use your own values here if you want to connect to your on-premises site, not the values shown in the example.*

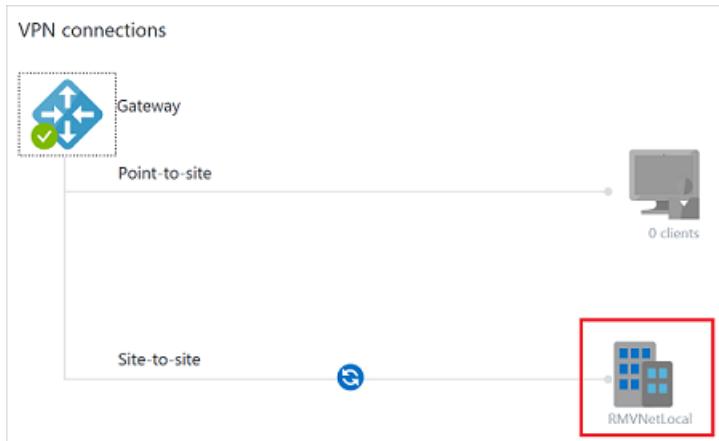
- **Configure BGP settings:** Use only when configuring BGP. Otherwise, don't select this.
- **Subscription:** Verify that the correct subscription is showing.
- **Resource Group:** Select the resource group that you want to use. You can either create a new resource group, or select one that you have already created.
- **Location:** Select the location that this object will be created in. You may want to select the same location that your VNet resides in, but you are not required to do so.

4. When you have finished specifying the values, click the **Create** button to create the gateway.

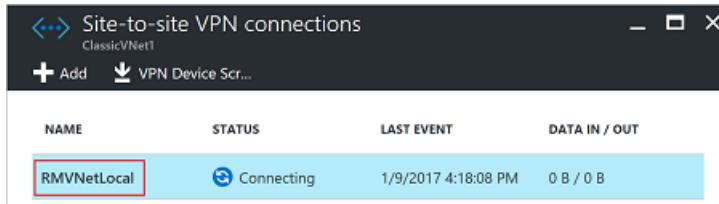
## Section 3 - Modify the classic VNet local site settings

In this section, you replace the placeholder IP address that you used when specifying the local site settings, with the Resource Manager VPN gateway IP address. This section uses the classic (SM) PowerShell cmdlets.

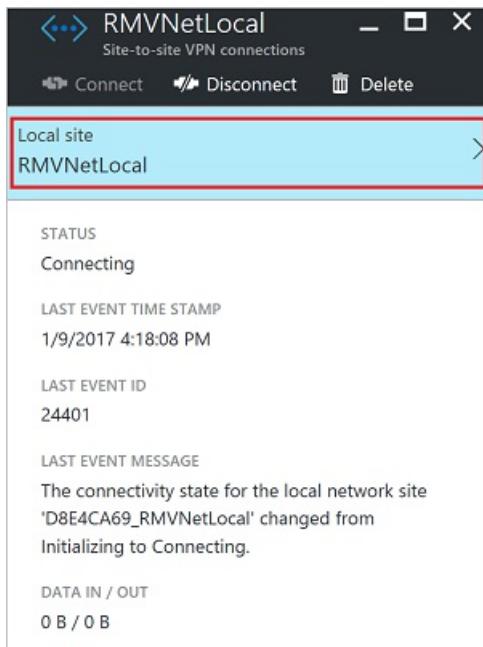
1. In the Azure portal, navigate to the classic virtual network.
2. On the page for your virtual network, click **Overview**.
3. In the **VPN connections** section, click the name of your local site in the graphic.



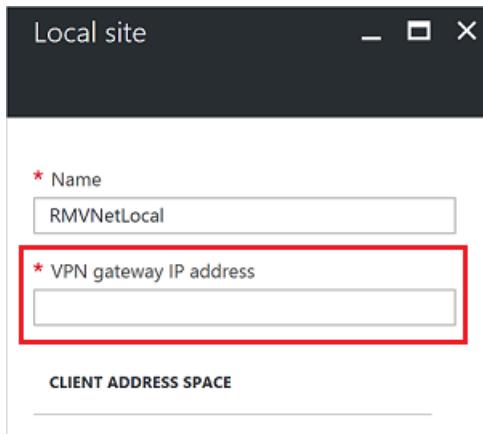
4. On the **Site-to-site VPN connections** page, click the name of the site.



5. On the connection page for your local site, click the name of the local site to open the **Local site** page.



6. On the **Local site** page, replace the **VPN gateway IP address** with the IP address of the Resource Manager gateway.



7. Click **OK** to update the IP address.

## Section 4 - Create Resource Manager to classic connection

In these steps, you configure the connection from the Resource Manager VNet to the classic VNet using the Azure portal.

1. In **All resources**, locate the local network gateway. In our example, the local network gateway is **ClassicVNetLocal**.
2. Click **Configuration** and verify that the IP address value is the VPN gateway for the classic VNet. Update, if needed, then click **Save**. Close the page.
3. In **All resources**, click the local network gateway.
4. Click **Connections** to open the Connections page.
5. On the **Connections** page, click **+** to add a connection.
6. On the **Add connection** page, name the connection. For example, 'RMtoClassic'.
7. **Site-to-Site** is already selected on this page.
8. Select the virtual network gateway that you want to associate with this site.
9. Create a **shared key**. This key is also used in the connection that you create from the classic VNet to the Resource Manager VNet. You can generate the key or make one up. In our example, we use 'abc123', but you can (and should) use something more complex.
10. Click **OK** to create the connection.

## Section 5 - Create classic to Resource Manager connection

In these steps, you configure the connection from the classic VNet to the Resource Manager VNet. These steps require PowerShell. You can't create this connection in the portal. Make sure you have downloaded and installed both the classic (SM) and Resource Manager (RM) PowerShell cmdlets.

### 1. Connect to your Azure account

Open the PowerShell console with elevated rights and log in to your Azure account. After logging in, your account settings are downloaded so that they are available to Azure PowerShell. The following cmdlet prompts you for the login credentials for your Azure Account for the Resource Manager deployment model:

```
Connect-AzAccount
```

Get a list of your Azure subscriptions.

```
Get-AzSubscription
```

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

```
Select-AzSubscription -SubscriptionName "Name of subscription"
```

Next, log in to use the classic PowerShell cmdlets (Service Management). Use the following command to add your Azure account for the classic deployment model:

```
Add-AzureAccount
```

Get a list of your subscriptions. This step may be necessary when adding the Service Management cmdlets, depending on your Azure module install.

```
Get-AzureSubscription
```

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

```
Select-AzureSubscription -SubscriptionName "Name of subscription"
```

### 2. View the network configuration file values

When you create a VNet in the Azure portal, the full name that Azure uses is not visible in the Azure portal. For example, a VNet that appears to be named 'ClassicVNet' in the Azure portal may have a much longer name in the network configuration file. The name might look something like: 'Group ClassicRG ClassicVNet'. In these steps, you download the network configuration file and view the values.

Create a directory on your computer and then export the network configuration file to the directory. In this example, the network configuration file is exported to C:\AzureNet.

```
Get-AzureVNetConfig -ExportToFile C:\AzureNet\NetworkConfig.xml
```

Open the file with a text editor and view the name for your classic VNet. Use the names in the network configuration file when running your PowerShell cmdlets.

- VNet names are listed as **VirtualNetworkSite name** =
- Site names are listed as **LocalNetworkSite name**=

### 3. Create the connection

Set the shared key and create the connection from the classic VNet to the Resource Manager VNet. You cannot set the shared key using the portal. Make sure you run these steps while logged in using the classic version of the PowerShell cmdlets. To do so, use **Add-AzureAccount**. Otherwise, you will not be able to set the '--AzureVNetGatewayKey'.

- In this example, **-VNetName** is the name of the classic VNet as found in your network configuration file.
- The **-LocalNetworkSiteName** is the name you specified for the local site, as found in your network configuration file.
- The **-SharedKey** is a value that you generate and specify. For this example, we used *abc123*, but you can generate something more complex. The important thing is that the value you specify here must be the same value that you specified when creating your Resource Manager to classic connection.

```
Set-AzureVNetGatewayKey -VNetName "Group ClassicRG ClassicVNet" `  
-LocalNetworkSiteName "172B9E16_RMVNetLocal" -SharedKey abc123
```

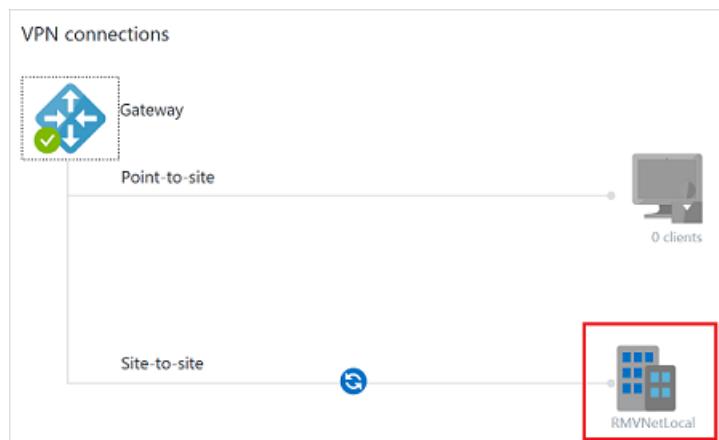
## Section 6 - Verify your connections

You can verify your connections by using the Azure portal or PowerShell. When verifying, you may need to wait a minute or two as the connection is being created. When a connection is successful, the connectivity state changes from 'Connecting' to 'Connected'.

### To verify the connection from your classic VNet to your Resource Manager VNet

In the Azure portal, you can view the connection status for a classic VNet VPN Gateway by navigating to the connection. The following steps show one way to navigate to your connection and verify.

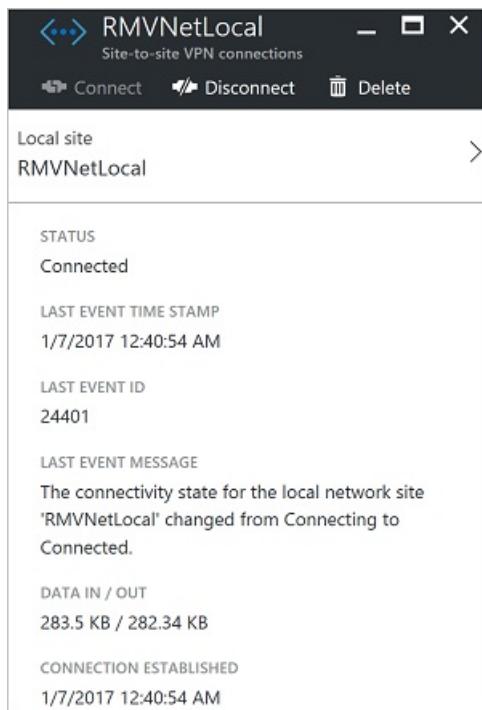
1. In the [Azure portal](#), click **All resources** and navigate to your classic virtual network.
2. On the virtual network blade, click **Overview** to access the **VPN connections** section of the blade.
3. On the VPN connections graphic, click the site.



4. On the **Site-to-site VPN connections** blade, view the information about your site.

NAME	STATUS	LAST EVENT	DATA IN / OUT
RMVNetLocal	Connected	1/7/2017 12:40:54 AM	177.97 KB / 177.72 KB

5. To view more information about the connection, click the name of the connection to open the **Site-to-site VPN Connection** blade.



#### To verify the connection from your Resource Manager VNet to your classic VNet

In the Azure portal, you can view the connection status of a Resource Manager VPN Gateway by navigating to the connection. The following steps show one way to navigate to your connection and verify.

1. In the [Azure portal](#), click **All resources** and navigate to your virtual network gateway.
2. On the blade for your virtual network gateway, click **Connections**. You can see the status of each connection.
3. Click the name of the connection that you want to verify to open **Essentials**. In Essentials, you can view more information about your connection. The **Status** is 'Succeeded' and 'Connected' when you have made a successful connection.

Essentials ^	
Resource group	Data in 2.35 KB
<b>Status</b> <b>Connected</b>	Data out 3.14 KB
Location	Virtual network
East US	
Subscription name	Virtual network gateway
Subscription ID	Local network gateway

## VNet-to-VNet FAQ

The VNet-to-VNet FAQ applies to VPN gateway connections. For information about VNet peering, see [Virtual network peering](#).

#### Does Azure charge for traffic between VNets?

VNet-to-VNet traffic within the same region is free for both directions when you use a VPN gateway connection. Cross-region VNet-to-VNet egress traffic is charged with the outbound inter-VNet data transfer rates based on the source regions. For more information, see [VPN Gateway pricing page](#). If you're connecting your VNets by

using VNet peering instead of a VPN gateway, see [Virtual network pricing](#).

**Does VNet-to-VNet traffic travel across the internet?**

No. VNet-to-VNet traffic travels across the Microsoft Azure backbone, not the internet.

**Can I establish a VNet-to-VNet connection across Azure Active Directory (AAD) tenants?**

Yes, VNet-to-VNet connections that use Azure VPN gateways work across AAD tenants.

**Is VNet-to-VNet traffic secure?**

Yes, it's protected by IPsec/IKE encryption.

**Do I need a VPN device to connect VNets together?**

No. Connecting multiple Azure virtual networks together doesn't require a VPN device unless cross-premises connectivity is required.

**Do my VNets need to be in the same region?**

No. The virtual networks can be in the same or different Azure regions (locations).

**If the VNets aren't in the same subscription, do the subscriptions need to be associated with the same Active Directory tenant?**

No.

**Can I use VNet-to-VNet to connect virtual networks in separate Azure instances?**

No. VNet-to-VNet supports connecting virtual networks within the same Azure instance. For example, you can't create a connection between global Azure and Chinese/German/US government Azure instances. Consider using a Site-to-Site VPN connection for these scenarios.

**Can I use VNet-to-VNet along with multi-site connections?**

Yes. Virtual network connectivity can be used simultaneously with multi-site VPNs.

**How many on-premises sites and virtual networks can one virtual network connect to?**

See the [Gateway requirements](#) table.

**Can I use VNet-to-VNet to connect VMs or cloud services outside of a VNet?**

No. VNet-to-VNet supports connecting virtual networks. It doesn't support connecting virtual machines or cloud services that aren't in a virtual network.

**Can a cloud service or a load-balancing endpoint span VNets?**

No. A cloud service or a load-balancing endpoint can't span across virtual networks, even if they're connected together.

**Can I use a PolicyBased VPN type for VNet-to-VNet or Multi-Site connections?**

No. VNet-to-VNet and Multi-Site connections require Azure VPN gateways with RouteBased (previously called dynamic routing) VPN types.

**Can I connect a VNet with a RouteBased VPN Type to another VNet with a PolicyBased VPN type?**

No, both virtual networks MUST use route-based (previously called dynamic routing) VPNs.

**Do VPN tunnels share bandwidth?**

Yes. All VPN tunnels of the virtual network share the available bandwidth on the Azure VPN gateway and the same VPN gateway uptime SLA in Azure.

**Are redundant tunnels supported?**

Redundant tunnels between a pair of virtual networks are supported when one virtual network gateway is configured as active-active.

**Can I have overlapping address spaces for VNet-to-VNet configurations?**

No. You can't have overlapping IP address ranges.

**Can there be overlapping address spaces among connected virtual networks and on-premises local sites?**

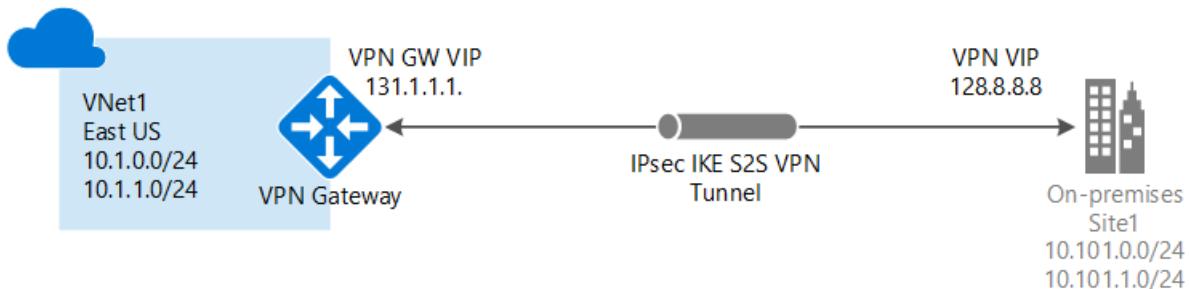
No. You can't have overlapping IP address ranges.

# Create a Site-to-Site connection in the Azure portal

1/10/2020 • 16 minutes to read • [Edit Online](#)

This article shows you how to use the Azure portal to create a Site-to-Site VPN gateway connection from your on-premises network to the VNet. The steps in this article apply to the Resource Manager deployment model. You can also create this configuration using a different deployment tool or deployment model by selecting a different option from the following list:

A Site-to-Site VPN gateway connection is used to connect your on-premises network to an Azure virtual network over an IPsec/IKE (IKEv1 or IKEv2) VPN tunnel. This type of connection requires a VPN device located on-premises that has an externally facing public IP address assigned to it. For more information about VPN gateways, see [About VPN gateway](#).



## Before you begin

Verify that you have met the following criteria before beginning your configuration:

- Make sure you have a compatible VPN device and someone who is able to configure it. For more information about compatible VPN devices and device configuration, see [About VPN Devices](#).
- Verify that you have an externally facing public IPv4 address for your VPN device.
- If you are unfamiliar with the IP address ranges located in your on-premises network configuration, you need to coordinate with someone who can provide those details for you. When you create this configuration, you must specify the IP address range prefixes that Azure will route to your on-premises location. None of the subnets of your on-premises network can overlap with the virtual network subnets that you want to connect to.

## Example values

The examples in this article use the following values. You can use these values to create a test environment, or refer to them to better understand the examples in this article. For more information about VPN Gateway settings in general, see [About VPN Gateway Settings](#).

- **Virtual network name:** VNet1
- **Address Space:** 10.1.0.0/16
- **Subscription:** The subscription you want to use
- **Resource Group:** TestRG1
- **Region:** East US
- **Subnet:** FrontEnd: 10.1.0.0/24, BackEnd: 10.1.1.0/24 (optional for this exercise)
- **Gateway subnet address range:** 10.1.255.0/27
- **Virtual network gateway name:** VNet1GW
- **Public IP address name:** VNet1GWIP
- **VPN type:** Route-based

- **Connection type:** Site-to-site (IPsec)
- **Gateway type:** VPN
- **Local network gateway name:** Site1
- **Connection name:** VNet1toSite1
- **Shared key:** For this example, we use abc123. But, you can use whatever is compatible with your VPN hardware. The important thing is that the values match on both sides of the connection.

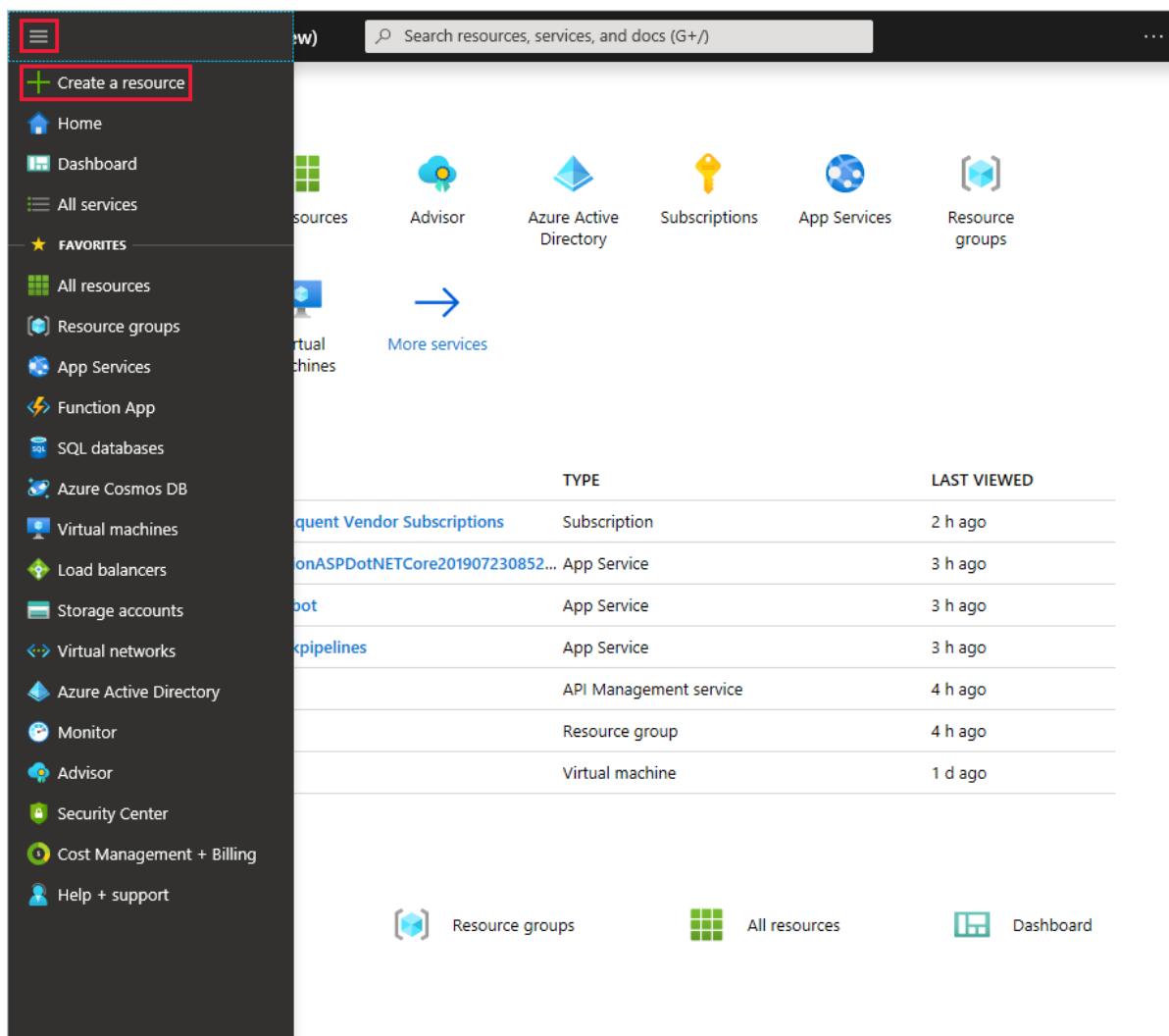
## 1. Create a virtual network

To create a VNet in the Resource Manager deployment model by using the Azure portal, follow the steps below. Use the **Example values** if you are using these steps as a tutorial. If you are not doing these steps as a tutorial, be sure to replace the values with your own. For more information about working with virtual networks, see the [Virtual Network Overview](#).

### NOTE

In order for this VNet to connect to an on-premises location you need to coordinate with your on-premises network administrator to carve out an IP address range that you can use specifically for this virtual network. If a duplicate address range exists on both sides of the VPN connection, traffic does not route the way you may expect it to. Additionally, if you want to connect this VNet to another VNet, the address space cannot overlap with other VNet. Take care to plan your network configuration accordingly.

1. From the [Azure portal](#) menu, select **Create a resource**.



2. In the **Search the marketplace** field, type 'virtual network'. Locate **Virtual network** from the returned list

and click to open the **Virtual Network** page.

3. Click **Create**. This opens the **Create virtual network** page.
4. On the **Create virtual network** page, configure the VNet settings. When you fill in the fields, the red exclamation mark becomes a green check mark when the characters entered in the field are valid. Use the following values:

- **Name:** VNet1
- **Address space:** 10.1.0.0/16
- **Subscription:** Verify that the subscription listed is the one you want to use. You can change subscriptions by using the drop-down.
- **Resource group:** TestRG1 (click **Create new** to create a new group)
- **Location:** East US
- **Subnet:** Frontend
- **Address range:** 10.1.0.0/24

**Create virtual network**

\* Name: VNet1 ✓

\* Address space ⓘ: 10.1.0.0/16 ✓  
10.1.0.0 - 10.1.255.255 (65536 addresses)

\* Subscription: ▾

\* Resource group: (New) TestRG1 ▾  
[Create new](#)

\* Location: (US) East US ▾

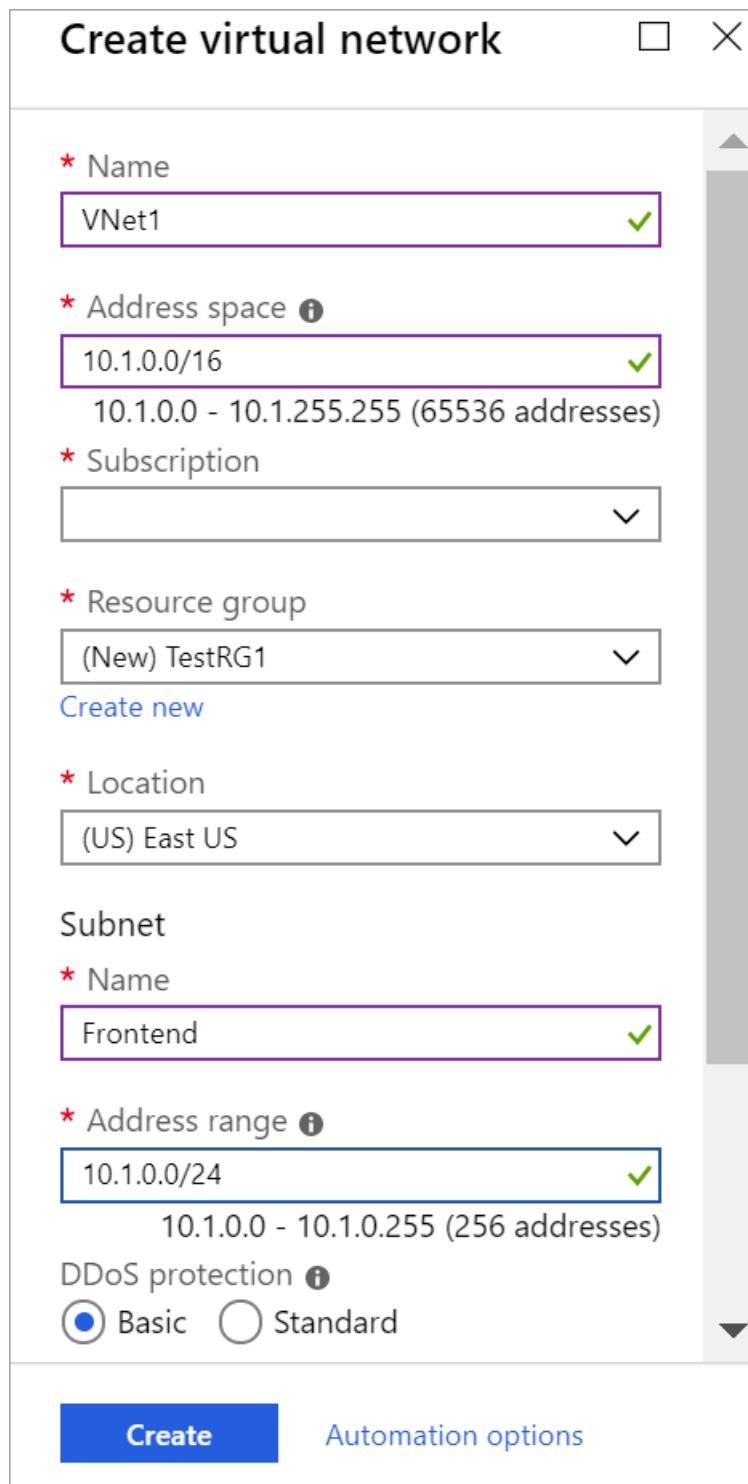
**Subnet**

\* Name: Frontend ✓

\* Address range ⓘ: 10.1.0.0/24 ✓  
10.1.0.0 - 10.1.0.255 (256 addresses)

DDoS protection ⓘ:  
 Basic  Standard ▾

**Create** [Automation options](#)



5. Leave DDoS as Basic, Service endpoints as Disabled, and Firewall as Disabled.

6. Click **Create** to create the VNet.

## 2. Create the VPN gateway

In this step, you create the virtual network gateway for your VNet. Creating a gateway can often take 45 minutes or more, depending on the selected gateway SKU.

The virtual network gateway uses specific subnet called the gateway subnet. The gateway subnet is part of the virtual network IP address range that you specify when configuring your virtual network. It contains the IP addresses that the virtual network gateway resources and services use.

When you create the gateway subnet, you specify the number of IP addresses that the subnet contains. The number of IP addresses needed depends on the VPN gateway configuration that you want to create. Some configurations require more IP addresses than others. We recommend that you create a gateway subnet that uses

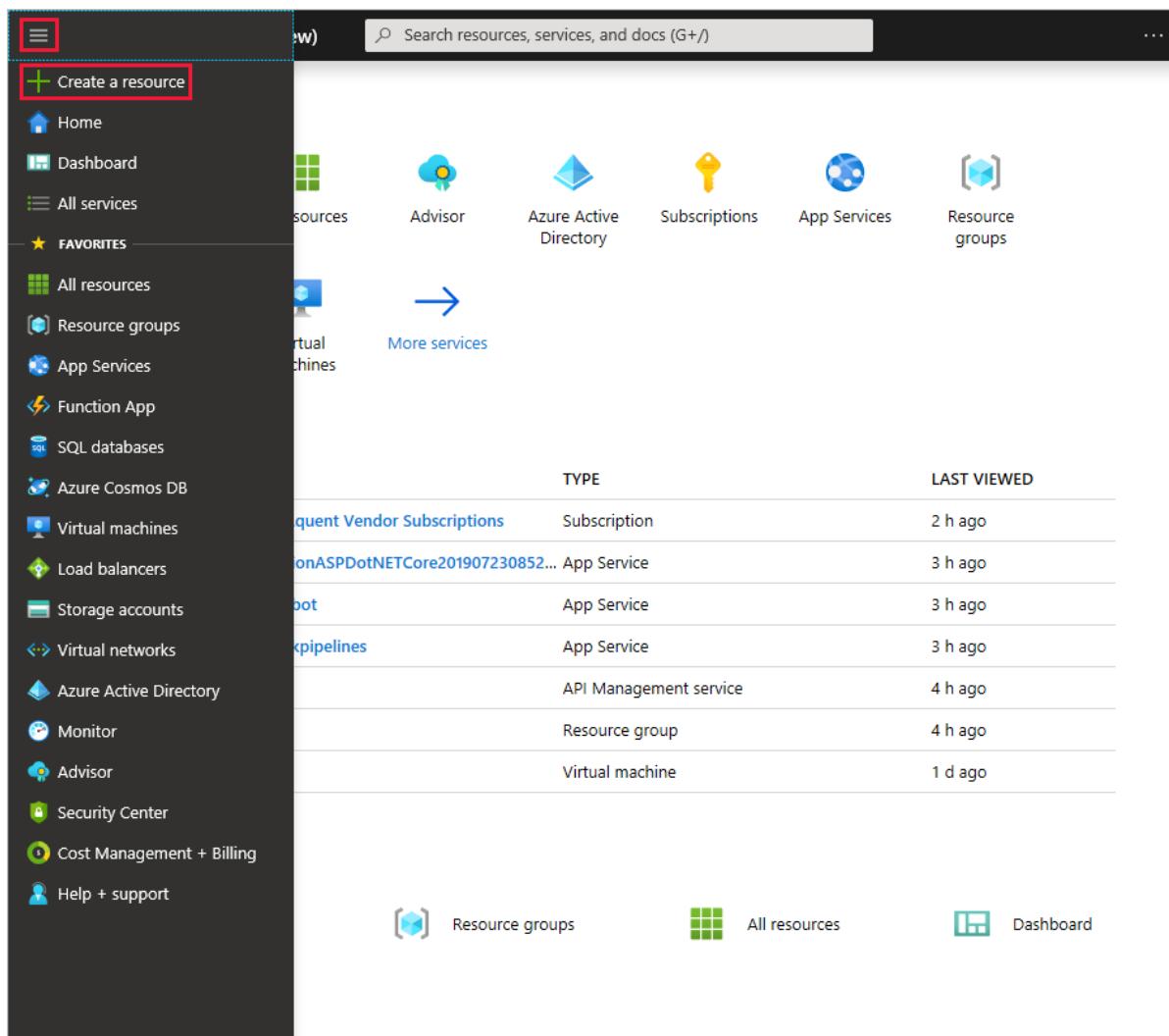
a /27 or /28.

If you see an error that specifies that the address space overlaps with a subnet, or that the subnet is not contained within the address space for your virtual network, check your VNet address range. You may not have enough IP addresses available in the address range you created for your virtual network. For example, if your default subnet encompasses the entire address range, there are no IP addresses left to create additional subnets. You can either adjust your subnets within the existing address space to free up IP addresses, or specify an additional address range and create the gateway subnet there.

### Example settings

- **Instance details > Region:** East US
- **Virtual Network > Virtual network:** VNet1
- **Instance details > Name:** VNet1GW
- **Instance details > Gateway type:** VPN
- **Instance details > VPN type:** Route-based
- **Virtual Network > Gateway subnet address range:** 10.1.255.0/27
- **Public IP address > Public IP address name:** VNet1GWIP

1. From the [Azure portal](#) menu, select **Create a resource**.



The screenshot shows the Azure portal's main dashboard. On the left, a dark sidebar lists various services like Home, Dashboard, All services, Favorites, and more. A red box highlights the '+ Create a resource' button at the top of this sidebar. The main area features a search bar at the top right and several service icons: Advisor, Azure Active Directory, Subscriptions, App Services, and Resource groups. Below these are links for 'More services' and 'Virtual machines'. A large table in the center lists recently viewed resources, categorized by Type and Last Viewed. At the bottom, there are navigation links for Resource groups, All resources, and Dashboard.

TYPE	LAST VIEWED
Recent Vendor Subscriptions	Subscription 2 h ago
monASPDotNETCore201907230852...	App Service 3 h ago
root	App Service 3 h ago
pipelines	App Service 3 h ago
	API Management service 4 h ago
	Resource group 4 h ago
	Virtual machine 1 d ago

2. In the **Search the Marketplace** field, type 'Virtual Network Gateway'. Locate **Virtual network gateway** in the search return and click the entry. On the **Virtual network gateway** page, click **Create**. This opens the **Create virtual network gateway** page.

## Create virtual network gateway

Azure has provided a planning and design guide to help you configure the various VPN gateway options. [Learn more.](#)

### Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*

Resource group ⓘ

TestRG1 (derived from virtual network's resource group)

### Instance details

Name \*

 VNet1GW ✓

Region \*

 (US) East US ✓

Gateway type \* ⓘ

VPN  ExpressRoute

VPN type \* ⓘ

Route-based  Policy-based

SKU \* ⓘ

 VpnGw1 ✓

Generation ⓘ

 Generation1 ✓

Virtual network \* ⓘ

 VNet1 ✓

[Create virtual network](#)

i Only virtual networks in the currently selected subscription and region are listed.

Gateway subnet address range \* ⓘ

 10.11.255.0/27 ✓

10.11.255.0 - 10.11.255.31 (32 addresses)

### Public IP address

Public IP address \* ⓘ

Create new  Use existing

Public IP address name \*

 VNet1GWpip ✓

Public IP address SKU

Basic

Assignment

Dynamic  Static

Enable active-active mode \* ⓘ

Enabled  Disabled

Configure BGP ASN \* ⓘ

Enabled  Disabled

3. On the **Create virtual network gateway** page, fill in the values for your virtual network gateway.

### Project details

- **Subscription:** Select the subscription you want to use from the dropdown.
- **Resource Group:** This setting is autofilled when you select your virtual network on this page.

### Instance details

- **Name:** Name your gateway. Naming your gateway not the same as naming a gateway subnet. It's the name of the gateway object you are creating.
- **Region:** Select the region in which you want to create this resource. The region for the gateway must be the same as the virtual network.
- **Gateway type:** Select **VPN**. VPN gateways use the virtual network gateway type **VPN**.

- **VPN type:** Select the VPN type that is specified for your configuration. Most configurations require a Route-based VPN type.
- **SKU:** Select the gateway SKU from the dropdown. The SKUs listed in the dropdown depend on the VPN type you select. For more information about gateway SKUs, see [Gateway SKUs](#).

**Virtual network:** Choose the virtual network to which you want to add this gateway.

**Gateway subnet address range:** This field only appears if your VNet doesn't have a gateway subnet. If possible, make the range /27 or larger (/26,/25 etc.). We don't recommend creating a range any smaller than /28. If you already have a gateway subnet, you can view GatewaySubnet details by navigating to your virtual network. Click **Subnets** to view the range. If you want to change the range, you can delete and recreate the GatewaySubnet.

**Public IP address:** This setting specifies the public IP address object that gets associated to the VPN gateway. The public IP address is dynamically assigned to this object when the VPN gateway is created. The only time the Public IP address changes is when the gateway is deleted and re-created. It doesn't change across resizing, resetting, or other internal maintenance/upgrades of your VPN gateway.

- **Public IP address:** Leave **Create new** selected.
- **Public IP address name:** In the text box, type a name for your public IP address instance.
- **Assignment:** VPN gateway supports only Dynamic.

**Active-Active mode:** Only select **Enable active-active mode** if you are creating an active-active gateway configuration. Otherwise, leave this setting unselected.

Leave **Configure BGP ASN** deselected, unless your configuration specifically requires this setting. If you do require this setting, the default ASN is 65515, although this can be changed.

4. Click **Review + create** to run validation. Once validation passes, click **Create** to deploy the VPN gateway. A gateway can take up to 45 minutes to fully create and deploy. You can see the deployment status on the Overview page for your gateway.

After the gateway is created, you can view the IP address that has been assigned to it by looking at the virtual network in the portal. The gateway appears as a connected device.

#### 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 Virtual Network gateway(VPN, Express Route gateway) to stop functioning as expected. For more information about network security groups, see [What is a network security group?](#)

## 3. Create the local network gateway

The local network gateway typically refers to your on-premises location. You give the site a name by which Azure can refer to it, then specify the IP address of the on-premises VPN device to which you will create a connection. You also specify the IP address prefixes that will be routed through the VPN gateway to the VPN device. The address prefixes you specify are the prefixes located on your on-premises network. If your on-premises network changes or you need to change the public IP address for the VPN device, you can easily update the values later.

#### Example values

- **Name:** Site1
- **Resource Group:** TestRG1
- **Location:** East US

1. From the [Azure portal](#) menu, select **Create a resource**.

The screenshot shows the Azure portal interface. On the left, there is a dark sidebar with a navigation menu. At the top of the sidebar is a red-bordered button labeled "Create a resource". Below it are several items: "Home", "Dashboard", "All services", "FAVORITES" (with a star icon), "All resources", "Resource groups", "App Services", "Function App", "SQL databases", "Azure Cosmos DB", "Virtual machines", "Load balancers", "Storage accounts", "Virtual networks", "Azure Active Directory", "Monitor", "Advisor", "Security Center", "Cost Management + Billing", and "Help + support". To the right of the sidebar, the main content area has a search bar at the top with the placeholder "Search resources, services, and docs (G+/" and a "..." button. Below the search bar are several service icons: "resources" (green grid), "Advisor" (cloud with sun), "Azure Active Directory" (blue diamond), "Subscriptions" (yellow key), "App Services" (blue globe), and "Resource groups" (hexagon). A large blue arrow points from the "More services" link to a list of recently viewed resources. This list includes: "requent Vendor Subscriptions" (Subscription, last viewed 2 h ago), "onASPDotNETCore201907230852..." (App Service, last viewed 3 h ago), "boot" (App Service, last viewed 3 h ago), "pipelines" (API Management service, last viewed 4 h ago), "Resource group" (Resource group, last viewed 4 h ago), and "Virtual machine" (Virtual machine, last viewed 1 d ago). At the bottom of the main content area are three buttons: "Resource groups" (hexagon icon), "All resources" (green grid icon), and "Dashboard" (blue square icon).

2. In the **Search the marketplace** field, type **Local network gateway**, then press **Enter** to search. This will return a list of results. Click **Local network gateway**, then click the **Create** button to open the **Create local network gateway** page.

**Create local network gate...**

---

\* Name

\* IP address i

Address space i

10.101.0.0/24	...
<a href="#">Add additional address range</a>	...

Configure BGP settings

---

\* Subscription

\* Resource group i  
   
[Create new](#)

\* Location

---

[Automation options](#)

3. On the **Create local network gateway page**, specify the values for your local network gateway.

- **Name:** Specify a name for your local network gateway object.
- **IP address:** This is the public IP address of the VPN device that you want Azure to connect to. Specify a valid public IP address. If you don't have the IP address right now, you can use the values shown in the example, but you'll need to go back and replace your placeholder IP address with the public IP address of your VPN device. Otherwise, Azure will not be able to connect.
- **Address Space** refers to the address ranges for the network that this local network represents. You can add multiple address space ranges. Make sure that the ranges you specify here do not overlap with ranges of other networks that you want to connect to. Azure will route the address range that you specify to the on-premises VPN device IP address. *Use your own values here if you want to connect to your on-premises site, not the values shown in the example.*
- **Configure BGP settings:** Use only when configuring BGP. Otherwise, don't select this.
- **Subscription:** Verify that the correct subscription is showing.

- **Resource Group:** Select the resource group that you want to use. You can either create a new resource group, or select one that you have already created.
  - **Location:** The location is the same as **Region** in other settings. Select the location that this object will be created in. You may want to select the same location that your VNet resides in, but you are not required to do so.
4. When you have finished specifying the values, click the **Create** button at the bottom of the page to create the local network gateway.

## 4. Configure your VPN device

Site-to-Site connections to an on-premises network require a VPN device. In this step, you configure your VPN device. When configuring your VPN device, you need the following:

- A shared key. This is the same shared key that you specify when creating your Site-to-Site VPN connection. In our examples, we use a basic shared key. We recommend that you generate a more complex key to use.
- The Public IP address of your virtual network gateway. You can view the public IP address by using the Azure portal, PowerShell, or CLI. To find the Public IP address of your VPN gateway using the Azure portal, navigate to **Virtual network gateways**, then click the name of your gateway.

### To download VPN device configuration scripts:

Depending on the VPN device that you have, you may be able to download a VPN device configuration script. For more information, see [Download VPN device configuration scripts](#).

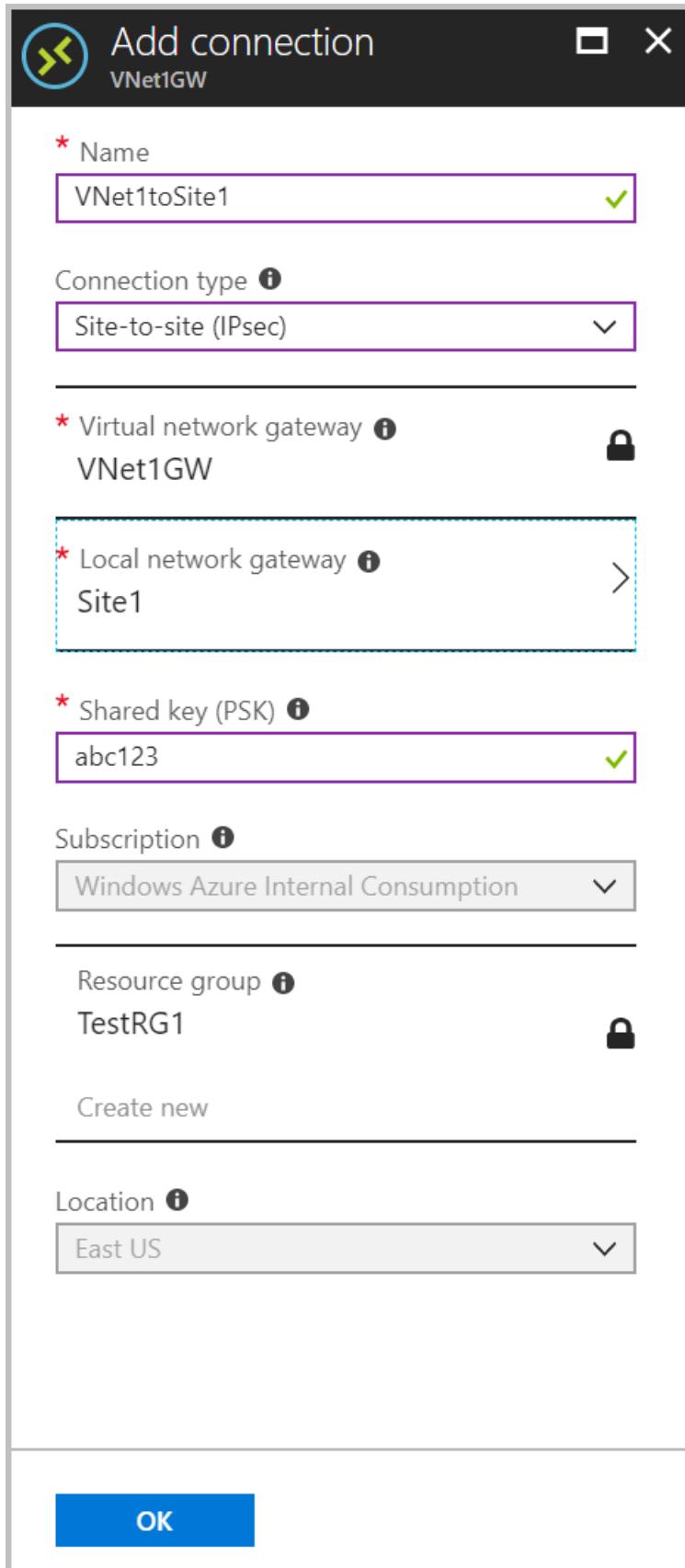
### See the following links for additional configuration information:

- For information about compatible VPN devices, see [VPN Devices](#).
- Before configuring your VPN device, check for any [Known device compatibility issues](#) for the VPN device that you want to use.
- For links to device configuration settings, see [Validated VPN Devices](#). The device configuration links are provided on a best-effort basis. It's always best to check with your device manufacturer for the latest configuration information. The list shows the versions we have tested. If your OS is not on that list, it is still possible that the version is compatible. Check with your device manufacturer to verify that OS version for your VPN device is compatible.
- For an overview of VPN device configuration, see [Overview of 3rd party VPN device configurations](#).
- For information about editing device configuration samples, see [Editing samples](#).
- For cryptographic requirements, see [About cryptographic requirements and Azure VPN gateways](#).
- For information about IPsec/IKE parameters, see [About VPN devices and IPsec/IKE parameters for Site-to-Site VPN gateway connections](#). This link shows information about IKE version, Diffie-Hellman Group, Authentication method, encryption and hashing algorithms, SA lifetime, PFS, and DPD, in addition to other parameter information that you need to complete your configuration.
- For IPsec/IKE policy configuration steps, see [Configure IPsec/IKE policy for S2S VPN or VNet-to-VNet connections](#).
- To connect multiple policy-based VPN devices, see [Connect Azure VPN gateways to multiple on-premises policy-based VPN devices using PowerShell](#).

## 5. Create the VPN connection

Create the Site-to-Site VPN connection between your virtual network gateway and your on-premises VPN device.

1. Open the page for your virtual network gateway. There are multiple ways to navigate. You can navigate to the gateway by going to **Name of your VNet** -> **Overview** -> **Connected devices** -> **Name of your gateway**.
2. On the page for the gateway, click **Connections**. At the top of the Connections page, click **+Add** to open the **Add connection** page.



3. On the **Add connection** page, configure the values for your connection.
  - **Name:** Name your connection.

- **Connection type:** Select **Site-to-site(IPSec)**.
  - **Virtual network gateway:** The value is fixed because you are connecting from this gateway.
  - **Local network gateway:** Click **Choose a local network gateway** and select the local network gateway that you want to use.
  - **Shared Key:** the value here must match the value that you are using for your local on-premises VPN device. The example uses 'abc123', but you can (and should) use something more complex. The important thing is that the value you specify here must be the same value that you specify when configuring your VPN device.
  - The remaining values for **Subscription**, **Resource Group**, and **Location** are fixed.
4. Click **OK** to create your connection. You'll see *Creating Connection* flash on the screen.
5. You can view the connection in the **Connections** page of the virtual network gateway. The Status will go from *Unknown* to *Connecting*, and then to *Succeeded*.

## 6. Verify the VPN connection

In the Azure portal, you can view the connection status of a Resource Manager VPN Gateway by navigating to the connection. The following steps show one way to navigate to your connection and verify.

1. In the [Azure portal](#) menu, select **All resources** or search for and select **All resources** from any page.
2. Select to your virtual network gateway.
3. On the blade for your virtual network gateway, click **Connections**. You can see the status of each connection.
4. Click the name of the connection that you want to verify to open **Essentials**. In Essentials, you can view more information about your connection. The **Status** is 'Succeeded' and 'Connected' when you have made a successful connection.

Essentials ^	
Resource group	Data in 2.35 KB
Status <b>Connected</b>	Data out 3.14 KB
Location	Virtual network
East US	
Subscription name	Virtual network gateway
Subscription ID	Local network gateway

## To connect to a virtual machine

You can connect to a VM that is deployed to your VNet by creating a Remote Desktop Connection to your VM. The best way to initially verify that you can connect to your VM is to connect by using its private IP address, rather than computer name. That way, you are testing to see if you can connect, not whether name resolution is configured properly.

1. Locate the private IP address. You can find the private IP address of a VM in multiple ways. Below, we show the steps for the Azure portal and for PowerShell.
- Azure portal - Locate your virtual machine in the Azure portal. View the properties for the VM. The private IP address is listed.
  - PowerShell - Use the example to view a list of VMs and private IP addresses from your resource groups. You don't need to modify this example before using it.

```

$VMs = Get-AzVM
$Nics = Get-AzNetworkInterface | Where VirtualMachine -ne $null

foreach($Nic in $Nics)
{
    $VM = $VMs | Where-Object -Property Id -eq $Nic.VirtualMachine.Id
    $Prv = $Nic.IpConfigurations | Select-Object -ExpandProperty PrivateIpAddress
    $Alloc = $Nic.IpConfigurations | Select-Object -ExpandProperty PrivateIpAllocationMethod
    Write-Output "$($VM.Name): $Prv,$Alloc"
}

```

2. Verify that you are connected to your VNet using the VPN connection.
3. Open **Remote Desktop Connection** by typing "RDP" or "Remote Desktop Connection" in the search box on the taskbar, then select Remote Desktop Connection. You can also open Remote Desktop Connection using the 'mstsc' command in PowerShell.
4. In Remote Desktop Connection, enter the private IP address of the VM. You can click "Show Options" to adjust additional settings, then connect.

#### To troubleshoot an RDP connection to a VM

If you are having trouble connecting to a virtual machine over your VPN connection, check the following:

- Verify that your VPN connection is successful.
- Verify that you are connecting to the private IP address for the VM.
- If you can connect to the VM using the private IP address, but not the computer name, verify that you have configured DNS properly. For more information about how name resolution works for VMs, see [Name Resolution for VMs](#).
- For more information about RDP connections, see [Troubleshoot Remote Desktop connections to a VM](#).

## How to reset a VPN gateway

Resetting an Azure VPN gateway is helpful if you lose cross-premises VPN connectivity on one or more Site-to-Site VPN tunnels. In this situation, your on-premises VPN devices are all working correctly, but are not able to establish IPsec tunnels with the Azure VPN gateways. For steps, see [Reset a VPN gateway](#).

## How to change a gateway SKU (resize a gateway)

For the steps to change a gateway SKU, see [Gateway SKUs](#).

## How to add an additional connection to a VPN gateway

You can add additional connections, provided that none of the address spaces overlap between connections.

1. To add an additional connection, navigate to the VPN gateway, then click **Connections** to open the Connections page.
2. Click **+Add** to add your connection. Adjust the connection type to reflect either VNet-to-VNet (if connecting to another VNet gateway), or Site-to-site.
3. If you are connecting using Site-to-site and you have not already created a local network gateway for the site you want to connect to, you can create a new one.
4. Specify the shared key that you want to use, then click **OK** to create the connection.

## Next steps

- For information about BGP, see the [BGP Overview](#) and [How to configure BGP](#).

- For information about forced tunneling, see [About forced tunneling](#).
- For information about Highly Available Active-Active connections, see [Highly Available cross-premises and VNet-to-VNet connectivity](#).
- For information about how to limit network traffic to resources in a virtual network, see [Network Security](#).
- For information about how Azure routes traffic between Azure, on-premises, and Internet resources, see [Virtual network traffic routing](#).
- For information about creating a Site-to-Site VPN connection using Azure Resource Manager template, see [Create a Site-to-Site VPN Connection](#).
- For information about creating a Vnet-to-Vnet VPN connection using Azure Resource Manager template, see [Deploy HBase geo replication](#).

# Connect a virtual network to an ExpressRoute circuit using the portal

11/13/2019 • 4 minutes to read • [Edit Online](#)

This article helps you create a connection to link a virtual network to an Azure ExpressRoute circuit using the Azure portal. The virtual networks that you connect to your Azure ExpressRoute circuit can either be in the same subscription, or they can be part of another subscription.

## Before you begin

- Review the [prerequisites](#), [routing requirements](#), and [workflows](#) before you begin configuration.
- You must have an active ExpressRoute circuit.
  - Follow the instructions to [create an ExpressRoute circuit](#) and have the circuit enabled by your connectivity provider.
  - Ensure that you have Azure private peering configured for your circuit. See the [Create and modify peering for an ExpressRoute circuit](#) article for peering and routing instructions.
  - Ensure that Azure private peering is configured and the BGP peering between your network and Microsoft is up so that you can enable end-to-end connectivity.
  - Ensure that you have a virtual network and a virtual network gateway created and fully provisioned. Follow the instructions to [create a virtual network gateway for ExpressRoute](#). A virtual network gateway for ExpressRoute uses the GatewayType 'ExpressRoute', not VPN.
- You can link up to 10 virtual networks to a standard ExpressRoute circuit. All virtual networks must be in the same geopolitical region when using a standard ExpressRoute circuit.
- A single VNet can be linked to up to four ExpressRoute circuits. Use the process below to create a new connection object for each ExpressRoute circuit you are connecting to. The ExpressRoute circuits can be in the same subscription, different subscriptions, or a mix of both.
- You can link a virtual network outside of the geopolitical region of the ExpressRoute circuit, or connect a larger number of virtual networks to your ExpressRoute circuit if you enabled the ExpressRoute premium add-on. Check the [FAQ](#) for more details on the premium add-on.
- You can [view a video](#) before beginning to better understand the steps.

## Connect a VNet to a circuit - same subscription

### NOTE

BGP configuration information will not show up if the layer 3 provider configured your peerings. If your circuit is in a provisioned state, you should be able to create connections.

### To create a connection

1. Ensure that your ExpressRoute circuit and Azure private peering have been configured successfully. Follow the instructions in [Create an ExpressRoute circuit](#) and [Create and modify peering for an ExpressRoute circuit](#). Your ExpressRoute circuit should look like the following image:

The figure shows three windows side-by-side:

- Left Window (Settings):** Shows the 'Essentials' tab with basic circuit information like Resource group (USWest-ER-Demo-RG), Provider (Equinix), and Status (Enabled). It also lists Peering location (Silicon Valley), Bandwidth (200 Mbps), and Service key.
- Middle Window (Peerings):** Shows the 'Peerings' table with three entries: Azure private (Enabled, 172.16.0.0/30, 172.16.0.4/30), Azure public (Disabled, -), and Microsoft (Disabled, -).
- Right Window (Peering Details):** Shows the 'Azure private' peering entry in more detail, including Primary Subnet (172.16.0.0/30) and Secondary Subnet (172.16.0.4/30).

2. You can now start provisioning a connection to link your virtual network gateway to your ExpressRoute circuit. Click **Connection** > **Add** to open the **Add connection** page, and then configure the values.

The figure shows three windows side-by-side:

- Left Window (Settings):** Shows the 'Connections' section with the 'Add' button highlighted.
- Middle Window (Connections):** Shows the 'Connections' table with no results.
- Right Window (Add connection):** Shows the configuration dialog for a new connection:
  - Name:** ER-VNet-Connection
  - Connection type:** ExpressRoute
  - Virtual network gateway:** Demo-VNet-GW
  - ExpressRoute circuit:** ER-Demo-Ckt-SV (locked)
  - Subscription:** ExpressRoute-Demo
  - Resource group:** USWest-ER-Demo-RG (locked)
  - Location:** West US

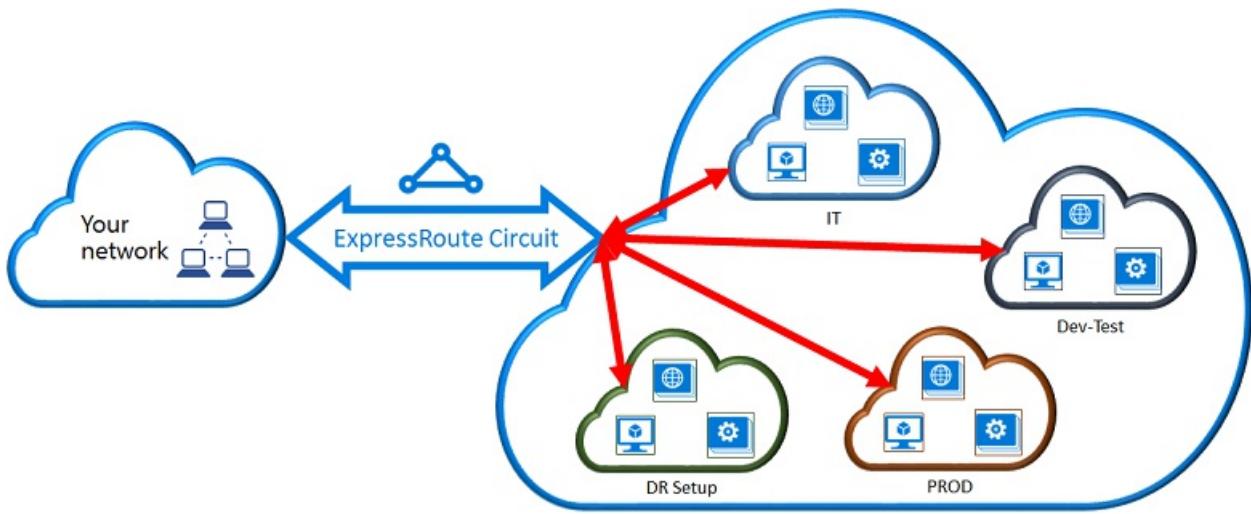
3. After your connection has been successfully configured, your connection object will show the information for the connection.

The figure shows two windows side-by-side:

- Left Window (Connections):** Shows the 'Connections' table with one entry: ER-VNet-Connection (Succeeded, ExpressRoute, Demo-VNet-GW).
- Right Window (ER-VNet-Connection):** Shows the details for the connection:
  - Essentials:** Resource group (USWest-ER-Demo-RG), Status (Succeeded), Location (West US), Subscription name (ExpressRoute-Demo), and Subscription ID.
  - Statistics:** Data in (0 B), Data out (0 B).
  - Details:** Virtual network (Demo-VNet), Virtual network gateway (Demo-VNet-GW (13.88.21.182)), and Circuit (ER-Demo-Ckt-SV).

## Connect a VNet to a circuit - different subscription

You can share an ExpressRoute circuit across multiple subscriptions. The figure below shows a simple schematic of how sharing works for ExpressRoute circuits across multiple subscriptions.



- Each of the smaller clouds within the large cloud is used to represent subscriptions that belong to different departments within an organization.
- Each of the departments within the organization can use their own subscription for deploying their services, but they can share a single ExpressRoute circuit to connect back to your on-premises network.
- A single department (in this example: IT) can own the ExpressRoute circuit. Other subscriptions within the organization can use the ExpressRoute circuit and authorizations associated to the circuit, including subscriptions linked to other Azure Active Directory tenants and Enterprise Agreement enrollments.

**NOTE**

Connectivity and bandwidth charges for the dedicated circuit will be applied to the ExpressRoute circuit owner. All virtual networks share the same bandwidth.

## Administration - About circuit owners and circuit users

The 'circuit owner' is an authorized Power User of the ExpressRoute circuit resource. The circuit owner can create authorizations that can be redeemed by 'circuit users'. Circuit users are owners of virtual network gateways that are not within the same subscription as the ExpressRoute circuit. Circuit users can redeem authorizations (one authorization per virtual network).

The circuit owner has the power to modify and revoke authorizations at any time. Revoking an authorization results in all link connections being deleted from the subscription whose access was revoked.

### Circuit owner operations

#### To create a connection authorization

The circuit owner creates an authorization. This results in the creation of an authorization key that can be used by a circuit user to connect their virtual network gateways to the ExpressRoute circuit. An authorization is valid for only one connection.

**NOTE**

Each connection requires a separate authorization.

1. In the ExpressRoute page, Click **Authorizations** and then type a **name** for the authorization and click **Save**.

The screenshot shows the 'Authorizations' blade for a specific circuit. A new authorization named 'S3Demo' is being added. The 'Save' button at the top left is highlighted with a red box.

- Once the configuration is saved, copy the **Resource ID** and the **Authorization Key**.

The screenshot shows the 'Authorizations' blade for the same circuit. The newly created authorization 'S3Demo' is listed with its details visible. The 'Resource ID' and 'Authorization Key' fields are highlighted with red boxes.

## To delete a connection authorization

You can delete a connection by selecting the **Delete** icon on the page for your connection.

## Circuit user operations

The circuit user needs the resource ID and an authorization key from the circuit owner.

## To redeem a connection authorization

- Click the **+New** button.

The screenshot shows the Microsoft Azure portal homepage. The 'New' button, which is used to start a new resource creation process, is highlighted with a red box.

- Search for "Connection" in the Marketplace, select it, and click **Create**.

A screenshot of the Microsoft Azure Marketplace search interface. The search bar at the top contains the text "Connection". Below the search bar, the results table has columns for NAME, PUBLISHER, and CATEGORY. The first result, "Connection" by Microsoft, is highlighted with a red box. To the right of the results table, there is a detailed description of a VPN connection, social sharing links, and useful links like Service overview, Documentation, and Pricing details. At the bottom right of the main pane, there is a "Create" button.

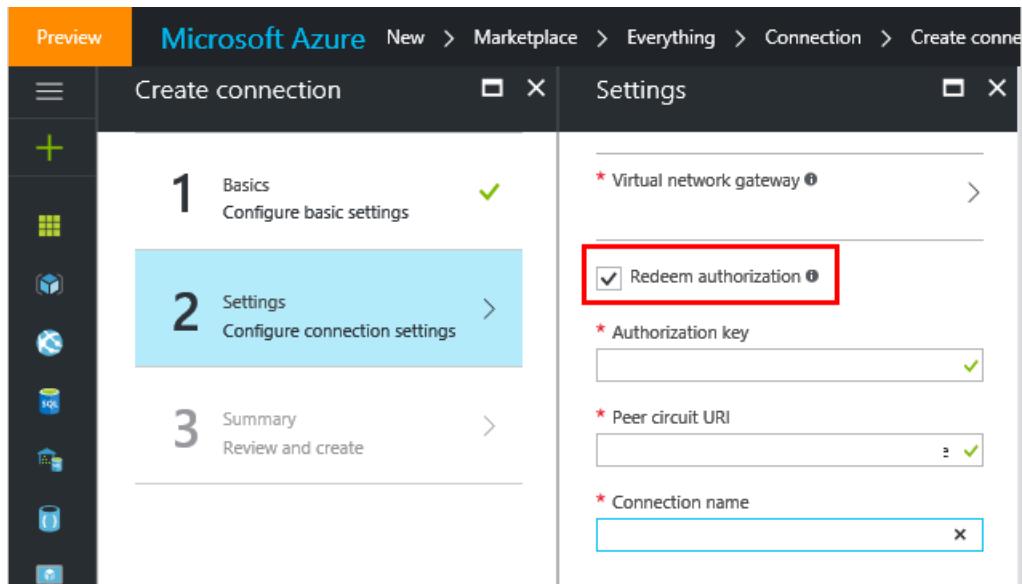
3. Make sure the **Connection type** is set to "ExpressRoute".

4. Fill in the details, then click **OK** in the Basics page.

A screenshot of the "Create connection" wizard in the Microsoft Azure portal. The left sidebar shows steps 1, 2, and 3. Step 1 is titled "Basics" and "Configure basic settings". Step 2 is titled "Settings" and "Configure connection settings". Step 3 is titled "Summary" and "Review and create". The right pane is titled "Basics" and contains the following fields: "Connection type" (set to "ExpressRoute"), "Subscription" (dropdown), "Resource group" (radio buttons for "Create new" and "Use existing", with "Use existing" selected and "Default-Networking" chosen), and "Location" (dropdown set to "West US").

5. In the **Settings** page, Select the **Virtual network gateway** and check the **Redeem authorization** check box.

6. Enter the **Authorization key** and the **Peer circuit URI** and give the connection a name. Click **OK**. The **Peer Circuit URI** is the Resource ID of the ExpressRoute circuit (which you can find under the Properties Setting pane of the ExpressRoute Circuit).



7. Review the information in the **Summary** page and click **OK**.

#### To release a connection authorization

You can release an authorization by deleting the connection that links the ExpressRoute circuit to the virtual network.

## Delete a connection to unlink a VNet

You can delete a connection and unlink your VNet to an ExpressRoute circuit by selecting the **Delete** icon on the page for your connection.

## Next steps

For more information about ExpressRoute, see the [ExpressRoute FAQ](#).

2 minutes to read

# Work with a virtual network TAP using the Azure CLI

11/19/2019 • 3 minutes to read • [Edit Online](#)

Azure virtual network TAP (Terminal Access Point) allows you to continuously stream your virtual machine network traffic to a network packet collector or analytics tool. The collector or analytics tool is provided by a [network virtual appliance](#) partner. For a list of partner solutions that are validated to work with virtual network TAP, see [partner solutions](#).

## Create a virtual network TAP resource

Read [prerequisites](#) before you create a virtual network TAP resource. You can run the commands that follow in the [Azure Cloud Shell](#), or by running the Azure command-line interface (CLI) from your computer. The Azure Cloud Shell is a free interactive shell, that doesn't require installing the Azure CLI on your computer. You must sign in to Azure with an account that has the appropriate [permissions](#). This article requires the Azure CLI version 2.0.46 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI 2.0](#).

Virtual network TAP is currently available as an extension. To install the extension you need to run

```
az extension add -n virtual-network-tap
```

If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

1. Retrieve the ID of your subscription into a variable that is used in a later step:

```
subscriptionId=$(az account show \
--query id \
--out tsv)
```

2. Set the subscription id that you will use to create a virtual network TAP resource.

```
az account set --subscription $subscriptionId
```

3. Re-register the subscription ID that you'll use to create a virtual network TAP resource. If you get a registration error when you create a TAP resource, run the following command:

```
az provider register --namespace Microsoft.Network --subscription $subscriptionId
```

4. If the destination for the virtual network TAP is the network interface on the network virtual appliance for collector or analytics tool -

- Retrieve the IP configuration of the network virtual appliance's network interface into a variable that is used in a later step. The ID is the end point that will aggregate the TAP traffic. The following example retrieves the ID of the `ipconfig1` IP configuration for a network interface named `myNetworkInterface`, in a resource group named `myResourceGroup`:

```
IpConfigId=$(az network nic ip-config show \
--name ipconfig1 \
--nic-name myNetworkInterface \
--resource-group myResourceGroup \
--query id \
--out tsv)
```

- Create the virtual network TAP in westcentralus azure region using the ID of the IP configuration as the destination and an optional port property. The port specifies the destination port on network interface IP configuration where the TAP traffic will be received :

```
az network vnet tap create \
--resource-group myResourceGroup \
--name myTap \
--destination $IpConfigId \
--port 4789 \
--location westcentralus
```

5. If the destination for the virtual network TAP is an azure internal load balancer:

- Retrieve the front end IP configuration of the Azure internal load balancer into a variable that is used in a later step. The ID is the end point that will aggregate the TAP traffic. The following example retrieves the ID of the *frontendipconfig1* front end IP configuration for a load balancer named *myInternalLoadBalancer*, in a resource group named *myResourceGroup*:

```
FrontendIpConfigId=$(az network lb frontend-ip show \
--name frontendipconfig1 \
--lb-name myInternalLoadBalancer \
--resource-group myResourceGroup \
--query id \
--out tsv)
```

- Create the virtual network TAP using the ID of the frontend IP configuration as the destination and an optional port property. The port specifies the destination port on front end IP configuration where the TAP traffic will be received :

```
az network vnet tap create \
--resource-group myResourceGroup \
--name myTap \
--destination $FrontendIpConfigId \
--port 4789 \
--location westcentralus
```

6. Confirm creation of the virtual network TAP:

```
az network vnet tap show \
--resource-group myResourceGroup \
--name myTap
```

## Add a TAP configuration to a network interface

1. Retrieve the ID of an existing virtual network TAP resource. The following example retrieves a virtual network TAP named *myTap* in a resource group named *myResourceGroup*:

```
tapId=$(az network vnet tap show \
--name myTap \
--resource-group myResourceGroup \
--query id \
--out tsv)
```

2. Create a TAP configuration on the network interface of the monitored virtual machine. The following example creates a TAP configuration for a network interface named *myNetworkInterface*:

```
az network nic vtap-config create \
--resource-group myResourceGroup \
--nic myNetworkInterface \
--vnet-tap $tapId \
--name mytapconfig \
--subscription subscriptionId
```

### 3. Confirm creation of the TAP configuration:

```
az network nic vtap-config show \
--resource-group myResourceGroup \
--nic-name myNetworkInterface \
--name mytapconfig \
--subscription subscriptionId
```

## Delete the TAP configuration on a network interface

```
az network nic vtap-config delete \
--resource-group myResourceGroup \
--nic myNetworkInterface \
--name myTapConfig \
--subscription subscriptionId
```

## List virtual network TAPs in a subscription

```
az network vnet tap list
```

## Delete a virtual network TAP in a resource group

```
az network vnet tap delete \
--resource-group myResourceGroup \
--name myTap
```

# Deploy the Azure Virtual Network container network interface plug-in

10/7/2019 • 5 minutes to read • [Edit Online](#)

The Azure Virtual Network container network interface (CNI) plug-in installs in an Azure virtual machine and brings virtual network capabilities to Kubernetes Pods and Docker containers. To learn more about the plug-in, see [Enable containers to use Azure Virtual Network capabilities](#). Additionally, the plug-in can be used with the Azure Kubernetes Service (AKS) by choosing the [Advanced Networking](#) option, which automatically places AKS containers in a virtual network.

## Deploy plug-in for ACS-Engine Kubernetes cluster

The ACS-Engine deploys a Kubernetes cluster with an Azure Resource Manager template. The cluster configuration is specified in a JSON file that is passed to the tool when generating the template. To learn more about the entire list of supported cluster settings and their descriptions, see [Microsoft Azure Container Service Engine - Cluster Definition](#). The plug-in is the default networking plug-in for clusters created using the ACS-Engine. The following network configuration settings are important when configuring the plug-in:

SETTING	DESCRIPTION
firstConsecutiveStaticIP	The IP address that is allocated to the Master node. This is a mandatory setting.
clusterSubnet under kubernetesConfig	CIDR of the virtual network subnet where the cluster is deployed, and from which IP addresses are allocated to Pods
vnetSubnetId under masterProfile	Specifies the Azure Resource Manager resource ID of the subnet where the cluster is to be deployed
vnetCidr	CIDR of the virtual network where the cluster is deployed
max_Pods under kubeletConfig	Maximum number of Pods on every agent virtual machine. For the plug-in, the default is 30. You can specify up to 250

### Example configuration

The json example that follows is for a cluster with the following properties:

- 1 Master node and 2 Agent nodes
- Is deployed in a subnet named *KubeClusterSubnet* (10.0.0.0/20), with both Master and Agent nodes residing in it.

```
{
  "apiVersion": "vlabs",
  "properties": {
    "orchestratorProfile": {
      "orchestratorType": "Kubernetes",
      "kubernetesConfig": {
        "clusterSubnet": "10.0.0.0/20" --> Subnet allocated for the cluster
      }
    },
    "masterProfile": {
      "count": 1,
      "dnsPrefix": "ACSKubeMaster",
      "vmSize": "Standard_A2",
      "vnetSubnetId": "/subscriptions/<subscription ID>/resourceGroups/<Resource Group Name>/providers/Microsoft.Network/virtualNetworks/<Vnet Name>/subnets/KubeClusterSubnet",
      "firstConsecutiveStaticIP": "10.0.1.50", --> IP address allocated to the Master node
      "vnetCidr": "10.0.0.0/16" --> Virtual network address space
    },
    "agentPoolProfiles": [
      {
        "name": "k8sagentpool1",
        "count": 2,
        "vmSize": "Standard_A2_v2",
        "vnetSubnetId": "/subscriptions/<subscription ID>/resourceGroups/<Resource Group Name>/providers/Microsoft.Network/virtualNetworks/<VNet Name>/subnets/KubeClusterSubnet",
        "availabilityProfile": "AvailabilitySet"
      }
    ],
    "linuxProfile": {
      "adminUsername": "KubeServerAdmin",
      "ssh": {
        "publicKeys": [
          {...}
        ]
      }
    },
    "servicePrincipalProfile": {
      "clientId": "dd438987-aa12-4754-b47d-375811889714",
      "secret": "azure123"
    }
  }
}
```

## Deploy plug-in for a Kubernetes cluster

Complete the following steps to install the plug-in on every Azure virtual machine in a Kubernetes cluster:

1. [Download and install the plug-in](#).
2. Pre-allocate a virtual network IP address pool on every virtual machine from which IP addresses will be assigned to Pods. Every Azure virtual machine comes with a primary virtual network private IP address on each network interface. The pool of IP addresses for Pods is added as secondary addresses (*ipconfigs*) on the virtual machine network interface, using one of the following options:
  - **CLI:** [Assign multiple IP addresses using the Azure CLI](#)
  - **PowerShell:** [Assign multiple IP addresses using PowerShell](#)
  - **Portal:** [Assign multiple IP addresses using the Azure portal](#)
  - **Azure Resource Manager template:** [Assign multiple IP addresses using templates](#)

Ensure that you add enough IP addresses for all of the Pods that you expect to bring up on the virtual machine.
3. Select the plug-in for providing networking for your cluster by passing Kubelet the `-network-plugin=cni`

command-line option during cluster creation. Kubernetes, by default, looks for the plug-in and the configuration file in the directories where they are already installed.

4. If you want your Pods to access the internet, add the following *iptables* rule on your Linux virtual machines to source-NAT internet traffic. In the following example, the specified IP range is 10.0.0.0/8.

```
iptables -t nat -A POSTROUTING -m iprange ! --dst-range 168.63.129.16 -m  
addrtype ! --dst-type local ! -d 10.0.0.0/8 -j MASQUERADE
```

The rules NAT traffic that is not destined to the specified IP ranges. The assumption is that all traffic outside the previous ranges is internet traffic. You can choose to specify the IP ranges of the virtual machine's virtual network, that of peered virtual networks, and on-premises networks.

Windows virtual machines automatically source NAT traffic that has a destination outside the subnet to which the virtual machine belongs. It is not possible to specify custom IP ranges.

After completing the previous steps, Pods brought up on the Kubernetes Agent virtual machines are automatically assigned private IP addresses from the virtual network.

## Deploy plug-in for Docker containers

1. [Download and install the plug-in](#).
2. Create Docker containers with the following command:

```
./docker-run.sh <container-name> <container-namespace> <image>
```

The containers automatically start receiving IP addresses from the allocated pool. If you want to load balance traffic to the Docker containers, they must be placed behind a software load balancer, and you must configure a load balancer probe, the same way you create a policy and probes for a virtual machine.

### CNI network configuration file

The CNI network configuration file is described in JSON format. It is, by default, present in `/etc/cni/net.d` for Linux and `c:\cni\netconf` for Windows. The file specifies the configuration of the plug-in and is different for Windows and Linux. The json that follows is a sample Linux configuration file, followed by an explanation for some of the key settings. You don't need to make any changes to the file:

```
{
  "cniVersion": "0.3.0",
  "name": "azure",
  "plugins": [
    {
      "type": "azure-vnet",
      "mode": "bridge",
      "bridge": "azure0",
      "ipam": {
        "type": "azure-vnet-ipam"
      }
    },
    {
      "type": "portmap",
      "capabilities": {
        "portMappings": true
      },
      "snat": true
    }
  ]
}
```

#### Settings explanation

- **cniVersion:** The Azure Virtual Network CNI plug-ins support versions 0.3.0 and 0.3.1 of the [CNI spec](#).
- **name:** Name of the network. This property can be set to any unique value.
- **type:** Name of the network plug-in. Set to *azure-vnet*.
- **mode:** Operational mode. This field is optional. The only mode supported is "bridge". For more information, see [operational modes](#).
- **bridge:** Name of the bridge that will be used to connect containers to a virtual network. This field is optional. If omitted, the plugin automatically picks a unique name, based on the master interface index.
- **ipam type:** Name of the IPAM plug-in. Always set to *azure-vnet-ipam*.

## Download and install the plug-in

Download the plug-in from [GitHub](#). Download the latest version for the platform that you're using:

- **Linux:** [azure-vnet-cni-linux-amd64-<version no.>.tgz](#)
- **Windows:** [azure-vnet-cni-windows-amd64-<version no.>.zip](#)

Copy the install script for [Linux](#) or [Windows](#) to your computer. Save the script to a `scripts` directory on your computer and name the file `install-cni-plugin.sh` for Linux, or `install-cni-plugin.ps1` for Windows. To install the plug-in, run the appropriate script for your platform, specifying the version of the plug-in you are using. For example, you might specify *v1.0.12-rc3*:

```
\$scripts/install-cni-plugin.sh [version]
```

```
scripts\install-cni-plugin.ps1 [version]
```

The script installs the plug-in under `/opt/cni/bin` for Linux and `c:\cni\bin` for Windows. The installed plug-in comes with a simple network configuration file that works after installation. It doesn't need to be updated. To learn more about the settings in the file, see [CNI network configuration file](#).

# Upgrade an IPv4 application to IPv6 in Azure virtual network - PowerShell (Preview)

1/16/2020 • 5 minutes to read • [Edit Online](#)

This article shows you how to add IPv6 connectivity to an existing IPv4 application in an Azure virtual network with a Standard Load Balancer and Public IP. The in-place upgrade includes:

- IPv6 address space for the virtual network and subnet
- a Standard Load Balancer with both IPv4 and IPv6 frontend configurations
- VMs with NICs that have both an IPv4 + IPv6 configuration
- IPv6 Public IP so the load balancer has Internet-facing IPv6 connectivity

## IMPORTANT

IPv6 support for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 6.9.0

or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

## Prerequisites

### Register the service

Before you deploy a dual stack application in Azure, you must configure your subscription for this preview feature using the following Azure PowerShell:

Register as follows:

```
Register-AzProviderFeature -FeatureName AllowIPv6VirtualNetwork -ProviderNamespace Microsoft.Network  
Register-AzProviderFeature -FeatureName AllowIPv6CAOnStandardLB -ProviderNamespace Microsoft.Network
```

It takes up to 30 minutes for feature registration to complete. You can check your registration status by running the following Azure PowerShell command: Check on the registration as follows:

```
Get-AzProviderFeature -FeatureName AllowIPv6VirtualNetwork -ProviderNamespace Microsoft.Network  
Get-AzProviderFeature -FeatureName AllowIPv6CAOnStandardLB -ProviderNamespace Microsoft.Network
```

After the registration is complete, run the following command:

```
Register-AzResourceProvider -ProviderNamespace Microsoft.Network
```

### Create a Standard Load Balancer

This article assumes that you deployed a Standard Load Balancer as described in [Quickstart: Create a Standard Load Balancer - Azure PowerShell](#).

## Retrieve the resource group

Before you can create your dual-stack virtual network, you must retrieve the resource group with [Get-AzResourceGroup](#).

```
$rg = Get-AzResourceGroup -ResourceGroupName "myResourceGroupSLB"
```

## Create an IPv6 IP addresses

Create a public IPv6 address with [New-AzPublicIpAddress](#) for your Standard Load Balancer. The following example creates an IPv6 public IP address named *PublicIP\_v6* in the *myResourceGroupSLB* resource group:

```
$PublicIP_v6 = New-AzPublicIpAddress `  
-Name "PublicIP_v6" `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-Sku Standard `  
-AllocationMethod Static `  
-IpAddressVersion IPv6
```

## Configure load balancer frontend

Retrieve the existing load balancer configuration and then add the new IPv6 IP address using [Add-AzLoadBalancerFrontendIpConfig](#) as follows:

```
# Retrieve the load balancer configuration
$lb = Get-AzLoadBalancer -ResourceGroupName $rg.ResourceGroupName -Name "MyLoadBalancer"
# Add IPv6 components to the local copy of the load balancer configuration
$lb | Add-AzLoadBalancerFrontendIpConfig ` 
    -Name "dsLbFrontEnd_v6" ` 
    -PublicIpAddress $PublicIP_v6
#Update the running load balancer with the new frontend
$lb | Set-AzLoadBalancer
```

## Configure load balancer backend pool

Create the backend pool on the local copy of the load balancer configuration and update the running load balancer with the new backend pool configuration as follows:

```
$lb | Add-AzLoadBalancerBackendAddressPoolConfig -Name "LbBackEndPool_v6"
# Update the running load balancer with the new backend pool
$lb | Set-AzLoadBalancer
```

## Configure load balancer rules

Retrieve the existing load balancer frontend and backend pool configuration and then add new load-balancing rules using [Add-AzLoadBalancerRuleConfig](#).

```
# Retrieve the updated (live) versions of the frontend and backend pool
$frontendIPv6 = Get-AzLoadBalancerFrontendIpConfig -Name "dsLbFrontEnd_v6" -LoadBalancer $lb
$backendPoolv6 = Get-AzLoadBalancerBackendAddressPoolConfig -Name "LbBackEndPool_v6" -LoadBalancer $lb
# Create new LB rule with the frontend and backend
$lb | Add-AzLoadBalancerRuleConfig ` 
    -Name "dsLBrule_v6" ` 
    -FrontendIpConfiguration $frontendIPv6 ` 
    -BackendAddressPool $backendPoolv6 ` 
    -Protocol Tcp ` 
    -FrontendPort 80 ` 
    -BackendPort 80
#Finalize all the load balancer updates on the running load balancer
$lb | Set-AzLoadBalancer
```

## Add IPv6 address ranges

Add IPv6 address ranges to the virtual network and subnet hosting the VMs as follows:

```

#Add IPv6 ranges to the VNET and subnet
#Retreive the VNET object
$vnet = Get-AzVirtualNetwork -ResourceGroupName $rg.ResourceGroupName -Name "myVnet"
#Add IPv6 prefix to the VNET
$vnet.addressspace.addressprefixes.add("ace:cab:deca::/48")
#Update the running VNET
$vnet | Set-AzVirtualNetwork

#Retriece the subnet object from the local copy of the VNET
$subnet= $vnet.subnets[0]
#Add IPv6 prefix to the Subnet (subnet of the VNET prefix, of course)
$subnet.addressprefix.add("ace:cab:deca::/64")
#Update the running VNET with the new subnet configuration
$vnet | Set-AzVirtualNetwork

```

## Add IPv6 configuration to NIC

Configure all of the VM NICs with an IPv6 address using [Add-AzNetworkInterfaceIpConfig](#) as follows:

```

#Retriece the NIC objects
$NIC_1 = Get-AzNetworkInterface -Name "myNic1" -ResourceGroupName $rg.ResourceGroupName
$NIC_2 = Get-AzNetworkInterface -Name "myNic2" -ResourceGroupName $rg.ResourceGroupName
$NIC_3 = Get-AzNetworkInterface -Name "myNic3" -ResourceGroupName $rg.ResourceGroupName
#Add an IPv6 IPconfig to NIC_1 and update the NIC on the running VM
$NIC_1 | Add-AzNetworkInterfaceIpConfig -Name MyIPv6Config -Subnet $vnet.Subnets[0] - 
PrivateIpAddressVersion IPv6 -LoadBalancerBackendAddressPool $backendPoolv6
$NIC_1 | Set-AzNetworkInterface
#Add an IPv6 IPconfig to NIC_2 and update the NIC on the running VM
$NIC_2 | Add-AzNetworkInterfaceIpConfig -Name MyIPv6Config -Subnet $vnet.Subnets[0] - 
PrivateIpAddressVersion IPv6 -LoadBalancerBackendAddressPool $backendPoolv6
$NIC_2 | Set-AzNetworkInterface
#Add an IPv6 IPconfig to NIC_3 and update the NIC on the running VM
$NIC_3 | Add-AzNetworkInterfaceIpConfig -Name MyIPv6Config -Subnet $vnet.Subnets[0] - 
PrivateIpAddressVersion IPv6 -LoadBalancerBackendAddressPool $backendPoolv6
$NIC_3 | Set-AzNetworkInterface

```

## View IPv6 dual stack virtual network in Azure portal

You can view the IPv6 dual stack virtual network in Azure portal as follows:

1. In the portal's search bar, enter *myVnet*.
2. When **myVnet** appears in the search results, select it. This launches the **Overview** page of the dual stack virtual network named *myVNet*. The dual stack virtual network shows the three NICs with both IPv4 and IPv6 configurations located in the dual stack subnet named *mySubnet*.

The screenshot shows the Azure portal interface for a virtual network named 'myVnet'. On the left, a sidebar menu includes options like Activity log, Access control (IAM), Tags, and Diagnose and solve problems. The main content area displays the 'Overview' tab for the resource group 'dualstackinplaceupgrade'. It shows the location as East US, the address space as 10.0.0.0/16, and DNS servers as Azure provided DNS service. Below this, a table titled 'Connected devices' lists three entries: MyNic1, MyNic2, and MyNic3, all categorized as Network interface. The IP Address column shows IPv6 addresses: 10.0.2.4 for MyNic1, 10.0.2.5 for MyNic2, and 10.0.2.6 for MyNic3. The Subnet column indicates they are all part of 'mySubnet'. A red box highlights the IP address of MyNic1 (ace:cab:deca::4).

Device	Type	IP Address	Subnet
MyNic1	Network interface	10.0.2.4	mySubnet
MyNic2	Network interface	10.0.2.5	mySubnet
MyNic3	Network interface	10.0.2.6	mySubnet
MyNic1	Network interface	ace:cab:deca::4	mySubnet
MyNic2	Network interface	ace:cab:deca::5	mySubnet
MyNic3	Network interface	ace:cab:deca::6	mySubnet

#### NOTE

The IPv6 for Azure virtual network is available in the Azure portal in read-only for this preview release.

## Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) command to remove the resource group, VM, and all related resources.

```
Remove-AzResourceGroup -Name MyAzureResourceGroupSLB
```

## Next steps

In this article, you updated an existing Standard Load Balancer with a IPv4 frontend IP configuration to a dual stack (IPv4 and IPv6) configuration. You also added IPv6 configurations to the NICs of the VMs in the backend pool and to the Virtual Network that hosts them. To learn more about IPv6 support in Azure virtual networks, see [What is IPv6 for Azure Virtual Network?](#)

# Add IPv6 to an IPv4 application in Azure virtual network - Azure CLI (Preview)

10/27/2019 • 4 minutes to read • [Edit Online](#)

This article shows you how to add IPv6 addresses to an application that is using IPv4 public IP address in an Azure virtual network for a Standard Load Balancer using Azure CLI. The in-place upgrade includes a virtual network and subnet, a Standard Load Balancer with IPv4 + IPV6 frontend configurations, VMs with NICs that have a IPv4 + IPv6 configurations, network security group, and public IPs.

## IMPORTANT

IPv6 support for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you decide to install and use Azure CLI locally instead, this quickstart requires you to use Azure CLI version 2.0.28 or later. To find your installed version, run `az --version`. See [Install Azure CLI](#) for install or upgrade info.

## Prerequisites

## Register the service

Before you deploy a dual stack application in Azure, you must configure your subscription for this preview feature using the following Azure CLI:

```
az feature register --name AllowIPv6VirtualNetwork --namespace Microsoft.Network  
az feature register --name AllowIPv6CAOnStandardLB --namespace Microsoft.Network
```

It takes up to 30 minutes for feature registration to complete. You can check your registration status by running the following Azure CLI command:

```
az feature show --name AllowIPv6VirtualNetwork --namespace Microsoft.Network  
az feature show --name AllowIPv6CAOnStandardLB --namespace Microsoft.Network
```

After the registration is complete, run the following command:

```
az provider register --namespace Microsoft.Network
```

## Create a Standard Load Balancer

This article assumes that you deployed a Standard Load Balancer as described in [Quickstart: Create a Standard Load Balancer - Azure CLI](#).

## Create IPv6 addresses

Create public IPv6 address with with [az network public-ip create](#) for your Standard Load Balancer. The following example creates an IPv6 public IP address named *PublicIP\_v6* in the *myResourceGroupSLB* resource group:

```
az network public-ip create \  
--name PublicIP_v6 \  
--resource-group MyResourceGroupSLB \  
--location EastUS \  
--sku Standard \  
--allocation-method static \  
--version IPv6
```

## Configure IPv6 load balancer frontend

Configure the load balancer with the new IPv6 IP address using [az network lb frontend-ip create](#) as follows:

```
az network lb frontend-ip create \  
--lb-name myLoadBalancer \  
--name dsLbFrontEnd_v6 \  
--resource-group MyResourceGroupSLB \  
--public-ip-address PublicIP_v6
```

## Configure IPv6 load balancer backend pool

Create the backend pool for NICs with IPv6 addresses using [az network lb address-pool create](#) as follows:

```
az network lb address-pool create \
--lb-name myLoadBalancer \
--name dsLbBackEndPool_v6 \
--resource-group MyResourceGroupSLB
```

## Configure IPv6 load balancer rules

Create IPv6 load balancer rules with [az network lb rule create](#).

```
az network lb rule create \
--lb-name myLoadBalancer \
--name dsLbRule_v6 \
--resource-group MyResourceGroupSLB \
--frontend-ip-name dsLbFrontEnd_v6 \
--protocol Tcp \
--frontend-port 80 \
--backend-port 80 \
--backend-pool-name dsLbBackEndPool_v6
```

## Add IPv6 address ranges

Add IPv6 address ranges to the virtual network and subnet hosting the load balancer as follows:

```
az network vnet update \
--name myVnet \
--resource-group MyResourceGroupSLB \
--address-prefixes "10.0.0.0/16" "ace:cab:deca::/48"

az network vnet subnet update \
--vnet-name myVnet \
--name mySubnet \
--resource-group MyResourceGroupSLB \
--address-prefixes "10.0.0.0/24" "ace:cab:deca:deed::/64"
```

## Add IPv6 configuration to NICs

Configure the VM NICs with an IPv6 address using [az network nic ip-config create](#) as follows:

```

az network nic ip-config create \
--name dsIp6Config_NIC1 \
--nic-name myNicVM1 \
--resource-group MyResourceGroupSLB \
--vnet-name myVnet \
--subnet mySubnet \
--private-ip-address-version IPv6 \
--lb-address-pools dsLbBackEndPool_v6 \
--lb-name dsLB

az network nic ip-config create \
--name dsIp6Config_NIC2 \
--nic-name myNicVM2 \
--resource-group MyResourceGroupSLB \
--vnet-name myVnet \
--subnet mySubnet \
--private-ip-address-version IPv6 \
--lb-address-pools dsLbBackEndPool_v6 \
--lb-name myLoadBalancer

az network nic ip-config create \
--name dsIp6Config_NIC3 \
--nic-name myNicVM3 \
--resource-group MyResourceGroupSLB \
--vnet-name myVnet \
--subnet mySubnet \
--private-ip-address-version IPv6 \
--lb-address-pools dsLbBackEndPool_v6 \
--lb-name myLoadBalancer

```

## View IPv6 dual stack virtual network in Azure portal

You can view the IPv6 dual stack virtual network in Azure portal as follows:

1. In the portal's search bar, enter *myVnet*.
2. When **myVnet** appears in the search results, select it. This launches the **Overview** page of the dual stack virtual network named *myVNet*. The dual stack virtual network shows the three NICs with both IPv4 and IPv6 configurations located in the dual stack subnet named *mySubnet*.

Device	Type	IP Address	Subnet
MyNic1	Network interface	10.0.2.4	mySubnet
MyNic2	Network interface	10.0.2.5	mySubnet
MyNic3	Network interface	10.0.2.6	mySubnet
MyNic1	Network interface	ace:cab:deca:4	mySubnet
MyNic2	Network interface	ace:cab:deca:5	mySubnet
MyNic3	Network interface	ace:cab:deca:6	mySubnet

**NOTE**

The IPv6 for Azure virtual network is available in the Azure portal in read-only for this preview release.

## Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) command to remove the resource group, VM, and all related resources.

```
Remove-AzResourceGroup -Name MyAzureResourceGroupSLB
```

## Next steps

In this article, you updated an existing Standard Load Balancer with a IPv4 frontend IP configuration to a dual stack (IPv4 and IPv6) configuration. You also added IPv6 configurations to the NICs of the VMs in the backend pool. To learn more about IPv6 support in Azure virtual networks, see [What is IPv6 for Azure Virtual Network?](#)

# Deploy an IPv6 dual stack application using Basic Load Balancer - PowerShell (Preview)

12/19/2019 • 9 minutes to read • [Edit Online](#)

This article shows you how to deploy a dual stack (IPv4 + IPv6) application with Basic Load Balancer using Azure PowerShell that includes a dual stack virtual network and subnet, a Basic Load Balancer with dual (IPv4 + IPv6) front-end configurations, VMs with NICs that have a dual IP configuration, network security group, and public IPs.

To deploy a dual stack (IPV4 + IPV6) application using Standard Load Balancer, see [Deploy an IPv6 dual stack application with Standard Load Balancer using Azure PowerShell](#).

## IMPORTANT

IPv6 support for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 6.9.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a

connection with Azure.

## Prerequisites

Before you deploy a dual stack application in Azure, you must configure your subscription for this preview feature using the following Azure PowerShell:

Register as follows:

```
Register-AzProviderFeature -FeatureName AllowIPv6VirtualNetwork -ProviderNamespace Microsoft.Network  
Register-AzProviderFeature -FeatureName AllowIPv6CAOnStandardLB -ProviderNamespace Microsoft.Network
```

It takes up to 30 minutes for feature registration to complete. You can check your registration status by running the following Azure PowerShell command: Check on the registration as follows:

```
Get-AzProviderFeature -FeatureName AllowIPv6VirtualNetwork -ProviderNamespace Microsoft.Network  
Get-AzProviderFeature -FeatureName AllowIPv6CAOnStandardLB -ProviderNamespace Microsoft.Network
```

After the registration is complete, run the following command:

```
Register-AzResourceProvider -ProviderNamespace Microsoft.Network
```

## Create a resource group

Before you can create your dual-stack virtual network, you must create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myRGDualStack* in the *east us* location:

```
$rg = New-AzResourceGroup `  
-ResourceGroupName "dsRG1" `  
-Location "east us"
```

## Create IPv4 and IPv6 public IP addresses

To access your virtual machines from the Internet, you need IPv4 and IPv6 public IP addresses for the load balancer. Create public IP addresses with [New-AzPublicIpAddress](#). The following example creates IPv4 and IPv6 public IP address named *dsPublicIP\_v4* and *dsPublicIP\_v6* in the *dsRG1* resource group:

```
$PublicIP_v4 = New-AzPublicIpAddress `  
-Name "dsPublicIP_v4" `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-AllocationMethod Dynamic `  
-IpAddressVersion IPv4  
  
$PublicIP_v6 = New-AzPublicIpAddress `  
-Name "dsPublicIP_v6" `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-AllocationMethod Dynamic `  
-IpAddressVersion IPv6
```

To access your virtual machines using a RDP connection, create a IPV4 public IP addresses for the virtual machines with [New-AzPublicIpAddress](#).

```
$RdpPublicIP_1 = New-AzPublicIpAddress ` 
-Name "RdpPublicIP_1" ` 
-ResourceGroupName $rg.ResourceGroupName ` 
-Location $rg.Location ` 
-AllocationMethod Dynamic ` 
-IpAddressVersion IPv4

$RdpPublicIP_2 = New-AzPublicIpAddress ` 
-Name "RdpPublicIP_2" ` 
-ResourceGroupName $rg.ResourceGroupName ` 
-Location $rg.Location ` 
-AllocationMethod Dynamic ` 
-IpAddressVersion IPv4
```

## Create Basic Load Balancer

In this section, you configure dual front-end IP (IPv4 and IPv6) and the back-end address pool for the load balancer and then create a Basic Load Balancer.

### Create front-end IP

Create a front-end IP with [New-AzLoadBalancerFrontendIpConfig](#). The following example creates IPv4 and IPv6 front-end IP configurations named *dsLbFrontEnd\_v4* and *dsLbFrontEnd\_v6*:

```
$frontendIPv4 = New-AzLoadBalancerFrontendIpConfig ` 
-Name "dsLbFrontEnd_v4" ` 
-PublicIpAddress $PublicIP_v4

$frontendIPv6 = New-AzLoadBalancerFrontendIpConfig ` 
-Name "dsLbFrontEnd_v6" ` 
-PublicIpAddress $PublicIP_v6
```

### Configure back-end address pool

Create a back-end address pool with [New-AzLoadBalancerBackendAddressPoolConfig](#). The VMs attach to this back-end pool in the remaining steps. The following example creates back-end address pools named *dsLbBackEndPool\_v4* and *dsLbBackEndPool\_v6* to include VMs with both IPV4 and IPV6 NIC configurations:

```
$backendPoolv4 = New-AzLoadBalancerBackendAddressPoolConfig ` 
-Name "dsLbBackEndPool_v4"

$backendPoolv6 = New-AzLoadBalancerBackendAddressPoolConfig ` 
-Name "dsLbBackEndPool_v6"
```

### Create a health probe

Use [Add-AzLoadBalancerProbeConfig](#) to create a health probe to monitor the health of the VMs.

```
$probe = New-AzLoadBalancerProbeConfig -Name MyProbe -Protocol tcp -Port 3389 -IntervalInSeconds 15 -ProbeCount 2
```

### Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port. To make sure only healthy VMs receive traffic, you can optionally define a health probe. Basic load balancer uses an IPv4 probe to assess health for both IPv4 and IPv6 endpoints on the VMs. Standard load balancer includes support for explicitly IPv6 health probes.

Create a load balancer rule with [Add-AzLoadBalancerRuleConfig](#). The following example creates load balancer rules named *dsLBrule\_v4* and *dsLBrule\_v6* and balances traffic on TCP port 80 to the IPv4 and IPv6 frontend IP configurations:

```
$lbrule_v4 = New-AzLoadBalancerRuleConfig `  
-Name "dsLBrule_v4" `  
-FrontendIpConfiguration $frontendIPv4 `  
-BackendAddressPool $backendPoolv4 `  
-Protocol Tcp `  
-FrontendPort 80 `  
-BackendPort 80 `  
-probe $probe  
  
$lbrule_v6 = New-AzLoadBalancerRuleConfig `  
-Name "dsLBrule_v6" `  
-FrontendIpConfiguration $frontendIPv6 `  
-BackendAddressPool $backendPoolv6 `  
-Protocol Tcp `  
-FrontendPort 80 `  
-BackendPort 80 `  
-probe $probe
```

## Create load balancer

Create the Basic Load Balancer with [New-AzLoadBalancer](#). The following example creates a public Basic Load Balancer named *myLoadBalancer* using the IPv4 and IPv6 frontend IP configurations, backend pools, and load-balancing rules that you created in the preceding steps:

```
$lb = New-AzLoadBalancer `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-Name "MyLoadBalancer" `  
-Sku "Basic" `  
-FrontendIpConfiguration $frontendIPv4,$frontendIPv6 `  
-BackendAddressPool $backendPoolv4,$backendPoolv6 `  
-LoadBalancingRule $lbrule_v4,$lbrule_v6
```

## Create network resources

Before you deploy some VMs and can test your balancer, you must create supporting network resources - availability set, network security group, virtual network, and virtual NICs.

### Create an availability set

To improve the high availability of your app, place your VMs in an availability set.

Create an availability set with [New-AzAvailabilitySet](#). The following example creates an availability set named *myAvailabilitySet*:

```
$avset = New-AzAvailabilitySet `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-Name "dsAVset" `  
-PlatformFaultDomainCount 2 `  
-PlatformUpdateDomainCount 2 `  
-Sku aligned
```

### Create network security group

Create a network security group for the rules that will govern inbound and outbound communication in your

VNET.

#### Create a network security group rule for port 3389

Create a network security group rule to allow RDP connections through port 3389 with [New-AzNetworkSecurityRuleConfig](#).

```
$rule1 = New-AzNetworkSecurityRuleConfig `  
-Name 'myNetworkSecurityGroupRuleRDP' `  
-Description 'Allow RDP' `  
-Access Allow `  
-Protocol Tcp `  
-Direction Inbound `  
-Priority 100 `  
-SourceAddressPrefix * `  
-SourcePortRange * `  
-DestinationAddressPrefix * `  
-DestinationPortRange 3389
```

#### Create a network security group rule for port 80

Create a network security group rule to allow internet connections through port 80 with [New-AzNetworkSecurityRuleConfig](#).

```
$rule2 = New-AzNetworkSecurityRuleConfig `  
-Name 'myNetworkSecurityGroupRuleHTTP' `  
-Description 'Allow HTTP' `  
-Access Allow `  
-Protocol Tcp `  
-Direction Inbound `  
-Priority 200 `  
-SourceAddressPrefix * `  
-SourcePortRange 80 `  
-DestinationAddressPrefix * `  
-DestinationPortRange 80
```

#### Create a network security group

Create a network security group with [New-AzNetworkSecurityGroup](#).

```
$nsg = New-AzNetworkSecurityGroup `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-Name "dsNSG1" `  
-SecurityRules $rule1,$rule2
```

#### Create a virtual network

Create a virtual network with [New-AzVirtualNetwork](#). The following example creates a virtual network named *myVnet* with *mySubnet*:

```
# Create dual stack subnet  
$subnet = New-AzVirtualNetworkSubnetConfig `  
-Name "dsSubnet" `  
-AddressPrefix "10.0.0.0/24","ace:cab:deca:deed::/64"  
  
# Create the virtual network  
$vnet = New-AzVirtualNetwork `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-Name "dsVnet" `  
-AddressPrefix "10.0.0.0/16","ace:cab:deca::/48" `  
-Subnet $subnet
```

## Create NICs

Create virtual NICs with [New-AzNetworkInterface](#). The following example creates two virtual NICs both with IPv4 and IPv6 configurations. (One virtual NIC for each VM you create for your app in the following steps).

```
$Ip4Config=New-AzNetworkInterfaceIpConfig `  
-Name dsIp4Config `  
-Subnet $vnet.subnets[0] `  
-PrivateIpAddressVersion IPv4 `  
-LoadBalancerBackendAddressPool $backendPoolv4 `  
-PublicIpAddress $RdpPublicIP_1  
  
$Ip6Config=New-AzNetworkInterfaceIpConfig `  
-Name dsIp6Config `  
-Subnet $vnet.subnets[0] `  
-PrivateIpAddressVersion IPv6 `  
-LoadBalancerBackendAddressPool $backendPoolv6  
  
$NIC_1 = New-AzNetworkInterface `  
-Name "dsNIC1" `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-NetworkSecurityGroupId $nsg.Id `  
-IpConfiguration $Ip4Config,$Ip6Config  
  
$Ip4Config=New-AzNetworkInterfaceIpConfig `  
-Name dsIp4Config `  
-Subnet $vnet.subnets[0] `  
-PrivateIpAddressVersion IPv4 `  
-LoadBalancerBackendAddressPool $backendPoolv4 `  
-PublicIpAddress $RdpPublicIP_2  
  
$NIC_2 = New-AzNetworkInterface `  
-Name "dsNIC2" `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-NetworkSecurityGroupId $nsg.Id `  
-IpConfiguration $Ip4Config,$Ip6Config
```

## Create virtual machines

Set an administrator username and password for the VMs with [Get-Credential](#):

```
$cred = get-credential -Message "DUAL STACK VNET SAMPLE: Please enter the Administrator credential to log into the VMs."
```

Now you can create the VMs with [New-AzVM](#). The following example creates two VMs and the required virtual network components if they do not already exist.

```

$vmsize = "Standard_A2"
$imagePublisher = "MicrosoftWindowsServer"
$imageOffer = "WindowsServer"
$imageSKU = "2019-Datacenter"

$vmName= "dsVM1"
$VMconfig1 = New-AzVMConfig -VMName $vmName -VMSize $vmsize -AvailabilitySetId $avset.Id 3> $null | Set-
AzVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred -ProvisionVMAgent 3> $null | Set-
AzVMSourceImage -PublisherName $ImagePublisher -Offer $imageOffer -Skus $imageSKU -Version "latest" 3> $null |
Set-AzVMOSDisk -Name "$vmName.vhd" -CreateOption fromImage 3> $null | Add-AzVMNetworkInterface -Id $NIC_1.Id
3> $null
$VM1 = New-AzVM -ResourceGroupName $rg.ResourceGroupName -Location $rg.Location -VM $VMconfig1

$vmName= "dsVM2"
$VMconfig2 = New-AzVMConfig -VMName $vmName -VMSize $vmsize -AvailabilitySetId $avset.Id 3> $null | Set-
AzVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred -ProvisionVMAgent 3> $null | Set-
AzVMSourceImage -PublisherName $ImagePublisher -Offer $imageOffer -Skus $imageSKU -Version "latest" 3> $null |
Set-AzVMOSDisk -Name "$vmName.vhd" -CreateOption fromImage 3> $null | Add-AzVMNetworkInterface -Id $NIC_2.Id
3> $null
$VM2 = New-AzVM -ResourceGroupName $rg.ResourceGroupName -Location $rg.Location -VM $VMconfig2

```

## Determine IP addresses of the IPv4 and IPv6 endpoints

Get all Network Interface Objects in the resource group to summarize the IP's used in this deployment with `get-AzNetworkInterface`. Also, get the Load Balancer's frontend addresses of the IPv4 and IPv6 endpoints with `get-AzpublicIpAddress`.

```

$rgName= "dsRG1"
$NICsInRG= get-AzNetworkInterface -resourceGroupName $rgName
write-host `nSummary of IPs in this Deployment:
write-host ****
foreach ($NIC in $NICsInRG) {

    $VMid= $NIC.virtualmachine.id
    $VMnamebits= $VMid.split("/")
    $VMname= $VMnamebits[($VMnamebits.count-1)]
    write-host `nPrivate IP addresses for $VMname
    $IPconfigsInNIC= $NIC.IPConfigurations
    foreach ($IPconfig in $IPconfigsInNIC) {

        $IPAddress= $IPconfig.privateipaddress
        write-host "      "$IPAddress
        IF ($IPconfig.PublicIpAddress.ID) {

            $IDbits= ($IPconfig.PublicIpAddress.ID).split("/")
            $PipName= $IDbits[($IDbits.count-1)]
            $PipObject= get-azPublicIpAddress -name $PipName -resourceGroup $rgName
            write-host "      "RDP address: $PipObject.IpAddress
        }
    }
}

write-host `nPublic IP addresses on Load Balancer:
(get-AzpublicIpAddress -resourcegroupname $rgName | where { $_.name -notlike "RdpPublicIP*" }).IpAddress

```

The following figure shows a sample output that lists the private IPv4 and IPv6 addresses of the two VMs, and the frontend IPv4 and IPv6 IP addresses of the Load Balancer.

## Summary of IPs in this Deployment:

---

### Private IP addresses for DsVM0

10.0.0.4  
RDP address: 40.118.190.180  
ace:cab:deca:deed::4

### Private IP addresses for DsVM1

10.0.0.5  
RDP address: 40.118.190.195  
ace:cab:deca:deed::5

### Public IP addresses on Load Balancer:

40.118.190.251  
2a01:111:f100:3000::a83e:19c3

## View IPv6 dual stack virtual network in Azure portal

You can view the IPv6 dual stack virtual network in Azure portal as follows:

1. In the portal's search bar, enter *dsVnet*.
2. When **myVirtualNetwork** appears in the search results, select it. This launches the **Overview** page of the dual stack virtual network named *dsVnet*. The dual stack virtual network shows the two NICs with both IPv4 and IPv6 configurations located in the dual stack subnet named *dsSubnet*.

Home > Resource groups > DsRG02 > VNET

**VNET**  
Virtual network

Search (Ctrl+/  
)

Overview      Refresh      Move      Delete

Resource group (change)  
DsRG02

Address space  
10.0.0.0/16, 1 more

Location  
West US

Subscription (change)

DNS servers  
Azure provided DNS service

Subscription ID

Tags (change)  
Click here to add tags

Connected devices

DEVICE	TYPE	IP ADDRESS	SUBNET
DsVM0	Network interface	10.0.0.4	DualStackSubnet
DsVM0	Network interface	ace:cab:deca:deed::4	DualStackSubnet
DsVM1	Network interface	10.0.0.5	DualStackSubnet
DsVM1	Network interface	ace:cab:deca:deed::5	DualStackSubnet

### NOTE

The IPv6 for Azure virtual network is available in the Azure portal in read-only for this preview release.

## Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) command to remove the resource group, VM, and all related resources.

```
Remove-AzResourceGroup -Name dsRG1
```

## Next steps

In this article, you created a Basic Load Balancer with a dual frontend IP configuration (IPv4 and IPv6). You also created a two virtual machines that included NICs with dual IP configurations (IPV4 + IPV6) that were added to the back-end pool of the load balancer. To learn more about IPv6 support in Azure virtual networks, see [What is IPv6 for Azure Virtual Network?](#)

# Deploy an IPv6 dual stack application using Basic Load Balancer - CLI (Preview)

12/19/2019 • 9 minutes to read • [Edit Online](#)

This article shows you how to deploy a dual stack (IPv4 + IPv6) application with Basic Load Balancer using Azure CLI that includes a dual stack virtual network with a dual stack subnet, a Basic Load Balancer with dual (IPv4 + IPv6) front-end configurations, VMs with NICs that have a dual IP configuration, dual network security group rules, and dual public IPs.

To deploy a dual stack (IPV4 + IPV6) application using Standard Load Balancer, see [Deploy an IPv6 dual stack application with Standard Load Balancer using Azure CLI](#).

## IMPORTANT

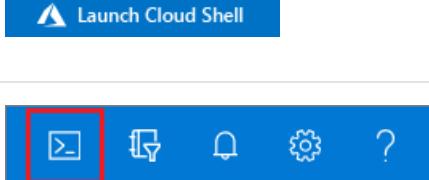
IPv6 dual stack for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

If you don't have an Azure subscription, create a [free account](#) now.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you decide to install and use Azure CLI locally instead, this quickstart requires you to use Azure CLI version 2.0.49 or later. To find your installed version, run `az --version`. See [Install Azure CLI](#) for install or upgrade info.

## Prerequisites

To use the IPv6 for Azure virtual network feature, you must configure your subscription using Azure CLI as follows:

```
az feature register --name AllowIPv6VirtualNetwork --namespace Microsoft.Network  
az feature register --name AllowIPv6CAOnStandardLB --namespace Microsoft.Network
```

It takes up to 30 minutes for feature registration to complete. You can check your registration status by running the following Azure CLI command:

```
az feature show --name AllowIPv6VirtualNetwork --namespace Microsoft.Network  
az feature show --name AllowIPv6CAOnStandardLB --namespace Microsoft.Network
```

After the registration is complete, run the following command:

```
az provider register --namespace Microsoft.Network
```

## Create a resource group

Before you can create your dual-stack virtual network, you must create a resource group with `az group create`. The following example creates a resource group named *DsResourceGroup01* in the *eastus* location:

```
az group create \  
--name DsResourceGroup01 \  
--location eastus
```

## Create IPv4 and IPv6 public IP addresses for load balancer

To access your IPv4 and IPv6 endpoints on the Internet, you need IPv4 and IPv6 public IP addresses for the load balancer. Create a public IP address with `az network public-ip create`. The following example creates IPv4 and IPv6 public IP address named *dsPublicIP\_v4* and *dsPublicIP\_v6* in the *DsResourceGroup01* resource group:

```
# Create an IPV4 IP address  
az network public-ip create \  
--name dsPublicIP_v4 \  
--resource-group DsResourceGroup01 \  
--location eastus \  
--sku BASIC \  
--allocation-method dynamic \  
--version IPv4  
  
# Create an IPV6 IP address  
az network public-ip create \  
--name dsPublicIP_v6 \  
--resource-group DsResourceGroup01 \  
--location eastus \  
--sku BASIC \  
--allocation-method dynamic \  
--version IPv6
```

## Create public IP addresses for VMs

To remotely access your VMs on the internet, you need IPv4 public IP addresses for the VMs. Create a public IP address with [az network public-ip create](#).

```
az network public-ip create \
--name dsVM0_remote_access \
--resource-group DsResourceGroup01 \
--location eastus \
--sku BASIC \
--allocation-method dynamic \
--version IPv4

az network public-ip create \
--name dsVM1_remote_access \
--resource-group DsResourceGroup01 \
--location eastus \
--sku BASIC \
--allocation-method dynamic \
--version IPv4
```

## Create Basic Load Balancer

In this section, you configure dual frontend IP (IPv4 and IPv6) and the back-end address pool for the load balancer and then create a Basic Load Balancer.

### Create load balancer

Create the Basic Load Balancer with [az network lb create](#) named **dsLB** that includes a frontend pool named **dsLbFrontEnd\_v4**, a backend pool named **dsLbBackEndPool\_v4** that is associated with the IPv4 public IP address **dsPublicIP\_v4** that you created in the preceding step.

```
az network lb create \
--name dsLB \
--resource-group DsResourceGroup01 \
--sku Basic \
--location eastus \
--frontend-ip-name dsLbFrontEnd_v4 \
--public-ip-address dsPublicIP_v4 \
--backend-pool-name dsLbBackEndPool_v4
```

### Create IPv6 frontend

Create an IPV6 frontend IP with [az network lb frontend-ip create](#). The following example creates a frontend IP configuration named *dsLbFrontEnd\_v6* and attaches the *dsPublicIP\_v6* address:

```
az network lb frontend-ip create \
--lb-name dsLB \
--name dsLbFrontEnd_v6 \
--resource-group DsResourceGroup01 \
--public-ip-address dsPublicIP_v6
```

### Configure IPv6 back-end address pool

Create a IPv6 back-end address pools with [az network lb address-pool create](#). The following example creates back-end address pool named *dsLbBackEndPool\_v6* to include VMs with IPv6 NIC configurations:

```
az network lb address-pool create \
--lb-name dsLB \
--name dsLbBackEndPool_v6 \
--resource-group DsResourceGroup01
```

## Create a health probe

Create a health probe with [az network lb probe create](#) to monitor the health of the virtual machines.

```
az network lb probe create -g DsResourceGroup01 --lb-name dsLB -n dsProbe --protocol tcp --port 3389
```

## Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port.

Create a load balancer rule with [az network lb rule create](#). The following example creates load balancer rules named *dsLBrule\_v4* and *dsLBrule\_v6* and balances traffic on *TCP* port *80* to the IPv4 and IPv6 frontend IP configurations:

```
az network lb rule create \
--lb-name dsLB \
--name dsLBrule_v4 \
--resource-group DsResourceGroup01 \
--frontend-ip-name dsLbFrontEnd_v4 \
--protocol Tcp \
--frontend-port 80 \
--backend-port 80 \
--probe-name dsProbe \
--backend-pool-name dsLbBackEndPool_v4
```

```
az network lb rule create \
--lb-name dsLB \
--name dsLBrule_v6 \
--resource-group DsResourceGroup01 \
--frontend-ip-name dsLbFrontEnd_v6 \
--protocol Tcp \
--frontend-port 80 \
--backend-port 80 \
--probe-name dsProbe \
--backend-pool-name dsLbBackEndPool_v6
```

# Create network resources

Before you deploy some VMs, you must create supporting network resources - availability set, network security group, virtual network, and virtual NICs.

## Create an availability set

To improve the availability of your app, place your VMs in an availability set.

Create an availability set with [az vm availability-set create](#). The following example creates an availability set named *dsAVset*:

```
az vm availability-set create \
--name dsAVset \
--resource-group DsResourceGroup01 \
--location eastus \
--platform-fault-domain-count 2 \
--platform-update-domain-count 2
```

## Create network security group

Create a network security group for the rules that will govern inbound and outbound communication in your VNET.

### Create a network security group

Create a network security group with [az network nsg create](#)

```
az network nsg create \
--name dsNSG1 \
--resource-group DsResourceGroup01 \
--location eastus
```

### Create a network security group rule for inbound and outbound connections

Create a network security group rule to allow RDP connections through port 3389, internet connection through port 80, and for outbound connections with [az network nsg rule create](#).

```

# Create inbound rule for port 3389
az network nsg rule create \
--name allowRdpIn \
--nsg-name dsNSG1 \
--resource-group DsResourceGroup01 \
--priority 100 \
--description "Allow Remote Desktop In" \
--access Allow \
--protocol "*" \
--direction Inbound \
--source-address-prefixes "*" \
--source-port-ranges "*" \
--destination-address-prefixes "*" \
--destination-port-ranges 3389

# Create inbound rule for port 80
az network nsg rule create \
--name allowHTTPIn \
--nsg-name dsNSG1 \
--resource-group DsResourceGroup01 \
--priority 200 \
--description "Allow HTTP In" \
--access Allow \
--protocol "*" \
--direction Inbound \
--source-address-prefixes "*" \
--source-port-ranges 80 \
--destination-address-prefixes "*" \
--destination-port-ranges 80

# Create outbound rule

az network nsg rule create \
--name allowAllOut \
--nsg-name dsNSG1 \
--resource-group DsResourceGroup01 \
--priority 300 \
--description "Allow All Out" \
--access Allow \
--protocol "*" \
--direction Outbound \
--source-address-prefixes "*" \
--source-port-ranges "*" \
--destination-address-prefixes "*" \
--destination-port-ranges "*"

```

## Create a virtual network

Create a virtual network with [az network vnet create](#). The following example creates a virtual network named *dsVNET* with subnets *dsSubNET\_v4* and *dsSubNET\_v6*:

```
# Create the virtual network
az network vnet create \
--name dsVNET \
--resource-group DsResourceGroup01 \
--location eastus \
--address-prefixes "10.0.0.0/16" "ace:cab:deca::/48"

# Create a single dual stack subnet

az network vnet subnet create \
--name dsSubNET \
--resource-group DsResourceGroup01 \
--vnet-name dsVNET \
--address-prefixes "10.0.0.0/24" "ace:cab:deca:deed::/64" \
--network-security-group dsNSG1
```

## Create NICs

Create virtual NICs for each VM with [az network nic create](#). The following example creates a virtual NIC for each VM. Each NIC has two IP configurations (1 IPv4 config, 1 IPv6 config). You create the IPV6 configuration with [az network nic ip-config create](#).

```

# Create NICs

az network nic create \
--name dsNIC0 \
--resource-group DsResourceGroup01 \
--network-security-group dsNSG1 \
--vnet-name dsVNET \
--subnet dsSubNet \
--private-ip-address-version IPv4 \
--lb-address-pools dsLbBackEndPool_v4 \
--lb-name dsLB \
--public-ip-address dsVM0_remote_access

az network nic create \
--name dsNIC1 \
--resource-group DsResourceGroup01 \
--network-security-group dsNSG1 \
--vnet-name dsVNET \
--subnet dsSubNet \
--private-ip-address-version IPv4 \
--lb-address-pools dsLbBackEndPool_v4 \
--lb-name dsLB \
--public-ip-address dsVM1_remote_access

# Create IPV6 configurations for each NIC

az network nic ip-config create \
--name dsIp6Config_NIC0 \
--nic-name dsNIC0 \
--resource-group DsResourceGroup01 \
--vnet-name dsVNET \
--subnet dsSubNet \
--private-ip-address-version IPv6 \
--lb-address-pools dsLbBackEndPool_v6 \
--lb-name dsLB

az network nic ip-config create \
--name dsIp6Config_NIC1 \
--nic-name dsNIC1 \
--resource-group DsResourceGroup01 \
--vnet-name dsVNET \
--subnet dsSubNet \
--private-ip-address-version IPv6 \
--lb-address-pools dsLbBackEndPool_v6 \
--lb-name dsLB

```

## Create virtual machines

Create the VMs with [az vm create](#). The following example creates two VMs and the required virtual network components if they do not already exist.

Create virtual machine *dsVM0* as follows:

```

az vm create \
--name dsVM0 \
--resource-group DsResourceGroup01 \
--nics dsNIC0 \
--size Standard_A2 \
--availability-set dsAVset \
--image MicrosoftWindowsServer:WindowsServer:2019-Datacenter:latest

```

Create virtual machine *dsVM1* as follows:

```
az vm create \
--name dsVM1 \
--resource-group DsResourceGroup01 \
--nics dsNIC1 \
--size Standard_A2 \
--availability-set dsAVset \
--image MicrosoftWindowsServer:WindowsServer:2019-Datacenter:latest
```

## View IPv6 dual stack virtual network in Azure portal

You can view the IPv6 dual stack virtual network in Azure portal as follows:

1. In the portal's search bar, enter *dsVnet*.
2. When **myVirtualNetwork** appears in the search results, select it. This launches the **Overview** page of the dual stack virtual network named *dsVnet*. The dual stack virtual network shows the two NICs with both IPv4 and IPv6 configurations located in the dual stack subnet named *dsSubnet*.

The screenshot shows the Azure portal interface for managing a virtual network. The left sidebar lists various settings and subnets. The main content area shows the overview of a virtual network named 'dsVnet'.

DEVICE	TYPE	IP ADDRESS	SUBNET
DsVM0	Network interface	10.0.0.4	DualStackSubnet
DsVM0	Network interface	ace:cab:deca:deed::4	DualStackSubnet
DsVM1	Network interface	10.0.0.5	DualStackSubnet
DsVM1	Network interface	ace:cab:deca:deed::5	DualStackSubnet

### NOTE

The IPv6 for Azure virtual network is available in the Azure portal in read-only for this preview release.

## Clean up resources

When no longer needed, you can use the [az group delete](#) command to remove the resource group, VM, and all related resources.

```
az group delete --name DsResourceGroup01
```

## Next steps

In this article, you created a Basic Load Balancer with a dual frontend IP configuration (IPv4 and IPv6). You also created a two virtual machines that included NICs with dual IP configurations (IPV4 + IPV6) that were added to the

back-end pool of the load balancer. To learn more about IPv6 support in Azure virtual networks, see [What is IPv6 for Azure Virtual Network?](#)

# Deploy an IPv6 dual stack application with Basic Load Balancer in Azure - Template (Preview)

8/23/2019 • 2 minutes to read • [Edit Online](#)

This article provides a list of IPv6 configuration tasks with the portion of the Azure Resource Manager VM template that applies to. Use the template described in this article to deploy a dual stack (IPv4 + IPv6) application with Basic Load Balancer that includes a dual stack virtual network with IPv4 and IPv6 subnets, a Basic Load Balancer with dual (IPv4 + IPv6) front-end configurations, VMs with NICs that have a dual IP configuration, network security group, and public IPs.

To deploy a dual stack (IPV4 + IPV6) application using Standard Load Balancer, see [Deploy an IPv6 dual stack application with Standard Load Balancer - Template](#).

## Required configurations

Search for the template sections in the template to see where they should occur.

### IPv6 addressSpace for the virtual network

Template section to add:

```
"addressSpace": {  
    "addressPrefixes": [  
        "[variables('vnetv4AddressRange')]",  
        "[variables('vnetv6AddressRange')]"  
    ]  
}
```

### IPv6 subnet within the IPv6 virtual network addressSpace

Template section to add:

```
{  
    "name": "V6Subnet",  
    "properties": {  
        "addressPrefix": "[variables('subnetv6AddressRange')]"  
    }  
}
```

### IPv6 configuration for the NIC

Template section to add:

```
{  
    "name": "ipconfig-v6",  
    "properties": {  
        "privateIPAllocationMethod": "Dynamic",  
        "privateIPAddressVersion": "IPv6",  
        "subnet": {  
            "id": "[variables('v6-subnet-id')]"  
        },  
        "loadBalancerBackendAddressPools": [  
            {  
                "id": "  
[concat(resourceId('Microsoft.Network/loadBalancers','loadBalancer'), '/backendAddressPools/LBBAP-v6')]"  
            }  
        ]  
    }  
}
```

## IPv6 network security group (NSG) rules

```
{  
    "name": "default-allow-rdp",  
    "properties": {  
        "description": "Allow RDP",  
        "protocol": "Tcp",  
        "sourcePortRange": "33819-33829",  
        "destinationPortRange": "5000-6000",  
        "sourceAddressPrefix": "ace:cab:deca:deed::/64",  
        "destinationAddressPrefix": "cab:ace:deca:deed::/64",  
        "access": "Allow",  
        "priority": 1003,  
        "direction": "Inbound"  
    }  
}
```

## Conditional configuration

If you're using a network virtual appliance, add IPv6 routes in the Route Table. Otherwise, this configuration is optional.

```
{  
    "type": "Microsoft.Network/routeTables",  
    "name": "v6route",  
    "apiVersion": "[variables('ApiVersion')]",  
    "location": "[resourceGroup().location]",  
    "properties": {  
        "routes": [  
            {  
                "name": "v6route",  
                "properties": {  
                    "addressPrefix": "ace:cab:deca:deed::/64",  
                    "nextHopType": "VirtualAppliance",  
                    "nextHopIpAddress": "deca:cab:ace:f00d::1"  
                }  
            }  
        ]  
    }  
}
```

## Optional configuration

### IPv6 Internet access for the virtual network

```
{  
    "name": "LBFE-v6",  
    "properties": {  
        "publicIPAddress": {  
            "id": "[resourceId('Microsoft.Network/publicIPAddresses','lbpublicip-v6')]"  
        }  
    }  
}
```

### IPv6 Public IP addresses

```
{  
    "apiVersion": "[variables('ApiVersion')]",  
    "type": "Microsoft.Network/publicIPAddresses",  
    "name": "lbpublicip-v6",  
    "location": "[resourceGroup().location]",  
    "properties": {  
        "publicIPAllocationMethod": "Dynamic",  
        "publicIPAddressVersion": "IPv6"  
    }  
}
```

## IPv6 Front end for Load Balancer

```
{  
    "name": "LBFE-v6",  
    "properties": {  
        "publicIPAddress": {  
            "id": "[resourceId('Microsoft.Network/publicIPAddresses', 'lbpublicip-v6')]"  
        }  
    }  
}
```

## IPv6 Back-end address pool for Load Balancer

```
"backendAddressPool": {  
    "id": "[concat(resourceId('Microsoft.Network/loadBalancers', 'loadBalancer'),  
    '/backendAddressPools/LBBAP-v6')]"  
},  
    "protocol": "Tcp",  
    "frontendPort": 8080,  
    "backendPort": 8080  
},  
    "name": "lbrule-v6"
```

## IPv6 load balancer rules to associate incoming and outgoing ports

```
{  
    "name": "ipconfig-v6",  
    "properties": {  
        "privateIPAllocationMethod": "Dynamic",  
        "privateIPAddressVersion": "IPv6",  
        "subnet": {  
            "id": "[variables('v6-subnet-id')]"  
        },  
        "loadBalancerBackendAddressPools": [  
            {  
                "id": "  
[concat(resourceId('Microsoft.Network/loadBalancers', 'loadBalancer'), '/backendAddressPools/LBBAP-v6')]"  
            }  
        ]  
    }  
}
```

## Sample VM template JSON

To deploy an IPv6 dual stack application with Basic Load Balancer in Azure virtual network using Azure Resource Manager template, view sample template [here](#).

## Next steps

You can find details about pricing for [public IP addresses](#), [network bandwidth](#), or [Load Balancer](#).

# Deploy an IPv6 dual stack application in Azure - PowerShell (Preview)

1/16/2020 • 9 minutes to read • [Edit Online](#)

This article shows you how to deploy a dual stack (IPv4 + IPv6) application using Standard Load Balancer in Azure that includes a dual stack virtual network and subnet, a Standard Load Balancer with dual (IPv4 + IPv6) front-end configurations, VMs with NICs that have a dual IP configuration, network security group, and public IPs.

## IMPORTANT

IPv6 support for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 6.9.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

## Prerequisites

Before you deploy a dual stack application in Azure, you must configure your subscription for this preview feature using the following Azure PowerShell:

Register as follows:

```
Register-AzProviderFeature -FeatureName AllowIPv6VirtualNetwork -ProviderNamespace Microsoft.Network  
Register-AzProviderFeature -FeatureName AllowIPv6CAOnStandardLB -ProviderNamespace Microsoft.Network
```

It takes up to 30 minutes for feature registration to complete. You can check your registration status by running the following Azure PowerShell command: Check on the registration as follows:

```
Get-AzProviderFeature -FeatureName AllowIPv6VirtualNetwork -ProviderNamespace Microsoft.Network  
Get-AzProviderFeature -FeatureName AllowIPv6CAOnStandardLB -ProviderNamespace Microsoft.Network
```

After the registration is complete, run the following command:

```
Register-AzResourceProvider -ProviderNamespace Microsoft.Network
```

## Create a resource group

Before you can create your dual-stack virtual network, you must create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myRGDualStack* in the *east us* location:

```
$rg = New-AzResourceGroup  
-ResourceGroupName "dsRG1"  
-Location "east us"
```

## Create IPv4 and IPv6 public IP addresses

To access your virtual machines from the Internet, you need IPv4 and IPv6 public IP addresses for the load balancer. Create public IP addresses with [New-AzPublicIpAddress](#). The following example creates IPv4 and IPv6 public IP address named *dsPublicIP\_v4* and *dsPublicIP\_v6* in the *dsRG1* resource group:

```
$PublicIP_v4 = New-AzPublicIpAddress  
-Name "dsPublicIP_v4"  
-ResourceGroupName $rg.ResourceGroupName  
-Location $rg.Location  
-AllocationMethod Static  
-IpAddressVersion IPv4  
-Sku Standard

$PublicIP_v6 = New-AzPublicIpAddress  
-Name "dsPublicIP_v6"  
-ResourceGroupName $rg.ResourceGroupName  
-Location $rg.Location  
-AllocationMethod Static  
-IpAddressVersion IPv6  
-Sku Standard
```

To access your virtual machines using a RDP connection, create a IPV4 public IP addresses for the virtual machines with [New-AzPublicIpAddress](#).

```
$RdpPublicIP_1 = New-AzPublicIpAddress `

-Name "RdpPublicIP_1" `

-ResourceGroupName $rg.ResourceGroupName `

-Location $rg.Location `

-AllocationMethod Static `

-Sku Standard `

-IpAddressVersion IPv4

$RdpPublicIP_2 = New-AzPublicIpAddress `

-Name "RdpPublicIP_2" `

-ResourceGroupName $rg.ResourceGroupName `

-Location $rg.Location `

-AllocationMethod Static `

-Sku Standard `

-IpAddressVersion IPv4
```

## Create Standard Load Balancer

In this section, you configure dual front-end IP (IPv4 and IPv6) and the back-end address pool for the load balancer and then create a Standard Load Balancer.

### Create front-end IP

Create a front-end IP with [New-AzLoadBalancerFrontendIpConfig](#). The following example creates IPv4 and IPv6 front-end IP configurations named *dsLbFrontEnd\_v4* and *dsLbFrontEnd\_v6*:

```
$frontendIPv4 = New-AzLoadBalancerFrontendIpConfig `

-Name "dsLbFrontEnd_v4" `

-PublicIpAddress $PublicIP_v4

$frontendIPv6 = New-AzLoadBalancerFrontendIpConfig `

-Name "dsLbFrontEnd_v6" `

-PublicIpAddress $PublicIP_v6
```

### Configure back-end address pool

Create a back-end address pool with [New-AzLoadBalancerBackendAddressPoolConfig](#). The VMs attach to this back-end pool in the remaining steps. The following example creates back-end address pools named *dsLbBackEndPool\_v4* and *dsLbBackEndPool\_v6* to include VMs with both IPV4 and IPv6 NIC configurations:

```
$backendPoolv4 = New-AzLoadBalancerBackendAddressPoolConfig `

-Name "dsLbBackEndPool_v4"

$backendPoolv6 = New-AzLoadBalancerBackendAddressPoolConfig `

-Name "dsLbBackEndPool_v6"
```

### Create a health probe

Use [Add-AzLoadBalancerProbeConfig](#) to create a health probe to monitor the health of the VMs.

```
$probe = New-AzLoadBalancerProbeConfig -Name MyProbe -Protocol tcp -Port 3389 -IntervalInSeconds 15 - `

-ProbeCount 2
```

### Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port. To make sure only healthy VMs receive traffic, you can optionally define a health probe. Basic

load balancer uses an IPv4 probe to assess health for both IPv4 and IPv6 endpoints on the VMs. Standard load balancer includes support for explicitly IPv6 health probes.

Create a load balancer rule with [Add-AzLoadBalancerRuleConfig](#). The following example creates load balancer rules named *dsLBrule\_v4* and *dsLBrule\_v6* and balances traffic on TCP port 80 to the IPv4 and IPv6 frontend IP configurations:

```
$lbrule_v4 = New-AzLoadBalancerRuleConfig `  
    -Name "dsLBrule_v4" `  
    -FrontendIpConfiguration $frontendIPv4 `  
    -BackendAddressPool $backendPoolv4 `  
    -Protocol Tcp `  
    -FrontendPort 80 `  
    -BackendPort 80 `  
    -probe $probe  
  
$lbrule_v6 = New-AzLoadBalancerRuleConfig `  
    -Name "dsLBrule_v6" `  
    -FrontendIpConfiguration $frontendIPv6 `  
    -BackendAddressPool $backendPoolv6 `  
    -Protocol Tcp `  
    -FrontendPort 80 `  
    -BackendPort 80 `  
    -probe $probe
```

## Create load balancer

Create a Standard Load Balancer with [New-AzLoadBalancer](#). The following example creates a public Standard Load Balancer named *myLoadBalancer* using the IPv4 and IPv6 frontend IP configurations, backend pools, and load-balancing rules that you created in the preceding steps:

```
$lb = New-AzLoadBalancer `  
    -ResourceGroupName $rg.ResourceGroupName `  
    -Location $rg.Location `  
    -Name "MyLoadBalancer" `  
    -Sku "Standard" `  
    -FrontendIpConfiguration $frontendIPv4,$frontendIPv6 `  
    -BackendAddressPool $backendPoolv4,$backendPoolv6 `  
    -LoadBalancingRule $lbrule_v4,$lbrule_v6
```

## Create network resources

Before you deploy some VMs and can test your balancer, you must create supporting network resources - availability set, network security group, virtual network, and virtual NICs.

### Create an availability set

To improve the high availability of your app, place your VMs in an availability set.

Create an availability set with [New-AzAvailabilitySet](#). The following example creates an availability set named *myAvailabilitySet*:

```
$avset = New-AzAvailabilitySet `  
    -ResourceGroupName $rg.ResourceGroupName `  
    -Location $rg.Location `  
    -Name "dsAVset" `  
    -PlatformFaultDomainCount 2 `  
    -PlatformUpdateDomainCount 2 `  
    -Sku aligned
```

## Create network security group

Create a network security group for the rules that will govern inbound and outbound communication in your VNET.

### Create a network security group rule for port 3389

Create a network security group rule to allow RDP connections through port 3389 with [New-AzNetworkSecurityRuleConfig](#).

```
$rule1 = New-AzNetworkSecurityRuleConfig `  
-Name 'myNetworkSecurityGroupRuleRDP' `  
-Description 'Allow RDP' `  
-Access Allow `  
-Protocol Tcp `  
-Direction Inbound `  
-Priority 100 `  
-SourceAddressPrefix * `  
-SourcePortRange * `  
-DestinationAddressPrefix * `  
-DestinationPortRange 3389
```

### Create a network security group rule for port 80

Create a network security group rule to allow internet connections through port 80 with [New-AzNetworkSecurityRuleConfig](#).

```
$rule2 = New-AzNetworkSecurityRuleConfig `  
-Name 'myNetworkSecurityGroupRuleHTTP' `  
-Description 'Allow HTTP' `  
-Access Allow `  
-Protocol Tcp `  
-Direction Inbound `  
-Priority 200 `  
-SourceAddressPrefix * `  
-SourcePortRange 80 `  
-DestinationAddressPrefix * `  
-DestinationPortRange 80
```

## Create a network security group

Create a network security group with [New-AzNetworkSecurityGroup](#).

```
$nsg = New-AzNetworkSecurityGroup `  
-ResourceGroupName $rg.ResourceGroupName `  
-Location $rg.Location `  
-Name "dsNSG1" `  
-SecurityRules $rule1,$rule2
```

## Create a virtual network

Create a virtual network with [New-AzVirtualNetwork](#). The following example creates a virtual network named *dsVnet* with *mySubnet*:

```

# Create dual stack subnet
$subnet = New-AzVirtualNetworkSubnetConfig ` 
-Name "dsSubnet" ` 
-AddressPrefix "10.0.0.0/24","ace:cab:deca:deed::/64"

# Create the virtual network
$vnet = New-AzVirtualNetwork ` 
-ResourceGroupName $rg.ResourceGroupName ` 
-Location $rg.Location ` 
-Name "dsVnet" ` 
-AddressPrefix "10.0.0.0/16","ace:cab:deca::/48" ` 
-Subnet $subnet

```

## Create NICs

Create virtual NICs with [New-AzNetworkInterface](#). The following example creates two virtual NICs both with IPv4 and IPv6 configurations. (One virtual NIC for each VM you create for your app in the following steps).

```

$Ip4Config=New-AzNetworkInterfaceIpConfig ` 
-Name dsIp4Config ` 
-Subnet $vnet.subnets[0] ` 
-PrivateIpAddressVersion IPv4 ` 
-LoadBalancerBackendAddressPool $backendPoolv4 ` 
-PublicIpAddress $RdpPublicIP_1

$Ip6Config=New-AzNetworkInterfaceIpConfig ` 
-Name dsIp6Config ` 
-Subnet $vnet.subnets[0] ` 
-PrivateIpAddressVersion IPv6 ` 
-LoadBalancerBackendAddressPool $backendPoolv6

$NIC_1 = New-AzNetworkInterface ` 
-Name "dsNIC1" ` 
-ResourceGroupName $rg.ResourceGroupName ` 
-Location $rg.Location ` 
-NetworkSecurityGroupId $nsg.Id ` 
-IpConfiguration $Ip4Config,$Ip6Config

$Ip4Config=New-AzNetworkInterfaceIpConfig ` 
-Name dsIp4Config ` 
-Subnet $vnet.subnets[0] ` 
-PrivateIpAddressVersion IPv4 ` 
-LoadBalancerBackendAddressPool $backendPoolv4 ` 
-PublicIpAddress $RdpPublicIP_2

$NIC_2 = New-AzNetworkInterface ` 
-Name "dsNIC2" ` 
-ResourceGroupName $rg.ResourceGroupName ` 
-Location $rg.Location ` 
-NetworkSecurityGroupId $nsg.Id ` 
-IpConfiguration $Ip4Config,$Ip6Config

```

## Create virtual machines

Set an administrator username and password for the VMs with [Get-Credential](#):

```
$cred = get-credential -Message "DUAL STACK VNET SAMPLE: Please enter the Administrator credential to log into the VMs."
```

Now you can create the VMs with [New-AzVM](#). The following example creates two VMs and the required virtual network components if they do not already exist.

```

$vmsize = "Standard_A2"
$imagePublisher = "MicrosoftWindowsServer"
$imageOffer = "WindowsServer"
$imageSKU = "2019-Datacenter"

$vmName= "dsVM1"
$VMconfig1 = New-AzVMConfig -VMName $vmName -VMSize $vmsize -AvailabilitySetId $avset.Id 3> $null | Set-
AzVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred -ProvisionVMAgent 3> $null | Set-
AzVMSourceImage -PublisherName $ImagePublisher -Offer $imageOffer -Skus $imageSKU -Version "latest" 3> $null |
Set-AzVMOSDisk -Name "$vmName.vhd" -CreateOption fromImage 3> $null | Add-AzVMNetworkInterface -Id $NIC_1.Id
3> $null
$VM1 = New-AzVM -ResourceGroupName $rg.ResourceGroupName -Location $rg.Location -VM $VMconfig1

$vmName= "dsVM2"
$VMconfig2 = New-AzVMConfig -VMName $vmName -VMSize $vmsize -AvailabilitySetId $avset.Id 3> $null | Set-
AzVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred -ProvisionVMAgent 3> $null | Set-
AzVMSourceImage -PublisherName $ImagePublisher -Offer $imageOffer -Skus $imageSKU -Version "latest" 3> $null |
Set-AzVMOSDisk -Name "$vmName.vhd" -CreateOption fromImage 3> $null | Add-AzVMNetworkInterface -Id $NIC_2.Id
3> $null
$VM2 = New-AzVM -ResourceGroupName $rg.ResourceGroupName -Location $rg.Location -VM $VMconfig2

```

## Determine IP addresses of the IPv4 and IPv6 endpoints

Get all Network Interface Objects in the resource group to summarize the IP's used in this deployment with `get-AzNetworkInterface`. Also, get the Load Balancer's frontend addresses of the IPv4 and IPv6 endpoints with `get-AzpublicIpAddress`.

```

$rgName= "dsRG1"
$NICsInRG= get-AzNetworkInterface -resourceGroupName $rgName
write-host `nSummary of IPs in this Deployment:
write-host ****
foreach ($NIC in $NICsInRG) {

    $VMid= $NIC.virtualmachine.id
    $VMnamebits= $VMid.split("/")
    $VMname= $VMnamebits[($VMnamebits.count-1)]
    write-host `nPrivate IP addresses for $VMname
    $IPconfigsInNIC= $NIC.IPconfigurations
    foreach ($IPconfig in $IPconfigsInNIC) {

        $IPaddress= $IPconfig.privateipaddress
        write-host "      $IPaddress
        IF ($IPconfig.PublicIpAddress.ID) {

            $IDbits= ($IPconfig.PublicIpAddress.ID).split("/")
            $PipName= $IDbits[($IDbits.count-1)]
            $PipObject= get-azPublicIpAddress -name $PipName -resourceGroup $rgName
            write-host "      RDP address: $PipObject.IpAddress
        }
    }
}

write-host `nPublic IP addresses on Load Balancer:
(get-AzpublicIpAddress -resourcegroupname $rgName | where { $_.name -notlike "RdpPublicIP*" }).IpAddress

```

The following figure shows a sample output that lists the private IPv4 and IPv6 addresses of the two VMs, and the frontend IPv4 and IPv6 IP addresses of the Load Balancer.

## Summary of IPs in this Deployment:

---

### Private IP addresses for DsVM0

10.0.0.4  
RDP address: 40.118.190.180  
ace:cab:deca:deed::4

### Private IP addresses for DsVM1

10.0.0.5  
RDP address: 40.118.190.195  
ace:cab:deca:deed::5

### Public IP addresses on Load Balancer:

40.118.190.251

2a01:111:f100:3000::a83e:19c3

## View IPv6 dual stack virtual network in Azure portal

You can view the IPv6 dual stack virtual network in Azure portal as follows:

1. In the portal's search bar, enter *dsVnet*.
2. When **dsVnet** appears in the search results, select it. This launches the **Overview** page of the dual stack virtual network named *dsVnet*. The dual stack virtual network shows the two NICs with both IPv4 and IPv6 configurations located in the dual stack subnet named *dsSubnet*.

The screenshot shows the Azure portal interface for managing a dual stack virtual network. The left sidebar contains navigation links for Home, Resource groups, DsRG02, VNET, Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Address space, Connected devices, Subnets, DDoS protection, Firewall, Security, DNS servers, Peerings, Service endpoints, and Properties. The main content area is titled 'VNET' and shows the following details:

- Resource group:** DsRG02
- Location:** West US
- Subscription:** (change)
- Address space:** 10.0.0.0/16, 1 more
- DNS servers:** Azure provided DNS service
- Subscription ID:** [REDACTED]
- Tags:** (change) Click here to add tags

**Connected devices:**

DEVICE	TYPE	IP ADDRESS	SUBNET
DsVM0	Network interface	10.0.0.4	DualStackSubnet
DsVM0	Network interface	ace:cab:deca:deed::4	DualStackSubnet
DsVM1	Network interface	10.0.0.5	DualStackSubnet
DsVM1	Network interface	ace:cab:deca:deed::5	DualStackSubnet

### NOTE

The IPv6 for Azure virtual network is available in the Azure portal in read-only for this preview release.

## Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) command to remove the resource group, VM, and all related resources.

```
Remove-AzResourceGroup -Name dsRG1
```

## Next steps

In this article, you created a Standard Load Balancer with a dual frontend IP configuration (IPv4 and IPv6). You also created a two virtual machines that included NICs with dual IP configurations (IPV4 + IPV6) that were added to the back-end pool of the load balancer. To learn more about IPv6 support in Azure virtual networks, see [What is IPv6 for Azure Virtual Network?](#)

# Deploy an IPv6 dual stack application in Azure virtual network - CLI (Preview)

12/19/2019 • 9 minutes to read • [Edit Online](#)

This article shows you how to deploy a dual stack (IPv4 + IPv6) application using Standard Load Balancer in Azure that includes a dual stack virtual network with a dual stack subnet, a Standard Load Balancer with dual (IPv4 + IPv6) front-end configurations, VMs with NICs that have a dual IP configuration, dual network security group rules, and dual public IPs.

## IMPORTANT

IPv6 dual stack for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

If you don't have an Azure subscription, create a [free account](#) now.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you decide to install and use Azure CLI locally instead, this quickstart requires you to use Azure CLI version 2.0.49 or later. To find your installed version, run `az --version`. See [Install Azure CLI](#) for install or upgrade info.

## Prerequisites

To use the IPv6 for Azure virtual network feature, you must configure your subscription using Azure CLI as follows:

```
az feature register --name AllowIPv6VirtualNetwork --namespace Microsoft.Network  
az feature register --name AllowIPv6CAOnStandardLB --namespace Microsoft.Network
```

It takes up to 30 minutes for feature registration to complete. You can check your registration status by running the following Azure CLI command:

```
az feature show --name AllowIPv6VirtualNetwork --namespace Microsoft.Network  
az feature show --name AllowIPv6CAOnStandardLB --namespace Microsoft.Network
```

After the registration is complete, run the following command:

```
az provider register --namespace Microsoft.Network
```

## Create a resource group

Before you can create your dual-stack virtual network, you must create a resource group with [az group create](#). The following example creates a resource group named *DsResourceGroup01* in the *eastus* location:

```
az group create \  
--name DsResourceGroup01 \  
--location eastus
```

## Create IPv4 and IPv6 public IP addresses for load balancer

To access your IPv4 and IPv6 endpoints on the Internet, you need IPv4 and IPv6 public IP addresses for the load balancer. Create a public IP address with [az network public-ip create](#). The following example creates IPv4 and IPv6 public IP address named *dsPublicIP\_v4* and *dsPublicIP\_v6* in the *DsResourceGroup01* resource group:

```
# Create an IPV4 IP address  
az network public-ip create \  
--name dsPublicIP_v4 \  
--resource-group DsResourceGroup01 \  
--location eastus \  
--sku STANDARD \  
--allocation-method static \  
--version IPv4  
  
# Create an IPV6 IP address  
az network public-ip create \  
--name dsPublicIP_v6 \  
--resource-group DsResourceGroup01 \  
--location eastus \  
--sku STANDARD \  
--allocation-method static \  
--version IPv6
```

## Create public IP addresses for VMs

To remotely access your VMs on the internet, you need IPv4 public IP addresses for the VMs. Create a public IP address with [az network public-ip create](#).

```
az network public-ip create \
--name dsVM0_remote_access \
--resource-group DsResourceGroup01 \
--location eastus \
--sku Standard \
--allocation-method static \
--version IPv4

az network public-ip create \
--name dsVM1_remote_access \
--resource-group DsResourceGroup01 \
--location eastus \
--sku Standard \
--allocation-method static \
--version IPv4
```

## Create Standard Load Balancer

In this section, you configure dual frontend IP (IPv4 and IPv6) and the back-end address pool for the load balancer and then create a Standard Load Balancer.

### Create load balancer

Create the Standard Load Balancer with [az network lb create](#) named **dsLB** that includes a frontend pool named **dsLbFrontEnd\_v4**, a backend pool named **dsLbBackEndPool\_v4** that is associated with the IPv4 public IP address **dsPublicIP\_v4** that you created in the preceding step.

```
az network lb create \
--name dsLB \
--resource-group DsResourceGroup01 \
--sku Standard \
--location eastus \
--frontend-ip-name dsLbFrontEnd_v4 \
--public-ip-address dsPublicIP_v4 \
--backend-pool-name dsLbBackEndPool_v4
```

### Create IPv6 frontend

Create an IPv6 frontend IP with [az network lb frontend-ip create](#). The following example creates a frontend IP configuration named *dsLbFrontEnd\_v6* and attaches the *dsPublicIP\_v6* address:

```
az network lb frontend-ip create \
--lb-name dsLB \
--name dsLbFrontEnd_v6 \
--resource-group DsResourceGroup01 \
--public-ip-address dsPublicIP_v6
```

### Configure IPv6 back-end address pool

Create a IPv6 back-end address pools with [az network lb address-pool create](#). The following example creates back-end address pool named *dsLbBackEndPool\_v6* to include VMs with IPv6 NIC configurations:

```
az network lb address-pool create \
--lb-name dsLB \
--name dsLbBackEndPool_v6 \
--resource-group DsResourceGroup01
```

## Create a health probe

Create a health probe with [az network lb probe create](#) to monitor the health of the virtual machines.

```
az network lb probe create -g DsResourceGroup01 --lb-name dsLB -n dsProbe --protocol tcp --port 3389
```

## Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port.

Create a load balancer rule with [az network lb rule create](#). The following example creates load balancer rules named *dsLBrule\_v4* and *dsLBrule\_v6* and balances traffic on *TCP* port *80* to the IPv4 and IPv6 frontend IP configurations:

```
az network lb rule create \
--lb-name dsLB \
--name dsLBrule_v4 \
--resource-group DsResourceGroup01 \
--frontend-ip-name dsLbFrontEnd_v4 \
--protocol Tcp \
--frontend-port 80 \
--backend-port 80 \
--probe-name dsProbe \
--backend-pool-name dsLbBackEndPool_v4
```

```
az network lb rule create \
--lb-name dsLB \
--name dsLBrule_v6 \
--resource-group DsResourceGroup01 \
--frontend-ip-name dsLbFrontEnd_v6 \
--protocol Tcp \
--frontend-port 80 \
--backend-port 80 \
--probe-name dsProbe \
--backend-pool-name dsLbBackEndPool_v6
```

# Create network resources

Before you deploy some VMs, you must create supporting network resources - availability set, network security group, virtual network, and virtual NICs.

## Create an availability set

To improve the availability of your app, place your VMs in an availability set.

Create an availability set with [az vm availability-set create](#). The following example creates an availability set named *dsAVset*:

```
az vm availability-set create \
--name dsAVset \
--resource-group DsResourceGroup01 \
--location eastus \
--platform-fault-domain-count 2 \
--platform-update-domain-count 2
```

## Create network security group

Create a network security group for the rules that will govern inbound and outbound communication in your VNet.

### Create a network security group

Create a network security group with [az network nsg create](#)

```
az network nsg create \
--name dsNSG1 \
--resource-group DsResourceGroup01 \
--location eastus
```

### Create a network security group rule for inbound and outbound connections

Create a network security group rule to allow RDP connections through port 3389, internet connection through port 80, and for outbound connections with [az network nsg rule create](#).

```
# Create inbound rule for port 3389
az network nsg rule create \
--name allowRdpIn \
--nsg-name dsNSG1 \
--resource-group DsResourceGroup01 \
--priority 100 \
--description "Allow Remote Desktop In" \
--access Allow \
--protocol "*" \
--direction Inbound \
--source-address-prefixes "*" \
--source-port-ranges "*" \
--destination-address-prefixes "*" \
--destination-port-ranges 3389

# Create inbound rule for port 80
az network nsg rule create \
--name allowHTTPIn \
--nsg-name dsNSG1 \
--resource-group DsResourceGroup01 \
--priority 200 \
--description "Allow HTTP In" \
--access Allow \
--protocol "*" \
--direction Inbound \
--source-address-prefixes "*" \
--source-port-ranges 80 \
--destination-address-prefixes "*" \
--destination-port-ranges 80

# Create outbound rule

az network nsg rule create \
--name allowAllOut \
--nsg-name dsNSG1 \
--resource-group DsResourceGroup01 \
--priority 300 \
--description "Allow All Out" \
--access Allow \
--protocol "*" \
--direction Outbound \
--source-address-prefixes "*" \
--source-port-ranges "*" \
--destination-address-prefixes "*" \
--destination-port-ranges "*"
```

## Create a virtual network

Create a virtual network with [az network vnet create](#). The following example creates a virtual network named *dsVNET* with subnets *dsSubNET\_v4* and *dsSubNET\_v6*:

```
# Create the virtual network
az network vnet create \
--name dsVNET \
--resource-group DsResourceGroup01 \
--location eastus \
--address-prefixes "10.0.0.0/16" "ace:cab:deca::/48"

# Create a single dual stack subnet

az network vnet subnet create \
--name dsSubNET \
--resource-group DsResourceGroup01 \
--vnet-name dsVNET \
--address-prefixes "10.0.0.0/24" "ace:cab:deca:deed::/64" \
--network-security-group dsNSG1
```

## Create NICs

Create virtual NICs for each VM with [az network nic create](#). The following example creates a virtual NIC for each VM. Each NIC has two IP configurations (1 IPv4 config, 1 IPv6 config). You create the IPV6 configuration with [az network nic ip-config create](#).

```

# Create NICs
az network nic create \
--name dsNIC0 \
--resource-group DsResourceGroup01 \
--network-security-group dsNSG1 \
--vnet-name dsVNET \
--subnet dsSubNet \
--private-ip-address-version IPv4 \
--lb-address-pools dsLbBackEndPool_v4 \
--lb-name dsLB \
--public-ip-address dsVM0_remote_access

az network nic create \
--name dsNIC1 \
--resource-group DsResourceGroup01 \
--network-security-group dsNSG1 \
--vnet-name dsVNET \
--subnet dsSubNet \
--private-ip-address-version IPv4 \
--lb-address-pools dsLbBackEndPool_v4 \
--lb-name dsLB \
--public-ip-address dsVM1_remote_access

# Create IPV6 configurations for each NIC

az network nic ip-config create \
--name dsIp6Config_NIC0 \
--nic-name dsNIC0 \
--resource-group DsResourceGroup01 \
--vnet-name dsVNET \
--subnet dsSubNet \
--private-ip-address-version IPv6 \
--lb-address-pools dsLbBackEndPool_v6 \
--lb-name dsLB

az network nic ip-config create \
--name dsIp6Config_NIC1 \
--nic-name dsNIC1 \
--resource-group DsResourceGroup01 \
--vnet-name dsVNET \
--subnet dsSubNet \
--private-ip-address-version IPv6 \
--lb-address-pools dsLbBackEndPool_v6 \
--lb-name dsLB

```

## Create virtual machines

Create the VMs with [az vm create](#). The following example creates two VMs and the required virtual network components if they do not already exist.

Create virtual machine *dsVM0* as follows:

```

az vm create \
--name dsVM0 \
--resource-group DsResourceGroup01 \
--nics dsNIC0 \
--size Standard_A2 \
--availability-set dsAVset \
--image MicrosoftWindowsServer:WindowsServer:2019-Datacenter:latest

```

Create virtual machine *dsVM1* as follows:

```
az vm create \
--name dsVM1 \
--resource-group DsResourceGroup01 \
--nics dsNIC1 \
--size Standard_A2 \
--availability-set dsAVset \
--image MicrosoftWindowsServer:WindowsServer:2019-Datacenter:latest
```

## View IPv6 dual stack virtual network in Azure portal

You can view the IPv6 dual stack virtual network in Azure portal as follows:

1. In the portal's search bar, enter *dsVnet*.
2. When **myVirtualNetwork** appears in the search results, select it. This launches the **Overview** page of the dual stack virtual network named *dsVnet*. The dual stack virtual network shows the two NICs with both IPv4 and IPv6 configurations located in the dual stack subnet named *dsSubnet*.

The screenshot shows the Azure portal interface for managing a virtual network. The URL in the address bar is [Home > Resource groups > DsRG02 > VNET](#). The main title is "VNET" with a subtitle "Virtual network". On the left, there's a sidebar with navigation links: Overview (which is selected and highlighted in blue), Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Address space, Connected devices, Subnets, DDoS protection, Firewall, Security, DNS servers, Peerings, Service endpoints, and Properties. The main content area shows the following details for the virtual network:

- Resource group (change):** DsRG02
- Location:** West US
- Subscription (change):** [Subscription ID]
- Tags (change):** Click here to add tags
- Connected devices:**

DEVICE	TYPE	IP ADDRESS	SUBNET
DsVM0	Network interface	10.0.0.4	DualStackSubnet
DsVM0	Network interface	ace:cab:deca:deed::4	DualStackSubnet
DsVM1	Network interface	10.0.0.5	DualStackSubnet
DsVM1	Network interface	ace:cab:deca:deed::5	DualStackSubnet

### NOTE

The IPv6 for Azure virtual network is available in the Azure portal in read-only for this preview release.

## Clean up resources

When no longer needed, you can use the [az group delete](#) command to remove the resource group, VM, and all related resources.

```
az group delete --name DsResourceGroup01
```

## Next steps

In this article, you created a Standard Load Balancer with a dual frontend IP configuration (IPv4 and IPv6). You also created a two virtual machines that included NICs with dual IP configurations (IPV4 + IPV6) that were added

to the back-end pool of the load balancer. To learn more about IPv6 support in Azure virtual networks, see [What is IPv6 for Azure Virtual Network?](#)

# Deploy an IPv6 dual stack application in Azure virtual network - Template (Preview)

8/23/2019 • 2 minutes to read • [Edit Online](#)

This article provides a list of IPv6 configuration tasks with the portion of the Azure Resource Manager VM template that applies to. Use the template described in this article to deploy a dual stack (IPv4 + IPv6) application using Standard Load Balancer in Azure that includes a dual stack virtual network with IPv4 and IPv6 subnets, a Standard Load Balancer with dual (IPv4 + IPv6) front-end configurations, VMs with NICs that have a dual IP configuration, network security group, and public IPs.

## Required configurations

Search for the template sections in the template to see where they should occur.

### IPv6 addressSpace for the virtual network

Template section to add:

```
"addressSpace": {  
    "addressPrefixes": [  
        "[variables('vnetv4AddressRange')]",  
        "[variables('vnetv6AddressRange')]"  
    ]  
}
```

### IPv6 subnet within the IPv6 virtual network addressSpace

Template section to add:

```
{  
    "name": "V6Subnet",  
    "properties": {  
        "addressPrefix": "[variables('subnetv6AddressRange')]"  
    }  
}
```

### IPv6 configuration for the NIC

Template section to add:

```
{  
    "name": "ipconfig-v6",  
    "properties": {  
        "privateIPAllocationMethod": "Dynamic",  
        "privateIPAddressVersion": "IPv6",  
        "subnet": {  
            "id": "[variables('v6-subnet-id')]"  
        },  
        "loadBalancerBackendAddressPools": [  
            {  
                "id": "  
[concat(resourceId('Microsoft.Network/loadBalancers', 'loadBalancer'), '/backendAddressPools/LBBAP-v6')]"  
            }  
        ]  
    }  
}
```

### IPv6 network security group (NSG) rules

```
{
    "name": "default-allow-rdp",
    "properties": {
        "description": "Allow RDP",
        "protocol": "Tcp",
        "sourcePortRange": "33819-33829",
        "destinationPortRange": "5000-6000",
        "sourceAddressPrefix": "ace:cab:deca:deed::/64",
        "destinationAddressPrefix": "cab:ace:deca:deed::/64",
        "access": "Allow",
        "priority": 1003,
        "direction": "Inbound"
    }
}
```

## Conditional configuration

If you're using a network virtual appliance, add IPv6 routes in the Route Table. Otherwise, this configuration is optional.

```
{
    "type": "Microsoft.Network/routeTables",
    "name": "v6route",
    "apiVersion": "[variables('ApiVersion')]",
    "location": "[resourceGroup().location]",
    "properties": {
        "routes": [
            {
                "name": "v6route",
                "properties": {
                    "addressPrefix": "ace:cab:deca:deed::/64",
                    "nextHopType": "VirtualAppliance",
                    "nextHopIpAddress": "deca:cab:ace:f00d::1"
                }
            }
        ]
    }
}
```

## Optional configuration

### IPv6 Internet access for the virtual network

```
{
    "name": "LBFE-v6",
    "properties": {
        "publicIPAddress": {
            "id": "[resourceId('Microsoft.Network/publicIPAddresses', 'lbppublicip-v6')]"
        }
    }
}
```

### IPv6 Public IP addresses

```
{
    "apiVersion": "[variables('ApiVersion')]",
    "type": "Microsoft.Network/publicIPAddresses",
    "name": "lbppublicip-v6",
    "location": "[resourceGroup().location]",
    "sku": {
        "name": "Standard"
    },
    "properties": {
        "publicIPAllocationMethod": "Static",
        "publicIPAddressVersion": "IPv6"
    }
}
```

## IPv6 Front end for Load Balancer

```
{  
    "name": "LBFE-v6",  
    "properties": {  
        "publicIPAddress": {  
            "id": "[resourceId('Microsoft.Network/publicIPAddresses','lbpublicip-v6')]"  
        }  
    }  
}
```

## IPv6 Back-end address pool for Load Balancer

```
"backendAddressPool": {  
    "id": "[concat(resourceId('Microsoft.Network/loadBalancers', 'loadBalancer'),  
    '/backendAddressPools/LBAP-v6')]"  
},  
    "protocol": "Tcp",  
    "frontendPort": 8080,  
    "backendPort": 8080  
},  
    "name": "lbrule-v6"
```

## IPv6 load balancer rules to associate incoming and outgoing ports

```
{  
    "name": "ipconfig-v6",  
    "properties": {  
        "privateIPAllocationMethod": "Dynamic",  
        "privateIPAddressVersion": "IPv6",  
        "subnet": {  
            "id": "[variables('v6-subnet-id')]"  
        },  
        "loadBalancerBackendAddressPools": [  
            {  
                "id": "  
[concat(resourceId('Microsoft.Network/loadBalancers', 'loadBalancer'), '/backendAddressPools/LBAP-v6')]"  
            }  
        ]  
    }  
}
```

## Sample VM template JSON

To deploy an IPv6 dual stack application in Azure virtual network using Azure Resource Manager template, view sample template [here](#).

## Next steps

You can find details about pricing for [public IP addresses](#), [network bandwidth](#), or [Load Balancer](#).

# Deploy an IPv6 dual stack application using Standard Internal Load Balancer in Azure - PowerShell (Preview)

10/14/2019 • 9 minutes to read • [Edit Online](#)

This article shows you how to deploy a dual stack (IPv4 + IPv6) application in Azure that includes a dual stack virtual network and subnet, a Standard Internal Load Balancer with dual (IPv4 + IPv6) front-end configurations, VMs with NICs that have a dual IP configuration, network security group, and public IPs.

## IMPORTANT

IPv6 support for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

The procedure to create an IPv6-capable Internal Load Balancer is nearly identical to the process for creating an Internet-facing IPv6 Load Balancer described [here](#). The only differences for creating an internal load balancer are in the front-end configuration as illustrated in the PowerShell example below:

```
$frontendIPv6 = New-AzLoadBalancerFrontendIpConfig `  
    -Name "dsLbFrontEnd_v6" `  
    -PrivateIpAddress "ace:cab:deca:deed::100" `  
    -PrivateIpAddressVersion "IPv6" `  
    -Subnet $DsSubnet
```

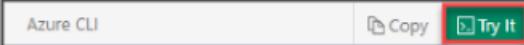
The changes that make the above an internal load balancer front-end configuration are:

- The `-PrivateIpAddressVersion` argument is specified as "IPv6"
- The `-PublicIpAddress` argument has been either omitted or replaced with `-PrivateIpAddress`. Note that the private address must be in the range of the Subnet IP space in which the internal load balancer will be deployed. If a static `-PrivateIpAddress` is omitted, the next free IPv6 address will be selected from the subnet in which the internal load Balancer is deployed.
- The dual stack subnet in which the internal load balancer will be deployed is specified with either a `-Subnet` or `-SubnetId` argument.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	

OPTION	EXAMPLE/LINK
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 6.9.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

## Prerequisites

Before you deploy a dual stack application in Azure, you must configure your subscription for this preview feature using the following Azure PowerShell:

Register as follows:

```
Register-AzProviderFeature -FeatureName AllowIPv6VirtualNetwork -ProviderNamespace Microsoft.Network
Register-AzProviderFeature -FeatureName AllowIPv6CAOnStandardLB -ProviderNamespace Microsoft.Network
```

It takes up to 30 minutes for feature registration to complete. You can check your registration status by running the following Azure PowerShell command: Check on the registration as follows:

```
Get-AzProviderFeature -FeatureName AllowIPv6VirtualNetwork -ProviderNamespace Microsoft.Network
Get-AzProviderFeature -FeatureName AllowIPv6CAOnStandardLB -ProviderNamespace Microsoft.Network
```

After the registration is complete, run the following command:

```
Register-AzResourceProvider -ProviderNamespace Microsoft.Network
```

## Create a resource group

Before you can create your dual-stack virtual network, you must create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *dsStd\_ILB\_RG* in the *east us* location:

```
$rg = New-AzResourceGroup ` 
-ResourceGroupName "dsStd_ILB_RG" ` 
-Location "east us"
```

## Create IPv4 and IPv6 public IP addresses

To access your virtual machines from the Internet, you need IPv4 and IPv6 public IP addresses for the VMs. Create public IP addresses with [New-AzPublicIpAddress](#). The following example creates IPv4 and IPv6 public IP address named *RdpPublicIP\_1* and *RdpPublicIP\_2* in the *dsStd\_ILB\_RG* resource group:

```
$RdpPublicIP_1 = New-AzPublicIpAddress `  
    -Name "RdpPublicIP_1" `  
    -ResourceGroupName $rg.ResourceGroupName `  
    -Location $rg.Location `  
    -AllocationMethod Static `  
    -IpAddressVersion IPv4 `  
    -sku Standard  
  
$RdpPublicIP_2 = New-AzPublicIpAddress `  
    -Name "RdpPublicIP_2" `  
    -ResourceGroupName $rg.ResourceGroupName `  
    -Location $rg.Location `  
    -AllocationMethod Static `  
    -IpAddressVersion IPv4 `  
    -sku Standard
```

## Create the virtual network and the subnet

Create a virtual network using [New-AzVirtualNetwork](#) with dual stack a subnet configuration using [New-AzVirtualNetworkSubnetConfig](#). The following example creates a virtual network named *dsVnet* with *dsSubnet*.

```
# Create dual stack subnet config  
$DsSubnet = New-AzVirtualNetworkSubnetConfig `  
    -Name "dsSubnet" `  
    -AddressPrefix "10.0.0.0/24", "ace:cab:deca:deed::/64"  
  
# Create the virtual network  
$vnet = New-AzVirtualNetwork `  
    -ResourceGroupName $rg.ResourceGroupName `  
    -Location $rg.Location `  
    -Name "dsVnet" `  
    -AddressPrefix "10.0.0.0/16", "ace:cab:deca::/48" `  
    -Subnet $DsSubnet  
  
#Refresh the fully populated subnet for use in load balancer frontend configuration  
$DsSubnet = get-AzVirtualNetworkSubnetconfig -name dsSubnet -VirtualNetwork $vnet
```

## Create Standard Load Balancer

In this section, you configure dual front-end IP (IPv4 and IPv6) and the back-end address pool for the load balancer and then create a Standard Load Balancer.

### Create front-end IP

Create a front-end IP with [New-AzLoadBalancerFrontendIpConfig](#). The following example creates IPv4 and IPv6 front-end IP configurations named *dsLbFrontEnd\_v4* and *dsLbFrontEnd\_v6*:

```
$frontendIPv4 = New-AzLoadBalancerFrontendIpConfig ` 
    -Name "dsLbFrontEnd_v4" ` 
    -PrivateIpAddress "10.0.0.100" ` 
    -PrivateIpAddressVersion "IPv4" ` 
    -Subnet $DsSubnet

$frontendIPv6 = New-AzLoadBalancerFrontendIpConfig ` 
    -Name "dsLbFrontEnd_v6" ` 
    -PrivateIpAddress "ace:cab:deca:deed::100" ` 
    -PrivateIpAddressVersion "IPv6" ` 
    -Subnet $DsSubnet
```

## Configure back-end address pool

Create a back-end address pool with [New-AzLoadBalancerBackendAddressPoolConfig](#). The VMs attach to this back-end pool in the remaining steps. The following example creates back-end address pools named *dsLbBackEndPool\_v4* and *dsLbBackEndPool\_v6* to include VMs with both IPV4 and IPv6 NIC configurations:

```
$backendPoolv4 = New-AzLoadBalancerBackendAddressPoolConfig -Name "dsLbBackEndPool_v4"

$backendPoolv6 = New-AzLoadBalancerBackendAddressPoolConfig -Name "dsLbBackEndPool_v6"
```

## Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port. To make sure only healthy VMs receive traffic, you can optionally define a health probe. Basic load balancer uses an IPv4 probe to assess health for both IPv4 and IPv6 endpoints on the VMs. Standard load balancer includes support for explicitly IPv6 health probes.

Create a load balancer rule with [Add-AzLoadBalancerRuleConfig](#). The following example creates load balancer rules named *dsLBrule\_v4* and *dsLBrule\_v6* and balances traffic on TCP port 80 to the IPv4 and IPv6 frontend IP configurations:

```
$lbrule_v4 = New-AzLoadBalancerRuleConfig ` 
    -Name "dsLBrule_v4" ` 
    -FrontendIpConfiguration $frontendIPv4 ` 
    -BackendAddressPool $backendPoolv4 ` 
    -Protocol Tcp ` 
    -FrontendPort 80 ` 
    -BackendPort 80

$lbrule_v6 = New-AzLoadBalancerRuleConfig ` 
    -Name "dsLBrule_v6" ` 
    -FrontendIpConfiguration $frontendIPv6 ` 
    -BackendAddressPool $backendPoolv6 ` 
    -Protocol Tcp ` 
    -FrontendPort 80 ` 
    -BackendPort 80
```

## Create load balancer

Create a Standard Load Balancer with [New-AzLoadBalancer](#). The following example creates a public Standard Load Balancer named *myInternalLoadBalancer* using the IPv4 and IPv6 frontend IP configurations, backend pools, and load-balancing rules that you created in the preceding steps:

```
$lb = New-AzLoadBalancer ` 
    -ResourceGroupName $rg.ResourceGroupName ` 
    -Location $rg.Location ` 
    -Name "MyInternalLoadBalancer" ` 
    -Sku "Standard" ` 
    -FrontendIpConfiguration $frontendIPv4,$frontendIPv6 ` 
    -BackendAddressPool $backendPoolv4,$backendPoolv6 ` 
    -LoadBalancingRule $lbrule_v4,$lbrule_v6
```

## Create network resources

Before you deploy some VMs and can test your balancer, you must create supporting network resources - availability set, network security group, and virtual NICs.

### Create an availability set

To improve the high availability of your application, place your VMs in an availability set.

Create an availability set with [New-AzAvailabilitySet](#). The following example creates an availability set named *dsAVset*:

```
$avset = New-AzAvailabilitySet ` 
    -ResourceGroupName $rg.ResourceGroupName ` 
    -Location $rg.Location ` 
    -Name "dsAVset" ` 
    -PlatformFaultDomainCount 2 ` 
    -PlatformUpdateDomainCount 2 ` 
    -Sku aligned
```

### Create network security group

Create a network security group for the rules that will govern inbound and outbound communication in your VNet.

#### Create a network security group rule for port 3389

Create a network security group rule to allow RDP connections through port 3389 with [New-AzNetworkSecurityRuleConfig](#).

```
$rule1 = New-AzNetworkSecurityRuleConfig ` 
    -Name 'myNetworkSecurityGroupRuleRDP' ` 
    -Description 'Allow RDP' ` 
    -Access Allow ` 
    -Protocol Tcp ` 
    -Direction Inbound ` 
    -Priority 100 ` 
    -SourceAddressPrefix * ` 
    -SourcePortRange * ` 
    -DestinationAddressPrefix * ` 
    -DestinationPortRange 3389
```

#### Create a network security group rule for port 80

Create a network security group rule to allow internet connections through port 80 with [New-AzNetworkSecurityRuleConfig](#).

```
$rule2 = New-AzNetworkSecurityRuleConfig `  
    -Name 'myNetworkSecurityGroupRuleHTTP' `  
    -Description 'Allow HTTP' `  
    -Access Allow `  
    -Protocol Tcp `  
    -Direction Inbound `  
    -Priority 200 `  
    -SourceAddressPrefix '*' `  
    -SourcePortRange 80 `  
    -DestinationAddressPrefix '*' `  
    -DestinationPortRange 80
```

### Create a network security group

Create a network security group with [New-AzNetworkSecurityGroup](#).

```
$nsg = New-AzNetworkSecurityGroup `  
    -ResourceGroupName $rg.ResourceGroupName `  
    -Location $rg.Location `  
    -Name "dsNSG1" `  
    -SecurityRules $rule1,$rule2
```

### Create NICs

Create virtual NICs with [New-AzNetworkInterface](#). The following example creates two virtual NICs both with IPv4 and IPv6 configurations. (One virtual NIC for each VM you create for your app in the following steps).

```

# Create the IPv4 configuration for NIC 1
$Ip4Config=New-AzNetworkInterfaceIpConfig ` 
    -Name dsIp4Config ` 
    -Subnet $vnet.subnets[0] ` 
    -PrivateIpAddressVersion IPv4 ` 
    -LoadBalancerBackendAddressPool $backendPoolv4 ` 
    -PublicIpAddress $RdpPublicIP_1

# Create the IPv6 configuration
$Ip6Config=New-AzNetworkInterfaceIpConfig ` 
    -Name dsIp6Config ` 
    -Subnet $vnet.subnets[0] ` 
    -PrivateIpAddressVersion IPv6 ` 
    -LoadBalancerBackendAddressPool $backendPoolv6

# Create NIC 1
$NIC_1 = New-AzNetworkInterface ` 
    -Name "dsNIC1" ` 
    -ResourceGroupName $rg.ResourceGroupName ` 
    -Location $rg.Location ` 
    -NetworkSecurityGroupId $nsg.Id ` 
    -IpConfiguration $Ip4Config,$Ip6Config

# Create the IPv4 configuration for NIC 2
$Ip4Config=New-AzNetworkInterfaceIpConfig ` 
    -Name dsIp4Config ` 
    -Subnet $vnet.subnets[0] ` 
    -PrivateIpAddressVersion IPv4 ` 
    -LoadBalancerBackendAddressPool $backendPoolv4 ` 
    -PublicIpAddress $RdpPublicIP_2

# Create NIC 2 reusing the IPv6 configuration from NIC 1
$NIC_2 = New-AzNetworkInterface ` 
    -Name "dsNIC2" ` 
    -ResourceGroupName $rg.ResourceGroupName ` 
    -Location $rg.Location ` 
    -NetworkSecurityGroupId $nsg.Id ` 
    -IpConfiguration $Ip4Config,$Ip6Config

```

## Create virtual machines

Set an administrator username and password for the VMs with [Get-Credential](#):

```
$cred = get-credential -Message "DUAL STACK VNET SAMPLE: Please enter the Administrator credential to log into the VM's"
```

Now you can create the VMs with [New-AzVM](#). The following example creates two VMs and the required virtual network components if they do not already exist.

```

$vmsize = "Standard_A2"
$imagePublisher = "MicrosoftWindowsServer"
$imageOffer = "WindowsServer"
$imageSKU = "2019-Datacenter"

$vmName= "dsVM1"
$VMconfig1 = New-AzVMConfig -VMName $vmName -VMSize $vmsize -AvailabilitySetId $avset.Id 3> $null | Set-
AzVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred -ProvisionVMAgent 3> $null | Set-
AzVMSourceImage -PublisherName $ImagePublisher -Offer $imageOffer -Skus $imageSKU -Version "latest" 3> $null |
Set-AzVMOSDisk -Name "$vmName.vhd" -CreateOption fromImage 3> $null | Add-AzVMNetworkInterface -Id $NIC_1.Id
3> $null
$VM1 = New-AzVM -ResourceGroupName $rg.ResourceGroupName -Location $rg.Location -VM $VMconfig1

$vmName= "dsVM2"
$VMconfig2 = New-AzVMConfig -VMName $vmName -VMSize $vmsize -AvailabilitySetId $avset.Id 3> $null | Set-
AzVMOperatingSystem -Windows -ComputerName $vmName -Credential $cred -ProvisionVMAgent 3> $null | Set-
AzVMSourceImage -PublisherName $ImagePublisher -Offer $imageOffer -Skus $imageSKU -Version "latest" 3> $null |
Set-AzVMOSDisk -Name "$vmName.vhd" -CreateOption fromImage 3> $null | Add-AzVMNetworkInterface -Id $NIC_2.Id
3> $null
$VM2 = New-AzVM -ResourceGroupName $rg.ResourceGroupName -Location $rg.Location -VM $VMconfig2

```

## View IPv6 dual stack virtual network in Azure portal

You can view the IPv6 dual stack virtual network in Azure portal as follows:

1. In the portal's search bar, enter *dsVnet*.
2. When **dsVnet** appears in the search results, select it. This launches the **Overview** page of the dual stack virtual network named *dsVnet*. The dual stack virtual network shows the two NICs with both IPv4 and IPv6 configurations located in the dual stack subnet named *dsSubnet*.

DEVICE	TYPE	IP ADDRESS	SUBNET
dsNIC1	Network interface	10.0.0.4	dsSubnet
dsNIC1	Network interface	ace:cab:deca:deed::4	dsSubnet
dsNIC2	Network interface	10.0.0.5	dsSubnet
dsNIC2	Network interface	ace:cab:deca:deed::5	dsSubnet

### NOTE

The IPv6 for Azure virtual network is available in the Azure portal in read-only for this preview release.

## Clean up resources

When no longer needed, you can use the [Remove-AzResourceGroup](#) command to remove the resource group, VM, and all related resources.

```
Remove-AzResourceGroup -Name dsStd_ILB_RG
```

## Next steps

In this article, you created a Standard Load Balancer with a dual frontend IP configuration (IPv4 and IPv6). You also created a two virtual machines that included NICs with dual IP configurations (IPV4 + IPV6) that were added to the back-end pool of the load balancer. To learn more about IPv6 support in Azure virtual networks, see [What is IPv6 for Azure Virtual Network?](#)

# Deploy virtual machine scale sets with IPv6 in Azure (Preview)

10/30/2019 • 2 minutes to read • [Edit Online](#)

This article shows you how to deploy a dual stack (IPv4 + IPv6) Virtual Machine Scale Set with a dual stack external load balancer in an Azure virtual network. The process to create an IPv6-capable virtual machine scale set is nearly identical to the process for creating individual VMs described [here](#). You'll start with the steps that are similar to ones described for individual VMs:

1. Create IPv4 and IPv6 Public IPs.
2. Create a dual stack load balancer.
3. Create network security group (NSG) rules.

The only step that is different from individual VMs is creating the network interface (NIC) configuration that uses the virtual machine scale set resource: `networkProfile/networkInterfaceConfigurations`. The JSON structure is similar to that of the `Microsoft.Network/networkInterfaces` object used for individual VMs with the addition of setting the NIC and the IPv4 `IpConfiguration` as the primary interface using the **“primary”: true** attribute as seen in the following example:

```

"networkProfile": {
    "networkInterfaceConfigurations": [
        {
            "name": "[variables('nicName')]",
            "properties": {
                "primary": true,
            }
        }
    ],
    "networkSecurityGroup": {
        "id": "[resourceId('Microsoft.Network/networkSecurityGroups', 'VmssNsg')]"
    },
    "ipConfigurations": [
        {
            "name": "[variables('ipConfigName')]",
            "properties": {
                "primary": true,
                "subnet": {
                    "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
'MyvirtualNetwork', 'Mysubnet')]"
                },
                "privateIPAddressVersion": "IPv4",
                "publicIPaddressConfiguration": {
                    "name": "pub1",
                    "properties": {
                        "idleTimeoutInMinutes": 15
                    }
                },
                "loadBalancerBackendAddressPools": [
                    {
                        "id": "[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
'loadBalancer', 'bePool'))]"
                    }
                ],
                "loadBalancerInboundNatPools": [
                    {
                        "id": "[resourceId('Microsoft.Network/loadBalancers/inboundNatPools',
'loadBalancer', 'natPool'))]"
                    }
                ]
            }
        },
        {
            "name": "[variables('ipConfigNameV6')]",
            "properties": {
                "subnet": {
                    "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets',
'MyvirtualNetwork', 'Mysubnet')]"
                },
                "privateIPAddressVersion": "IPv6",
                "loadBalancerBackendAddressPools": [
                    {
                        "id": "[resourceId('Microsoft.Network/loadBalancers/backendAddressPools',
'loadBalancer', 'bePoolv6'))]"
                    }
                ],
                "loadBalancerInboundNatPools": [
                    {
                        "id": "[resourceId('Microsoft.Network/loadBalancers/inboundNatPools',
'loadBalancer', 'natPoolv6'))]"
                    }
                ]
            }
        }
    ]
}

```

## Sample virtual machine scale set template JSON

To deploy a dual stack (IPv4 + IPv6) Virtual Machine Scale Set with dual stack external Load Balancer and virtual

network view sample template [here](#).

## Next steps

To learn more about IPv6 support in Azure virtual networks, see [What is IPv6 for Azure Virtual Network?](#).

# Reserve public IPv6 address prefix

10/18/2019 • 2 minutes to read • [Edit Online](#)

IPv6 for Azure Virtual Network (VNet) enables you to host applications in Azure with IPv6 and IPv4 connectivity both within a virtual network and to and from the Internet. In addition to reserving individual IPv6 addresses, you can reserve contiguous ranges of Azure IPv6 addresses (known as IP Prefix) for your use. This article describes how to create IPv6 public IP addresses and address ranges using Azure PowerShell and CLI.

## IMPORTANT

IPv6 for Azure Virtual Network is currently in public preview. This preview is provided without a service level agreement and is not recommended for production workloads. Certain features may not be supported or may have constrained capabilities. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

## Create a single reserved IPv6 public IP

### Using Azure PowerShell

You can create a single reserved (static) IPv6 Public IP address using Azure PowerShell with [New-AzPublicIpAddress](#) as follows:

```
$myOwnIPv6Address = New-AzPublicIpAddress `  
-name PIPv6_WestUS `  
-ResourceGroup MyRG `  
-Location "West US" `  
-Sku Standard `  
-allocationMethod static `  
-IpAddressVersion IPv6
```

### Using Azure CLI

You can create a single reserved (static) IPv6 Public IP address Azure CLI with [az network public-ip create](#) as follows:

```
az network public-ip create \  
--name dsPublicIP_v6 \  
--resource-group UpgradeInPlace_CLI_RG1 \  
--location WestUS \  
--sku Standard \  
--allocation-method static \  
--version IPv6
```

## Create a reserved IPv6 prefix (range)

To reserve an IPv6 prefix, add the IP address family of IPv6 to the same command used for creating IPv4 prefixes. The following commands create a prefix of size /125 ( 8 IPv6 addresses).

### Using Azure PowerShell

You can create a public IPv6 address using Azure CLI with [az network public-ip create](#) as follows:

```
$myOwnIPv6Prefix = New-AzPublicIpPrefix `  
-name IPv6PrefixWestUS `  
-ResourceGroupName MyRG `  
-Location "West US" `  
-Sku Standard `  
-IpAddressVersion IPv6 `  
-PrefixLength 125
```

## Using Azure CLI

You can create a public IPv6 address using Azure CLI as follows:

```
az network public-ip prefix create \  
--name IPv6PrefixWestUS \  
--resource-group MyRG \  
--location WestUS \  
--version IPv6 \  
--length 125
```

## Allocate a public IP address from a reserved IPv6 Prefix

### Using Azure PowerShell

You create a static IPv6 Public IP from a reserved prefix by adding the `-PublicIpPrefix` argument when creating the public IP using Azure PowerShell. The following example assumes that a prefix was created and stored in a PowerShell variable named: `$myOwnIPv6Prefix`.

```
$MyIPv6PublicIPFromMyReservedPrefix = New-AzPublicIpAddress \  
-name PIPv6_fromPrefix `  
-ResourceGroup DsStdLb04 `  
-Location "West Central US" `  
-Sku Standard `  
-allocationMethod static `  
-IpAddressVersion IPv6 `  
-PublicIpPrefix $myOwnIPv6Prefix
```

### Using Azure CLI

The following example assumes that a prefix was created and stored in a CLI variable named: `IPv6PrefixWestUS`.

```
az network public-ip create \  
--name dsPublicIP_v6 \  
--resource-group UpgradeInPlace_CLI_RG1 \  
--location WestUS \  
--sku Standard \  
--allocation-method static \  
--version IPv6 \  
--public-ip-prefix IPv6PrefixWestUS
```

## Next steps

- Learn more about [IPv6 address prefix](#).
- Learn more about [IPv6 addresses](#).

# Create, change, or delete a public IP address

12/9/2019 • 11 minutes to read • [Edit Online](#)

Learn about a public IP address and how to create, change, and delete one. A public IP address is a resource with its own configurable settings. Assigning a public IP address to an Azure resource that supports public IP addresses enables:

- Inbound communication from the Internet to the resource, such as Azure Virtual Machines (VM), Azure Application Gateways, Azure Load Balancers, Azure VPN Gateways, and others. You can still communicate with some resources, such as VMs, from the Internet, if a VM doesn't have a public IP address assigned to it, as long as the VM is part of a load balancer back-end pool, and the load balancer is assigned a public IP address. To determine whether a resource for a specific Azure service can be assigned a public IP address, or whether it can be communicated with through the public IP address of a different Azure resource, see the documentation for the service.
- Outbound connectivity to the Internet using a predictable IP address. For example, a virtual machine can communicate outbound to the Internet without a public IP address assigned to it, but its address is network address translated by Azure to an unpredictable public address, by default. Assigning a public IP address to a resource enables you to know which IP address is used for the outbound connection. Though predictable, the address can change, depending on the assignment method chosen. For more information, see [Create a public IP address](#). To learn more about outbound connections from Azure resources, see [Understand outbound connections](#).

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.31 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

The account you log into, or connect to Azure with, must be assigned to the [network contributor](#) role or to a

custom role that is assigned the appropriate actions listed in [Permissions](#).

Public IP addresses have a nominal charge. To view the pricing, read the [IP address pricing](#) page.

## Create a public IP address

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Enter *public ip address* in the *Search the Marketplace* box. When **Public IP address** appears in the search results, select it.
3. Under **Public IP address**, select **Create**.
4. Enter, or select values for the following settings, under **Create public IP address**, then select **Create**:

SETTING	REQUIRED?	DETAILS
IP Version	Yes	Select IPv4 or IPv6 or Both. Selecting Both will result in 2 Public IP addresses being created- 1 IPv4 address and 1 IPv6 address. Learn more about <a href="#">IPv6 in Azure VNETs</a> .

SETTING	REQUIRED?	DETAILS
SKU	Yes	<p>All public IP addresses created before the introduction of SKUs are <b>Basic</b> SKU public IP addresses. You cannot change the SKU after the public IP address is created. A standalone virtual machine, virtual machines within an availability set, or virtual machine scale sets can use Basic or Standard SKUs. Mixing SKUs between virtual machines within availability sets or scale sets or standalone VMs is not allowed.</p> <p><b>Basic</b> SKU: If you are creating a public IP address in a region that supports availability zones, the <b>Availability zone</b> setting is set to <i>None</i> by default. Basic Public IPs do not support Availability zones.</p> <p><b>Standard</b> SKU: A Standard SKU public IP can be associated to a virtual machine or a load balancer front end. If you're creating a public IP address in a region that supports availability zones, the <b>Availability zone</b> setting is set to <i>Zone-redundant</i> by default. For more information about availability zones, see the <b>Availability zone</b> setting. The standard SKU is required if you associate the address to a Standard load balancer. To learn more about standard load balancers, see <a href="#">Azure load balancer standard SKU</a>. When you assign a standard SKU public IP address to a virtual machine's network interface, you must explicitly allow the intended traffic with a <a href="#">network security group</a>.</p> <p>Communication with the resource fails until you create and associate a network security group and explicitly allow the desired traffic.</p>
Name	Yes	The name must be unique within the resource group you select.

SETTING	REQUIRED?	DETAILS
IP address assignment	Yes	<p><b>Dynamic:</b> Dynamic addresses are assigned only after a public IP address is associated to an Azure resource, and the resource is started for the first time. Dynamic addresses can change if they're assigned to a resource, such as a virtual machine, and the virtual machine is stopped (deallocated), and then restarted. The address remains the same if a virtual machine is rebooted or stopped (but not deallocated). Dynamic addresses are released when a public IP address resource is dissociated from a resource it is associated to. <b>Static:</b> Static addresses are assigned when a public IP address is created. Static addresses are not released until a public IP address resource is deleted. If the address is not associated to a resource, you can change the assignment method after the address is created. If the address is associated to a resource, you may not be able to change the assignment method. If you select <i>IPv6</i> for the <b>IP version</b>, the assignment method must be <i>Dynamic</i> for Basic SKU. Standard SKU addresses are <i>Static</i> for both IPv4 and IPv6.</p>
Idle timeout (minutes)	No	How many minutes to keep a TCP or HTTP connection open without relying on clients to send keep-alive messages. If you select <i>IPv6</i> for <b>IP Version</b> , this value can't be changed.

Setting	Required?	Details
DNS name label	No	Must be unique within the Azure location you create the name in (across all subscriptions and all customers). Azure automatically registers the name and IP address in its DNS so you can connect to a resource with the name. Azure appends a default subnet such as <i>location.cloudapp.azure.com</i> (where location is the location you select) to the name you provide, to create the fully qualified DNS name. If you choose to create both address versions, the same DNS name is assigned to both the IPv4 and IPv6 addresses. Azure's default DNS contains both IPv4 A and IPv6 AAAA name records and responds with both records when the DNS name is looked up. The client chooses which address (IPv4 or IPv6) to communicate with. Instead of, or in addition to, using the DNS name label with the default suffix, you can use the Azure DNS service to configure a DNS name with a custom suffix that resolves to the public IP address. For more information, see <a href="#">Use Azure DNS with an Azure public IP address</a> .
Name (Only visible if you select IP Version of <b>Both</b> )	Yes, if you select IP Version of <b>Both</b>	The name must be different than the name you enter for the first <b>Name</b> in this list. If you choose to create both an IPv4 and an IPv6 address, the portal creates two separate public IP address resources, one with each IP address version assigned to it.
IP address assignment (Only visible if you select IP Version of <b>Both</b> )	Yes, if you select IP Version of <b>Both</b>	Same restrictions as IP Address Assignment above
Subscription	Yes	Must exist in the same <a href="#">subscription</a> as the resource to which you'll associate the Public IP's.
Resource group	Yes	Can exist in the same, or different, <a href="#">resource group</a> as the resource to which you'll associate the Public IP's.
Location	Yes	Must exist in the same <a href="#">location</a> , also referred to as region, as the resource to which you'll associate the Public IP's.

Setting	Required?	Details
Availability zone	No	This setting only appears if you select a supported location. For a list of supported locations, see <a href="#">Availability zones overview</a> . If you selected the <b>Basic</b> SKU, <i>None</i> is automatically selected for you. If you prefer to guarantee a specific zone, you may select a specific zone. Either choice is not zone-redundant. If you selected the <b>Standard</b> SKU: Zone-redundant is automatically selected for you and makes your data path resilient to zone failure. If you prefer to guarantee a specific zone, which is not resilient to zone failure, you may select a specific zone.

## Commands

Though the portal provides the option to create two public IP address resources (one IPv4 and one IPv6), the following CLI and PowerShell commands create one resource with an address for one IP version or the other. If you want two public IP address resources, one for each IP version, you must run the command twice, specifying different names and IP versions for the public IP address resources.

Tool	Command
CLI	<a href="#">az network public-ip create</a>
PowerShell	<a href="#">New-AzPublicIpAddress</a>

## View, change settings for, or delete a public IP address

- In the box that contains the text *Search resources* at the top of the Azure portal, type *public ip address*. When **Public IP addresses** appear in the search results, select it.
- Select the name of the public IP address you want to view, change settings for, or delete from the list.
- Complete one of the following options, depending on whether you want to view, delete, or change the public IP address.
  - View:** The **Overview** section shows key settings for the public IP address, such as the network interface it's associated to (if the address is associated to a network interface). The portal does not display the version of the address (IPv4 or IPv6). To view the version information, use the PowerShell or CLI command to view the public IP address. If the IP address version is IPv6, the assigned address is not displayed by the portal, PowerShell, or the CLI.
  - Delete:** To delete the public IP address, select **Delete** in the **Overview** section. If the address is currently associated to an IP configuration, it cannot be deleted. If the address is currently associated with a configuration, select **Dissociate** to dissociate the address from the IP configuration.
  - Change:** select **Configuration**. Change settings using the information in step 4 of [Create a public IP address](#). To change the assignment for an IPv4 address from static to dynamic, you must first dissociate the public IPv4 address from the IP configuration it's associated to. You can then change the assignment method to dynamic and select **Associate** to associate the IP address to the same IP configuration, a different configuration, or you can leave it dissociated. To dissociate a public IP address, in the **Overview** section, select **Dissociate**.

### WARNING

When you change the assignment method from static to dynamic, you lose the IP address that was assigned to the public IP address. While the Azure public DNS servers maintain a mapping between static or dynamic addresses and any DNS name label (if you defined one), a dynamic IP address can change when the virtual machine is started after being in the stopped (deallocated) state. To prevent the address from changing, assign a static IP address.

## Commands

TOOL	COMMAND
CLI	<code>az network public-ip list</code> to list public IP addresses, <code>az network public-ip show</code> to show settings; <code>az network public-ip update</code> to update; <code>az network public-ip delete</code> to delete
PowerShell	<code>Get-AzPublicIpAddress</code> to retrieve a public IP address object and view its settings, <code>Set-AzPublicIpAddress</code> to update settings; <code>Remove-AzPublicIpAddress</code> to delete

## Assign a public IP address

Learn how to assign a public IP address to the following resources:

- A [Windows](#) or [Linux](#) VM (when creating), or to an [existing VM](#)
- [Internet-facing Load Balancer](#)
- [Azure Application Gateway](#)
- [Site-to-site connection using an Azure VPN Gateway](#)
- [Azure Virtual Machine Scale Set](#)

## Permissions

To perform tasks on public IP addresses, your account must be assigned to the [network contributor](#) role or to a [custom](#) role that is assigned the appropriate actions listed in the following table:

ACTION	NAME
Microsoft.Network/publicIPAddresses/read	Read a public IP address
Microsoft.Network/publicIPAddresses/write	Create or update a public IP address
Microsoft.Network/publicIPAddresses/delete	Delete a public IP address
Microsoft.Network/publicIPAddresses/join/action	Associate a public IP address to a resource

## Next steps

- Create a public IP address using [PowerShell](#) or [Azure CLI](#) sample scripts, or using [Azure Resource Manager templates](#)
- Create and apply [Azure policy](#) for public IP addresses

# Create, change, or delete a public IP address prefix

5/20/2019 • 5 minutes to read • [Edit Online](#)

Learn about a public IP address prefix and how to create, change, and delete one. A public IP address prefix is a contiguous range of addresses based on the number of public IP addresses you specify. The addresses are assigned to your subscription. When you create a public IP address resource, you can assign a static public IP address from the prefix and associate the address to virtual machines, load balancers, or other resources, to enable internet connectivity. If you're not familiar with public IP address prefixes, see [Public IP address prefix overview](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.41 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI 2.0](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

The account you log into, or connect to Azure with, must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Permissions](#).

Public IP address prefixes have a charge. For details, see [pricing](#).

## Create a public IP address prefix

1. At the top, left corner of the portal, select **+ Create a resource**.
2. Enter *public ip address prefix* in the *Search the Marketplace* box. When **Public IP address prefix** appears in the search results, select it.
3. Under **Public IP address prefix**, select **Create**.

4. Enter, or select values for the following settings, under **Create public IP address prefix**, then select

**Create:**

SETTING	REQUIRED?	DETAILS
Subscription	Yes	Must exist in the same <a href="#">subscription</a> as the resource you want to associate the public IP address to.
Resource group	Yes	Can exist in the same, or different, <a href="#">resource group</a> as the resource you want to associate the public IP address to.
Name	Yes	The name must be unique within the resource group you select.
Region	Yes	Must exist in the same <a href="#">region</a> as the public IP addresses you'll assign addresses from the range.
Prefix size	Yes	The size of the prefix you need. A /28 or 16 IP addresses is the default.

## Commands

TOOL	COMMAND
CLI	<a href="#">az network public-ip prefix create</a>
PowerShell	<a href="#">New-AzPublicIpPrefix</a>

## Create a static public IP address from a prefix

Once you create a prefix, you must create static IP addresses from the prefix. In order to do this, follow steps below.

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *public ip address prefix*. When **Public IP address prefixes** appear in the search results, select it.
2. Select the prefix you want to create public IPs from.
3. When it appears in the search results, select it and click on **+Add IP address** in the Overview section.
4. Enter or select values for the following settings under **Create public IP address**. Since a prefix is for Standard SKU, IPv4, and static, you only need to provide the following information:

SETTING	REQUIRED?	DETAILS
Name	Yes	The name of the public IP address must be unique within the resource group you select.

SETTING	REQUIRED?	DETAILS
Idle timeout (minutes)	No	How many minutes to keep a TCP or HTTP connection open without relying on clients to send keep-alive messages.
DNS name label	No	Must be unique within the Azure region you create the name in (across all subscriptions and all customers). Azure automatically registers the name and IP address in its DNS so you can connect to a resource with the name. Azure appends a default subnet such as <i>location.cloudapp.azure.com</i> (where location is the location you select) to the name you provide, to create the fully qualified DNS name. For more information, see <a href="#">Use Azure DNS with an Azure public IP address</a> .

Alternatively you may use the CLI and PS commands below with the --public-ip-prefix (CLI) and -PublicIpPrefix (PS) parameters, to create a Public IP address resource.

TOOL	COMMAND
CLI	<a href="#">az network public-ip create</a>
PowerShell	<a href="#">New-AzPublicIpAddress</a>

## View or delete a prefix

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *public ip address prefix*. When **Public IP address prefixes** appear in the search results, select it.
2. Select the name of the public IP address prefix that you want to view, change settings for, or delete from the list.
3. Complete one of the following options, depending on whether you want to view, delete, or change the public IP address prefix.
  - **View:** The **Overview** section shows key settings for the public IP address prefix, such as prefix.
  - **Delete:** To delete the public IP address prefix, select **Delete** in the **Overview** section. If addresses within the prefix are associated to public IP address resources, you must first delete the public IP address resources. See [delete a public IP address](#).

## Commands

TOOL	COMMAND
CLI	<a href="#">az network public-ip prefix list</a> to list public IP addresses, <a href="#">az network public-ip prefix show</a> to show settings; <a href="#">az network public-ip prefix update</a> to update; <a href="#">az network public-ip prefix delete</a> to delete

TOOL	COMMAND
PowerShell	<a href="#">Get-AzPublicIpPrefix</a> to retrieve a public IP address object and view its settings, <a href="#">Set-AzPublicIpPrefix</a> to update settings; <a href="#">Remove-AzPublicIpPrefix</a> to delete

## Permissions

To perform tasks on public IP address prefixes, your account must be assigned to the [network contributor](#) role or to a [custom](#) role that is assigned the appropriate actions listed in the following table:

ACTION	NAME
Microsoft.Network/publicIPPrefixes/read	Read a public IP address prefix
Microsoft.Network/publicIPPrefixes/write	Create or update a public IP address prefix
Microsoft.Network/publicIPPrefixes/delete	Delete a public IP address prefix
Microsoft.Network/publicIPPrefixes/join/action	Create a public IP address from a prefix

## Next steps

- Learn about scenarios and benefits of using a [public IP prefix](#)

# Add, change, or remove IP addresses for an Azure network interface

1/23/2020 • 15 minutes to read • [Edit Online](#)

Learn how to add, change, and remove public and private IP addresses for a network interface. Private IP addresses assigned to a network interface enable a virtual machine to communicate with other resources in an Azure virtual network and connected networks. A private IP address also enables outbound communication to the Internet using an unpredictable IP address. A [Public IP address](#) assigned to a network interface enables inbound communication to a virtual machine from the Internet. The address also enables outbound communication from the virtual machine to the Internet using a predictable IP address. For details, see [Understanding outbound connections in Azure](#).

If you need to create, change, or delete a network interface, read the [Manage a network interface](#) article. If you need to add network interfaces to or remove network interfaces from a virtual machine, read the [Add or remove network interfaces](#) article.

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.31 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

The account you log into, or connect to Azure with, must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Network interface permissions](#).

## Add IP addresses

You can add as many [private](#) and [public IPv4](#) addresses as necessary to a network interface, within the limits

listed in the [Azure limits](#) article. You can add a private IPv6 address to one [secondary IP configuration](#) (as long as there are no existing secondary IP configurations) for an existing network interface. Each network interface may have at most one IPv6 private address. You can optionally add a public IPv6 address to an IPv6 network interface configuration. See [IPv6](#) for details about using IPv6 addresses.

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appear in the search results, select it.
2. Select the network interface you want to add an IPv4 address for from the list.
3. Under **SETTINGS**, select **IP configurations**.
4. Under **IP configurations**, select **+ Add**.
5. Specify the following, then select **OK**:

SETTING	REQUIRED?	DETAILS
Name	Yes	Must be unique for the network interface
Type	Yes	Since you're adding an IP configuration to an existing network interface, and each network interface must have a <a href="#">primary</a> IP configuration, your only option is <b>Secondary</b> .
Private IP address assignment method	Yes	<b>Dynamic:</b> Azure assigns the next available address for the subnet address range the network interface is deployed in. <b>Static:</b> You assign an unused address for the subnet address range the network interface is deployed in.
Public IP address	No	<b>Disabled:</b> No public IP address resource is currently associated to the IP configuration. <b>Enabled:</b> Select an existing IPv4 Public IP address, or create a new one. To learn how to create a public IP address, read the <a href="#">Public IP addresses</a> article.

6. Manually add secondary private IP addresses to the virtual machine operating system by completing the instructions in the [Assign multiple IP addresses to virtual machine operating systems](#) article. See [private](#) IP addresses for special considerations before manually adding IP addresses to a virtual machine operating system. Do not add any public IP addresses to the virtual machine operating system.

## Commands

TOOL	COMMAND
CLI	<a href="#">az network nic ip-config create</a>
PowerShell	<a href="#">Add-AzNetworkInterfaceIpConfig</a>

## Change IP address settings

You may need to change the assignment method of an IPv4 address, change the static IPv4 address, or change the public IP address assigned to a network interface. If you're changing the private IPv4 address of a secondary IP configuration associated with a secondary network interface in a virtual machine (learn more about [primary and secondary network interfaces](#)), place the virtual machine into the stopped (deallocated) state before completing the following steps:

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appear in the search results, select it.
2. Select the network interface that you want to view or change IP address settings for from the list.
3. Under **SETTINGS**, select **IP configurations**.
4. Select the IP configuration you want to modify from the list.
5. Change the settings, as desired, using the information about the settings in step 5 of [Add an IP configuration](#).
6. Select **Save**.

#### NOTE

If the primary network interface has multiple IP configurations and you change the private IP address of the primary IP configuration, you must manually reassign the primary and secondary IP addresses to the network interface within Windows (not required for Linux). To manually assign IP addresses to a network interface within an operating system, see [Assign multiple IP addresses to virtual machines](#). For special considerations before manually adding IP addresses to a virtual machine operating system, see [private](#) IP addresses. Do not add any public IP addresses to the virtual machine operating system.

## Commands

TOOL	COMMAND
CLI	<a href="#">az network nic ip-config update</a>
PowerShell	<a href="#">Set-AzNetworkInterfaceIpConfig</a>

## Remove IP addresses

You can remove [private](#) and [public](#) IP addresses from a network interface, but a network interface must always have at least one private IPv4 address assigned to it.

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appear in the search results, select it.
2. Select the network interface that you want to remove IP addresses from the list.
3. Under **SETTINGS**, select **IP configurations**.
4. Right-select a [secondary](#) IP configuration (you cannot delete the [primary](#) configuration) that you want to delete, select **Delete**, then select **Yes**, to confirm the deletion. If the configuration had a public IP address resource associated to it, the resource is dissociated from the IP configuration, but the resource is not deleted.

## Commands

TOOL	COMMAND
CLI	<a href="#">az network nic ip-config delete</a>
PowerShell	<a href="#">Remove-AzNetworkInterfaceIpConfig</a>

# IP configurations

Private and (optionally) public IP addresses are assigned to one or more IP configurations assigned to a network interface. There are two types of IP configurations:

## Primary

Each network interface is assigned one primary IP configuration. A primary IP configuration:

- Has a private IPv4 address assigned to it. You cannot assign a private IPv6 address to a primary IP configuration.
- May also have a public IPv4 address assigned to it. You cannot assign a public IPv6 address to a primary (IPv4) IP configuration.

## Secondary

In addition to a primary IP configuration, a network interface may have zero or more secondary IP configurations assigned to it. A secondary IP configuration:

- Must have a private IPv4 or IPv6 address assigned to it. If the address is IPv6, the network interface can only have one secondary IP configuration. If the address is IPv4, the network interface may have multiple secondary IP configurations assigned to it. To learn more about how many private and public IPv4 addresses can be assigned to a network interface, see the [Azure limits](#) article.
- May also have a public IPv4 or IPv6 address assigned to it. Assigning multiple IPv4 addresses to a network interface is helpful in scenarios such as:
  - Hosting multiple websites or services with different IP addresses and SSL certificates on a single server.
  - A virtual machine serving as a network virtual appliance, such as a firewall or load balancer.
  - The ability to add any of the private IPv4 addresses for any of the network interfaces to an Azure Load Balancer back-end pool. In the past, only the primary IPv4 address for the primary network interface could be added to a back-end pool. To learn more about how to load balance multiple IPv4 configurations, see the [Load balancing multiple IP configurations](#) article.
  - The ability to load balance one IPv6 address assigned to a network interface. To learn more about how to load balance to a private IPv6 address, see the [Load balance IPv6 addresses](#) article.

# Address types

You can assign the following types of IP addresses to an [IP configuration](#):

## Private

Private IPv4 or IPv6 addresses enable a virtual machine to communicate with other resources in a virtual network or other connected networks.

By default, the Azure DHCP servers assign the private IPv4 address for the [primary IP configuration](#) of the Azure network interface to the network interface within the virtual machine operating system. Unless necessary, you should never manually set the IP address of a network interface within the virtual machine's operating system.

### WARNING

If the IPv4 address set as the primary IP address of a network interface within a virtual machine's operating system is ever different than the private IPv4 address assigned to the primary IP configuration of the primary network interface attached to a virtual machine within Azure, you lose connectivity to the virtual machine.

There are scenarios where it's necessary to manually set the IP address of a network interface within the virtual machine's operating system. For example, you must manually set the primary and secondary IP addresses of a

Windows operating system when adding multiple IP addresses to an Azure virtual machine. For a Linux virtual machine, you may only need to manually set the secondary IP addresses. See [Add IP addresses to a VM operating system](#) for details. If you ever need to change the address assigned to an IP configuration, it's recommended that you:

1. Ensure that the virtual machine is receiving an address from the Azure DHCP servers. Once you have, change the assignment of the IP address back to DHCP within the operating system and restart the virtual machine.
2. Stop (deallocate) the virtual machine.
3. Change the IP address for the IP configuration within Azure.
4. Start the virtual machine.
5. [Manually configure](#) the secondary IP addresses within the operating system (and also the primary IP address within Windows) to match what you set within Azure.

By following the previous steps, the private IP address assigned to the network interface within Azure, and within a virtual machine's operating system, remain the same. To keep track of which virtual machines within your subscription that you've manually set IP addresses within an operating system for, consider adding an Azure [tag](#) to the virtual machines. You might use "IP address assignment: Static", for example. This way, you can easily find the virtual machines within your subscription that you've manually set the IP address for within the operating system.

In addition to enabling a virtual machine to communicate with other resources within the same, or connected virtual networks, a private IP address also enables a virtual machine to communicate outbound to the Internet. Outbound connections are source network address translated by Azure to an unpredictable public IP address. To learn more about Azure outbound Internet connectivity, read the [Azure outbound Internet connectivity](#) article. You cannot communicate inbound to a virtual machine's private IP address from the Internet. If your outbound connections require a predictable public IP address, associate a public IP address resource to a network interface.

## Public

Public IP addresses assigned through a public IP address resource enable inbound connectivity to a virtual machine from the Internet. Outbound connections to the Internet use a predictable IP address. See [Understanding outbound connections in Azure](#) for details. You may assign a public IP address to an IP configuration, but aren't required to. If you don't assign a public IP address to a virtual machine by associating a public IP address resource, the virtual machine can still communicate outbound to the Internet. In this case, the private IP address is source network address translated by Azure to an unpredictable public IP address. To learn more about public IP address resources, see [Public IP address resource](#).

There are limits to the number of private and public IP addresses that you can assign to a network interface. For details, read the [Azure limits](#) article.

### NOTE

Azure translates a virtual machine's private IP address to a public IP address. As a result, a virtual machine's operating system is unaware of any public IP address assigned to it, so there is no need to ever manually assign a public IP address within the operating system.

## Assignment methods

Public and private IP addresses are assigned using one of the following assignment methods:

### Dynamic

Dynamic private IPv4 and IPv6 (optionally) addresses are assigned by default.

- **Public only:** Azure assigns the address from a range unique to each Azure region. To learn which ranges are assigned to each region, see [Microsoft Azure Datacenter IP Ranges](#). The address can change when a virtual machine is stopped (deallocated), then started again. You cannot assign a public IPv6 address to an IP configuration using either assignment method.
- **Private only:** Azure reserves the first four addresses in each subnet address range, and doesn't assign the addresses. Azure assigns the next available address to a resource from the subnet address range. For example, if the subnet's address range is 10.0.0.0/16, and addresses 10.0.0.0.4-10.0.0.14 are already assigned (.0-.3 are reserved), Azure assigns 10.0.0.15 to the resource. Dynamic is the default allocation method. Once assigned, dynamic IP addresses are only released if a network interface is deleted, assigned to a different subnet within the same virtual network, or the allocation method is changed to static, and a different IP address is specified. By default, Azure assigns the previous dynamically assigned address as the static address when you change the allocation method from dynamic to static.

## Static

You can (optionally) assign a public or private static IPv4 or IPv6 address to an IP configuration. To learn more about how Azure assigns static public IPv4 addresses, see [Public IP addresses](#).

- **Public only:** Azure assigns the address from a range unique to each Azure region. You can download the list of ranges (prefixes) for the Azure [Public](#), [US government](#), [China](#), and [Germany](#) clouds. The address doesn't change until the public IP address resource it's assigned to is deleted, or the assignment method is changed to dynamic. If the public IP address resource is associated to an IP configuration, it must be dissociated from the IP configuration before changing its assignment method.
- **Private only:** You select and assign an address from the subnet's address range. The address you assign can be any address within the subnet address range that is not one of the first four addresses in the subnet's address range and is not currently assigned to any other resource in the subnet. Static addresses are only released if a network interface is deleted. If you change the allocation method to static, Azure dynamically assigns the previously assigned dynamic IP address as the static address, even if the address isn't the next available address in the subnet's address range. The address also changes if the network interface is assigned to a different subnet within the same virtual network, but to assign the network interface to a different subnet, you must first change the allocation method from static to dynamic. Once you've assigned the network interface to a different subnet, you can change the allocation method back to static, and assign an IP address from the new subnet's address range.

## IP address versions

You can specify the following versions when assigning addresses:

### IPv4

Each network interface must have one [primary](#) IP configuration with an assigned [private IPv4](#) address. You can add one or more [secondary](#) IP configurations that each have an IPv4 private and (optionally) an IPv4 [public](#) IP address.

### IPv6

You can assign zero or one private [IPv6](#) address to one secondary IP configuration of a network interface. The network interface cannot have any existing secondary IP configurations. Each network interface may have at most one IPv6 private address. You can optionally add a public IPv6 address to an IPv6 network interface configuration.

**NOTE**

Though you can create a network interface with an IPv6 address using the portal, you can't add an existing network interface to a new, or existing virtual machine, using the portal. Use PowerShell or the Azure CLI to create a network interface with a private IPv6 address, then attach the network interface when creating a virtual machine. You cannot attach a network interface with a private IPv6 address assigned to it to an existing virtual machine. You cannot add a private IPv6 address to an IP configuration for any network interface attached to a virtual machine using any tools (portal, CLI, or PowerShell).

You can't assign a public IPv6 address to a primary or secondary IP configuration.

## SKUs

A public IP address is created with the basic or standard SKU. For more information about SKU differences, see [Manage public IP addresses](#).

**NOTE**

When you assign a standard SKU public IP address to a virtual machine's network interface, you must explicitly allow the intended traffic with a [network security group](#). Communication with the resource fails until you create and associate a network security group and explicitly allow the desired traffic.

## Next steps

To create a virtual machine with different IP configurations, read the following articles:

TASK	TOOL
Create a VM with multiple network interfaces	<a href="#">CLI</a> , <a href="#">PowerShell</a>
Create a single NIC VM with multiple IPv4 addresses	<a href="#">CLI</a> , <a href="#">PowerShell</a>
Create a single NIC VM with a private IPv6 address (behind an Azure Load Balancer)	<a href="#">CLI</a> , <a href="#">PowerShell</a> , <a href="#">Azure Resource Manager template</a>

# Manage Azure DDoS Protection Standard using the Azure portal

12/20/2019 • 12 minutes to read • [Edit Online](#)

Learn how to enable and disable distributed denial of service (DDoS) protection, and use telemetry to mitigate a DDoS attack with Azure DDoS Protection Standard. DDoS Protection Standard protects Azure resources such as virtual machines, load balancers, and application gateways that have an Azure public IP address assigned to it. To learn more about DDoS Protection Standard and its capabilities, see [DDoS Protection Standard overview](#).

Before completing any steps in this tutorial, log in to the Azure portal at <https://portal.azure.com> with an account assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Permissions](#).

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Create a DDoS protection plan

A DDoS protection plan defines a set of virtual networks that have DDoS protection standard enabled, across subscriptions. You can configure one DDoS protection plan for your organization and link virtual networks from multiple subscriptions to the same plan. The DDoS Protection Plan itself is also associated with a subscription, that you select during the creation of the plan. The DDoS Protection Plan works across regions and subscriptions.

Example -you can create the plan in Region East-US and link to subscription #1 in your tenant. The same plan can be linked to virtual networks from other subscriptions in different regions, across your tenant. The subscription the plan is associated to incurs the monthly recurring bill for the plan, as well as overage charges, in case the number of protected public IP addresses exceed 100. For more information on DDoS pricing, see [pricing details](#).

Creation of more than one plan is not required for most organizations. A plan cannot be moved between subscriptions. If you want to change the subscription a plan is in, you have to [delete the existing plan](#) and create a new one.

1. Select **Create a resource** in the upper left corner of the Azure portal.
2. Search for *DDoS*. When **DDos protection plan** appears in the search results, select it.
3. Select **Create**.
4. Enter or select your own values, or enter, or select the following example values, and then select **Create**:

SETTING	VALUE
Name	myDdosProtectionPlan
Subscription	Select your subscription.
Resource group	Select <b>Create new</b> and enter <i>myResourceGroup</i>
Location	East US

## Enable DDoS for a new virtual network

1. Select **Create a resource** in the upper left corner of the Azure portal.

2. Select **Networking**, and then select **Virtual network**.
3. Enter or select your own values, or enter or select the following example values, accept the remaining defaults, and then select **Create**:

SETTING	VALUE
Name	myVirtualNetwork
Subscription	Select your subscription.
Resource group	Select <b>Use existing</b> , and then select <b>myResourceGroup</b>
Location	East US
DDos protection	Select <b>Standard</b> and then under <b>DDoS protection</b> , select <b>myDdosProtectionPlan</b> . The plan you select can be in the same, or different subscription than the virtual network, but both subscriptions must be associated to the same Azure Active Directory tenant.

You cannot move a virtual network to another resource group or subscription when DDoS Standard is enabled for the virtual network. If you need to move a virtual network with DDoS Standard enabled, disable DDoS Standard first, move the virtual network, and then enable DDoS standard. After the move, the auto-tuned policy thresholds for all the protected public IP addresses in the virtual network are reset.

## Enable DDoS for an existing virtual network

1. Create a DDoS protection plan by completing the steps in [Create a DDoS protection plan](#), if you don't have an existing DDoS protection plan.
2. Select **Create a resource** in the upper left corner of the Azure portal.
3. Enter the name of the virtual network that you want to enable DDoS Protection Standard for in the **Search resources, services, and docs box** at the top of the portal. When the name of the virtual network appears in the search results, select it.
4. Select **DDoS protection**, under **SETTINGS**.
5. Select **Standard**. Under **DDoS protection plan**, select an existing DDoS protection plan, or the plan you created in step 1, and then select **Save**. The plan you select can be in the same, or different subscription than the virtual network, but both subscriptions must be associated to the same Azure Active Directory tenant.

### Commands

- Azure CLI: [az network ddos-protection create](#)
- Powershell: [New-AzDdosProtectionPlan](#)

## Disable DDoS for a virtual network

1. Enter the name of the virtual network you want to disable DDoS protection standard for in the **Search resources, services, and docs box** at the top of the portal. When the name of the virtual network appears in the search results, select it.
2. Select **DDoS protection**, under **SETTINGS**.
3. Select **Basic** under **DDoS protection plan** and then select **Save**.

### Commands

- Azure CLI: [az network ddos-protection delete](#)

- Powershell: [Remove-AzDdosProtectionPlan](#)

## Work with DDoS protection plans

1. Select **All services** on the top, left of the portal.
2. Enter *DDoS* in the **Filter** box. When **DDoS protection plans** appear in the results, select it.
3. Select the protection plan you want to view from the list.
4. All virtual networks associated to the plan are listed.
5. If you want to delete a plan, you must first dissociate all virtual networks from it. To dissociate a plan from a virtual network, see [Disable DDoS for a virtual network](#).

## Configure alerts for DDoS protection metrics

You can select any of the available DDoS protection metrics to alert you when there's an active mitigation during an attack, using the Azure Monitor alert configuration. When the conditions are met, the address specified receives an alert email:

1. Select **All services** on the top, left of the portal.
2. Enter *Monitor* in the **Filter** box. When **Monitor** appears in the results, select it.
3. Select **Metrics** under **SHARED SERVICES**.
4. Enter, or select your own values, or enter the following example values, accept the remaining defaults, and then select **OK**:

SETTING	VALUE
Name	myDdosAlert
Subscription	Select the subscription that contains the public IP address you want to receive alerts for.
Resource group	Select the resource group that contains the public IP address you want to receive alerts for.
Resource	Select the public IP address that contains the public IP address you want to receive alerts for. DDoS monitors public IP addresses assigned to resources within a virtual network. If you don't have any resources with public IP addresses in the virtual network, you must first create a resource with a public IP address. You can monitor the public IP address of all resources deployed through Resource Manager (not classic) listed in <a href="#">Virtual network for Azure services</a> , except for Azure App Service Environments and Azure VPN Gateway. To continue with this tutorial, you can quickly create a <a href="#">Windows</a> or <a href="#">Linux</a> virtual machine.
Metric	Under DDoS attack or not
Threshold	1 - <b>1</b> means you are under attack. <b>0</b> means you are not under attack.
Period	Select whatever value you choose.

SETTING	VALUE
Notify via Email	Check the checkbox
Additional administrator	Enter your email address if you're not an email owner, contributor, or reader for the subscription.

Within a few minutes of attack detection, you receive an email from Azure Monitor metrics that looks similar to the following picture:

The screenshot shows the Azure Monitor Metrics dashboard. At the top, there are three large numbers: 'Alerts fired' (1), 'Activity log errors' (0), and 'Service Health' (0). Below these are three status indicators: 'Service Issues' (0), 'Planned Maintenance' (0), and 'Health Advisories' (0). Under 'Service Health', there are two tabs: 'Alert sources (1)' (underlined) and 'Application Insights (2)'. Below this is a search bar labeled 'Filter alerts...'. A table follows, with columns: NAME, STATUS, CONDITION, and SOURCE. One row is highlighted in blue, showing 'DDoS Attack -App Gateway...' with a warning icon, 'Failed locations >= 3', and 'Metrics' under SOURCE.

To simulate a DDoS attack to validate your alert, see [Validate DDoS detection](#).

You can also learn more about [configuring webhooks](#) and [logic apps](#) for creating alerts.

## Use DDoS protection telemetry

Telemetry for an attack is provided through Azure Monitor in real time. The telemetry is available only for the duration that a public IP address is under mitigation. You don't see telemetry before or after an attack is mitigated.

1. Select **All services** on the top, left of the portal.
2. Enter **Monitor** in the **Filter** box. When **Monitor** appears in the results, select it.
3. Select **Metrics**, under **SHARED SERVICES**.
4. Select the **Subscription** and **Resource group** that contain the public IP address that you want telemetry for.
5. Select **Public IP Address** for **Resource type**, then select the specific public IP address you want telemetry for.
6. A series of **Available Metrics** appear on the left side of the screen. These metrics, when selected, are graphed in the **Azure Monitor Metrics Chart** on the overview screen.
7. Select the **aggregation** type as **Max**

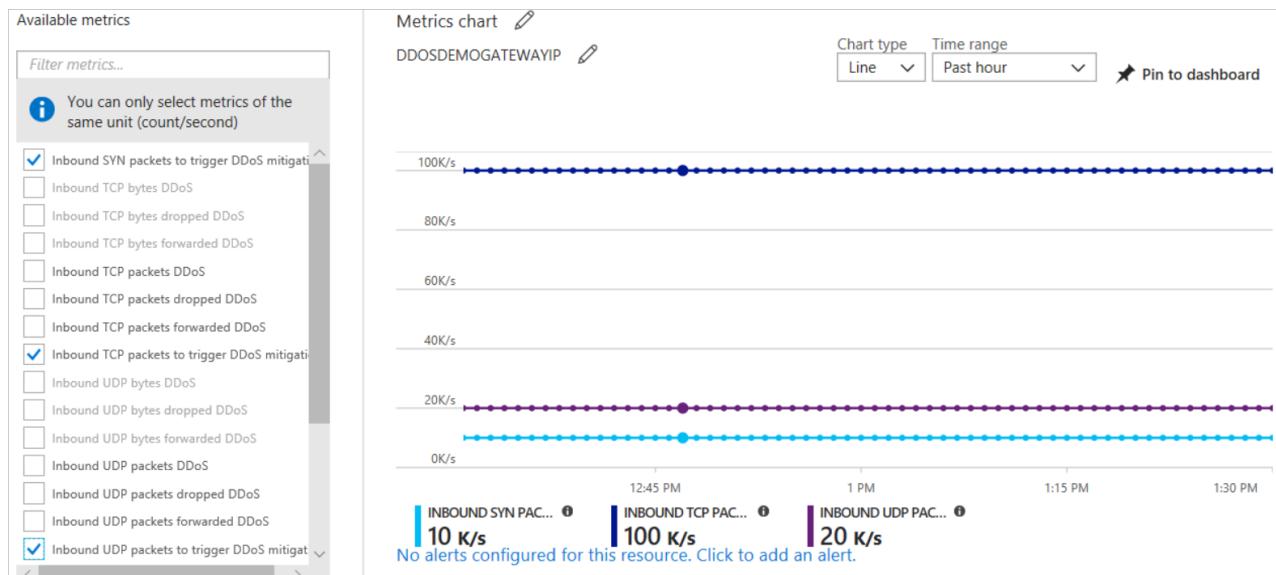
The metric names present different packet types, and bytes vs. packets, with a basic construct of tag names on each metric as follows:

- **Dropped tag name** (for example, **Inbound Packets Dropped DDoS**): The number of packets dropped/scrubbed by the DDoS protection system.
- **Forwarded tag name** (for example **Inbound Packets Forwarded DDoS**): The number of packets forwarded by the DDoS system to the destination VIP – traffic that was not filtered.
- **No tag name** (for example **Inbound Packets DDoS**): The total number of packets that came into the scrubbing system – representing the sum of the packets dropped and forwarded.

To simulate a DDoS attack to validate telemetry, see [Validate DDoS detection](#).

## View DDoS mitigation policies

DDoS Protection Standard applies three auto-tuned mitigation policies (TCP SYN, TCP & UDP) for each public IP address of the protected resource, in the virtual network that has DDoS enabled. You can view the policy thresholds by selecting the **Inbound TCP packets to trigger DDoS mitigation** and **Inbound UDP packets to trigger DDoS mitigation** metrics with **aggregation** type as 'Max', as shown in the following picture:



Policy thresholds are auto-configured via Azure machine learning-based network traffic profiling. Only when the policy threshold is breached does DDoS mitigation occur for the IP address under attack.

## Configure DDoS attack analytics

Azure DDoS Protection standard provides detailed attack insights and visualization with DDoS Attack Analytics. Customers protecting their virtual networks against DDoS attacks have detailed visibility into attack traffic and actions taken to mitigate the attack via attack mitigation reports & mitigation flow logs.

## Configure DDoS attack mitigation reports

Attack mitigation reports uses the Netflow protocol data which is aggregated to provide detailed information about the attack on your resource. Anytime a public IP resource is under attack, the report generation will start as soon as the mitigation starts. There will be an incremental report generated every 5 mins and a post-mitigation report for the whole mitigation period. This is to ensure that in an event the DDoS attack continues for a longer duration of time, you will be able to view the most current snapshot of mitigation report every 5 minutes and a complete summary once the attack mitigation is over.

1. Select **All services** on the top, left of the portal.
2. Enter **Monitor** in the **Filter** box. When **Monitor** appears in the results, select it.
3. Under **SETTINGS**, select **Diagnostic Settings**.
4. Select the **Subscription** and **Resource group** that contain the public IP address you want to log.
5. Select **Public IP Address** for **Resource type**, then select the specific public IP address you want to log metrics for.
6. Select **Turn on diagnostics to collect the DDoSMitigationReports log** and then select as many of the following options as you require:
  - **Archive to a storage account**: Data is written to an Azure Storage account. To learn more about this option, see [Archive diagnostic logs](#).

- **Stream to an event hub:** Allows a log receiver to pick up logs using an Azure Event Hub. Event hubs enable integration with Splunk or other SIEM systems. To learn more about this option, see [Stream diagnostic logs to an event hub](#).
- **Send to Log Analytics:** Writes logs to the Azure Monitor service. To learn more about this option, see [Collect logs for use in Azure Monitor logs](#).

Both the incremental & post-attack mitigation reports include the following fields

- Attack vectors
- Traffic statistics
- Reason for dropped packets
- Protocols involved
- Top 10 source countries or regions
- Top 10 source ASNs

## Configure DDoS attack mitigation flow logs

Attack Mitigation Flow Logs allow you to review the dropped traffic, forwarded traffic and other interesting datapoints during an active DDoS attack in near-real time. You can ingest the constant stream of this data into your SIEM systems via event hub for near-real time monitoring, take potential actions and address the need of your defense operations.

1. Select **All services** on the top, left of the portal.
2. Enter *Monitor* in the **Filter** box. When **Monitor** appears in the results, select it.
3. Under **SETTINGS**, select **Diagnostic Settings**.
4. Select the **Subscription** and **Resource group** that contain the public IP address you want to log.
5. Select **Public IP Address** for **Resource type**, then select the specific public IP address you want to log metrics for.
6. Select **Turn on diagnostics to collect the DDoSMitigationFlowLogs log** and then select as many of the following options as you require:
  - **Archive to a storage account:** Data is written to an Azure Storage account. To learn more about this option, see [Archive diagnostic logs](#).
  - **Stream to an event hub:** Allows a log receiver to pick up logs using an Azure Event Hub. Event hubs enable integration with Splunk or other SIEM systems. To learn more about this option, see [Stream diagnostic logs to an event hub](#).
  - **Send to Log Analytics:** Writes logs to the Azure Monitor service. To learn more about this option, see [Collect logs for use in Azure Monitor logs](#).
7. To view the flow logs data in Azure analytics dashboard, you can import the sample dashboard from <https://github.com/Anupamvi/Azure-DDoS-Protection/raw/master/flowlogsbyip.zip>

Flow logs will have the following fields:

- Source IP
- Destination IP
- Source Port
- Destination port
- Protocol type
- Action taken during mitigation

## Validate DDoS detection

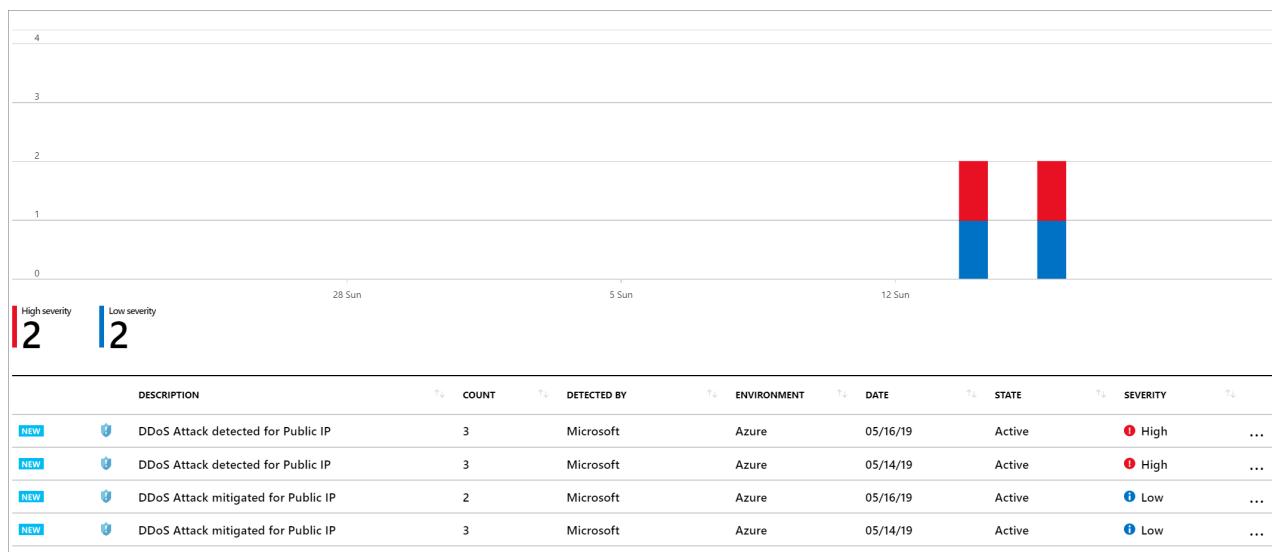
Microsoft has partnered with [BreakingPoint Cloud](#) to build an interface where you can generate traffic against DDoS Protection-enabled public IP addresses for simulations. The BreakPoint Cloud simulation allows you to:

- Validate how Microsoft Azure DDoS Protection protects your Azure resources from DDoS attacks
- Optimize your incident response process while under DDoS attack
- Document DDoS compliance
- Train your network security teams

## View DDoS protection alerts in Azure Security Center

Azure Security Center provides a list of [security alerts](#), with information to help investigate and remediate problems. With this feature, you get a unified view of alerts, including DDoS attack-related alerts and the actions taken to mitigate the attack in near-time. There are two specific alerts that you will see for any DDoS attack detection and mitigation:

- **DDoS Attack detected for Public IP:** This alert is generated when the DDoS protection service detects that one of your public IP addresses is the target of a DDoS attack.
- **DDoS Attack mitigated for Public IP:** This alert is generated when an attack on the public IP address has been mitigated. To view the alerts, open **Security Center** in the Azure portal. Under **Threat Protection**, select **Security alerts**. The following screenshot shows an example of the DDoS attack alerts.



The alerts include general information about the public IP address that's under attack, geo and threat intelligence information, and remediations steps.

## Permissions

To work with DDoS protection plans, your account must be assigned to the [network contributor](#) role or to a [custom](#) role that is assigned the appropriate actions listed in the following table:

ACTION	NAME
Microsoft.Network/ddosProtectionPlans/read	Read a DDoS protection plan
Microsoft.Network/ddosProtectionPlans/write	Create or update a DDoS protection plan
Microsoft.Network/ddosProtectionPlans/delete	Delete a DDoS protection plan

ACTION	NAME
Microsoft.Network/ddosProtectionPlans/join/action	Join a DDoS protection plan

To enable DDoS protection for a virtual network, your account must also be assigned the appropriate [actions for virtual networks](#).

## Next steps

- Create and apply [Azure policy](#) for virtual networks

# Create, change, or delete a network security group

1/14/2020 • 13 minutes to read • [Edit Online](#)

Security rules in network security groups enable you to filter the type of network traffic that can flow in and out of virtual network subnets and network interfaces. If you're not familiar with network security groups, see [Network security group overview](#) to learn more about them and complete the [Filter network traffic](#) tutorial to gain some experience with network security groups.

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.28 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

The account you log into, or connect to Azure with must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Permissions](#).

## Work with network security groups

You can create, [view all](#), [view details of](#), [change](#), and [delete](#) a network security group. You can also [associate](#) or [dissociate](#) a network security group from a network interface or subnet.

### Create a network security group

There is a limit to how many network security groups you can create per Azure location and subscription. For details, see [Azure limits](#).

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Select **Networking**, then select **network security group**.
3. Enter a **Name** for the network security group, select your **Subscription**, create a new **Resource group**, or

select an existing resource group, select a **Location**, and then select **Create**.

## Commands

- Azure CLI: [az network nsg create](#)
- PowerShell: [New-AzNetworkSecurityGroup](#)

## View all network security groups

In the search box at the top of the portal, enter *network security groups*. When **network security groups** appear in the search results, select it. The network security groups that exist in your subscription are listed.

## Commands

- Azure CLI: [az network nsg list](#)
- PowerShell: [Get-AzNetworkSecurityGroup](#)

## View details of a network security group

1. In the search box at the top of the portal, enter *network security groups*. When **network security groups** appear in the search results, select it.
2. Select the network security group in the list that you want to view details for. Under **SETTINGS** you can view the **Inbound security rules** and **Outbound security rules**, the **Network interfaces** and **Subnets** the network security group is associated to. You can also enable or disable **Diagnostic logs** and view **Effective security rules**. To learn more, see [Diagnostic logs](#) and [View effective security rules](#).
3. To learn more about the common Azure settings listed, see the following articles:
  - [Activity log](#)
  - [Access control \(IAM\)](#)
  - [Tags](#)
  - [Locks](#)
  - [Automation script](#)

## Commands

- Azure CLI: [az network nsg show](#)
- PowerShell: [Get-AzNetworkSecurityGroup](#)

## Change a network security group

1. In the search box at the top of the portal, enter *network security groups* in the search box. When **network security groups** appear in the search results, select it.
2. Select the network security group you want to change. The most common changes are [adding](#) or [removing](#) security rules and [Associating or dissociating a network security group to or from a subnet or network interface](#).

## Commands

- Azure CLI: [az network nsg update](#)
- PowerShell: [Set-AzNetworkSecurityGroup](#)

## Associate or dissociate a network security group to or from a subnet or network interface

To associate a network security group to, or dissociate a network security group from a network interface, see [Associate a network security group to, or dissociate a network security group from a network interface](#). To associate a network security group to, or dissociate a network security group from a subnet, see [Change subnet settings](#).

## Delete a network security group

If a network security group is associated to any subnets or network interfaces, it cannot be deleted. Dissociate a

network security group from all subnets and network interfaces before attempting to delete it.

1. In the search box at the top of the portal, enter *network security groups* in the search box. When **network security groups** appear in the search results, select it.
2. Select the network security group you want to delete from the list.
3. Select **Delete**, and then select **Yes**.

## Commands

- Azure CLI: [az network nsg delete](#)
- PowerShell: [Remove-AzNetworkSecurityGroup](#)

## Work with security rules

A network security group contains zero or more security rules. You can create, [view all](#), [view details of](#), [change](#), and [delete](#) a security rule.

### Create a security rule

There is a limit to how many rules per network security group can create per Azure location and subscription. For details, see [Azure limits](#).

1. In the search box at the top of the portal, enter *network security groups* in the search box. When **network security groups** appear in the search results, select it.
2. Select the network security group from the list that you want to add a security rule to.
3. Select **Inbound security rules** under **SETTINGS**. Several existing rules are listed. Some of the rules you may not have added. When a network security group is created, several default security rules are created in it. To learn more, see [default security rules](#). You can't delete default security rules, but you can override them with rules that have a higher priority.
4. Select **+ Add**. Select or add values for the following settings and then select **OK**:

SETTING	VALUE	DETAILS
---------	-------	---------

Setting	Value	Details
Source	Select <b>Any</b> , <b>Application security group</b> , <b>IP Addresses</b> , or <b>Service Tag</b> for inbound security rules. If you're creating an outbound security rule, the options are the same as options listed for <b>Destination</b> .	If you select <b>Application security group</b> , then select one or more existing application security groups that exist in the same region as the network interface. Learn how to <a href="#">create an application security group</a> . If you select <b>Application security group</b> for both the <b>Source</b> and <b>Destination</b> , the network interfaces within both application security groups must be in the same virtual network. If you select <b>IP Addresses</b> , then specify <b>Source IP addresses/CIDR ranges</b> . You can specify a single value or comma-separated list of multiple values. An example of multiple values is 10.0.0.0/16, 192.188.1.1. There are limits to the number of values you can specify. See <a href="#">Azure limits</a> for details. If you select <b>Service Tag</b> , then select one service tag. A service tag is a predefined identifier for a category of IP addresses. To learn more about available service tags, and what each tag represents, see <a href="#">Service tags</a> . If the IP address you specify is assigned to an Azure virtual machine, ensure that you specify the private IP, not the public IP address assigned to the virtual machine. Security rules are processed after Azure translates the public IP address to a private IP address for inbound security rules, and before Azure translates a private IP address to a public IP address for outbound rules. To learn more about public and private IP addresses in Azure, see <a href="#">IP address types</a> .
Source port ranges	Specify a single port, such as 80, a range of ports, such as 1024-65535, or a comma-separated list of single ports and/or port ranges, such as 80, 1024-65535. Enter an asterisk to allow traffic on any port.	The ports and ranges specify which ports traffic is allowed or denied by the rule. There are limits to the number of ports you can specify. See <a href="#">Azure limits</a> for details.

Setting	Value	Details
Destination	Select <b>Any</b> , <b>Application security group</b> , <b>IP addresses</b> , or <b>Virtual Network</b> for outbound security rules. If you're creating an inbound security rule, the options are the same as options listed for <b>Source</b> .	If you select <b>Application security group</b> you must then select one or more existing application security groups that exist in the same region as the network interface. Learn how to <a href="#">create an application security group</a> . If you select <b>Application security group</b> , then select one existing application security group that exists in the same region as the network interface. If you select <b>IP addresses</b> , then specify <b>Destination IP addresses/CIDR ranges</b> . Similar to <b>Source</b> and <b>Source IP addresses/CIDR ranges</b> , you can specify a single, or multiple addresses or ranges, and there are limits to the number you can specify. Selecting <b>Virtual network</b> , which is a service tag, means that traffic is allowed to all IP addresses within the address space of the virtual network. If the IP address you specify is assigned to an Azure virtual machine, ensure that you specify the private IP, not the public IP address assigned to the virtual machine. Security rules are processed after Azure translates the public IP address to a private IP address for inbound security rules, and before Azure translates a private IP address to a public IP address for outbound rules. To learn more about public and private IP addresses in Azure, see <a href="#">IP address types</a> .
Destination port ranges	Specify a single value, or comma-separated list of values.	Similar to <b>Source port ranges</b> , you can specify a single, or multiple ports and ranges, and there are limits to the number you can specify.
Protocol	Select <b>Any</b> , <b>TCP</b> , <b>UDP</b> or <b>ICMP</b> .	
Action	Select <b>Allow</b> or <b>Deny</b> .	
Priority	Enter a value between 100-4096 that is unique for all security rules within the network security group.	Rules are processed in priority order. The lower the number, the higher the priority. It's recommended that you leave a gap between priority numbers when creating rules, such as 100, 200, 300. Leaving gaps makes it easier to add rules in the future that you may need to make higher or lower than existing rules.

Setting	Value	Details
Name	A unique name for the rule within the network security group.	The name can be up to 80 characters. It must begin with a letter or number, end with a letter, number, or underscore, and may contain only letters, numbers, underscores, periods, or hyphens.
Description	An optional description.	

## Commands

- Azure CLI: [az network nsg rule create](#)
- PowerShell: [New-AzNetworkSecurityRuleConfig](#)

## View all security rules

A network security group contains zero or multiple rules. To learn more about the information listed when viewing rules, see [Network security group overview](#).

1. In the search box at the top of the portal, enter *network security groups*. When **network security groups** appear in the search results, select it.
2. Select the network security group from the list that you want to view rules for.
3. Select **Inbound security rules** or **Outbound security rules** under **SETTINGS**.

The list contains any rules you have created and the network security group [default security rules](#).

## Commands

- Azure CLI: [az network nsg rule list](#)
- PowerShell: [Get-AzNetworkSecurityRuleConfig](#)

## View details of a security rule

1. In the search box at the top of the portal, enter *network security groups*. When **network security groups** appear in the search results, select it.
2. Select the network security group you want to view details of a security rule for.
3. Select **Inbound security rules** or **Outbound security rules** under **SETTINGS**.
4. Select the rule you want to view details for. For a detailed explanation of all settings, see [security rule settings](#).

## Commands

- Azure CLI: [az network nsg rule show](#)
- PowerShell: [Get-AzNetworkSecurityRuleConfig](#)

## Change a security rule

1. Complete the steps in [View details of a security rule](#).
2. Change the settings as desired, and then select **Save**. For a detailed explanation of all settings, see [security rule settings](#).

## Commands

- Azure CLI: [az network nsg rule update](#)
- PowerShell: [Set-AzNetworkSecurityRuleConfig](#)

## Delete a security rule

1. Complete the steps in [View details of a security rule](#).

2. Select **Delete**, and then select **Yes**.

## Commands

- Azure CLI: [az network nsg rule delete](#)
- PowerShell: [Remove-AzNetworkSecurityRuleConfig](#)

# Work with application security groups

An application security group contains zero or more network interfaces. To learn more, see [application security groups](#). All network interfaces in an application security group must exist in the same virtual network. To learn how to add a network interface to an application security group, see [Add a network interface to an application security group](#).

## Create an application security group

1. Select **+** **Create a resource** on the upper, left corner of the Azure portal.
2. In the **Search the Marketplace** box, enter *Application security group*. When **Application security group** appears in the search results, select it, select **Application security group** again under **Everything**, and then select **Create**.
3. Enter, or select, the following information, and then select **Create**:

SETTING	VALUE
Name	The name must be unique within a resource group.
Subscription	Select your subscription.
Resource group	Select an existing resource group, or create a new one.
Location	Select a location

## Commands

- Azure CLI: [az network asg create](#)
- PowerShell: [New-AzApplicationSecurityGroup](#)

## View all application security groups

1. Select **All services** on the upper, left corner of the Azure portal.
2. Enter *application security groups* in the **All services Filter** box, and then select **Application security groups** when it appears in the search results.

## Commands

- Azure CLI: [az network asg list](#)
- PowerShell: [Get-AzApplicationSecurityGroup](#)

## View details of a specific application security group

1. Select **All services** on the upper, left corner of the Azure portal.
2. Enter *application security groups* in the **All services Filter** box, and then select **Application security groups** when it appears in the search results.
3. Select the application security group that you want to view the details of.

## Commands

- Azure CLI: [az network asg show](#)
- PowerShell: [Get-AzApplicationSecurityGroup](#)

## Change an application security group

1. Select **All services** on the upper, left corner of the Azure portal.
  2. Enter *application security groups* in the **All services Filter** box, and then select **Application security groups** when it appears in the search results.
  3. Select the application security group that you want to change settings for. You can add or remove tags, or assign or remove permissions to the application security group.
- Azure CLI: [az network asg update](#)
  - PowerShell: No PowerShell cmdlet.

## Delete an application security group

You cannot delete an application security group if it has any network interfaces in it. Remove all network interfaces from the application security group by either changing network interface settings, or deleting the network interfaces. For details, see [Add to or remove a network interface from application security groups](#) or [delete a network interface](#).

1. Select **All services** on the upper, left corner of the Azure portal.
2. Enter *application security groups* in the **All services Filter** box, and then select **Application security groups** when it appears in the search results.
3. Select the application security group that you want to delete.
4. Select **Delete**, and then select **Yes** to delete the application security group.

## Commands

- Azure CLI: [az network asg delete](#)
- PowerShell: [Remove-AzApplicationSecurityGroup](#)

# Permissions

To perform tasks on network security groups, security rules, and application security groups, your account must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate permissions listed in the following tables:

## Network security group

ACTION	NAME
Microsoft.Network/networkSecurityGroups/read	Get network security group
Microsoft.Network/networkSecurityGroups/write	Create or update network security group
Microsoft.Network/networkSecurityGroups/delete	Delete network security group
Microsoft.Network/networkSecurityGroups/join/action	Associate a network security group to a subnet or network interface

## Network security group rule

ACTION	NAME
Microsoft.Network/networkSecurityGroups/rules/read	Get rule

ACTION	NAME
Microsoft.Network/networkSecurityGroups/rules/write	Create or update rule
Microsoft.Network/networkSecurityGroups/rules/delete	Delete rule

## Application security group

ACTION	NAME
Microsoft.Network/applicationSecurityGroups/joinIpConfiguration/action	Join an IP configuration to an application security group
Microsoft.Network/applicationSecurityGroups/joinNetworkSecurityRule/action	Join a security rule to an application security group
Microsoft.Network/applicationSecurityGroups/read	Get an application security group
Microsoft.Network/applicationSecurityGroups/write	Create or update an application security group
Microsoft.Network/applicationSecurityGroups/delete	Delete an application security group

## Next steps

- Create a network or application security group using [PowerShell](#) or [Azure CLI](#) sample scripts, or using [Azure Resource Manager templates](#)
- Create and apply [Azure policy](#) for virtual networks

# Partnering with Azure DDoS Protection Standard

1/28/2020 • 4 minutes to read • [Edit Online](#)

This article describes partnering opportunities enabled by the Azure DDoS Protection Standard. This article is designed to help product managers and business development roles understand the investment paths and provide insight into the partnering value propositions.

## Background

Distributed denial of service (DDoS) attacks are one of the top availability and security concerns voiced by customers moving their applications to the cloud. With extortion and hacktivism being the common motivations behind DDoS attacks, they have been consistently increasing in type, scale, and frequency of occurrence as they are relatively easy and cheap to launch.

Azure DDoS Protection provides countermeasures against the most sophisticated DDoS threats, leveraging the global scale of Azure networking. The service provides enhanced DDoS mitigation capabilities for applications and resources deployed in virtual networks.

Technology partners can protect their customers' resources natively with Azure DDoS Protection Standard to address the availability and reliability concerns due to DDoS attacks.

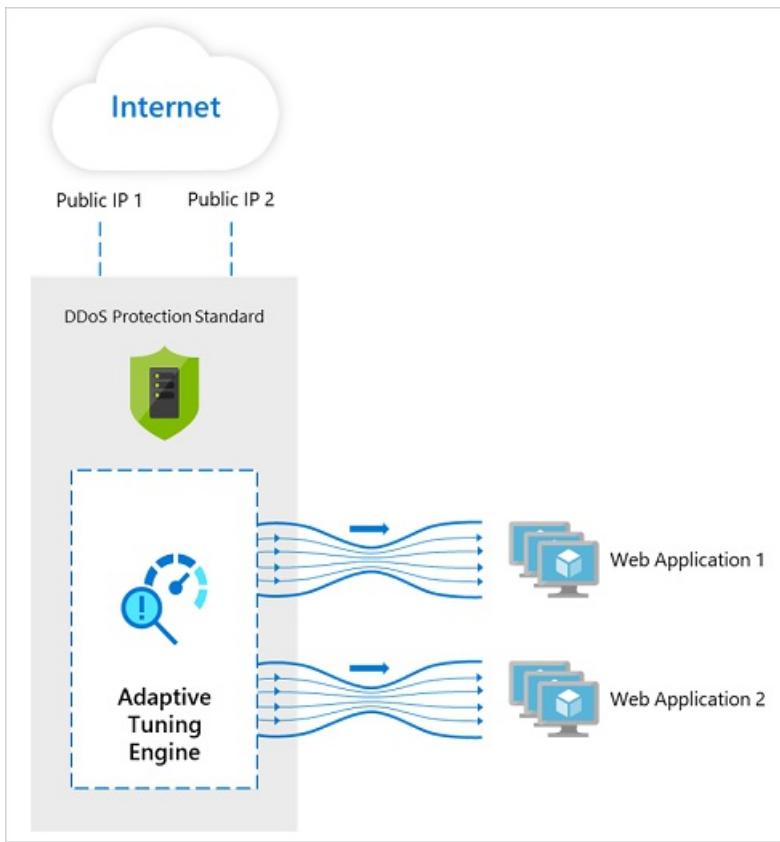
## Introduction to Azure DDoS Protection Standard

Azure DDoS Protection Standard provides enhanced DDoS mitigation capabilities against Layer 3 and Layer 4 DDoS attacks. The following are the key features of DDoS Protection Standard service.

### Adaptive real-time tuning

For every protected application, Azure DDoS Protection Standard automatically tunes the DDoS mitigation policy thresholds based on the application's traffic profile patterns. The service accomplishes this customization by using two insights:

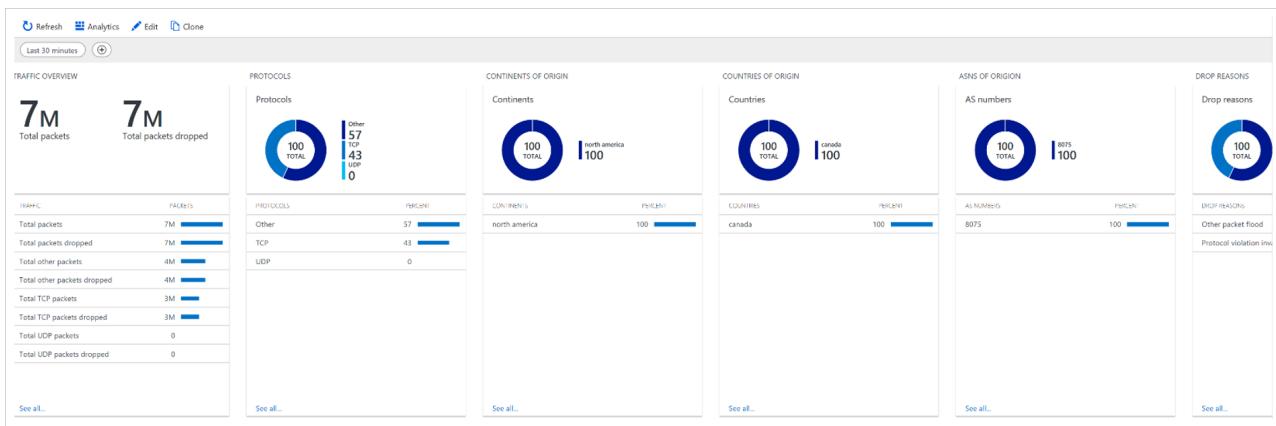
- Automatic learning of per-customer (per-IP) traffic patterns for Layer 3 and 4.
- Minimizing false positives, considering that the scale of Azure allows it to absorb a significant amount of traffic.



## Attack analytics, telemetry, monitoring, and alerting

Azure DDoS Protection identifies and mitigates DDoS attacks without any user intervention.

- If the protected resource is in the subscription covered under Azure Security Center, DDoS Protection Standard automatically sends an alert to Security Center whenever a DDoS attack is detected and mitigated against the protected application.
- Alternatively, to get notified when there's an active mitigation for a protected public IP, you can [configure an alert](#) on the metric Under DDoS attack or not.
- You can additionally choose to create alerts for the other DDoS metrics and [configure attack analytics](#) to understand the scale of the attack, traffic being dropped, attack vectors, top contributors, and other details.



## DDoS rapid response (DRR)

DDoS Protection Standard customers have access to [Rapid Response team](#) during an active attack. DRR can help with attack investigation, custom mitigations during an attack, and post-attack analysis.

## SLA guarantee and cost protection

DDoS Protection Standard service is covered by a 99.99% SLA, and cost protection provides resource credits for scale out during a documented attack. For more information, see [SLA for Azure DDoS Protection](#).

# Featured partner scenarios

The following are key benefits you can derive by integrating with the Azure DDoS Protection Standard:

- Partners' offered services (load balancer, web application firewall, firewall, etc.) to their customers are automatically protected (white labeled) by Azure DDoS Protection Standard in the back end.
- Partners have access to Azure DDoS Protection Standard attack analytics and telemetry that they can integrate with their own products, offering a unified customer experience.
- Partners have access to DDoS rapid response support even in the absence of Azure rapid response, for DDoS related issues.
- Partners' protected applications are backed by a DDoS SLA guarantee and cost protection in the event of DDoS attacks.

## Technical integration overview

Azure DDoS Protection Standard partnering opportunities are made available via Azure portal, APIs, and CLI/PS.

### Integrate with DDoS Protection Standard

The following steps are required for partners to configure integration with Azure DDoS Protection Standard:

- Create a DDoS Protection Plan in your desired (partner) subscription. For step-by-step instructions, see [Create a DDoS Standard Protection plan](#).

#### NOTE

Only 1 DDoS Protection Plan needs to be created for a given tenant.

- Deploy a service with public endpoint in your (partner) subscriptions, such as load balancer, firewalls, and web application firewall.
- Enable Azure DDoS Protection Standard on the virtual network of the service that has public endpoints using DDoS Protection Plan created in the first step. For step-by-step instructions, see [Enable DDoS Standard Protection plan](#)

#### IMPORTANT

After Azure DDoS Protection Standard is enabled on a virtual network, all public IPs within that virtual network are automatically protected. The origin of these public IPs can be either within Azure (client subscription) or outside of Azure.

- Optionally, integrate Azure DDoS Protection Standard telemetry and attack analytics in your application-specific customer-facing dashboard. For more information about using telemetry, see [Use DDoS Protection telemetry](#). For more information about configuring attack analytics, see [Configure DDoS attack analytics](#)

### Onboarding guides and technical documentation

- [Azure DDoS Protection product page](#)
- [Azure DDoS Protection documentation](#)
- [Azure DDoS Protection API reference](#)
- [Azure virtual network API reference](#)

### Get help

- If you have questions about application, service, or product integrations with Azure DDoS Protection Standard, reach out to the [Azure security community](#).
- Follow discussions on [Stack Overflow](#).

## **Get to market**

- The primary program for partnering with Microsoft is the [Microsoft Partner Network](#). – Microsoft Graph Security integrations fall into the [MPN Independent Software Vendor \(ISV\)](#) track.
- [Microsoft Intelligent Security Association](#) is the program specifically for Microsoft Security Partners to help enrich your security products and improve customer discoverability of your integrations to Microsoft Security products.

## **Next steps**

View existing partner integrations:

- [Barracuda WAF-as-a-service](#)
- [Azure Cloud WAF from Radware](#)

# Filter network traffic with a network security group using PowerShell

6/5/2019 • 8 minutes to read • [Edit Online](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

You can filter network traffic inbound to and outbound from a virtual network subnet with a network security group. Network security groups contain security rules that filter network traffic by IP address, port, and protocol. Security rules are applied to resources deployed in a subnet. In this article, you learn how to:

- Create a network security group and security rules
- Create a virtual network and associate a network security group to a subnet
- Deploy virtual machines (VM) into a subnet
- Test traffic filters

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.

4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

## Create a network security group

A network security group contains security rules. Security rules specify a source and destination. Sources and destinations can be application security groups.

### Create application security groups

First create a resource group for all the resources created in this article with [New-AzResourceGroup](#). The following example creates a resource group in the *eastus* location:

```
New-AzResourceGroup -ResourceGroupName myResourceGroup -Location EastUS
```

Create an application security group with [New-AzApplicationSecurityGroup](#). An application security group enables you to group servers with similar port filtering requirements. The following example creates two application security groups.

```
$webAsg = New-AzApplicationSecurityGroup `  
    -ResourceGroupName myResourceGroup `  
    -Name myAsgWebServers `  
    -Location eastus  
  
$mgmtAsg = New-AzApplicationSecurityGroup `  
    -ResourceGroupName myResourceGroup `  
    -Name myAsgMgmtServers `  
    -Location eastus
```

### Create security rules

Create a security rule with [New-AzNetworkSecurityRuleConfig](#). The following example creates a rule that allows traffic inbound from the internet to the *myWebServers* application security group over ports 80 and 443:

```
$webRule = New-AzNetworkSecurityRuleConfig ` 
    -Name "Allow-Web-All" ` 
    -Access Allow ` 
    -Protocol Tcp ` 
    -Direction Inbound ` 
    -Priority 100 ` 
    -SourceAddressPrefix Internet ` 
    -SourcePortRange * ` 
    -DestinationApplicationSecurityGroupId $webAsg.id ` 
    -DestinationPortRange 80,443
```

The following example creates a rule that allows traffic inbound from the internet to the `*myMgmtServers*` application security group over port 3389:

```
$mgmtRule = New-AzNetworkSecurityRuleConfig ` 
    -Name "Allow-RDP-All" ` 
    -Access Allow ` 
    -Protocol Tcp ` 
    -Direction Inbound ` 
    -Priority 110 ` 
    -SourceAddressPrefix Internet ` 
    -SourcePortRange * ` 
    -DestinationApplicationSecurityGroupId $mgmtAsg.id ` 
    -DestinationPortRange 3389
```

In this article, RDP (port 3389) is exposed to the internet for the `myAsgMgmtServers` VM. For production environments, instead of exposing port 3389 to the internet, it's recommended that you connect to Azure resources that you want to manage using a [VPN](#) or [private](#) network connection.

## Create a network security group

Create a network security group with [New-AzNetworkSecurityGroup](#). The following example creates a network security group named `myNsg`:

```
$nsg = New-AzNetworkSecurityGroup ` 
    -ResourceGroupName myResourceGroup ` 
    -Location eastus ` 
    -Name myNsg ` 
    -SecurityRules $webRule,$mgmtRule
```

## Create a virtual network

Create a virtual network with [New-AzVirtualNetwork](#). The following example creates a virtual named `myVirtualNetwork`:

```
$virtualNetwork = New-AzVirtualNetwork ` 
    -ResourceGroupName myResourceGroup ` 
    -Location EastUS ` 
    -Name myVirtualNetwork ` 
    -AddressPrefix 10.0.0.0/16
```

Create a subnet configuration with [New-AzVirtualNetworkSubnetConfig](#), and then write the subnet configuration to the virtual network with [Set-AzVirtualNetwork](#). The following example adds a subnet named `mySubnet` to the virtual network and associates the `myNsg` network security group to it:

```
Add-AzVirtualNetworkSubnetConfig ` 
-Name mySubnet ` 
-VirtualNetwork $virtualNetwork ` 
-AddressPrefix "10.0.2.0/24" ` 
-NetworkSecurityGroup $nsg
$virtualNetwork | Set-AzVirtualNetwork
```

## Create virtual machines

Before creating the VMs, retrieve the virtual network object with the subnet with [Get-AzVirtualNetwork](#):

```
$virtualNetwork = Get-AzVirtualNetwork ` 
-Name myVirtualNetwork ` 
-ResourceGroupName myResourceGroup
```

Create a public IP address for each VM with [New-AzPublicIpAddress](#):

```
$publicIpWeb = New-AzPublicIpAddress ` 
-AllocationMethod Dynamic ` 
-ResourceGroupName myResourceGroup ` 
-Location eastus ` 
-Name myVmWeb

$publicIpMgmt = New-AzPublicIpAddress ` 
-AllocationMethod Dynamic ` 
-ResourceGroupName myResourceGroup ` 
-Location eastus ` 
-Name myVmMgmt
```

Create two network interfaces with [New-AzNetworkInterface](#), and assign a public IP address to the network interface. The following example creates a network interface, associates the *myVmWeb* public IP address to it, and makes it a member of the *myAsgWebServers* application security group:

```
$webNic = New-AzNetworkInterface ` 
-Location eastus ` 
-Name myVmWeb ` 
-ResourceGroupName myResourceGroup ` 
-SubnetId $virtualNetwork.Subnets[0].Id ` 
-ApplicationSecurityGroupId $webAsg.Id ` 
-PublicIpAddressId $publicIpWeb.Id
```

The following example creates a network interface, associates the *myVmMgmt* public IP address to it, and makes it a member of the *myAsgMgmtServers* application security group:

```
$mgmtNic = New-AzNetworkInterface ` 
-Location eastus ` 
-Name myVmMgmt ` 
-ResourceGroupName myResourceGroup ` 
-SubnetId $virtualNetwork.Subnets[0].Id ` 
-ApplicationSecurityGroupId $mgmtAsg.Id ` 
-PublicIpAddressId $publicIpMgmt.Id
```

Create two VMs in the virtual network so you can validate traffic filtering in a later step.

Create a VM configuration with [New-AzVMConfig](#), then create the VM with [New-AzVM](#). The following example creates a VM that will serve as a web server. The `-AsJob` option creates the VM in the background, so you can continue to the next step:

```

# Create user object
$cred = Get-Credential -Message "Enter a username and password for the virtual machine."

$webVmConfig = New-AzVMConfig ` 
    -VMName myVmWeb ` 
    -VMSize Standard_DS1_V2 | ` 
Set-AzVMOperatingSystem -Windows ` 
    -ComputerName myVmWeb ` 
    -Credential $cred | ` 
Set-AzVMSourceImage ` 
    -PublisherName MicrosoftWindowsServer ` 
    -Offer WindowsServer ` 
    -Skus 2016-Datacenter ` 
    -Version latest | ` 
Add-AzVMNetworkInterface ` 
    -Id $webNic.Id 
New-AzVM ` 
    -ResourceGroupName myResourceGroup ` 
    -Location eastus ` 
    -VM $webVmConfig ` 
    -AsJob

```

Create a VM to serve as a management server:

```

# Create user object
$cred = Get-Credential -Message "Enter a username and password for the virtual machine."

# Create the web server virtual machine configuration and virtual machine.
$mgmtVmConfig = New-AzVMConfig ` 
    -VMName myVmMgmt ` 
    -VMSize Standard_DS1_V2 | ` 
Set-AzVMOperatingSystem -Windows ` 
    -ComputerName myVmMgmt ` 
    -Credential $cred | ` 
Set-AzVMSourceImage ` 
    -PublisherName MicrosoftWindowsServer ` 
    -Offer WindowsServer ` 
    -Skus 2016-Datacenter ` 
    -Version latest | ` 
Add-AzVMNetworkInterface ` 
    -Id $mgmtNic.Id 
New-AzVM ` 
    -ResourceGroupName myResourceGroup ` 
    -Location eastus ` 
    -VM $mgmtVmConfig

```

The virtual machine takes a few minutes to create. Don't continue with the next step until Azure finishes creating the VM.

## Test traffic filters

Use [Get-AzPublicIpAddress](#) to return the public IP address of a VM. The following example returns the public IP address of the *myVmMgmt* VM:

```

Get-AzPublicIpAddress ` 
    -Name myVmMgmt ` 
    -ResourceGroupName myResourceGroup ` 
    | Select IpAddress

```

Use the following command to create a remote desktop session with the *myVmMgmt* VM from your local

computer. Replace <publicIpAddress> with the IP address returned from the previous command.

```
mstsc /v:<publicIpAddress>
```

Open the downloaded RDP file. If prompted, select **Connect**.

Enter the user name and password you specified when creating the VM (you may need to select **More choices**, then **Use a different account**, to specify the credentials you entered when you created the VM), then select **OK**. You may receive a certificate warning during the sign-in process. Select **Yes** to proceed with the connection.

The connection succeeds, because port 3389 is allowed inbound from the internet to the *myAsgMgmtServers* application security group that the network interface attached to the *myVmMgmt* VM is in.

Use the following command to create a remote desktop connection to the *myVmWeb* VM, from the *myVmMgmt* VM, with the following command, from PowerShell:

```
mstsc /v:myvmWeb
```

The connection succeeds because a default security rule within each network security group allows traffic over all ports between all IP addresses within a virtual network. You can't create a remote desktop connection to the *myVmWeb* VM from the internet because the security rule for the *myAsgWebServers* doesn't allow port 3389 inbound from the internet.

Use the following command to install Microsoft IIS on the *myVmWeb* VM from PowerShell:

```
Install-WindowsFeature -name Web-Server -IncludeManagementTools
```

After the IIS installation is complete, disconnect from the *myVmWeb* VM, which leaves you in the *myVmMgmt* VM remote desktop connection. To view the IIS welcome screen, open an internet browser and browse to <http://myVmWeb>.

Disconnect from the *myVmMgmt* VM.

On your computer, enter the following command from PowerShell to retrieve the public IP address of the *myVmWeb* server:

```
Get-AzPublicIpAddress ` 
-Name myVmWeb ` 
-ResourceGroupName myResourceGroup ` 
| Select IPAddress
```

To confirm that you can access the *myVmWeb* web server from outside of Azure, open an internet browser on your computer and browse to <http://<public-ip-address-from-previous-step>>. The connection succeeds, because port 80 is allowed inbound from the internet to the *myAsgWebServers* application security group that the network interface attached to the *myVmWeb* VM is in.

## Clean up resources

When no longer needed, you can use [Remove-AzResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

## Next steps

In this article, you created a network security group and associated it to a virtual network subnet. To learn more about network security groups, see [Network security group overview](#) and [Manage a network security group](#).

Azure routes traffic between subnets by default. You may instead, choose to route traffic between subnets through a VM, serving as a firewall, for example. To learn how, see [Create a route table](#).

# Filter network traffic with a network security group using the Azure CLI

4/25/2019 • 7 minutes to read • [Edit Online](#)

You can filter network traffic inbound to and outbound from a virtual network subnet with a network security group. Network security groups contain security rules that filter network traffic by IP address, port, and protocol. Security rules are applied to resources deployed in a subnet. In this article, you learn how to:

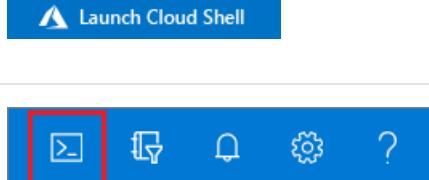
- Create a network security group and security rules
- Create a virtual network and associate a network security group to a subnet
- Deploy virtual machines (VM) into a subnet
- Test traffic filters

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this article requires that you are running the Azure CLI version 2.0.28 or later. To find the version, run `az --version`. If you need to install or upgrade, see [Install Azure CLI](#).

## Create a network security group

A network security group contains security rules. Security rules specify a source and destination. Sources and destinations can be application security groups.

## Create application security groups

First create a resource group for all the resources created in this article with [az group create](#). The following example creates a resource group in the *eastus* location:

```
az group create \
--name myResourceGroup \
--location eastus
```

Create an application security group with [az network asg create](#). An application security group enables you to group servers with similar port filtering requirements. The following example creates two application security groups.

```
az network asg create \
--resource-group myResourceGroup \
--name myAsgWebServers \
--location eastus

az network asg create \
--resource-group myResourceGroup \
--name myAsgMgmtServers \
--location eastus
```

## Create a network security group

Create a network security group with [az network nsg create](#). The following example creates a network security group named *myNsg*:

```
# Create a network security group
az network nsg create \
--resource-group myResourceGroup \
--name myNsg
```

## Create security rules

Create a security rule with [az network nsg rule create](#). The following example creates a rule that allows traffic inbound from the internet to the *myWebServers* application security group over ports 80 and 443:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNsg \
--name Allow-Web-All \
--access Allow \
--protocol Tcp \
--direction Inbound \
--priority 100 \
--source-address-prefix Internet \
--source-port-range "*" \
--destination-asgs "myAsgWebServers" \
--destination-port-range 80 443
```

The following example creates a rule that allows traffic inbound from the Internet to the *myMgmtServers* application security group over port 22:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNsg \
--name Allow-SSH-All \
--access Allow \
--protocol Tcp \
--direction Inbound \
--priority 110 \
--source-address-prefix Internet \
--source-port-range "*" \
--destination-asgs "myAsgMgmtServers" \
--destination-port-range 22
```

In this article, SSH (port 22) is exposed to the internet for the *myAsgMgmtServers* VM. For production environments, instead of exposing port 22 to the internet, it's recommended that you connect to Azure resources that you want to manage using a [VPN](#) or [private](#) network connection.

## Create a virtual network

Create a virtual network with [az network vnet create](#). The following example creates a virtual named *myVirtualNetwork*:

```
az network vnet create \
--name myVirtualNetwork \
--resource-group myResourceGroup \
--address-prefixes 10.0.0.0/16
```

Add a subnet to a virtual network with [az network vnet subnet create](#). The following example adds a subnet named *mySubnet* to the virtual network and associates the *myNsg* network security group to it:

```
az network vnet subnet create \
--vnet-name myVirtualNetwork \
--resource-group myResourceGroup \
--name mySubnet \
--address-prefix 10.0.0.0/24 \
--network-security-group myNsg
```

## Create virtual machines

Create two VMs in the virtual network so you can validate traffic filtering in a later step.

Create a VM with [az vm create](#). The following example creates a VM that will serve as a web server. The `--asgs myAsgWebServers` option causes Azure to make the network interface it creates for the VM a member of the *myAsgWebServers* application security group.

The `--nsg ""` option is specified to prevent Azure from creating a default network security group for the network interface Azure creates when it creates the VM. To streamline this article, a password is used. Keys are typically used in production deployments. If you use keys, you must also configure SSH agent forwarding for the remaining steps. For more information, see the documentation for your SSH client. Replace `<replace-with-your-password>` in the following command with a password of your choosing.

```
adminPassword=<replace-with-your-password>

az vm create \
--resource-group myResourceGroup \
--name myVmWeb \
--image UbuntuLTS \
--vnet-name myVirtualNetwork \
--subnet mySubnet \
--nsg "" \
--asgs myAsgWebServers \
--admin-username azureuser \
--admin-password $adminPassword
```

The VM takes a few minutes to create. After the VM is created, output similar to the following example is returned:

```
{
  "fqdns": "",
  "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVmWeb",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "13.90.242.231",
  "resourceGroup": "myResourceGroup"
}
```

Take note of the **publicIpAddress**. This address is used to access the VM from the internet in a later step. Create a VM to serve as a management server:

```
az vm create \
--resource-group myResourceGroup \
--name myVmMgmt \
--image UbuntuLTS \
--vnet-name myVirtualNetwork \
--subnet mySubnet \
--nsg "" \
--asgs myAsgMgmtServers \
--admin-username azureuser \
--admin-password $adminPassword
```

The VM takes a few minutes to create. After the VM is created, note the **publicIpAddress** in the returned output. This address is used to access the VM in the next step. Don't continue with the next step until Azure finishes creating the VM.

## Test traffic filters

Use the command that follows to create an SSH session with the *myVmMgmt* VM. Replace <publicIpAddress> with the public IP address of your VM. In the example above, the IP address is 13.90.242.231.

```
ssh azureuser@<publicIpAddress>
```

When prompted for a password, enter the password you entered in [Create VMs](#).

The connection succeeds, because port 22 is allowed inbound from the Internet to the *myAsgMgmtServers* application security group that the network interface attached to the *myVmMgmt* VM is in.

Use the following command to SSH to the *myVmWeb* VM from the *myVmMgmt* VM:

```
ssh azureuser@myVmWeb
```

The connection succeeds because a default security rule within each network security group allows traffic over all ports between all IP addresses within a virtual network. You can't SSH to the *myVmWeb* VM from the Internet because the security rule for the *myAsgWebServers* doesn't allow port 22 inbound from the Internet.

Use the following commands to install the nginx web server on the *myVmWeb* VM:

```
# Update package source  
sudo apt-get -y update  
  
# Install NGINX  
sudo apt-get -y install nginx
```

The *myVmWeb* VM is allowed outbound to the Internet to retrieve nginx because a default security rule allows all outbound traffic to the Internet. Exit the *myVmWeb* SSH session, which leaves you at the `username@myVmMgmt:~$` prompt of the *myVmMgmt* VM. To retrieve the nginx welcome screen from the *myVmWeb* VM, enter the following command:

```
curl myVmWeb
```

Logout of the *myVmMgmt* VM. To confirm that you can access the *myVmWeb* web server from outside of Azure, enter `curl <publicIpAddress>` from your own computer. The connection succeeds, because port 80 is allowed inbound from the Internet to the *myAsgWebServers* application security group that the network interface attached to the *myVmWeb* VM is in.

## Clean up resources

When no longer needed, use [az group delete](#) to remove the resource group and all of the resources it contains.

```
az group delete --name myResourceGroup --yes
```

## Next steps

In this article, you created a network security group and associated it to a virtual network subnet. To learn more about network security groups, see [Network security group overview](#) and [Manage a network security group](#).

Azure routes traffic between subnets by default. You may instead, choose to route traffic between subnets through a VM, serving as a firewall, for example. To learn how, see [Create a route table](#).

# Route network traffic with a route table using PowerShell

10/30/2019 • 9 minutes to read • [Edit Online](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Azure automatically routes traffic between all subnets within a virtual network, by default. You can create your own routes to override Azure's default routing. The ability to create custom routes is helpful if, for example, you want to route traffic between subnets through a network virtual appliance (NVA). In this article, you learn how to:

- Create a route table
- Create a route
- Create a virtual network with multiple subnets
- Associate a route table to a subnet
- Create an NVA that routes traffic
- Deploy virtual machines (VM) into different subnets
- Route traffic from one subnet to another through an NVA

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.

2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

## Create a route table

Before you can create a route table, create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myResourceGroup* for all resources created in this article.

```
New-AzResourceGroup -ResourceGroupName myResourceGroup -Location EastUS
```

Create a route table with [New-AzRouteTable](#). The following example creates a route table named *myRouteTablePublic*.

```
$routeTablePublic = New-AzRouteTable ` 
-Name 'myRouteTablePublic' ` 
-ResourceGroupName myResourceGroup ` 
-location EastUS
```

## Create a route

Create a route by retrieving the route table object with [Get-AzRouteTable](#), create a route with [Add-AzRouteConfig](#), then write the route configuration to the route table with [Set-AzRouteTable](#).

```
Get-AzRouteTable ` 
-ResourceGroupName "myResourceGroup" ` 
-Name "myRouteTablePublic" ` 
| Add-AzRouteConfig ` 
-Name "ToPrivateSubnet" ` 
-AddressPrefix 10.0.1.0/24 ` 
-NextHopType "VirtualAppliance" ` 
-NextHopIpAddress 10.0.2.4 ` 
| Set-AzRouteTable
```

## Associate a route table to a subnet

Before you can associate a route table to a subnet, you have to create a virtual network and subnet. Create a virtual network with [New-AzVirtualNetwork](#). The following example creates a virtual network named *myVirtualNetwork* with the address prefix *10.0.0.0/16*.

```
$virtualNetwork = New-AzVirtualNetwork ` 
-ResourceGroupName myResourceGroup ` 
-Location EastUS ` 
-Name myVirtualNetwork ` 
-AddressPrefix 10.0.0.0/16
```

Create three subnets by creating three subnet configurations with [New-AzVirtualNetworkSubnetConfig](#). The

following example creates three subnet configurations for *Public*, *Private*, and *DMZ* subnets:

```
$subnetConfigPublic = Add-AzVirtualNetworkSubnetConfig `  
    -Name Public `  
    -AddressPrefix 10.0.0.0/24 `  
    -VirtualNetwork $virtualNetwork  
  
$subnetConfigPrivate = Add-AzVirtualNetworkSubnetConfig `  
    -Name Private `  
    -AddressPrefix 10.0.1.0/24 `  
    -VirtualNetwork $virtualNetwork  
  
$subnetConfigDmz = Add-AzVirtualNetworkSubnetConfig `  
    -Name DMZ `  
    -AddressPrefix 10.0.2.0/24 `  
    -VirtualNetwork $virtualNetwork
```

Write the subnet configurations to the virtual network with [Set-AzVirtualNetwork](#), which creates the subnets in the virtual network:

```
$virtualNetwork | Set-AzVirtualNetwork
```

Associate the *myRouteTablePublic* route table to the *Public* subnet with [Set-AzVirtualNetworkSubnetConfig](#) and then write the subnet configuration to the virtual network with [Set-AzVirtualNetwork](#).

```
Set-AzVirtualNetworkSubnetConfig `  
    -VirtualNetwork $virtualNetwork `  
    -Name 'Public' `  
    -AddressPrefix 10.0.0.0/24 `  
    -RouteTable $routeTablePublic | `  
Set-AzVirtualNetwork
```

## Create an NVA

An NVA is a VM that performs a network function, such as routing, firewalling, or WAN optimization.

Before creating a VM, create a network interface.

### Create a network interface

Before creating a network interface, you have to retrieve the virtual network Id with [Get-AzVirtualNetwork](#), then the subnet Id with [Get-AzVirtualNetworkSubnetConfig](#). Create a network interface with [New-AzNetworkInterface](#) in the *DMZ* subnet with IP forwarding enabled:

```

# Retrieve the virtual network object into a variable.
$virtualNetwork=Get-AzVirtualNetwork ` 
    -Name myVirtualNetwork ` 
    -ResourceGroupName myResourceGroup

# Retrieve the subnet configuration into a variable.
$subnetConfigDmz = Get-AzVirtualNetworkSubnetConfig ` 
    -Name DMZ ` 
    -VirtualNetwork $virtualNetwork

# Create the network interface.
$nic = New-AzNetworkInterface ` 
    -ResourceGroupName myResourceGroup ` 
    -Location EastUS ` 
    -Name 'myVmNva' ` 
    -SubnetId $subnetConfigDmz.Id ` 
    -EnableIPForwarding

```

## Create a VM

To create a VM and attach an existing network interface to it, you must first create a VM configuration with [New-AzVMConfig](#). The configuration includes the network interface created in the previous step. When prompted for a username and password, select the user name and password you want to log into the VM with.

```

# Create a credential object.
$cred = Get-Credential -Message "Enter a username and password for the VM."

# Create a VM configuration.
$vmConfig = New-AzVMConfig ` 
    -VMName 'myVmNva' ` 
    -VMSize Standard_DS2 | ` 
    Set-AzVMOperatingSystem -Windows ` 
        -ComputerName 'myVmNva' ` 
        -Credential $cred | ` 
    Set-AzVMSourceImage ` 
        -PublisherName MicrosoftWindowsServer ` 
        -Offer WindowsServer ` 
        -Skus 2016-Datacenter ` 
        -Version latest | ` 
    Add-AzVMNetworkInterface -Id $nic.Id

```

Create the VM using the VM configuration with [New-AzVM](#). The following example creates a VM named *myVmNva*.

```

$vmNva = New-AzVM ` 
    -ResourceGroupName myResourceGroup ` 
    -Location EastUS ` 
    -VM $vmConfig ` 
    -AsJob

```

The `-AsJob` option creates the VM in the background, so you can continue to the next step.

## Create virtual machines

Create two VMs in the virtual network so you can validate that traffic from the *Public* subnet is routed to the *Private* subnet through the network virtual appliance in a later step.

Create a VM in the *Public* subnet with [New-AzVM](#). The following example creates a VM named *myVmPublic* in the *Public* subnet of the *myVirtualNetwork* virtual network.

```
New-AzVm ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "East US" ` 
-VirtualNetworkName "myVirtualNetwork" ` 
-SubnetName "Public" ` 
-ImageName "Win2016Datacenter" ` 
-Name "myVmPublic" ` 
-AsJob
```

Create a VM in the *Private* subnet.

```
New-AzVm ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "East US" ` 
-VirtualNetworkName "myVirtualNetwork" ` 
-SubnetName "Private" ` 
-ImageName "Win2016Datacenter" ` 
-Name "myVmPrivate"
```

The VM takes a few minutes to create. Don't continue with the next step until the VM is created and Azure returns output to PowerShell.

## Route traffic through an NVA

Use [Get-AzPublicIpAddress](#) to return the public IP address of the *myVmPrivate* VM. The following example returns the public IP address of the *myVmPrivate* VM:

```
Get-AzPublicIpAddress ` 
-Name myVmPrivate ` 
-ResourceGroupName myResourceGroup ` 
| Select IpAddress
```

Use the following command to create a remote desktop session with the *myVmPrivate* VM from your local computer. Replace `<publicIpAddress>` with the IP address returned from the previous command.

```
mstsc /v:<publicIpAddress>
```

Open the downloaded RDP file. If prompted, select **Connect**.

Enter the user name and password you specified when creating the VM (you may need to select **More choices**, then **Use a different account**, to specify the credentials you entered when you created the VM), then select **OK**. You may receive a certificate warning during the sign-in process. Select **Yes** to proceed with the connection.

In a later step, the `tracert.exe` command is used to test routing. Tracert uses the Internet Control Message Protocol (ICMP), which is denied through the Windows Firewall. Enable ICMP through the Windows firewall by entering the following command from PowerShell on the *myVmPrivate* VM:

```
New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4
```

Though trace route is used to test routing in this article, allowing ICMP through the Windows Firewall for production deployments is not recommended.

You enabled IP forwarding within Azure for the VM's network interface in Enable IP forwarding. Within the VM, the operating system, or an application running within the VM, must also be able to forward network traffic. Enable IP forwarding within the operating system of the *myVmNva*.

From a command prompt on the *myVmPrivate* VM, remote desktop to the *myVmNva*:

```
mstsc /v:myvmnva
```

To enable IP forwarding within the operating system, enter the following command in PowerShell from the *myVmNva* VM:

```
Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters -Name IpEnableRouter -Value 1
```

Restart the *myVmNva* VM, which also disconnects the remote desktop session.

While still connected to the *myVmPrivate* VM, create a remote desktop session to the *myVmPublic* VM, after the *myVmNva* VM restarts:

```
mstsc /v:myVmPublic
```

Enable ICMP through the Windows firewall by entering the following command from PowerShell on the *myVmPublic* VM:

```
New-NetFirewallRule -DisplayName "Allow ICMPv4-In" -Protocol ICMPv4
```

To test routing of network traffic to the *myVmPrivate* VM from the *myVmPublic* VM, enter the following command from PowerShell on the *myVmPublic* VM:

```
tracert myVmPrivate
```

The response is similar to the following example:

```
Tracing route to myVmPrivate.vpgub4nqnocezhjgurw44dnxrc.bx.internal.cloudapp.net [10.0.1.4]
over a maximum of 30 hops:
1    <1 ms      *      1 ms  10.0.2.4
2      1 ms      1 ms  1 ms  10.0.1.4

Trace complete.
```

You can see that the first hop is 10.0.2.4, which is the NVA's private IP address. The second hop is 10.0.1.4, the private IP address of the *myVmPrivate* VM. The route added to the *myRouteTablePublic* route table and associated to the *Public* subnet caused Azure to route the traffic through the NVA, rather than directly to the *Private* subnet.

Close the remote desktop session to the *myVmPublic* VM, which leaves you still connected to the *myVmPrivate* VM.

To test routing of network traffic to the *myVmPublic* VM from the *myVmPrivate* VM, enter the following command from a command prompt on the *myVmPrivate* VM:

```
tracert myVmPublic
```

The response is similar to the following example:

```
Tracing route to myVmPublic.vpgub4nqnocezhjgurw44dnxrc.bx.internal.cloudapp.net [10.0.0.4]
over a maximum of 30 hops:

1      1 ms      1 ms      1 ms  10.0.0.4

Trace complete.
```

You can see that traffic is routed directly from the *myVmPrivate* VM to the *myVmPublic* VM. By default, Azure routes traffic directly between subnets.

Close the remote desktop session to the *myVmPrivate* VM.

## Clean up resources

When no longer needed, use [Remove-AzResourcegroup](#) to remove the resource group and all of the resources it contains.

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

## Next steps

In this article, you created a route table and associated it to a subnet. You created a simple network virtual appliance that routed traffic from a public subnet to a private subnet. Deploy a variety of pre-configured network virtual appliances that perform network functions such as firewall and WAN optimization from the [Azure Marketplace](#). To learn more about routing, see [Routing overview](#) and [Manage a route table](#).

While you can deploy many Azure resources within a virtual network, resources for some Azure PaaS services cannot be deployed into a virtual network. You can still restrict access to the resources of some Azure PaaS services to traffic only from a virtual network subnet though. To learn how, see [Restrict network access to PaaS resources](#).

# Route network traffic with a route table using the Azure CLI

4/25/2019 • 7 minutes to read • [Edit Online](#)

Azure automatically routes traffic between all subnets within a virtual network, by default. You can create your own routes to override Azure's default routing. The ability to create custom routes is helpful if, for example, you want to route traffic between subnets through a network virtual appliance (NVA). In this article, you learn how to:

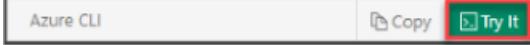
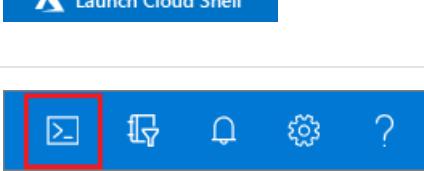
- Create a route table
- Create a route
- Create a virtual network with multiple subnets
- Associate a route table to a subnet
- Create an NVA that routes traffic
- Deploy virtual machines (VM) into different subnets
- Route traffic from one subnet to another through an NVA

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this quickstart requires that you are running the Azure CLI version 2.0.28 or later. To find the version, run `az --version`. If you need to install or upgrade, see [Install Azure CLI](#).

## Create a route table

Before you can create a route table, create a resource group with [az group create](#) for all resources created in this article.

```
# Create a resource group.  
az group create \  
  --name myResourceGroup \  
  --location eastus
```

Create a route table with [az network route-table create](#). The following example creates a route table named *myRouteTablePublic*.

```
# Create a route table  
az network route-table create \  
  --resource-group myResourceGroup \  
  --name myRouteTablePublic
```

## Create a route

Create a route in the route table with [az network route-table route create](#).

```
az network route-table route create \  
  --name ToPrivateSubnet \  
  --resource-group myResourceGroup \  
  --route-table-name myRouteTablePublic \  
  --address-prefix 10.0.1.0/24 \  
  --next-hop-type VirtualAppliance \  
  --next-hop-ip-address 10.0.2.4
```

## Associate a route table to a subnet

Before you can associate a route table to a subnet, you have to create a virtual network and subnet. Create a virtual network with one subnet with [az network vnet create](#).

```
az network vnet create \  
  --name myVirtualNetwork \  
  --resource-group myResourceGroup \  
  --address-prefix 10.0.0.0/16 \  
  --subnet-name Public \  
  --subnet-prefix 10.0.0.0/24
```

Create two additional subnets with [az network vnet subnet create](#).

```
# Create a private subnet.  
az network vnet subnet create \  
  --vnet-name myVirtualNetwork \  
  --resource-group myResourceGroup \  
  --name Private \  
  --address-prefix 10.0.1.0/24  
  
# Create a DMZ subnet.  
az network vnet subnet create \  
  --vnet-name myVirtualNetwork \  
  --resource-group myResourceGroup \  
  --name DMZ \  
  --address-prefix 10.0.2.0/24
```

Associate the *myRouteTablePublic* route table to the *Public* subnet with [az network vnet subnet update](#).

```
az network vnet subnet update \  
  --vnet-name myVirtualNetwork \  
  --name Public \  
  --resource-group myResourceGroup \  
  --route-table myRouteTablePublic
```

## Create an NVA

An NVA is a VM that performs a network function, such as routing, firewalling, or WAN optimization.

Create an NVA in the *DMZ* subnet with [az vm create](#). When you create a VM, Azure creates and assigns a public IP address to the VM, by default. The `--public-ip-address ""` parameter instructs Azure not to create and assign a public IP address to the VM, since the VM doesn't need to be connected to from the internet. If SSH keys do not already exist in a default key location, the command creates them. To use a specific set of keys, use the `--ssh-key-value` option.

```
az vm create \  
  --resource-group myResourceGroup \  
  --name myVmNva \  
  --image UbuntuLTS \  
  --public-ip-address "" \  
  --subnet DMZ \  
  --vnet-name myVirtualNetwork \  
  --generate-ssh-keys
```

The VM takes a few minutes to create. Do not continue to the next step until Azure finishes creating the VM and returns output about the VM.

For a network interface to be able to forward network traffic sent to it, that is not destined for its own IP address, IP forwarding must be enabled for the network interface. Enable IP forwarding for the network interface with [az network nic update](#).

```
az network nic update \  
  --name myVmNvaVMNic \  
  --resource-group myResourceGroup \  
  --ip-forwarding true
```

Within the VM, the operating system, or an application running within the VM, must also be able to forward network traffic. Enable IP forwarding within the VM's operating system with [az vm extension set](#):

```
az vm extension set \
--resource-group myResourceGroup \
--vm-name myVmNva \
--name customScript \
--publisher Microsoft.Azure.Extensions \
--settings '{"commandToExecute":"sudo sysctl -w net.ipv4.ip_forward=1"}'
```

The command may take up to a minute to execute.

## Create virtual machines

Create two VMs in the virtual network so you can validate that traffic from the *Public* subnet is routed to the *Private* subnet through the NVA in a later step.

Create a VM in the *Public* subnet with [az vm create](#). The `--no-wait` parameter enables Azure to execute the command in the background so you can continue to the next command. To streamline this article, a password is used. Keys are typically used in production deployments. If you use keys, you must also configure SSH agent forwarding. For more information, see the documentation for your SSH client. Replace `<replace-with-your-password>` in the following command with a password of your choosing.

```
adminPassword="<replace-with-your-password>"  
  
az vm create \  
--resource-group myResourceGroup \  
--name myVmPublic \  
--image UbuntuLTS \  
--vnet-name myVirtualNetwork \  
--subnet Public \  
--admin-username azureuser \  
--admin-password $adminPassword \  
--no-wait
```

Create a VM in the *Private* subnet.

```
az vm create \  
--resource-group myResourceGroup \  
--name myVmPrivate \  
--image UbuntuLTS \  
--vnet-name myVirtualNetwork \  
--subnet Private \  
--admin-username azureuser \  
--admin-password $adminPassword
```

The VM takes a few minutes to create. After the VM is created, the Azure CLI shows information similar to the following example:

```
{  
  "fqdns": "",  
  "id": "/subscriptions/00000000-0000-0000-0000-  
0000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVmPrivate",  
  "location": "eastus",  
  "macAddress": "00-0D-3A-23-9A-49",  
  "powerState": "VM running",  
  "privateIpAddress": "10.0.1.4",  
  "publicIpAddress": "13.90.242.231",  
  "resourceGroup": "myResourceGroup"  
}
```

Take note of the **publicIpAddress**. This address is used to access the VM from the internet in a later step.

## Route traffic through an NVA

Use the following command to create an SSH session with the *myVmPrivate* VM. Replace <publicIpAddress> with the public IP address of your VM. In the example above, the IP address is 13.90.242.231.

```
ssh azureuser@<publicIpAddress>
```

When prompted for a password, enter the password you selected in [Create virtual machines](#).

Use the following command to install trace route on the *myVmPrivate* VM:

```
sudo apt-get install traceroute
```

Use the following command to test routing for network traffic to the *myVmPublic* VM from the *myVmPrivate* VM.

```
traceroute myVmPublic
```

The response is similar to the following example:

```
traceroute to myVmPublic (10.0.0.4), 30 hops max, 60 byte packets
1 10.0.0.4 (10.0.0.4) 1.404 ms 1.403 ms 1.398 ms
```

You can see that traffic is routed directly from the *myVmPrivate* VM to the *myVmPublic* VM. Azure's default routes, route traffic directly between subnets.

Use the following command to SSH to the *myVmPublic* VM from the *myVmPrivate* VM:

```
ssh azureuser@myVmPublic
```

Use the following command to install trace route on the *myVmPublic* VM:

```
sudo apt-get install traceroute
```

Use the following command to test routing for network traffic to the *myVmPrivate* VM from the *myVmPublic* VM.

```
traceroute myVmPrivate
```

The response is similar to the following example:

```
traceroute to myVmPrivate (10.0.1.4), 30 hops max, 60 byte packets
1 10.0.2.4 (10.0.2.4) 0.781 ms 0.780 ms 0.775 ms
2 10.0.1.4 (10.0.0.4) 1.404 ms 1.403 ms 1.398 ms
```

You can see that the first hop is 10.0.2.4, which is the NVA's private IP address. The second hop is 10.0.1.4, the private IP address of the *myVmPrivate* VM. The route added to the *myRouteTablePublic* route table and associated to the *Public* subnet caused Azure to route the traffic through the NVA, rather than directly to the *Private* subnet.

Close the SSH sessions to both the *myVmPublic* and *myVmPrivate* VMs.

## Clean up resources

When no longer needed, use [az group delete](#) to remove the resource group and all of the resources it contains.

```
az group delete --name myResourceGroup --yes
```

## Next steps

In this article, you created a route table and associated it to a subnet. You created a simple NVA that routed traffic from a public subnet to a private subnet. Deploy a variety of pre-configured NVAs that perform network functions such as firewall and WAN optimization from the [Azure Marketplace](#). To learn more about routing, see [Routing overview](#) and [Manage a route table](#).

While you can deploy many Azure resources within a virtual network, resources for some Azure PaaS services cannot be deployed into a virtual network. You can still restrict access to the resources of some Azure PaaS services to traffic only from a virtual network subnet though. To learn how, see [Restrict network access to PaaS resources](#).

# Create, change, or delete a route table

1/14/2020 • 11 minutes to read • [Edit Online](#)

Azure automatically routes traffic between Azure subnets, virtual networks, and on-premises networks. If you want to change any of Azure's default routing, you do so by creating a route table. If you're new to routing in virtual networks, you can learn more about it in the [routing overview](#) or by completing a [tutorial](#).

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and sign in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools pre-installed and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.31 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

The account you sign into, or connect to Azure with, must be assigned to the [network contributor](#) role or to a [custom role](#) that is assigned the appropriate actions listed in [Permissions](#).

## Create a route table

There is a limit to how many route tables you can create per Azure location and subscription. For details, see [Azure limits](#).

1. In the top-left corner of the portal, select **+** **Create a resource**.
2. Select **Networking**, then select **Route table**.
3. Enter a **Name** for the route table, select your **Subscription**, create a new **Resource group**, or select an existing resource group, select a **Location**, then select **Create**. If you plan to associate the route table to a subnet in a virtual network that is connected to your on-premises network through a VPN gateway, and you disable **Virtual network gateway route propagation**, your on-premises routes are not propagated to the network interfaces in the subnet.

## Create route table - commands

- Azure CLI: [az network route-table create](#)
- PowerShell: [New-AzRouteTable](#)

## View route tables

In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it. The route tables that exist in your subscription are listed.

## View route table - commands

- Azure CLI: [az network route-table list](#)
- PowerShell: [Get-AzRouteTable](#)

## View details of a route table

1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it.
2. Select the route table in the list that you want to view details for. Under **SETTINGS**, you can view the **Routes** in the route table and the **Subnets** the route table is associated to.
3. To learn more about common Azure settings, see the following information:
  - [Activity log](#)
  - [Access control \(IAM\)](#)
  - [Tags](#)
  - [Locks](#)
  - [Automation script](#)

## View details of route table - commands

- Azure CLI: [az network route-table show](#)
- PowerShell: [Get-AzRouteTable](#)

## Change a route table

1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it.
2. Select the route table you want to change. The most common changes are [adding](#) or [removing](#) routes and [associating](#) route tables to, or [dissociating](#) route tables from subnets.

## Change a route table - commands

- Azure CLI: [az network route-table update](#)
- PowerShell: [Set-AzRouteTable](#)

## Associate a route table to a subnet

A subnet can have zero or one route table associated to it. A route table can be associated to zero or multiple subnets. Since route tables are not associated to virtual networks, you must associate a route table to each subnet you want the route table associated to. All traffic leaving the subnet is routed based on routes you've created within route tables, [default routes](#), and routes propagated from an on-premises network, if the virtual network is connected to an Azure virtual network gateway (ExpressRoute or VPN). You can only associate a route table to subnets in virtual networks that exist in the same Azure location and subscription as the route table.

1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks**

- appear in the search results, select it.
2. Select the virtual network in the list that contains the subnet you want to associate a route table to.
  3. Select **Subnets** under **SETTINGS**.
  4. Select the subnet you want to associate the route table to.
  5. Select **Route table**, select the route table you want to associate to the subnet, then select **Save**.
- If your virtual network is connected to an Azure VPN gateway, do not associate a route table to the [gateway subnet](#) that includes a route with a destination of 0.0.0.0/0. Doing so can prevent the gateway from functioning properly. For more information about using 0.0.0.0/0 in a route, see [Virtual network traffic routing](#).
- #### Associate a route table - commands
- Azure CLI: [az network vnet subnet update](#)
  - PowerShell: [Set-AzVirtualNetworkSubnetConfig](#)
- ## Dissociate a route table from a subnet
- When you dissociate a route table from a subnet, Azure routes traffic based on its [default routes](#).
1. In the search box at the top of the portal, enter *virtual networks* in the search box. When **Virtual networks** appear in the search results, select it.
  2. Select the virtual network that contains the subnet you want to dissociate a route table from.
  3. Select **Subnets** under **SETTINGS**.
  4. Select the subnet you want to dissociate the route table from.
  5. Select **Route table**, select **None**, then select **Save**.
- #### Dissociate a route table - commands
- Azure CLI: [az network vnet subnet update](#)
  - PowerShell: [Set-AzVirtualNetworkSubnetConfig](#)
- ## Delete a route table
- If a route table is associated to any subnets, it cannot be deleted. [Dissociate](#) a route table from all subnets before attempting to delete it.
1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it.
  2. Select ... on the right-side of the route table you want to delete.
  3. Select **Delete**, and then select **Yes**.
- #### Delete a route table - commands
- Azure CLI: [az network route-table delete](#)
  - PowerShell: [Remove-AzRouteTable](#)
- ## Create a route
- There is a limit to how many routes per route table can create per Azure location and subscription. For details, see [Azure limits](#).
1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it.
  2. Select the route table from the list that you want to add a route to.
  3. Select **Routes**, under **SETTINGS**.
  4. Select **+ Add**.

5. Enter a unique **Name** for the route within the route table.
6. Enter the **Address prefix**, in CIDR notation, that you want to route traffic to. The prefix cannot be duplicated in more than one route within the route table, though the prefix can be within another prefix. For example, if you defined 10.0.0.0/16 as a prefix in one route, you can still define another route with the 10.0.0.0/24 address prefix. Azure selects a route for traffic based on longest prefix match. To learn more about how Azure selects routes, see [Routing overview](#).
7. Select a **Next hop type**. For a detailed description of all next hop types, see [Routing overview](#).
8. Enter an IP address for **Next hop address**. You can only enter an address if you selected *Virtual appliance* for **Next hop type**.
9. Select **OK**.

#### Create a route - commands

- Azure CLI: [az network route-table route create](#)
- PowerShell: [New-AzRouteConfig](#)

## View routes

A route table contains zero or multiple routes. To learn more about the information listed when viewing routes, see [Routing overview](#).

1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it.
2. Select the route table from the list that you want to view routes for.
3. Select **Routes** under **SETTINGS**.

#### View routes - commands

- Azure CLI: [az network route-table route list](#)
- PowerShell: [Get-AzRouteConfig](#)

## View details of a route

1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it.
2. Select the route table you want to view details of a route for.
3. Select **Routes**.
4. Select the route you want to view details of.

#### View details of a route - commands

- Azure CLI: [az network route-table route show](#)
- PowerShell: [Get-AzRouteConfig](#)

## Change a route

1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it.
2. Select the route table you want to change a route for.
3. Select **Routes**.
4. Select the route you want to change.
5. Change existing settings to their new settings, then select **Save**.

#### Change a route - commands

- Azure CLI: [az network route-table route update](#)

- PowerShell: [Set-AzRouteConfig](#)

## Delete a route

1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appear in the search results, select it.
2. Select the route table you want to delete a route for.
3. Select **Routes**.
4. From the list of routes, select ... on the right-side of the route you want to delete.
5. Select **Delete**, then select **Yes**.

### Delete a route - commands

- Azure CLI: [az network route-table route delete](#)
- PowerShell: [Remove-AzRouteConfig](#)

## View effective routes

The effective routes for each network interface attached to a virtual machine are a combination of route tables that you've created, Azure's default routes, and any routes propagated from on-premises networks via BGP through an Azure virtual network gateway. Understanding the effective routes for a network interface is helpful when troubleshooting routing problems. You can view the effective routes for any network interface that is attached to a running virtual machine.

1. In the search box at the top of the portal, enter the name of a virtual machine you want to view effective routes for. If you don't know the name of a virtual machine, enter *virtual machines* in the search box. When **Virtual machines** appear in the search results, select it and select a virtual machine from the list.
2. Select **Networking** under **SETTINGS**.
3. Select the name of a network interface.
4. Select **Effective routes** under **SUPPORT + TROUBLESHOOTING**.
5. Review the list of effective routes to determine if the correct route exists for where you want to route traffic to.  
Learn more about next hop types that you see in this list in [Routing overview](#).

### View effective routes - commands

- Azure CLI: [az network nic show-effective-route-table](#)
- PowerShell: [Get-AzEffectiveRouteTable](#)

## Validate routing between two endpoints

You can determine the next hop type between a virtual machine and the IP address of another Azure resource, an on-premises resource, or a resource on the Internet. Determining Azure's routing is helpful when troubleshooting routing problems. To complete this task, you must have an existing Network Watcher. If you don't have an existing Network Watcher, create one by completing the steps in [Create a Network Watcher instance](#).

1. In the search box at the top of the portal, enter *network watcher* in the search box. When **Network Watcher** appears in the search results, select it.
2. Select **Next hop** under **NETWORK DIAGNOSTIC TOOLS**.
3. Select your **Subscription** and the **Resource group** of the source virtual machine you want to validate routing from.
4. Select the **Virtual machine**, **Network interface** attached to the virtual machine, and **Source IP address** assigned to the network interface that you want to validate routing from.
5. Enter the **Destination IP address** that you want to validate routing to.
6. Select **Next hop**.

7. After a short wait, information is returned that tells you the next hop type and the ID of the route that routed the traffic. Learn more about next hop types that you see returned in [Routing overview](#).

#### Validate routing between two endpoints - commands

- Azure CLI: [az network watcher show-next-hop](#)
- PowerShell: [Get-AzNetworkWatcherNextHop](#)

## Permissions

To perform tasks on route tables and routes, your account must be assigned to the [network contributor](#) role or to a [custom](#) role that is assigned the appropriate actions listed in the following table:

ACTION	NAME
Microsoft.Network/routeTables/read	Read a route table
Microsoft.Network/routeTables/write	Create or update a route table
Microsoft.Network/routeTables/delete	Delete a route table
Microsoft.Network/routeTables/join/action	Associate a route table to a subnet
Microsoft.Network/routeTables/routes/read	Read a route
Microsoft.Network/routeTables/routes/write	Create or update a route
Microsoft.Network/routeTables/routes/delete	Delete a route
Microsoft.Network/networkInterfaces/effectiveRouteTable/acti on	Get the effective route table for a network interface
Microsoft.Network/networkWatchers/nextHop/action	Gets the next hop from a VM

## Next steps

- Create a route table using [PowerShell](#) or [Azure CLI](#) sample scripts, or using [Azure Resource Manager templates](#)
- Create and apply [Azure policy](#) for virtual networks

# Restrict network access to PaaS resources with virtual network service endpoints using PowerShell

11/19/2019 • 10 minutes to read • [Edit Online](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Virtual network service endpoints enable you to limit network access to some Azure service resources to a virtual network subnet. You can also remove internet access to the resources. Service endpoints provide direct connection from your virtual network to supported Azure services, allowing you to use your virtual network's private address space to access the Azure services. Traffic destined to Azure resources through service endpoints always stays on the Microsoft Azure backbone network. In this article, you learn how to:

- Create a virtual network with one subnet
- Add a subnet and enable a service endpoint
- Create an Azure resource and allow network access to it from only a subnet
- Deploy a virtual machine (VM) to each subnet
- Confirm access to a resource from a subnet
- Confirm access is denied to a resource from a subnet and the internet

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.

2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

## Create a virtual network

Before creating a virtual network, you have to create a resource group for the virtual network, and all other resources created in this article. Create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myResourceGroup*:

```
New-AzResourceGroup -ResourceGroupName myResourceGroup -Location EastUS
```

Create a virtual network with [New-AzVirtualNetwork](#). The following example creates a virtual network named *myVirtualNetwork* with the address prefix `10.0.0.0/16`.

```
$virtualNetwork = New-AzVirtualNetwork `  
    -ResourceGroupName myResourceGroup `  
    -Location EastUS `  
    -Name myVirtualNetwork `  
    -AddressPrefix 10.0.0.0/16
```

Create a subnet configuration with [New-AzVirtualNetworkSubnetConfig](#). The following example creates a subnet configuration for a subnet named *Public*:

```
$subnetConfigPublic = Add-AzVirtualNetworkSubnetConfig `  
    -Name Public `  
    -AddressPrefix 10.0.0.0/24 `  
    -VirtualNetwork $virtualNetwork
```

Create the subnet in the virtual network by writing the subnet configuration to the virtual network with [Set-AzVirtualNetwork](#):

```
$virtualNetwork | Set-AzVirtualNetwork
```

## Enable a service endpoint

You can enable service endpoints only for services that support service endpoints. View service endpoint-enabled services available in an Azure location with [Get-AzVirtualNetworkAvailableEndpointService](#). The following example returns a list of service-endpoint-enabled services available in the *eastus* region. The list of services returned will grow over time as more Azure services become service endpoint enabled.

```
Get-AzVirtualNetworkAvailableEndpointService -Location eastus | Select Name
```

Create an additional subnet in the virtual network. In this example, a subnet named *Private* is created with a service endpoint for *Microsoft.Storage*:

```
$subnetConfigPrivate = Add-AzVirtualNetworkSubnetConfig ` 
    -Name Private ` 
    -AddressPrefix 10.0.1.0/24 ` 
    -VirtualNetwork $virtualNetwork ` 
    -ServiceEndpoint Microsoft.Storage

$virtualNetwork | Set-AzVirtualNetwork
```

## Restrict network access for a subnet

Create network security group security rules with [New-AzNetworkSecurityRuleConfig](#). The following rule allows outbound access to the public IP addresses assigned to the Azure Storage service:

```
$rule1 = New-AzNetworkSecurityRuleConfig ` 
    -Name Allow-Storage-All ` 
    -Access Allow ` 
    -DestinationAddressPrefix Storage ` 
    -DestinationPortRange * ` 
    -Direction Outbound ` 
    -Priority 100 ` 
    -Protocol * ` 
    -SourceAddressPrefix VirtualNetwork ` 
    -SourcePortRange *
```

The following rule denies access to all public IP addresses. The previous rule overrides this rule, due to its higher priority, which allows access to the public IP addresses of Azure Storage.

```
$rule2 = New-AzNetworkSecurityRuleConfig ` 
    -Name Deny-Internet-All ` 
    -Access Deny ` 
    -DestinationAddressPrefix Internet ` 
    -DestinationPortRange * ` 
    -Direction Outbound ` 
    -Priority 110 ` 
    -Protocol * ` 
    -SourceAddressPrefix VirtualNetwork ` 
    -SourcePortRange *
```

The following rule allows Remote Desktop Protocol (RDP) traffic inbound to the subnet from anywhere. Remote desktop connections are allowed to the subnet, so that you can confirm network access to a resource in a later step.

```
$rule3 = New-AzNetworkSecurityRuleConfig ` 
    -Name Allow-RDP-All ` 
    -Access Allow ` 
    -DestinationAddressPrefix VirtualNetwork ` 
    -DestinationPortRange 3389 ` 
    -Direction Inbound ` 
    -Priority 120 ` 
    -Protocol * ` 
    -SourceAddressPrefix * ` 
    -SourcePortRange *
```

Create a network security group with [New-AzNetworkSecurityGroup](#). The following example creates a network security group named *myNsgPrivate*.

```
$nsg = New-AzNetworkSecurityGroup `  
    -ResourceGroupName myResourceGroup `  
    -Location EastUS `  
    -Name myNsgPrivate `  
    -SecurityRules $rule1,$rule2,$rule3
```

Associate the network security group to the *Private* subnet with [Set-AzVirtualNetworkSubnetConfig](#) and then write the subnet configuration to the virtual network. The following example associates the *myNsgPrivate* network security group to the *Private* subnet:

```
Set-AzVirtualNetworkSubnetConfig `  
    -VirtualNetwork $VirtualNetwork `  
    -Name Private `  
    -AddressPrefix 10.0.1.0/24 `  
    -ServiceEndpoint Microsoft.Storage `  
    -NetworkSecurityGroup $nsg  
  
$virtualNetwork | Set-AzVirtualNetwork
```

## Restrict network access to a resource

The steps necessary to restrict network access to resources created through Azure services enabled for service endpoints varies across services. See the documentation for individual services for specific steps for each service. The remainder of this article includes steps to restrict network access for an Azure Storage account, as an example.

### Create a storage account

Create an Azure storage account with [New-AzStorageAccount](#). Replace

```
<replace-with-your-unique-storage-account-name>
```

 with a name that is unique across all Azure locations, between 3-24 characters in length, using only numbers and lower-case letters.

```
$storageAcctName = '<replace-with-your-unique-storage-account-name>'  
  
New-AzStorageAccount `  
    -Location EastUS `  
    -Name $storageAcctName `  
    -ResourceGroupName myResourceGroup `  
    -SkuName Standard_LRS `  
    -Kind StorageV2
```

After the storage account is created, retrieve the key for the storage account into a variable with [Get-AzStorageAccountKey](#):

```
$storageAcctKey = (Get-AzStorageAccountKey `  
    -ResourceGroupName myResourceGroup `  
    -AccountName $storageAcctName).Value[0]
```

The key is used to create a file share in a later step. Enter `$storageAcctKey` and note the value, as you'll also need to manually enter it in a later step when you map the file share to a drive in a VM.

### Create a file share in the storage account

Create a context for your storage account and key with [New-AzStorageContext](#). The context encapsulates the storage account name and account key:

```
$storageContext = New-AzStorageContext $storageAcctName $storageAcctKey
```

Create a file share with [New-AzStorageShare](#):

```
$share = New-AzStorageShare my-file-share -Context $storageContext
```

### Deny all network access to a storage account

By default, storage accounts accept network connections from clients in any network. To limit access to selected networks, change the default action to *Deny* with [Update-AzStorageAccountNetworkRuleSet](#). Once network access is denied, the storage account is not accessible from any network.

```
Update-AzStorageAccountNetworkRuleSet  
-ResourceGroupName "myresourcegroup"  
-Name $storageAcctName  
-DefaultAction Deny
```

### Enable network access from a subnet

Retrieve the created virtual network with [Get-AzVirtualNetwork](#) and then retrieve the private subnet object into a variable with [Get-AzVirtualNetworkSubnetConfig](#):

```
$privateSubnet = Get-AzVirtualNetwork  
-ResourceGroupName "myResourceGroup"  
-Name "myVirtualNetwork"  
| Get-AzVirtualNetworkSubnetConfig  
-Name "Private"
```

Allow network access to the storage account from the *Private* subnet with [Add-AzStorageAccountNetworkRule](#).

```
Add-AzStorageAccountNetworkRule  
-ResourceGroupName "myresourcegroup"  
-Name $storageAcctName  
-VirtualNetworkResourceId $privateSubnet.Id
```

## Create virtual machines

To test network access to a storage account, deploy a VM to each subnet.

### Create the first virtual machine

Create a virtual machine in the *Public* subnet with [New-AzVM](#). When running the command that follows, you are prompted for credentials. The values that you enter are configured as the user name and password for the VM.

The `-AsJob` option creates the VM in the background, so that you can continue to the next step.

```
New-AzVm  
-ResourceGroupName "myResourceGroup"  
-Location "East US"  
-VirtualNetworkName "myVirtualNetwork"  
-SubnetName "Public"  
-Name "myVmPublic"  
-AsJob
```

Output similar to the following example output is returned:

ID	Name	PSJobTypeName	State	HasMoreData	Location	Command
--	---	-----	-----	-----	-----	-----
1	Long Running...	AzureLongRun...	Running	True	localhost	New-AzVM

## Create the second virtual machine

Create a virtual machine in the *Private* subnet:

```
New-AzVm ` 
    -ResourceGroupName "myResourceGroup" ` 
    -Location "East US" ` 
    -VirtualNetworkName "myVirtualNetwork" ` 
    -SubnetName "Private" ` 
    -Name "myVmPrivate"
```

It takes a few minutes for Azure to create the VM. Do not continue to the next step until Azure finishes creating the VM and returns output to PowerShell.

## Confirm access to storage account

Use [Get-AzPublicIpAddress](#) to return the public IP address of a VM. The following example returns the public IP address of the *myVmPrivate* VM:

```
Get-AzPublicIpAddress ` 
    -Name myVmPrivate ` 
    -ResourceGroupName myResourceGroup ` 
    | Select IpAddress
```

Replace `<publicIpAddress>` in the following command, with the public IP address returned from the previous command, and then enter the following command:

```
mstsc /v:<publicIpAddress>
```

A Remote Desktop Protocol (.rdp) file is created and downloaded to your computer. Open the downloaded rdp file. If prompted, select **Connect**. Enter the user name and password you specified when creating the VM. You may need to select **More choices**, then **Use a different account**, to specify the credentials you entered when you created the VM. Select **OK**. You may receive a certificate warning during the sign-in process. If you receive the warning, select **Yes** or **Continue**, to proceed with the connection.

On the *myVmPrivate* VM, map the Azure file share to drive Z using PowerShell. Before running the commands that follow, replace `<storage-account-key>` and `<storage-account-name>` with values from you supplied or retrieved in [Create a storage account](#).

```
$acctKey = ConvertTo-SecureString -String "<storage-account-key>" -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential -ArgumentList "Azure\<storage-account-name>", $acctKey
New-PSDrive -Name Z -PSProvider FileSystem -Root "\\\<storage-account-name>.file.core.windows.net\my-file-share" -Credential $credential
```

PowerShell returns output similar to the following example output:

Name	Used (GB)	Free (GB)	Provider	Root
---	-----	-----	-----	-----
Z			FileSystem	\vnt.file.core.windows.net\my-f...

The Azure file share successfully mapped to the Z drive.

Confirm that the VM has no outbound connectivity to any other public IP addresses:

```
ping bing.com
```

You receive no replies, because the network security group associated to the *Private* subnet does not allow outbound access to public IP addresses other than the addresses assigned to the Azure Storage service.

Close the remote desktop session to the *myVmPrivate* VM.

## Confirm access is denied to storage account

Get the public IP address of the *myVmPublic* VM:

```
Get-AzPublicIpAddress ` 
-Name myVmPublic ` 
-ResourceGroupName myResourceGroup ` 
| Select IpAddress
```

Replace `<publicIpAddress>` in the following command, with the public IP address returned from the previous command, and then enter the following command:

```
mstsc /v:<publicIpAddress>
```

On the *myVmPublic* VM, attempt to map the Azure file share to drive Z. Before running the commands that follow, replace `<storage-account-key>` and `<storage-account-name>` with values from you supplied or retrieved in [Create a storage account](#).

```
$acctKey = ConvertTo-SecureString -String "<storage-account-key>" -AsPlainText -Force
$credential = New-Object System.Management.Automation.PSCredential -ArgumentList "Azure\<storage-account-name>", $acctKey
New-PSDrive -Name Z -PSPrinter FileSystem -Root "\\\<storage-account-name>.file.core.windows.net\my-file-share" -Credential $credential
```

Access to the share is denied, and you receive a `New-PSDrive : Access is denied` error. Access is denied because the *myVmPublic* VM is deployed in the *Public* subnet. The *Public* subnet does not have a service endpoint enabled for Azure Storage, and the storage account only allows network access from the *Private* subnet, not the *Public* subnet.

Close the remote desktop session to the *myVmPublic* VM.

From your computer, attempt to view the file shares in the storage account with the following command:

```
Get-AzStorageFile ` 
-ShareName my-file-share ` 
-Context $storageContext
```

Access is denied, and you receive a *Get-AzStorageFile : The remote server returned an error: (403) Forbidden. HTTP Status Code: 403 - HTTP Error Message: This request is not authorized to perform this operation* error, because your computer is not in the *Private* subnet of the *MyVirtualNetwork* virtual network.

## Clean up resources

When no longer needed, you can use [Remove-AzResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

## Next steps

In this article, you enabled a service endpoint for a virtual network subnet. You learned that service endpoints can be enabled for resources deployed with multiple Azure services. You created an Azure Storage account and limited network access to the storage account to only resources within a virtual network subnet. To learn more about service endpoints, see [Service endpoints overview](#) and [Manage subnets](#).

If you have multiple virtual networks in your account, you may want to connect two virtual networks together so the resources within each virtual network can communicate with each other. To learn how, see [Connect virtual networks](#).

# Restrict network access to PaaS resources with virtual network service endpoints using the Azure CLI

11/19/2019 • 9 minutes to read • [Edit Online](#)

Virtual network service endpoints enable you to limit network access to some Azure service resources to a virtual network subnet. You can also remove internet access to the resources. Service endpoints provide direct connection from your virtual network to supported Azure services, allowing you to use your virtual network's private address space to access the Azure services. Traffic destined to Azure resources through service endpoints always stays on the Microsoft Azure backbone network. In this article, you learn how to:

- Create a virtual network with one subnet
- Add a subnet and enable a service endpoint
- Create an Azure resource and allow network access to it from only a subnet
- Deploy a virtual machine (VM) to each subnet
- Confirm access to a resource from a subnet
- Confirm access is denied to a resource from a subnet and the internet

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select <b>Try It</b> in the upper-right corner of a code block. Selecting <b>Try It</b> doesn't automatically copy the code to Cloud Shell.	
Go to <a href="https://shell.azure.com">https://shell.azure.com</a> , or select the <b>Launch Cloud Shell</b> button to open Cloud Shell in your browser.	
Select the <b>Cloud Shell</b> button on the menu bar at the upper right in the <a href="#">Azure portal</a> .	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl+Shift+V** on Windows and Linux or by selecting **Cmd+Shift+V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this quickstart requires that you are running the Azure CLI version

2.0.28 or later. To find the version, run `az --version`. If you need to install or upgrade, see [Install Azure CLI](#).

## Create a virtual network

Before creating a virtual network, you have to create a resource group for the virtual network, and all other resources created in this article. Create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location.

```
az group create \
--name myResourceGroup \
--location eastus
```

Create a virtual network with one subnet with [az network vnet create](#).

```
az network vnet create \
--name myVirtualNetwork \
--resource-group myResourceGroup \
--address-prefix 10.0.0.0/16 \
--subnet-name Public \
--subnet-prefix 10.0.0.0/24
```

## Enable a service endpoint

You can enable service endpoints only for services that support service endpoints. View service endpoint-enabled services available in an Azure location with [az network vnet list-endpoint-services](#). The following example returns a list of service-endpoint-enabled services available in the *eastus* region. The list of services returned will grow over time, as more Azure services become service endpoint enabled.

```
az network vnet list-endpoint-services \
--location eastus \
--out table
```

Create an additional subnet in the virtual network with [az network vnet subnet create](#). In this example, a service endpoint for *Microsoft.Storage* is created for the subnet:

```
az network vnet subnet create \
--vnet-name myVirtualNetwork \
--resource-group myResourceGroup \
--name Private \
--address-prefix 10.0.1.0/24 \
--service-endpoints Microsoft.Storage
```

## Restrict network access for a subnet

Create a network security group with [az network nsg create](#). The following example creates a network security group named *myNsgPrivate*.

```
az network nsg create \
--resource-group myResourceGroup \
--name myNsgPrivate
```

Associate the network security group to the *Private* subnet with [az network vnet subnet update](#). The following example associates the *myNsgPrivate* network security group to the *Private* subnet:

```
az network vnet subnet update \
--vnet-name myVirtualNetwork \
--name Private \
--resource-group myResourceGroup \
--network-security-group myNsgPrivate
```

Create security rules with [az network nsg rule create](#). The rule that follows allows outbound access to the public IP addresses assigned to the Azure Storage service:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNsgPrivate \
--name Allow-Storage-All \
--access Allow \
--protocol "*" \
--direction Outbound \
--priority 100 \
--source-address-prefix "VirtualNetwork" \
--source-port-range "*" \
--destination-address-prefix "Storage" \
--destination-port-range "*"
```

Each network security group contains several [default security rules](#). The rule that follows overrides a default security rule that allows outbound access to all public IP addresses. The `--destination-address-prefix "Internet"` option denies outbound access to all public IP addresses. The previous rule overrides this rule, due to its higher priority, which allows access to the public IP addresses of Azure Storage.

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNsgPrivate \
--name Deny-Internet-All \
--access Deny \
--protocol "*" \
--direction Outbound \
--priority 110 \
--source-address-prefix "VirtualNetwork" \
--source-port-range "*" \
--destination-address-prefix "Internet" \
--destination-port-range "*"
```

The following rule allows SSH traffic inbound to the subnet from anywhere. The rule overrides a default security rule that denies all inbound traffic from the internet. SSH is allowed to the subnet so that connectivity can be tested in a later step.

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNsgPrivate \
--name Allow-SSH-All \
--access Allow \
--protocol Tcp \
--direction Inbound \
--priority 120 \
--source-address-prefix "*" \
--source-port-range "*" \
--destination-address-prefix "VirtualNetwork" \
--destination-port-range "22"
```

## Restrict network access to a resource

The steps necessary to restrict network access to resources created through Azure services enabled for service endpoints varies across services. See the documentation for individual services for specific steps for each service. The remainder of this article includes steps to restrict network access for an Azure Storage account, as an example.

## Create a storage account

Create an Azure storage account with [az storage account create](#). Replace

<replace-with-your-unique-storage-account-name> with a name that is unique across all Azure locations, between 3-24 characters in length, using only numbers and lower-case letters.

```
storageAcctName=<replace-with-your-unique-storage-account-name>

az storage account create \
--name $storageAcctName \
--resource-group myResourceGroup \
--sku Standard_LRS \
--kind StorageV2
```

After the storage account is created, retrieve the connection string for the storage account into a variable with [az storage account show-connection-string](#). The connection string is used to create a file share in a later step.

```
saConnectionString=$(az storage account show-connection-string \
--name $storageAcctName \
--resource-group myResourceGroup \
--query 'connectionString' \
--out tsv)
```

View the contents of the variable and note the value for **AccountKey** returned in the output, because it's used in a later step.

```
echo $saConnectionString
```

## Create a file share in the storage account

Create a file share in the storage account with [az storage share create](#). In a later step, this file share is mounted to confirm network access to it.

```
az storage share create \
--name my-file-share \
--quota 2048 \
--connection-string $saConnectionString > /dev/null
```

## Deny all network access to a storage account

By default, storage accounts accept network connections from clients in any network. To limit access to selected networks, change the default action to *Deny* with [az storage account update](#). Once network access is denied, the storage account is not accessible from any network.

```
az storage account update \
--name $storageAcctName \
--resource-group myResourceGroup \
--default-action Deny
```

## Enable network access from a subnet

Allow network access to the storage account from the *Private* subnet with [az storage account network-rule add](#).

```
az storage account network-rule add \
--resource-group myResourceGroup \
--account-name $storageAcctName \
--vnet-name myVirtualNetwork \
--subnet Private
```

## Create virtual machines

To test network access to a storage account, deploy a VM to each subnet.

### Create the first virtual machine

Create a VM in the *Public* subnet with [az vm create](#). If SSH keys do not already exist in a default key location, the command creates them. To use a specific set of keys, use the `--ssh-key-value` option.

```
az vm create \
--resource-group myResourceGroup \
--name myVmPublic \
--image UbuntuLTS \
--vnet-name myVirtualNetwork \
--subnet Public \
--generate-ssh-keys
```

The VM takes a few minutes to create. After the VM is created, the Azure CLI shows information similar to the following example:

```
{
  "fqdns": "",
  "id": "/subscriptions/00000000-0000-0000-0000-
0000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVmPublic",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.4",
  "publicIpAddress": "13.90.242.231",
  "resourceGroup": "myResourceGroup"
}
```

Take note of the **publicIpAddress** in the returned output. This address is used to access the VM from the internet in a later step.

### Create the second virtual machine

```
az vm create \
--resource-group myResourceGroup \
--name myVmPrivate \
--image UbuntuLTS \
--vnet-name myVirtualNetwork \
--subnet Private \
--generate-ssh-keys
```

The VM takes a few minutes to create. After creation, take note of the **publicIpAddress** in the output returned. This address is used to access the VM from the internet in a later step.

## Confirm access to storage account

SSH into the *myVmPrivate* VM. Replace `<publicIpAddress>` with the public IP address of your *myVmPrivate* VM.

```
ssh <publicIpAddress>
```

Create a folder for a mount point:

```
sudo mkdir /mnt/MyAzureFileShare
```

Mount the Azure file share to the directory you created. Before running the following command, replace `<storage-account-name>` with the account name and `<storage-account-key>` with the key you retrieved in [Create a storage account](#).

```
sudo mount --types cifs //<storage-account-name>.file.core.windows.net/my-file-share /mnt/MyAzureFileShare --  
options vers=3.0,username=<storage-account-name>,password=<storage-account-  
key>,dir_mode=0777,file_mode=0777,serverino
```

You receive the `user@myVmPrivate:~$` prompt. The Azure file share successfully mounted to `/mnt/MyAzureFileShare`.

Confirm that the VM has no outbound connectivity to any other public IP addresses:

```
ping bing.com -c 4
```

You receive no replies, because the network security group associated to the *Private* subnet does not allow outbound access to public IP addresses other than the addresses assigned to the Azure Storage service.

Exit the SSH session to the *myVmPrivate* VM.

## Confirm access is denied to storage account

Use the following command to create an SSH session with the *myVmPublic* VM. Replace `<publicIpAddress>` with the public IP address of your *myVmPublic* VM:

```
ssh <publicIpAddress>
```

Create a directory for a mount point:

```
sudo mkdir /mnt/MyAzureFileShare
```

Attempt to mount the Azure file share to the directory you created. This article assumes you deployed the latest version of Ubuntu. If you are using earlier versions of Ubuntu, see [Mount on Linux](#) for additional instructions about mounting file shares. Before running the following command, replace `<storage-account-name>` with the account name and `<storage-account-key>` with the key you retrieved in [Create a storage account](#):

```
sudo mount --types cifs //<storage-account-name>.file.core.windows.net/my-file-share /mnt/MyAzureFileShare --  
options vers=3.0,username=<storage-account-name>,password=<storage-account-  
key>,dir_mode=0777,file_mode=0777,serverino
```

Access is denied, and you receive a `mount error(13): Permission denied` error, because the *myVmPublic* VM is deployed within the *Public* subnet. The *Public* subnet does not have a service endpoint enabled for Azure Storage, and the storage account only allows network access from the *Private* subnet, not the *Public* subnet.

Exit the SSH session to the *myVmPublic* VM.

From your computer, attempt to view the shares in your storage account with `az storage share list`. Replace `<account-name>` and `<account-key>` with the storage account name and key from [Create a storage account](#):

```
az storage share list \
--account-name <account-name> \
--account-key <account-key>
```

Access is denied and you receive a *This request is not authorized to perform this operation* error, because your computer is not in the *Private* subnet of the *MyVirtualNetwork* virtual network.

## Clean up resources

When no longer needed, use `az group delete` to remove the resource group and all of the resources it contains.

```
az group delete --name myResourceGroup --yes
```

## Next steps

In this article, you enabled a service endpoint for a virtual network subnet. You learned that service endpoints can be enabled for resources deployed with multiple Azure services. You created an Azure Storage account and limited network access to the storage account to only resources within a virtual network subnet. To learn more about service endpoints, see [Service endpoints overview](#) and [Manage subnets](#).

If you have multiple virtual networks in your account, you may want to connect two virtual networks together so the resources within each virtual network can communicate with each other. To learn how, see [Connect virtual networks](#).

# Create, change, or delete service endpoint policy using the Azure portal

2/26/2020 • 3 minutes to read • [Edit Online](#)

Service endpoint policies enable you to filter virtual network traffic to specific Azure resources, over service endpoints. If you're not familiar with service endpoint policies, see [service endpoint policies overview](#) to learn more.

In this tutorial, you learn how to:

- Create a service endpoint policy
- Create a service endpoint policy definition
- Create a virtual network with a subnet
- Associate a service endpoint policy to a subnet

If you don't have an Azure subscription, create a [free account](#) before you begin.

## Sign in to Azure

Sign in to the Azure portal at <https://portal.azure.com>.

## Create a service endpoint policy

1. Select + **Create a resource** on the upper, left corner of the Azure portal.
2. In search pane, type "service endpoint policy" and select **Service endpoint policy** and then select **Create**.

The screenshot shows the Azure portal interface for creating a new 'Service endpoint policy'. The top navigation bar includes 'Dashboard', 'New', and 'Service endpoint policy'. The main title 'Service endpoint policy' is displayed above a Microsoft logo and a 'Create' button. Below the title, there are two tabs: 'Overview' (which is selected) and 'Plans'. A detailed description of VNet service endpoints is provided, mentioning their purpose in extending private address space and identity to Azure services via direct connections. At the bottom, there are 'Useful Links' and 'Documentation' links.

3. Enter, or select, the following information in **Basics**

- Subscription : Select your subscription for policy
- Resource group : Select **Create new** and enter *myResourceGroup*
- Name : myEndpointPolicy
- Location : Central US

Home > MySEP-RG > New > Service endpoint policy > Create a service endpoint policy

## Create a service endpoint policy

**Basics** Policy definitions Tags Review + create

Create service endpoint policies to allow traffic to specific azure resources from your virtual network over service endpoints. [Learn more](#)

**Project details**

Subscription \*

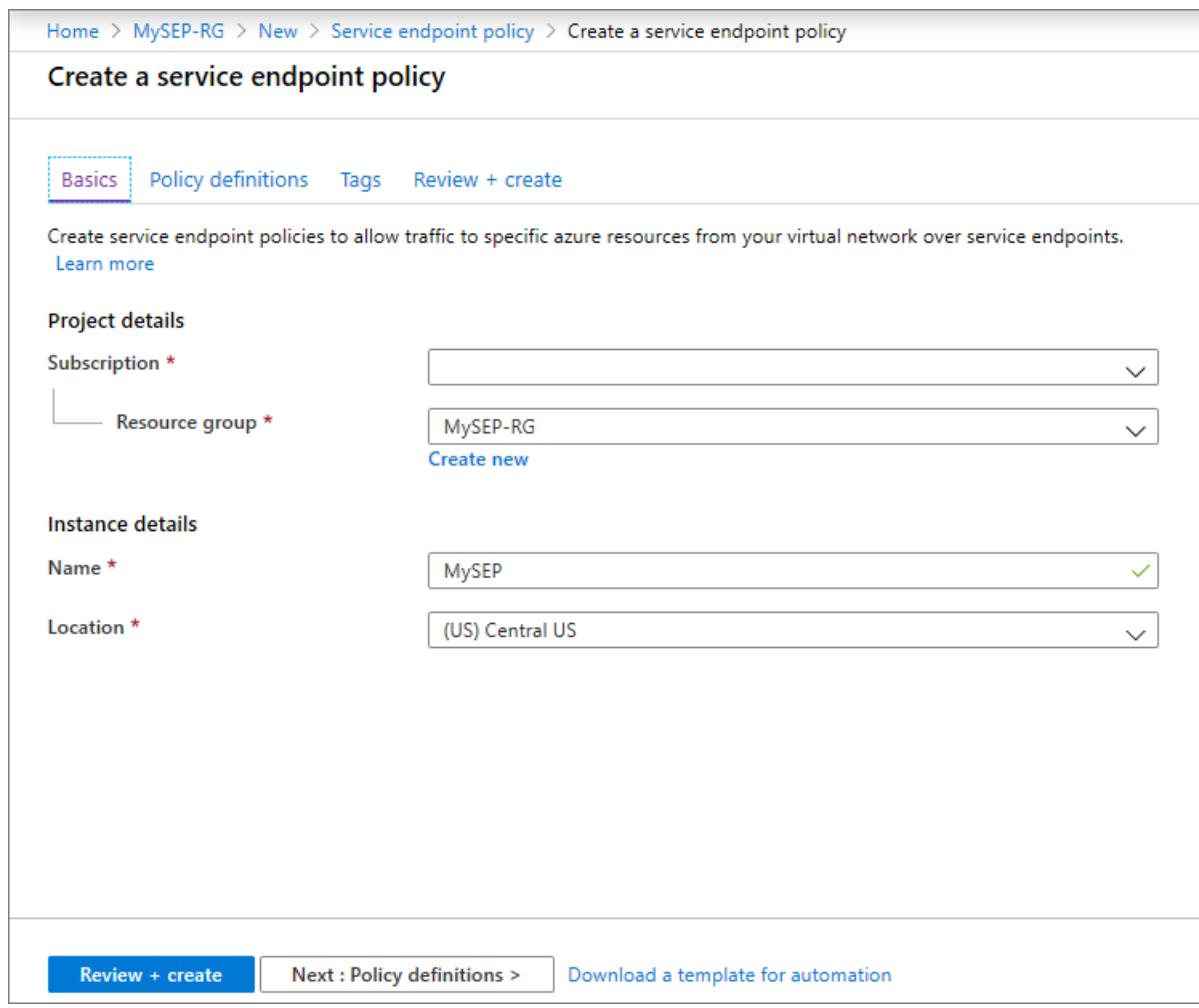
Resource group \*  MySEP-RG [Create new](#)

**Instance details**

Name \*  MySEP ✓

Location \*  (US) Central US

[Review + create](#) [Next : Policy definitions >](#) [Download a template for automation](#)



4. Select + **Add** under **Resources** and enter or select the following information in **Add a resource** pane

- Service : Only **Microsoft.Storage** is available with Service Endpoint Policies
- Scope : Select one out of **Single Account**, **All accounts in subscription** and **All accounts in resource group**
- Subscription : Select your subscription for storage account. Policy and storage accounts can be in different subscriptions.
- Resource group : Select your resource group. Required, if scope is set as, "All accounts in resource group" or "Single account".
- Resource : Select your Azure Storage resource under the selected Subscription or Resource Group
- Click on **Add** button at bottom to finish adding the resource

**Create a service endpoint policy**

**Basics** Policy definitions Tags Review + create

This policy will allow access only to the azure service resources listed

**Resources** + Add a resource

Service	Allowed Resources	Re
Add a resource to get started		

Service \* Microsoft.Storage

Scope \* Single account

Subscription \*

Resource group \* MySEP-RG

Resource \* myteststgacc123

Review + create < Previous Next : Tags > Do Add

- Add more resources by repeating the above steps as needed

5. Optional: Enter or select, the following information in **Tags**:

- Key : Select your key for the policy. Ex: Dept
- Value : Enter value pair for the key. Ex: Finance

6. Select **Review + Create**. Validate the information and Click **Create**. To make further edits, click **Previous**.

**Create a service endpoint policy**

✓ Validation passed

Basics Policy definitions Tags Review + create

**Basics**

Subscription	NON-PROD - Tesing - EAPv2
Resource group	MySEP-RG
Region	(US) Central US
Name	MySEP

**Resources**

Microsoft.Storage	myteststgacc123 (Storage account)
-------------------	-----------------------------------

For this policy to take effect, you will need to associate it to one or more subnets that have virtual network service endpoints. Please visit a virtual network in (US) Central US region and then select the subnets to which you would like to associate this policy.

Create < Previous Download a template for automation

## View endpoint policies

1. In the *All services* box in the portal, begin typing *service endpoint policies*. Select **Service Endpoint**

## Policies.

2. Under **Subscriptions**, select your subscription and resource group, as shown in the following picture

The screenshot shows the 'Service endpoint policies' dashboard under the 'Microsoft' category. At the top, there are filters for 'Subscription' (3 of 21 selected), 'Resource group' (all), 'Location' (all), and a search bar ('Filter by name...'). Below the filters, it says 'Showing 1 to 3 of 3 records.' A table lists one policy: 'MySEP' (Service endpoint policy) in 'MySEP-RG' (Resource group) located in 'Central US' (Location) under 'NON-PROD - Testing' (Subscription). There are columns for Name, Type, Resource group, Location, and Subscription.

3. Select the policy and click on **Policy Definitions** to view or add more policy definitions.

The screenshot shows the 'MySEP - Policy definitions' page. On the left, a sidebar lists options like Overview, Activity log, Access control (IAM), Tags, Settings (with Policy definitions selected), Associated subnets, Properties, Locks, Export template, Support + troubleshooting, and New support request. The main area has a search bar ('Search resources') and a table with columns: Service, Allowed Resources, Resource Group, Subscription name, and Subscription ID. One entry is listed: Microsoft.Storage (Service) with myteststgacc123 (Allowed Resources), MySEP-RG (Resource Group), and MySEP-RG (Subscription name and ID).

4. Select **Associated subnets** to view the subnets the policy is associated. If no subnet is associated yet, follow the instructions in the next step.

The screenshot shows the 'MySEP - Associated subnets' page. The sidebar includes the same set of options as the previous page. The main area features a search bar ('Search subnets') and a table with columns: Virtual network | Subnet, Resource group, and Service endpoints. A message at the bottom says 'Begin with associating subnet with this service endpoint policy.'

5. Associate a policy to a subnet

## WARNING

Ensure that all the resources accessed from the subnet are added to the policy definition before associating the policy to the given subnet. Once the policy is associated, only access to the *allow listed* resources will be allowed over service endpoints.

Also ensure that no managed Azure services exist in the subnet that is being associated to the service endpoint policy

- Before you can associate a policy to a subnet, you have to create a virtual network and subnet. Please refer to the [Create a Virtual Network](#) article for help with this.
- Once you have the virtual network and subnet are setup, you need to configure Virtual Network Service Endpoints for Azure Storage. On the Virtual Network blade, select **Service endpoints**, and in the next pane select **Microsoft.Storage** and under **Subnets** select the desired VNet or Subnet
- Now, you can either choose to select the Service Endpoint Policy from the drop-down in the above pane if you have already created Service Endpoint policies before configuring Service Endpoint for the Subnet as shown below

The screenshot shows the Azure portal interface for managing service endpoints. On the left, the navigation menu includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Address space, Connected devices, Subnets, DDoS protection, Firewall, Security, DNS servers, Peerings, Service endpoints), Private endpoints, Properties, and Locks. The 'Service endpoints' option under Settings is currently selected. On the right, a modal window titled 'Add service endpoints' is open. It has a 'Service' dropdown set to 'Microsoft.Storage'. Below it, a 'Service endpoint policies' section shows 'MySEP' selected. There is also a note explaining the impact of switching to private IP addresses and a warning about existing IP firewall rules. At the bottom of the modal is a blue 'Add' button.

- OR if you are associating Service Endpoint policies after Service Endpoints are already configured, you can choose to associate the subnet from within the Service Endpoint Policy blade by navigating to the **Associated Subnets** pane as shown below

The screenshot shows the Azure portal interface for managing a Service Endpoint Policy (MySEP). On the left, there's a navigation sidebar with links like Overview, Activity log, Access control (IAM), Tags, Settings (Policy definitions, Associated subnets selected), Properties, Locks, and Export template. Below that is Support + troubleshooting and New support request.

The main area is titled "Edit subnet association" for "MySEP - Associated subnets". It shows a "Virtual network \*" dropdown set to "myVnet". A note says: "Select subnets from this virtual network to which the policy will be applied. Unselecting a subnet will disassociate the policy from that subnet. You can only apply this policy to subnets with service endpoints that support the services in this policy." A table lists one subnet: "mySubnet1" with "Service endpoints" "Microsoft.Storage". At the bottom is a blue "Apply" button.

### WARNING

Access to Azure Storage resources in all regions will be restricted as per Service Endpoint Policy from this subnet.

## Next steps

In this tutorial, you created a service endpoint policy and associated it to a subnet. To learn more about service endpoint policies, see [service endpoint policies overview](#).

# Virtual appliance scenario

4/26/2019 • 8 minutes to read • [Edit Online](#)

A common scenario among larger Azure customer is the need to provide a two-tiered application exposed to the Internet, while allowing access to the back tier from an on-premises datacenter. This document will walk you through a scenario using User Defined Routes (UDR), a VPN Gateway, and network virtual appliances to deploy a two-tier environment that meets the following requirements:

- Web application must be accessible from the public Internet only.
- Web server hosting the application must be able to access a backend application server.
- All traffic from the Internet to the web application must go through a firewall virtual appliance. This virtual appliance will be used for Internet traffic only.
- All traffic going to the application server must go through a firewall virtual appliance. This virtual appliance will be used for access to the backend end server, and access coming in from the on-premises network via a VPN Gateway.
- Administrators must be able to manage the firewall virtual appliances from their on-premises computers, by using a third firewall virtual appliance used exclusively for management purposes.

This is a standard perimeter network (also knowns as DMZ) scenario with a DMZ and a protected network. Such scenario can be constructed in Azure by using NSGs,firewall virtual appliances, or a combination of both. The table below shows some of the pros and cons between NSGs and firewall virtual appliances.

	PROS	CONS
NSG	No cost. Integrated into Azure RBAC. Rules can be created in Azure Resource Manager templates.	Complexity could vary in larger environments.
Firewall	Full control over data plane. Central management through firewall console.	Cost of firewall appliance. Not integrated with Azure RBAC.

The solution below uses firewall virtual appliances to implement a perimeter network (DMZ)/protected network scenario.

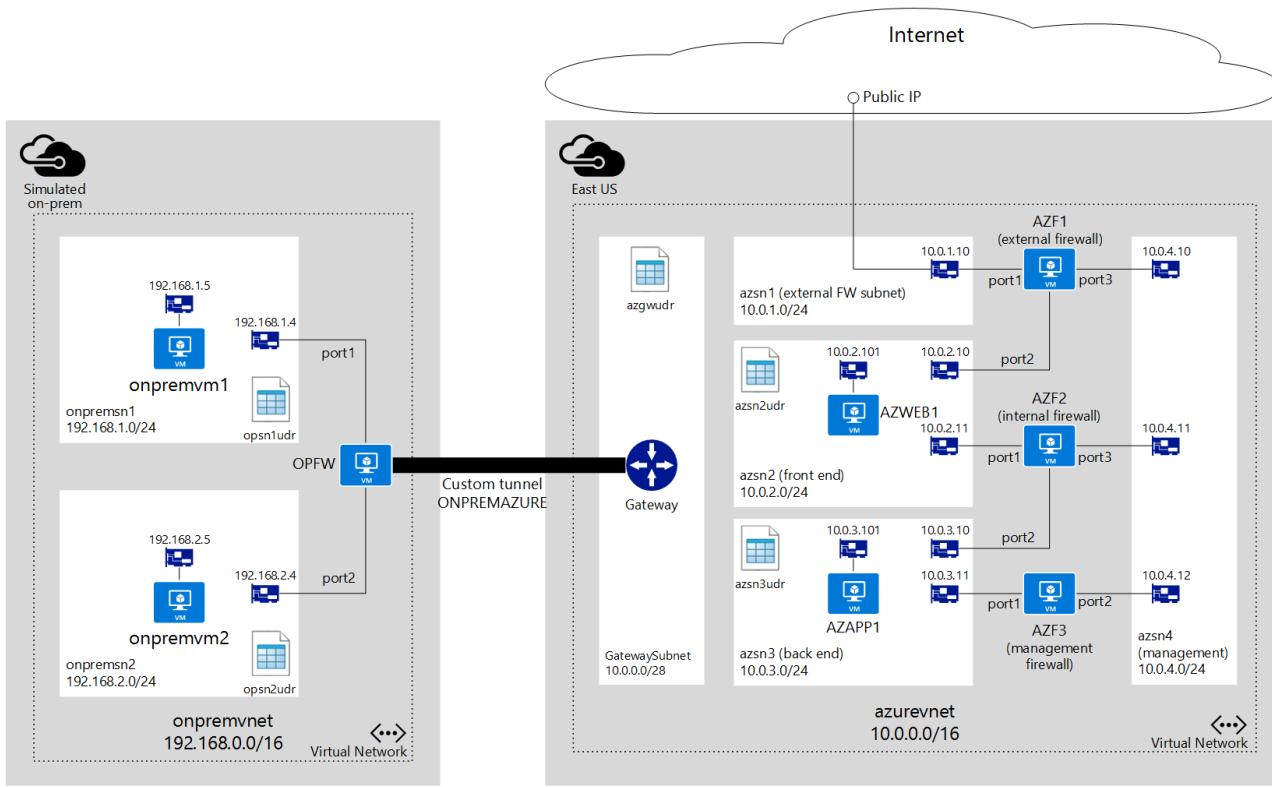
## Considerations

You can deploy the environment explained above in Azure using different features available today, as follows.

- **Virtual network (VNet).** An Azure VNet acts in similar fashion to an on-premises network, and can be segmented into one or more subnets to provide traffic isolation, and separation of concerns.
- **Virtual appliance.** Several partners provide virtual appliances in the Azure Marketplace that can be used for the three firewalls described above.
- **User Defined Routes (UDR).** Route tables can contain UDRs used by Azure networking to control the flow of packets within a VNet. These route tables can be applied to subnets. One of the newest features in Azure is the ability to apply a route table to the GatewaySubnet, providing the ability to forward all traffic coming into the Azure VNet from a hybrid connection to a virtual appliance.
- **IP Forwarding.** By default, the Azure networking engine forward packets to virtual network interface cards (NICs) only if the packet destination IP address matches the NIC IP address. Therefore, if a UDR defines that a

packet must be sent to a given virtual appliance, the Azure networking engine would drop that packet. To ensure the packet is delivered to a VM (in this case a virtual appliance) that is not the actual destination for the packet, you need to enable IP Forwarding for the virtual appliance.

- **Network Security Groups (NSGs).** The example below does not make use of NSGs, but you could use NSGs applied to the subnets and/or NICs in this solution to further filter the traffic in and out of those subnets and NICs.



In this example there is a subscription that contains the following:

- 2 resource groups, not shown in the diagram.
  - **ONPREMRG**. Contains all resources necessary to simulate an on-premises network.
  - **AZURERG**. Contains all resources necessary for the Azure virtual network environment.
- A VNet named **onpremvnet** used to mimic an on-premises datacenter segmented as listed below.
  - **onpremsn1**. Subnet containing a virtual machine (VM) running Ubuntu to mimic an on-premises server.
  - **onpremsn2**. Subnet containing a VM running Ubuntu to mimic an on-premises computer used by an administrator.
- There is one firewall virtual appliance named **OPFW** on **onpremvnet** used to maintain a tunnel to **azurevnet**.
- A VNet named **azurevnet** segmented as listed below.
  - **azsn1**. External firewall subnet used exclusively for the external firewall. All Internet traffic will come in through this subnet. This subnet only contains a NIC linked to the external firewall.
  - **azsn2**. Front end subnet hosting a VM running as a web server that will be accessed from the Internet.
  - **azsn3**. Backend subnet hosting a VM running a backend application server that will be accessed by the front end web server.
  - **azsn4**. Management subnet used exclusively to provide management access to all firewall virtual appliances. This subnet only contains a NIC for each firewall virtual appliance used in the solution.
  - **GatewaySubnet**. Azure hybrid connection subnet required for ExpressRoute and VPN Gateway to provide connectivity between Azure VNets and other networks.
- There are 3 firewall virtual appliances in the **azurevnet** network.
  - **AZF1**. External firewall exposed to the public Internet by using a public IP address resource in Azure. You need to ensure you have a template from the Marketplace, or directly from your appliance vendor, that provisions a 3-NIC virtual appliance.

- **AZF2.** Internal firewall used to control traffic between **azsn2** and **azsn3**. This is also a 3-NIC virtual appliance.
- **AZF3.** Management firewall accessible to administrators from the on-premises datacenter, and connected to a management subnet used to manage all firewall appliances. You can find 2-NIC virtual appliance templates in the Marketplace, or request one directly from your appliance vendor.

## User Defined Routing (UDR)

Each subnet in Azure can be linked to a UDR table used to define how traffic initiated in that subnet is routed. If no UDRs are defined, Azure uses default routes to allow traffic to flow from one subnet to another. To better understand UDRs, visit [What are User Defined Routes and IP Forwarding](#).

To ensure communication is done through the right firewall appliance, based on the last requirement above, you need to create the following route table containing UDRs in **azurevnet**.

### **azgwudr**

In this scenario, the only traffic flowing from on-premises to Azure will be used to manage the firewalls by connecting to **AZF3**, and that traffic must go through the internal firewall, **AZF2**. Therefore, only one route is necessary in the **GatewaySubnet** as shown below.

DESTINATION	NEXT HOP	EXPLANATION
10.0.4.0/24	10.0.3.11	Allows on-premises traffic to reach management firewall <b>AZF3</b>

### **azsn2udr**

DESTINATION	NEXT HOP	EXPLANATION
10.0.3.0/24	10.0.2.11	Allows traffic to the backend subnet hosting the application server through <b>AZF2</b>
0.0.0.0/0	10.0.2.10	Allows all other traffic to be routed through <b>AZF1</b>

### **azsn3udr**

DESTINATION	NEXT HOP	EXPLANATION
10.0.2.0/24	10.0.3.10	Allows traffic to <b>azsn2</b> to flow from app server to the webserver through <b>AZF2</b>

You also need to create route tables for the subnets in **onpremvnet** to mimic the on-premises datacenter.

### **onpremsn1udr**

DESTINATION	NEXT HOP	EXPLANATION
192.168.2.0/24	192.168.1.4	Allows traffic to <b>onpremsn2</b> through <b>OPFW</b>

### **onpremsn2udr**

DESTINATION	NEXT HOP	EXPLANATION
10.0.3.0/24	192.168.2.4	Allows traffic to the backed subnet in Azure through <b>OPFW</b>
192.168.1.0/24	192.168.2.4	Allows traffic to <b>onpremsn1</b> through <b>OPFW</b>

## IP Forwarding

UDR and IP Forwarding are features that you can use in combination to allow virtual appliances to be used to control traffic flow in an Azure VNet. A virtual appliance is nothing more than a VM that runs an application used to handle network traffic in some way, such as a firewall or a NAT device.

This virtual appliance VM must be able to receive incoming traffic that is not addressed to itself. To allow a VM to receive traffic addressed to other destinations, you must enable IP Forwarding for the VM. This is an Azure setting, not a setting in the guest operating system. Your virtual appliance still needs to run some type of application to handle the incoming traffic, and route it appropriately.

To learn more about IP Forwarding, visit [What are User Defined Routes and IP Forwarding](#).

As an example, imagine you have the following setup in an Azure vnet:

- Subnet **onpremsn1** contains a VM named **onpremvm1**.
- Subnet **onpremsn2** contains a VM named **onpremvm2**.
- A virtual appliance named **OPFW** is connected to **onpremsn1** and **onpremsn2**.
- A user defined route linked to **onpremsn1** specifies that all traffic to **onpremsn2** must be sent to **OPFW**.

At this point, if **onpremvm1** tries to establish a connection with **onpremvm2**, the UDR will be used and traffic will be sent to **OPFW** as the next hop. Keep in mind that the actual packet destination is not being changed, it still says **onpremvm2** is the destination.

Without IP Forwarding enabled for **OPFW**, the Azure virtual networking logic will drop the packets, since it only allows packets to be sent to a VM if the VM's IP address is the destination for the packet.

With IP Forwarding, the Azure virtual network logic will forward the packets to OPFW, without changing its original destination address. **OPFW** must handle the packets and determine what to do with them.

For the scenario above to work, you must enable IP Forwarding on the NICs for **OPFW**, **AZF1**, **AZF2**, and **AZF3** that are used for routing (all NICs except the ones linked to the management subnet).

## Firewall Rules

As described above, IP Forwarding only ensures packets are sent to the virtual appliances. Your appliance still needs to decide what to do with those packets. In the scenario above, you will need to create the following rules in your appliances:

### OPFW

OPFW represents an on-premises device containing the following rules:

- **Route:** All traffic to 10.0.0.0/16 (**azurevnet**) must be sent through tunnel **ONPREMAZURE**.
- **Policy:** Allow all bidirectional traffic between **port2** and **ONPREMAZURE**.

### AZF1

AZF1 represents an Azure virtual appliance containing the following rules:

- **Policy:** Allow all bidirectional traffic between **port1** and **port2**.

## AZF2

AZF2 represents an Azure virtual appliance containing the following rules:

- **Route:** All traffic to 10.0.0.0/16 (**onpremvnet**) must be sent to the Azure gateway IP address (i.e. 10.0.0.1) through **port1**.
- **Policy:** Allow all bidirectional traffic between **port1** and **port2**.

## Network Security Groups (NSGs)

In this scenario, NSGs are not being used. However, you could apply NSGs to each subnet to restrict incoming and outgoing traffic. For instance, you could apply the following NSG rules to the external FW subnet.

### Incoming

- Allow all TCP traffic from the Internet to port 80 on any VM in the subnet.
- Deny all other traffic from the Internet.

### Outgoing

- Deny all traffic to the Internet.

## High level steps

To deploy this scenario, follow the high level steps below.

1. Login to your Azure Subscription.
2. If you want to deploy a VNet to mimic the on-premises network, provision the resources that are part of **ONPREMRG**.
3. Provision the resources that are part of **AZURERG**.
4. Provision the tunnel from **onpremvnet** to **azurevnet**.
5. Once all resources are provisioned, sign in to **onpremvn2** and ping 10.0.3.101 to test connectivity between **onpremsn2** and **azsn3**.

2 minutes to read

# Create a virtual machine with a static public IP address using the Azure portal

1/16/2020 • 3 minutes to read • [Edit Online](#)

You can create a virtual machine with a static public IP address. A public IP address enables you to communicate to a virtual machine from the internet. Assign a static public IP address, rather than a dynamic address, to ensure that the address never changes. Learn more about [static public IP addresses](#). To change a public IP address assigned to an existing virtual machine from dynamic to static, or to work with private IP addresses, see [Add, change, or remove IP addresses](#). Public IP addresses have a [nominal charge](#), and there is a [limit](#) to the number of public IP addresses that you can use per subscription.

## Sign in to Azure

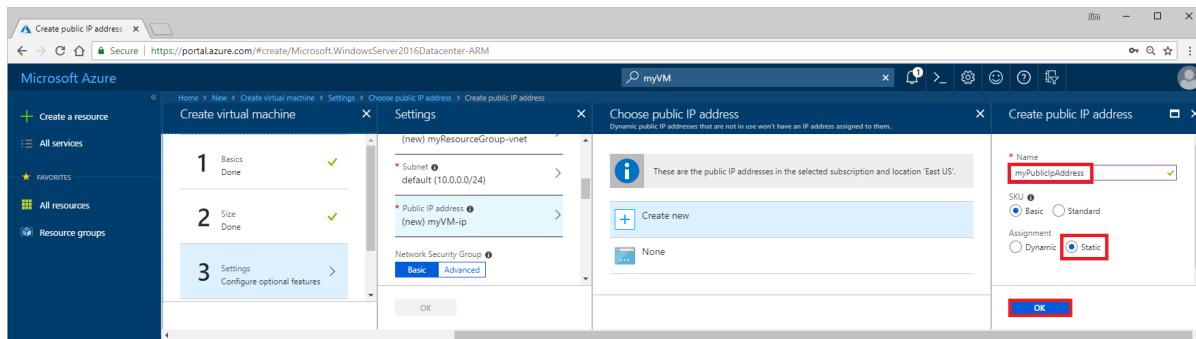
Sign in to the Azure portal at <https://portal.azure.com>.

## Create a virtual machine

1. Select + **Create a resource** found on the upper, left corner of the Azure portal.
2. Select **Compute**, and then select **Windows Server 2016 VM**, or another operating system of your choosing.
3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **OK**:

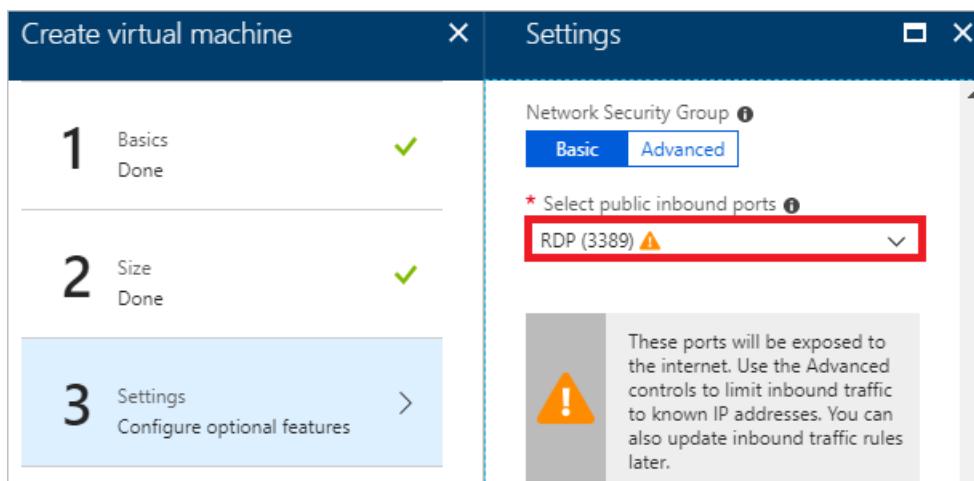
SETTING	VALUE
Name	myVM
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the <a href="#">defined complexity requirements</a> .
Subscription	Select your subscription.
Resource group	Select <b>Use existing</b> and select <b>myResourceGroup</b> .
Location	Select <b>East US</b>

4. Select a size for the VM and then select **Select**.
5. Under **Settings**, select **Public IP address**.
6. Enter *myPublicIpAddress*, select **Static**, and then select **OK**, as shown in the following picture:

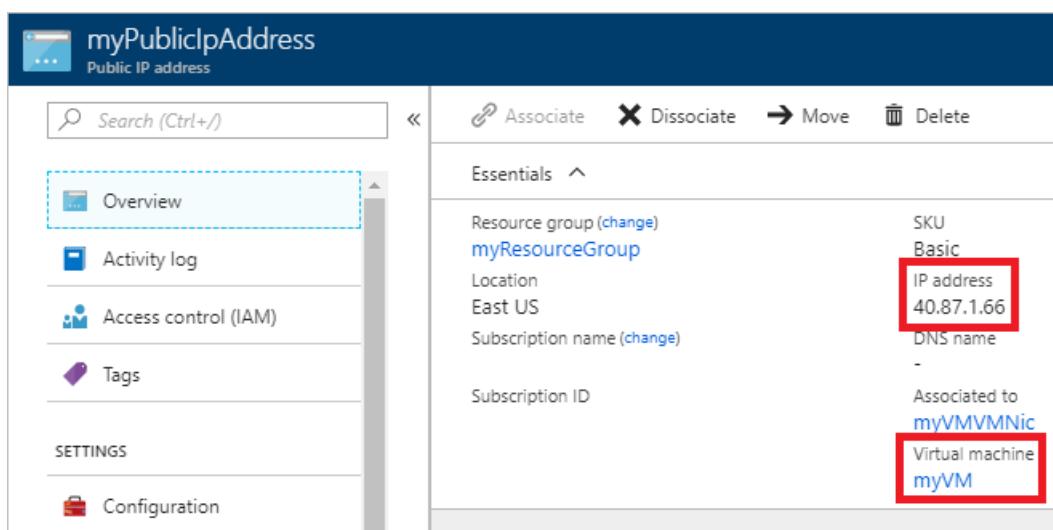


If the public IP address must be a standard SKU, select **Standard** under **SKU**. Learn more about [Public IP address SKUs](#). If the virtual machine will be added to the back-end pool of a public Azure Load Balancer, the SKU of the virtual machine's public IP address must match the SKU of the load balancer's public IP address. For details, see [Azure Load Balancer](#).

7. Select a port, or no ports under **Select public inbound ports**. Portal 3389 is selected, to enable remote access to the Windows Server virtual machine from the internet. Opening port 3389 from the internet is not recommended for production workloads.



8. Accept the remaining default settings and select **OK**.
9. On the **Summary** page, select **Create**. The virtual machine takes a few minutes to deploy.
10. Once the virtual machine is deployed, enter *myPublicIpAddress* in the search box at the top of the portal. When **myPublicIpAddress** appears in the search results, select it.
11. You can view the public IP address that is assigned, and that the address is assigned to the **myVM** virtual machine, as shown in the following picture:



Azure assigned a public IP address from addresses used in the region you created the virtual machine in. You can download the list of ranges (prefixes) for the Azure [Public](#), [US government](#), [China](#), and [Germany](#) clouds.

12. Select **Configuration** to confirm that the assignment is **Static**.

The screenshot shows the Azure portal interface for managing a Public IP address. The left sidebar lists several options: Overview, Activity log, Access control (IAM), Tags, SETTINGS, Configuration (which is highlighted with a red box), Properties, Locks, and Automation script. The main content area is titled 'myPublicIpAddress - Configuration'. It includes a search bar, save and discard buttons, and a message box stating: 'This static public IP address is associated to the IP configuration 'ipconfigmyVM', in the network interface 'myVMVMNic'. You must dissociate it from the network interface before changing its assignment.' Below this, the 'Assignment' section is shown with two radio buttons: 'Dynamic' (unchecked) and 'Static' (checked). A red box highlights the 'Assignment' section. Further down, the 'IP address' is listed as 40.87.1.66, 'Idle timeout (minutes)' is set to 4, and there is a field for 'DNS name label (optional)' containing '.eastus.cloudapp.azure.com'. A note at the bottom suggests using Azure DNS with a link to 'Try Azure DNS now'.

#### WARNING

Do not modify the IP address settings within the virtual machine's operating system. The operating system is unaware of Azure public IP addresses. Though you can add private IP address settings to the operating system, we recommend not doing so unless necessary, and not until after reading [Add a private IP address to an operating system](#).

## Clean up resources

When no longer needed, delete the resource group and all of the resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal. When you see **myResourceGroup** in the search results, select it.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

## Next steps

- Learn more about [public IP addresses](#) in Azure
- Learn more about all [public IP address settings](#)
- Learn more about [private IP addresses](#) and assigning a [static private IP address](#) to an Azure virtual machine
- Learn more about creating [Linux](#) and [Windows](#) virtual machines

# Create a virtual machine with a static public IP address using PowerShell

1/16/2020 • 3 minutes to read • [Edit Online](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

You can create a virtual machine with a static public IP address. A public IP address enables you to communicate to a virtual machine from the internet. Assign a static public IP address, rather than a dynamic address, to ensure that the address never changes. Learn more about [static public IP addresses](#). To change a public IP address assigned to an existing virtual machine from dynamic to static, or to work with private IP addresses, see [Add, change, or remove IP addresses](#). Public IP addresses have a [nominal charge](#), and there is a [limit](#) to the number of public IP addresses that you can use per subscription.

## Create a virtual machine

You can complete the following steps from your local computer or by using the Azure Cloud Shell. To use your local computer, ensure you have the [Azure PowerShell installed](#). To use the Azure Cloud Shell, select **Try It** in the top right corner of any command box that follows. The Cloud Shell signs you into Azure.

1. If using the Cloud Shell, skip to step 2. Open a command session and sign into Azure with `Connect-AzAccount`.
2. Create a resource group with the `New-AzResourceGroup` command. The following example creates a resource group in the East US Azure region:

```
New-AzResourceGroup -Name myResourceGroup -Location EastUS
```

3. Create a virtual machine with the `New-AzVm` command. The `-AllocationMethod "Static"` option assigns a static public IP address to the virtual machine. The following example creates a Windows Server virtual machine with a static, basic SKU public IP address named *myPublicIpAddress*. When prompted, provide a username and password to be used as the sign in credentials for the virtual machine:

```
New-AzVm `  
-ResourceGroupName "myResourceGroup" `  
-Name "myVM" `  
-Location "East US" `  
-PublicIpAddressName "myPublicIpAddress" `  
-AllocationMethod "Static"
```

If the public IP address must be a standard SKU, you have to [create a public IP address](#), [create a network interface](#), [assign the public IP address to the network interface](#), and then [create a virtual machine with the network interface](#), in separate steps. Learn more about [Public IP address SKUs](#). If the virtual machine will be added to the back-end pool of a public Azure Load Balancer, the SKU of the virtual machine's public IP address must match the SKU of the load balancer's public IP address. For details, see [Azure Load Balancer](#).

4. View the public IP address assigned and confirm that it was created as a static address, with [Get-AzPublicIpAddress](#):

```
Get-AzPublicIpAddress `  
-ResourceGroupName "myResourceGroup" `  
-Name "myPublicIpAddress" `  
| Select "IpAddress", "PublicIpAllocationMethod" `  
| Format-Table
```

Azure assigned a public IP address from addresses used in the region you created the virtual machine in. You can download the list of ranges (prefixes) for the Azure [Public](#), [US government](#), [China](#), and [Germany](#) clouds.

#### WARNING

Do not modify the IP address settings within the virtual machine's operating system. The operating system is unaware of Azure public IP addresses. Though you can add private IP address settings to the operating system, we recommend not doing so unless necessary, and not until after reading [Add a private IP address to an operating system](#).

## Clean up resources

When no longer needed, you can use [Remove-AzResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

## Next steps

- Learn more about [public IP addresses](#) in Azure
- Learn more about all [public IP address settings](#)
- Learn more about [private IP addresses](#) and assigning a [static private IP address](#) to an Azure virtual machine
- Learn more about creating [Linux](#) and [Windows](#) virtual machines

# Create a virtual machine with a static public IP address using the Azure CLI

1/16/2020 • 2 minutes to read • [Edit Online](#)

You can create a virtual machine with a static public IP address. A public IP address enables you to communicate to a virtual machine from the internet. Assign a static public IP address, rather than a dynamic address, to ensure that the address never changes. Learn more about [static public IP addresses](#). To change a public IP address assigned to an existing virtual machine from dynamic to static, or to work with private IP addresses, see [Add, change, or remove IP addresses](#). Public IP addresses have a [nominal charge](#), and there is a [limit](#) to the number of public IP addresses that you can use per subscription.

## Create a virtual machine

You can complete the following steps from your local computer or by using the Azure Cloud Shell. To use your local computer, ensure you have the [Azure CLI installed](#). To use the Azure Cloud Shell, select **Try It** in the top right corner of any command box that follows. The Cloud Shell signs you into Azure.

1. If using the Cloud Shell, skip to step 2. Open a command session and sign into Azure with `az login`.
2. Create a resource group with the `az group create` command. The following example creates a resource group in the East US Azure region:

```
az group create --name myResourceGroup --location eastus
```

3. Create a virtual machine with the `az vm create` command. The `--public-ip-address-allocation=static` option assigns a static public IP address to the virtual machine. The following example creates an Ubuntu virtual machine with a static, basic SKU public IP address named `myPublicIpAddress`:

```
az vm create \
  --resource-group myResourceGroup \
  --name myVM \
  --image UbuntuLTS \
  --admin-username azureuser \
  --generate-ssh-keys \
  --public-ip-address myPublicIpAddress \
  --public-ip-address-allocation static
```

If the public IP address must be a standard SKU, add `--public-ip-sku Standard` to the previous command. Learn more about [Public IP address SKUs](#). If the virtual machine will be added to the back-end pool of a public Azure Load Balancer, the SKU of the virtual machine's public IP address must match the SKU of the load balancer's public IP address. For details, see [Azure Load Balancer](#).

4. View the public IP address assigned and confirm that it was created as a static, basic SKU address, with `az network public-ip show`:

```
az network public-ip show \
  --resource-group myResourceGroup \
  --name myPublicIpAddress \
  --query [ipAddress,publicIpAllocationMethod,sku] \
  --output table
```

Azure assigned a public IP address from addresses used in the region you created the virtual machine in. You can download the list of ranges (prefixes) for the Azure [Public](#), [US government](#), [China](#), and [Germany](#) clouds.

#### WARNING

Do not modify the IP address settings within the virtual machine's operating system. The operating system is unaware of Azure public IP addresses. Though you can add private IP address settings to the operating system, we recommend not doing so unless necessary, and not until after reading [Add a private IP address to an operating system](#).

## Clean up resources

When no longer needed, you can use `az group delete` to remove the resource group and all of the resources it contains:

```
az group delete --name myResourceGroup --yes
```

## Next steps

- Learn more about [public IP addresses](#) in Azure
- Learn more about all [public IP address settings](#)
- Learn more about [private IP addresses](#) and assigning a [static private IP address](#) to an Azure virtual machine
- Learn more about creating [Linux](#) and [Windows](#) virtual machines

# Associate a public IP address to a virtual machine

1/3/2020 • 10 minutes to read • [Edit Online](#)

In this article, you learn how to associate a public IP address to an existing virtual machine (VM). If you want to connect to a VM from the internet, the VM must have a public IP address associated to it. If you want to create a new VM with a public IP address, you can do so using the [Azure portal](#), the [Azure command-line interface \(CLI\)](#), or [PowerShell](#). Public IP addresses have a nominal fee. For details, see [pricing](#). There is a limit to the number of public IP addresses that you can use per subscription. For details, see [limits](#).

You can use the [Azure portal](#), the Azure [command-line interface \(CLI\)](#), or [PowerShell](#) to associate a public IP address to a VM.

## Azure portal

1. Sign in to the [Azure portal](#).
2. Browse to, or search for the virtual machine that you want to add the public IP address to and then select it.
3. Under **Settings**, select **Networking**, and then select the network interface you want to add the public IP address to, as shown in the following picture:

The screenshot shows the Azure portal interface for managing a virtual machine named 'myVM'. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, and Networking. The Networking option is highlighted with a red box. The main content area shows the 'myVM - Networking' page. At the top, there are buttons for Attach network interface and Detach network interface. Below that, the 'Network Interface' section displays 'myVMVMNic'. It shows the Virtual network/subnet as 'myVm2VNET/myVm2Subnet', Public IP as 'None', and Private IP as '10.0.0.5'. The 'Effective security rules' and 'Topology' tabs are also visible. Under the 'Inbound port rules' tab, it lists one rule: 'Network security group myVMNSG (attached to network interface: myVMVMNic) Impacts 0 subnets, 1 network interfaces'. A table below shows the rule details: Priority 1000 and Name 'rdp'.

### NOTE

Public IP addresses are associated to network interfaces attached to a VM. In the previous picture, the VM only has one network interface. If the VM had multiple network interfaces, they would all appear, and you'd select the network interface you want to associate the public IP address to.

4. Select **IP configurations** and then select an IP configuration, as shown in the following picture:

Home > myVMVMNic - IP configurations

## myVMVMNic - IP configurations

Network interface

Search (Ctrl+ /)

Add Save Discard

Overview Activity log Access control (IAM) Tags Settings IP configurations DNS servers Network security group Properties

IP forwarding settings

IP forwarding  Disabled  Enabled

Virtual network myVm2VNET

IP configurations

\* Subnet myVm2Subnet (10.0.0.0/24)

Search IP configurations

NAME	IP VERSIO...	TYPE	PRIVATE IP ADDRESS	PUBLIC IP ADDRESS
ipconfigmyVM	IPv4	Primary	10.0.0.5 (Dynamic)	-

#### NOTE

Public IP addresses are associated to IP configurations for a network interface. In the previous picture, the network interface has one IP configuration. If the network interface had multiple IP configurations, they would all appear in the list, and you'd select the IP configuration that you want to associate the public IP address to.

- Select **Enabled**, then select **IP address (Configure required settings)**. Choose an existing public IP address, which automatically closes the **Choose public IP address** box. If you don't have any available public IP addresses listed, you need to create one. To learn how, see [Create a public IP address](#). Select **Save**, as shown in the picture that follows, and then close the box for the IP configuration.

Home > myVMVMNic - IP configurations > ipconfigmyVM > Choose public IP address

ipconfigmyVM myVMVMNic

Save  Discard

Public IP address settings

Public IP address  Enabled

\* IP address  Configure required settings

X

Choose public IP address

Dynamic public IP addresses that are not in use won't have an IP address assigned to them.

**i** These are the public IP addresses in the selected subscription and location 'East US'.

Create new

myVMPublicIP myResourceGroup

#### NOTE

The public IP addresses that appear are those that exist in the same region as the VM. If you have multiple public IP addresses created in the region, all will appear here. If any are grayed out, it's because the address is already associated to a different resource.

- View the public IP address assigned to the IP configuration, as shown in the picture that follows. It may take a few seconds for an IP address to appear.

The screenshot shows the Azure portal's IP configurations page for a specific network interface. On the left, there's a sidebar with options like Overview, Activity log, Access control (IAM), Tags, Settings, IP configurations (which is selected), DNS servers, Network security group, and Properties. The main area displays IP forwarding settings (disabled), the virtual network (myVm2VNET), and the IP configurations. One configuration is selected, showing its name (ipconfigmyVM), IP version (IPv4), type (Primary), private IP address (10.0.0.5 (Dynamic)), and public IP address (52.179.3.114 (myVMPublicIP)). The public IP address is highlighted with a red box.

#### NOTE

The address is assigned from a pool of addresses used in each Azure region. To see a list of address pools used in each region, see [Microsoft Azure Datacenter IP Ranges](#). The address assigned can be any address in the pools used for the region. If you need the address to be assigned from a specific pool in the region, use a [Public IP address prefix](#).

7. Allow network traffic to the VM with security rules in a network security group.

## Azure CLI

Install the [Azure CLI](#), or use the Azure Cloud Shell. The Azure Cloud Shell is a free Bash shell that you can run directly within the Azure portal. It has the Azure CLI preinstalled and configured to use with your account. Select the **Try it** button in the CLI commands that follow. Selecting **Try it** invokes a Cloud Shell that you can sign in to your Azure account with.

1. If using the CLI locally in Bash, sign in to Azure with `az login`.
2. A public IP address is associated to an IP configuration of a network interface attached to a VM. Use the `az network nic-ip-config update` command to associate a public IP address to an IP configuration. The following example associates an existing public IP address named *myVMPublicIP* to the IP configuration named *ipconfigmyVM* of an existing network interface named *myVMVMNic* that exists in a resource group named *myResourceGroup*.

```
az network nic ip-config update \
--name ipconfigmyVM \
--nic-name myVMVMNic \
--resource-group myResourceGroup \
--public-ip-address myVMPublicIP
```

- If you don't have an existing public IP address, use the `az network public-ip create` command to create one. For example, the following command creates a public IP address named *myVMPublicIP* in a resource group named *myResourceGroup*.

```
az network public-ip create --name myVMPublicIP --resource-group myResourceGroup
```

#### NOTE

The previous command creates a public IP address with default values for several settings that you may want to customize. To learn more about all public IP address settings, see [Create a public IP address](#). The address is assigned from a pool of public IP addresses used for each Azure region. To see a list of address pools used in each region, see [Microsoft Azure Datacenter IP Ranges](#).

- If you don't know the name of a network interface attached to your VM, use the [az vm nic list](#) command to view them. For example, the following command lists the names of the network interfaces attached to a VM named *myVM* in a resource group named *myResourceGroup*:

```
az vm nic list --vm-name myVM --resource-group myResourceGroup
```

The output includes one or more lines that are similar to the following example:

```
"id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkInterfaces/myVMVMnic",
```

In the previous example, *myVMVMNic* is the name of the network interface.

- If you don't know the name of an IP configuration for a network interface, use the [az network nic ip-config list](#) command to retrieve them. For example, the following command lists the names of the IP configurations for a network interface named *myVMVMNic* in a resource group named *myResourceGroup*:

```
az network nic ip-config list --nic-name myVMVMNic --resource-group myResourceGroup --out table
```

3. View the public IP address assigned to the IP configuration with the [az vm list-ip-addresses](#) command. The following example shows the IP addresses assigned to an existing VM named *myVM* in a resource group named *myResourceGroup*.

```
az vm list-ip-addresses --name myVM --resource-group myResourceGroup --out table
```

#### NOTE

The address is assigned from a pool of addresses used in each Azure region. To see a list of address pools used in each region, see [Microsoft Azure Datacenter IP Ranges](#). The address assigned can be any address in the pools used for the region. If you need the address to be assigned from a specific pool in the region, use a [Public IP address prefix](#).

4. [Allow network traffic to the VM](#) with security rules in a network security group.

## PowerShell

Install [PowerShell](#), or use the Azure Cloud Shell. The Azure Cloud Shell is a free shell that you can run directly within the Azure portal. It has PowerShell preinstalled and configured to use with your account. Select the **Try it** button in the PowerShell commands that follow. Selecting **Try it** invokes a Cloud Shell that you can sign in to your Azure account with.

1. If using PowerShell locally, sign in to Azure with `Connect-AzAccount`.

2. A public IP address is associated to an IP configuration of a network interface attached to a VM. Use the [Get-AzVirtualNetwork](#) and [Get-AzVirtualNetworkSubnetConfig](#) commands to get the virtual network and subnet that the network interface is in. Next, use the [Get-AzNetworkInterface](#) command to get a network interface and the [Get-AzPublicIpAddress](#) command to get an existing public IP address. Then use the [Set-AzNetworkInterfaceIpConfig](#) command to associate the public IP address to the IP configuration and the [Set-AzNetworkInterface](#) command to write the new IP configuration to the network interface.

The following example associates an existing public IP address named *myVMPublicIP* to the IP configuration named *ipconfigmyVM* of an existing network interface named *myVMVMNic* that exists in a subnet named *myVMSubnet* in a virtual network named *myVMVNet*. All resources are in a resource group named *myResourceGroup*.

```
$vnet = Get-AzVirtualNetwork -Name myVMVNet -ResourceGroupName myResourceGroup  
$subnet = Get-AzVirtualNetworkSubnetConfig -Name myVMSubnet -VirtualNetwork $vnet  
$nic = Get-AzNetworkInterface -Name myVMVMNic -ResourceGroupName myResourceGroup  
$pip = Get-AzPublicIpAddress -Name myVMPublicIP -ResourceGroupName myResourceGroup  
$nic | Set-AzNetworkInterfaceIpConfig -Name ipconfigmyVM -PublicIPAddress $pip -Subnet $subnet  
$nic | Set-AzNetworkInterface
```

- If you don't have an existing public IP address, use the [New-AzPublicIpAddress](#) command to create one. For example, the following command creates a *dynamic* public IP address named *myVMPublicIP* in a resource group named *myResourceGroup* in the *eastus* region.

```
New-AzPublicIpAddress -Name myVMPublicIP -ResourceGroupName myResourceGroup -AllocationMethod Dynamic -Location eastus
```

#### NOTE

The previous command creates a public IP address with default values for several settings that you may want to customize. To learn more about all public IP address settings, see [Create a public IP address](#). The address is assigned from a pool of public IP addresses used for each Azure region. To see a list of address pools used in each region, see [Microsoft Azure Datacenter IP Ranges](#).

- If you don't know the name of a network interface attached to your VM, use the [Get-AzVM](#) command to view them. For example, the following command lists the names of the network interfaces attached to a VM named *myVM* in a resource group named *myResourceGroup*:

```
$vm = Get-AzVM -name myVM -ResourceGroupName myResourceGroup  
$vm.NetworkProfile
```

The output includes one or more lines that are similar to the example that follows. In the example output, *myVMVMNic* is the name of the network interface.

```
"id": "/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkInterfaces/myVMVM  
nic",
```

- If you don't know the name of the virtual network or subnet that the network interface is in, use the [Get-AzNetworkInterface](#) command to view the information. For example, the following command gets the virtual network and subnet information for a network interface named *myVMVMNic* in a resource group named *myResourceGroup*:

```
$nic = Get-AzNetworkInterface -Name myVMVMNic -ResourceGroupName myResourceGroup  
$ipConfigs = $nic.IpConfigurations  
$ipConfigs.Subnet | Select Id
```

The output includes one or more lines that are similar to the example that follows. In the example output, *myVMVNET* is the name of the virtual network and *myVMSubnet* is the name of the subnet.

```
"/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetworks/myVMVNET/  
subnets/myVMSubnet",
```

- If you don't know the name of an IP configuration for a network interface, use the [Get-AzNetworkInterface](#) command to retrieve them. For example, the following command lists the names of the IP configurations for a network interface named *myVMVMNic* in a resource group named *myResourceGroup*:

```
$nic = Get-AzNetworkInterface -Name myVMVMNic -ResourceGroupName myResourceGroup  
$nic.IPCConfigurations
```

The output includes one or more lines that are similar to the example that follows. In the example output, *ipconfigmyVM* is the name of an IP configuration.

```
Id      : /subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkInterfaces/myVMVN  
ic/ipConfigurations/ipconfigmyVM
```

3. View the public IP address assigned to the IP configuration with the [Get-AzPublicIpAddress](#) command. The following example shows the address assigned to a public IP address named *myVMPublicIP* in a resource group named *myResourceGroup*.

```
Get-AzPublicIpAddress -Name myVMPublicIP -ResourceGroupName myResourceGroup | Select IPAddress
```

If you don't know the name of the public IP address assigned to an IP configuration, run the following commands to get it:

```
$nic = Get-AzNetworkInterface -Name myVMVMNic -ResourceGroupName myResourceGroup  
$nic.IPCConfigurations  
$address = $nic.IPCConfigurations.PublicIpAddress  
$address | Select Id
```

The output includes one or more lines that are similar to the example that follows. In the example output, *myVMPublicIP* is the name of the public IP address assigned to the IP configuration.

```
"/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Network/publicIPAddresses/myVMPublicIP"
```

**NOTE**

The address is assigned from a pool of addresses used in each Azure region. To see a list of address pools used in each region, see [Microsoft Azure Datacenter IP Ranges](#). The address assigned can be any address in the pools used for the region. If you need the address to be assigned from a specific pool in the region, use a [Public IP address prefix](#).

4. [Allow network traffic to the VM](#) with security rules in a network security group.

## Allow network traffic to the VM

Before you can connect to the public IP address from the internet, ensure that you have the necessary ports open in any network security group that you might have associated to the network interface, the subnet the network interface is in, or both. Though security groups filter traffic to the private IP address of the network interface, once inbound internet traffic arrives at the public IP address, Azure translates the public address to the private IP address, so if a network security group prevents the traffic flow, the communication with the public IP address fails. You can view the effective security rules for a network interface and its subnet using the [Portal](#), [CLI](#), or [PowerShell](#).

## Next steps

Allow inbound internet traffic to your VM with a network security group. To learn how to create a network security group, see [Work with network security groups](#). To learn more about network security groups, see [Security groups](#).

# Dissociate a public IP address from an Azure VM

12/6/2019 • 4 minutes to read • [Edit Online](#)

In this article, you learn how to dissociate a public IP address from an Azure virtual machine (VM).

You can use the [Azure portal](#), the Azure [command-line interface](#) (CLI), or [PowerShell](#) to dissociate a public IP address from a VM.

## Azure portal

1. Sign in to the [Azure portal](#).
2. Browse to, or search for the virtual machine that you want to disassociate the public IP address from and then select it.
3. In the VM page, select **Overview**, select the public IP address as shown in the following picture:

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with navigation links like 'Dashboard', 'Resource groups', and 'myResourceGroup'. Under 'myResourceGroup', 'myVM' is selected. The main content area has a title bar with 'myVM' and several actions: 'Connect', 'Start', 'Restart', 'Stop', 'Capture', 'Delete', and 'Refresh'. Below this is a blue banner with the text 'Advisor (1 of 3): Enable virtual machine replication to protect your applications from regional outage →'. The main content area is titled 'Overview' and contains detailed information about the VM, including its resource group, status, location, subscription, computer name, operating system, size, and tags. On the right side, there's a table showing the Public IP address as 52.170.252.185, which is also highlighted with a red box. Other details in the table include Private IP address (192.168.1.4), Public IP address (IPv6), Private IP address (IPv6), Virtual network/subnet (myVnet/mySubnet), DNS name (myvm-0467f8.eastus.cloudapp.azure.com), and Scale Set (N/A).

4. In the public IP address page, select **Overview**, and then select **Dissociate**, as shown in the following picture:

This screenshot shows the Microsoft Azure portal interface, specifically the 'myPublicIpAddress' page under 'myResourceGroup'. The left sidebar shows 'myPublicIpAddress' selected. The main content area has a title bar with 'myPublicIpAddress' and actions: 'Associate', 'Dissociate', 'Move', 'Delete', and 'Refresh'. Below this is a blue banner with the text 'Associate' and 'Dissociate'. The main content area is titled 'Overview' and contains information about the public IP address, including its resource group, location, subscription, and tags. The 'Dissociate' button is highlighted with a red box.

5. In **Dissociate public IP address**, select **Yes**.

## Azure CLI

Install the [Azure CLI](#), or use the Azure Cloud Shell. The Azure Cloud Shell is a free Bash shell that you can run directly within the Azure portal. It has the Azure CLI preinstalled and configured to use with your account. Select the **Try it** button in the CLI commands that follow. Selecting **Try it** invokes a Cloud Shell that you can sign in to your Azure account with.

1. If using the CLI locally in Bash, sign in to Azure with `az login`.
2. A public IP address is associated to an IP configuration of a network interface attached to a VM. Use the `az`

`network nic-ip-config update` command to dissociate a public IP address from an IP configuration. The following example dissociates a public IP address named *myVMPublicIP* from the IP configuration named *ipconfigmyVM* of an existing network interface named *myVMVMNic* that is attached to a VM named *myVM* in a resource group named *myResourceGroup*.

```
az network nic ip-config update \
--name ipconfigmyVM \
--resource-group myResourceGroup \
--nic-name myVMVMNic \
--remove PublicIpAddress
```

If you don't know the name of a network interface attached to your VM, use the `az vm nic list` command to view them. For example, the following command lists the names of the network interfaces attached to a VM named *myVM* in a resource group named *myResourceGroup*:

```
az vm nic list --vm-name myVM --resource-group myResourceGroup
```

The output includes one or more lines that are similar to the following example:

```
"id": "/subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkInterfaces/myVMVMNic",
```

In the previous example, *myVMVMNic* is the name of the network interface.

- If you don't know the name of an IP configuration for a network interface, use the `az network nic ip-config list` command to retrieve them. For example, the following command lists the names of the public IP configurations for a network interface named *myVMVMNic* in a resource group named *myResourceGroup*:

```
az network nic ip-config list --nic-name myVMVMNic --resource-group myResourceGroup --out table
```

- If you don't know the name of a public IP configuration for a network interface, use the `az network nic ip-config show` command to retrieve them. For example, the following command lists the names of the public IP configurations for a network interface named *myVMVMNic* in a resource group named *myResourceGroup*:

```
az network nic ip-config show --name ipconfigmyVM --nic-name myVMVMNic --resource-group myResourceGroup --query publicIPAddress.id
```

## PowerShell

Install [PowerShell](#), or use the Azure Cloud Shell. The Azure Cloud Shell is a free shell that you can run directly within the Azure portal. It has PowerShell preinstalled and configured to use with your account. Select the **Try it** button in the PowerShell commands that follow. Selecting **Try it** invokes a Cloud Shell that you can sign in to your Azure account with.

1. If using PowerShell locally, sign in to Azure with `Connect-AzAccount`.
2. A public IP address is associated to an IP configuration of a network interface attached to a VM. Use the `Get-AzNetworkInterface` command to get a network interface. Set the Public IP address value to null and then use the `Set-AzNetworkInterface` command to write the new IP configuration to the network interface.

The following example dissociates a public IP address named *myVMPublicIP* from a network interface

named *myVMVMNic* that is attached to a VM named *myVM*. All resources are in a resource group named *myResourceGroup*.

```
$nic = Get-AzNetworkInterface -Name myVMVMNic -ResourceGroupName myResourceGroup  
$nic.IpConfigurations.publicIpAddress.id = $null  
Set-AzNetworkInterface -NetworkInterface $nic
```

- If you don't know the name of a network interface attached to your VM, use the [Get-AzVM](#) command to view them. For example, the following command lists the names of the network interfaces attached to a VM named *myVM* in a resource group named *myResourceGroup*:

```
$vm = Get-AzVM -name myVM -ResourceGroupName myResourceGroup  
$vm.NetworkProfile
```

The output includes one or more lines that are similar to the example that follows. In the example output, *myVMVMNic* is the name of the network interface.

```
"id": "/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkInterfaces/myVMVMNic",
```

- If you don't know the name of an IP configuration for a network interface, use the [Get-AzNetworkInterface](#) command to retrieve them. For example, the following command lists the names of the IP configurations for a network interface named *myVMVMNic* in a resource group named *myResourceGroup*:

```
$nic = Get-AzNetworkInterface -Name myVMVMNic -ResourceGroupName myResourceGroup  
$nic.IpConfigurations.id
```

The output includes one or more lines that are similar to the example that follows. In the example output, *ipconfigmyVM* is the name of an IP configuration.

```
"id": "/subscriptions/00000000-0000-0000-0000-  
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Network/networkInterfaces/myVMVMNic/ipCo  
nfigurations/ipconfigmyVM"
```

## Next steps

- Learn how to [associate a public IP address to a VM](#).

# Configure private IP addresses for a virtual machine using the Azure portal

11/19/2019 • 4 minutes to read • [Edit Online](#)

A virtual machine (VM) is automatically assigned a private IP address from a range that you specify, based on the subnet it is deployed in. The address is retained by a VM until the VM is deleted. Azure dynamically assigns the next available private IP address from the subnet you create a VM in. Assign a static IP address if you want a specific IP address from the subnet assigned to the VM.

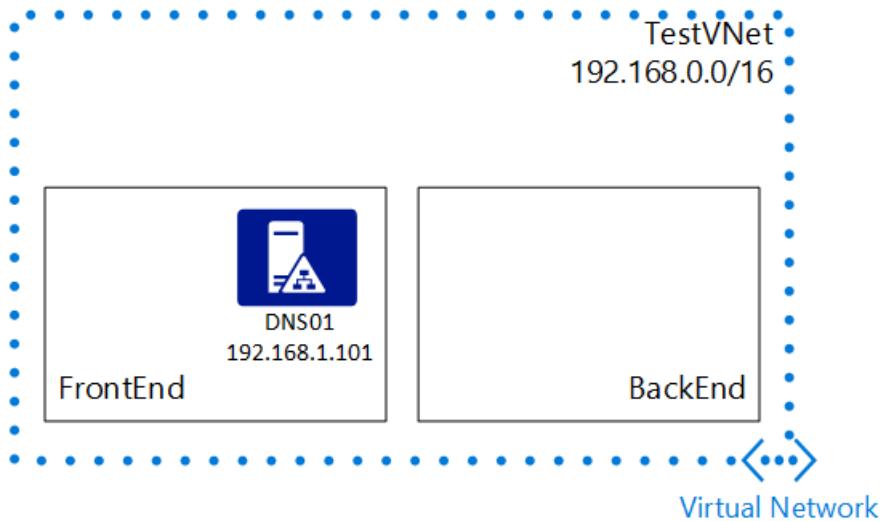
## IMPORTANT

Before you work with Azure resources, it's important to understand that Azure currently has two deployment models: Azure Resource Manager and classic. Make sure you understand [deployment models and tools](#) before you work with any Azure resource. You can view the documentation for different tools by clicking the tabs at the top of this article.

This article covers the Resource Manager deployment model. You can also [manage static private IP address in the classic deployment model](#).

## Scenario

To better illustrate how to configure a static IP address for a VM, this document will use the scenario below.



In this scenario you will create a VM named **DNS01** in the **FrontEnd** subnet, and set it to use a static IP address of **192.168.1.101**.

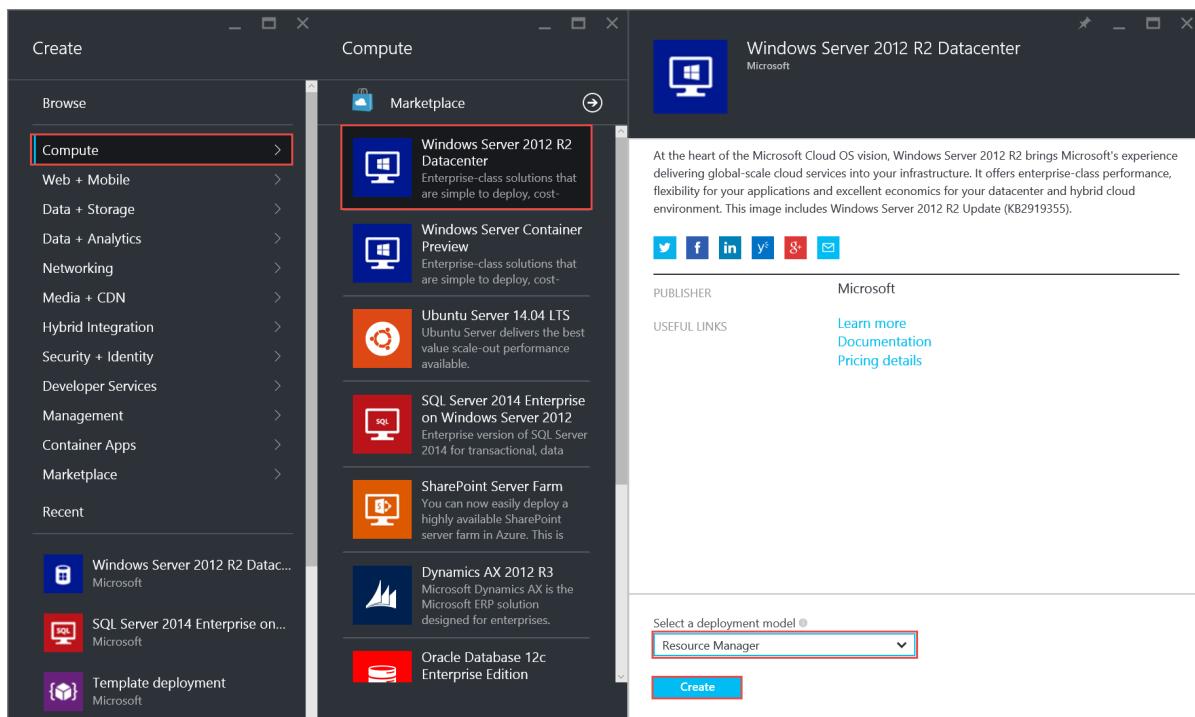
The following sample steps expect a simple environment already created. If you want to run the steps as they are displayed in this document, first build the test environment described in [Create a virtual network](#).

## How to create a VM for testing static private IP addresses

You cannot set a static private IP address during the creation of a VM in the Resource Manager deployment mode by using the Azure portal. You must create the VM first, then set its private IP to be static.

To create a VM named **DNS01** in the **FrontEnd** subnet of a VNet named **TestVNet**, follow these steps:

- From a browser, navigate to <https://portal.azure.com> and, if necessary, sign in with your Azure account.
- Click **Create a resource > Compute > Windows Server 2012 R2 Datacenter**, notice that the **Select a deployment model** list already shows **Resource Manager**, and then click **Create**, as seen in the following figure.



- In the **Basics** pane, enter the name of the VM to create (*DNS01* in the scenario), the local administrator account, and password, as seen in the following figure.

<b>Name</b>	DNS01
<b>User name</b>	adminuser
<b>Password</b>	*****
<b>Subscription</b>	Microsoft Azure Internal Consumption
<b>Resource group</b>	Select existing
<b>Location</b>	Central US

- Make sure the **Location** selected is *Central US*, then click **Select existing** under **Resource group**, then click **Resource group** again, then click *TestRG*, and then click **OK**.

Basics

\* Name  
DNS01

\* User name  
adminuser

\* Password  
\*\*\*\*\*

\* Subscription  
Microsoft Azure Internal Consumption

\* Resource group  
TestRG

Create new

\* Location  
Central US

OK

The screenshot shows the 'Basics' configuration step of a virtual machine creation wizard. It includes fields for Name (DNS01), User name (adminuser), Password (redacted), Subscription (Microsoft Azure Internal Consumption), Resource group (TestRG, highlighted with a red box), Location (Central US), and an 'OK' button at the bottom.

5. In the **Choose a size** pane, select **A1 Standard**, and then click **Select**.

Create virtual machine

1 Basics Done ✓

2 Size Choose virtual machine size >

3 Settings Configure optional features >

4 Summary Windows Server 2012 R2 Data... >

Choose a size

Browse the available sizes and their features

Prices presented below are estimated retail prices that include both Azure infrastructure and applicable third-party software costs. Prices do not reflect applicable discounts for your subscription and may include currency conversions.

★ Recommended | [View all](#)

D1 Standard	A1 Standard
1 Core	1 Core
3.5 GB	1.75 GB
2 Data disks	2 Data disks
2x500 Max IOPS	2x500 Max IOPS
50 GB Local SSD	Load balancing
Load balancing	Auto scale
118.30 USD/MONTH (ESTIMATED)	66.96 USD/MONTH (ESTIMATED)

Select

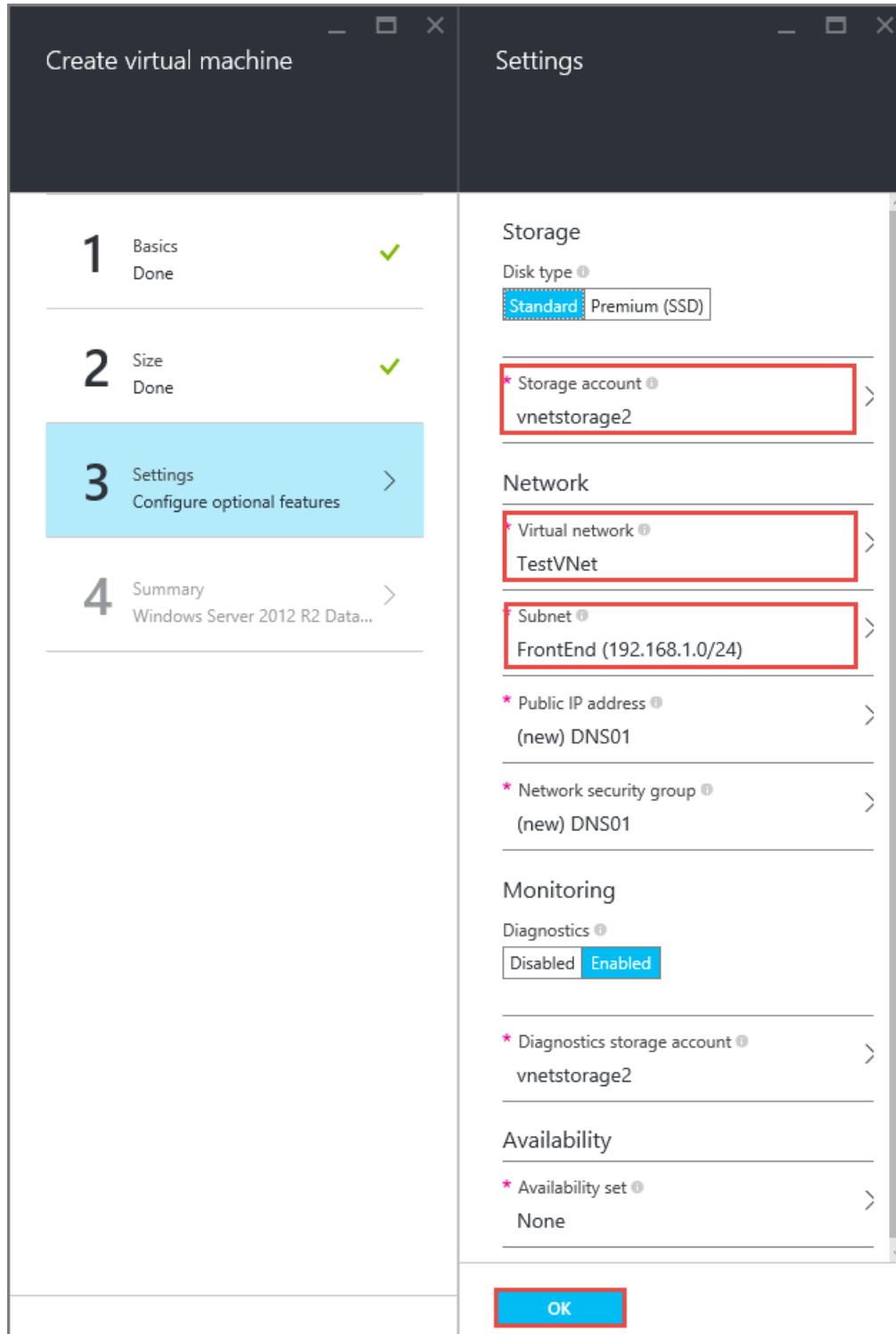
The screenshot shows the 'Choose a size' pane of the Azure virtual machine creation wizard. It lists 'D1 Standard' and 'A1 Standard' sizes. The 'A1 Standard' row is highlighted with a red box. The 'Select' button is located at the bottom of the pane.

6. In the **Settings** pane, be sure the properties are set with the following values, and then click **OK**.

-**Storage account:** vnetstorage

• **Network:** TestVNet

• **Subnet:** FrontEnd



7. In the **Summary** pane, click **OK**. Notice the following tile displayed in your dashboard.



It's recommended that you do not statically assign the private IP assigned to the Azure virtual machine within the operating system of a VM, unless necessary, such as when [assigning multiple IP addresses to a Windows VM](#). If you do manually set the private IP address within the operating system, ensure that it is the same address as the private IP address assigned to the Azure [network interface](#), or you can lose connectivity to the virtual machine. Learn more about [private IP address settings](#). You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's operating system.

## How to retrieve static private IP address information for a VM

To view the static private IP address information for the VM created with the steps above, execute the following steps.

1. From the Azure portal, click **BROWSE ALL > Virtual machines > DNS01 > All settings > Network interfaces** and then click on the only network interface listed.

A screenshot of the Azure portal showing the "Network interfaces" page for a VM named "DNS01". The table lists one network interface, "dns01995", with its details: Public IP Address is 40.122.203.66 and Private IP Address is 192.168.1.6. The entire row for "dns01995" is highlighted with a red box.

2. In the **Network interface** pane, click **All settings > IP addresses** and notice the **Assignment** and **IP address** values.

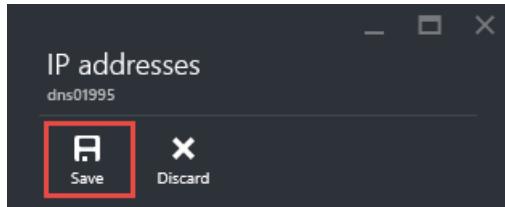
A screenshot of the Azure portal showing the "dns01995" network interface settings. The "IP addresses" section is open, showing the "Public IP address settings" and "Private IP address settings". The "Public IP address" is listed as "Enabled" (40.122.203.66). The "Private IP address" is listed as "192.168.1.6". The "Assignment" dropdown is set to "Dynamic". A red box highlights the "IP addresses" link in the center navigation menu and the "Static" option in the "Assignment" dropdown.

## How to add a static private IP address to an existing VM

To add a static private IP address to the VM created using the steps above, follow these steps:

1. From the **IP addresses** pane shown above, click **Static** under **Assignment**.

2. Type 192.168.1.101 for **IP address**, and then click **Save**.



Public IP address settings

Public IP address

IP address

DNS01 (40.122.203.66)



Private IP address settings

Virtual network

TestVNet

Subnet

FrontEnd (192.168.1.0/24)



Assignment

\* IP address

192.168.1.101



### NOTE

If after clicking **Save**, you notice that the assignment is still set to **Dynamic**, it means the IP address you typed is already in use. Try a different IP address.

It's recommended that you do not statically assign the private IP assigned to the Azure virtual machine within the operating system of a VM, unless necessary, such as when [assigning multiple IP addresses to a Windows VM](#). If you do manually set the private IP address within the operating system, ensure that it is the same address as the private IP address assigned to the Azure [network interface](#), or you can lose connectivity to the virtual machine. Learn more about [private IP address](#) settings. You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's operating system.

## How to remove a static private IP address from a VM

To remove the static private IP address from the VM created above, complete the following step:

From the **IP addresses** pane shown above, click **Dynamic** under **Assignment**, and then click **Save**.

## Set IP addresses within the operating system

It's recommended that you do not statically assign the private IP assigned to the Azure virtual machine within the operating system of a VM, unless necessary, such as when [assigning multiple IP addresses to a Windows VM](#). If you do manually set the private IP address within the operating system, ensure that it is the same address as the private IP address assigned to the Azure [network interface](#), or you can lose connectivity to the virtual machine. Learn more about [private IP address](#) settings. You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's operating system.

## Next steps

Learn about managing [IP address settings](#).

# Create a virtual machine with a static private IP address using PowerShell

11/19/2019 • 3 minutes to read • [Edit Online](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

You can create a virtual machine (VM) with a static private IP address. Assign a static private IP address, rather than a dynamic address, if you want to select which address from a subnet is assigned to a VM. Learn more about [static private IP addresses](#). To change a private IP address assigned to an existing VM from dynamic to static, or to work with public IP addresses, see [Add, change, or remove IP addresses](#).

## Create a virtual machine

You can complete the following steps from your local computer or by using the Azure Cloud Shell. To use your local computer, ensure you have the [Azure PowerShell installed](#). To use the Azure Cloud Shell, select **Try It** in the top right corner of any command box that follows. The Cloud Shell signs you into Azure.

1. If using the Cloud Shell, skip to step 2. Open a command session and sign into Azure with [Connect-AzAccount](#).
2. Create a resource group with the [New-AzResourceGroup](#) command. The following example creates a resource group in the East US Azure region:

```
$RgName = "myResourceGroup"  
$Location = "eastus"  
New-AzResourceGroup -Name $RgName -Location $Location
```

3. Create a subnet configuration and virtual network with the [New-AzVirtualNetworkSubnetConfig](#) and [New-AzVirtualNetwork](#) commands:

```
# Create a subnet configuration  
$SubnetConfig = New-AzVirtualNetworkSubnetConfig `  
-Name MySubnet `  
-AddressPrefix 10.0.0.0/24  
  
# Create a virtual network  
$VNet = New-AzVirtualNetwork `  
-ResourceGroupName $RgName `  
-Location $Location `  
-Name MyVNet `  
-AddressPrefix 10.0.0.0/16 `  
-Subnet $subnetConfig  
  
# Get the subnet object for use in a later step.  
$Subnet = Get-AzVirtualNetworkSubnetConfig -Name $SubnetConfig.Name -VirtualNetwork $VNet
```

4. Create a network interface in the virtual network and assign a private IP address from the subnet to the network interface with the [New-AzNetworkInterfaceIpConfig](#) and [New-AzNetworkInterface](#) commands:

```
$IpConfigName1 = "IPConfig-1"
$IpConfig1      = New-AzNetworkInterfaceIpConfig ` 
    -Name $IpConfigName1 ` 
    -Subnet $Subnet ` 
    -PrivateIpAddress 10.0.0.4 ` 
    -Primary

$NIC = New-AzNetworkInterface ` 
    -Name MyNIC ` 
    -ResourceGroupName $RgName ` 
    -Location $Location ` 
    -IpConfiguration $IpConfig1
```

5. Create a VM configuration with [New-AzVMConfig](#), and then create the VM with [New-AzVM](#). When prompted, provide a username and password to be used as the sign in credentials for the VM:

```
$VirtualMachine = New-AzVMConfig -VMName MyVM -VMSize "Standard_DS3"
$VirtualMachine = Set-AzVMOperatingSystem -VM $VirtualMachine -Windows -ComputerName MyServerVM - 
ProvisionVMAgent -EnableAutoUpdate
$VirtualMachine = Add-AzVMNetworkInterface -VM $VirtualMachine -Id $NIC.Id
$VirtualMachine = Set-AzVMSourceImage -VM $VirtualMachine -PublisherName 'MicrosoftWindowsServer' -Offer 
'WindowsServer' -Skus '2012-R2-Datacenter' -Version latest
New-AzVM -ResourceGroupName $RgName -Location $Location -VM $VirtualMachine -Verbose
```

#### WARNING

Though you can add private IP address settings to the operating system, we recommend not doing so until after reading [Add a private IP address to an operating system](#).

#### IMPORTANT

To access the VM from the internet, you must assign a public IP address to the VM. You can also change a dynamic private IP address assignment to a static assignment. For details, see [Add or change IP addresses](#). Additionally, it's recommended that you limit the network traffic to your VM by associating a network security group to the network interface, the subnet you created the network interface in, or both. For details, see [Manage network security groups](#).

## Clean up resources

When no longer needed, you can use [Remove-AzResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzResourceGroup -Name myResourceGroup -Force
```

## Next steps

- Learn more about [private IP addresses](#) and assigning a [static private IP address](#) to an Azure virtual machine.
- Learn more about creating [Linux](#) and [Windows](#) virtual machines.

# Configure private IP addresses for a virtual machine using the Azure CLI

11/22/2019 • 5 minutes to read • [Edit Online](#)

A virtual machine (VM) is automatically assigned a private IP address from a range that you specify, based on the subnet it is deployed in. The address is retained by a VM until the VM is deleted. Azure dynamically assigns the next available private IP address from the subnet you create a VM in. Assign a static IP address if you want a specific IP address from the subnet assigned to the VM.

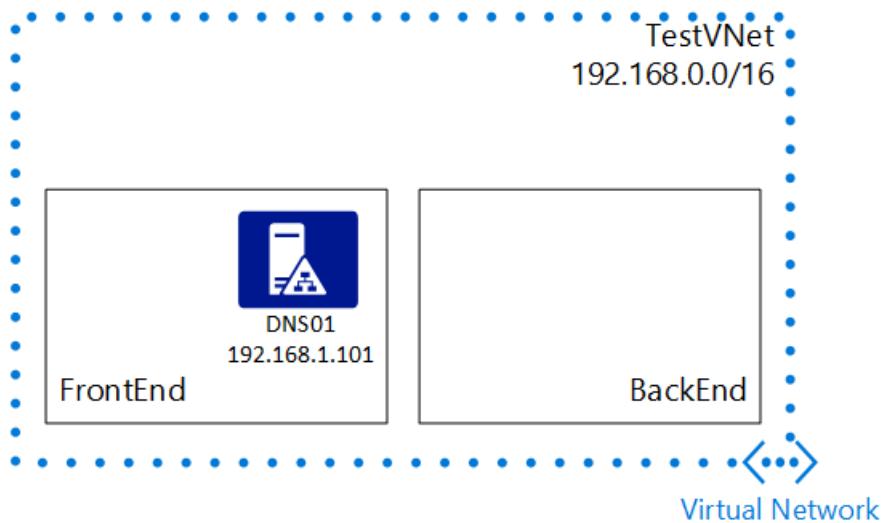
## IMPORTANT

Before you work with Azure resources, it's important to understand that Azure currently has two deployment models: Azure Resource Manager and classic. Make sure you understand [deployment models and tools](#) before you work with any Azure resource. You can view the documentation for different tools by clicking the tabs at the top of this article.

This article covers the Resource Manager deployment model. You can also [manage static private IP address in the classic deployment model](#).

## Scenario

To better illustrate how to configure a static IP address for a VM, this document will use the scenario below.



In this scenario you will create a VM named **DNS01** in the **FrontEnd** subnet, and set it to use a static IP address of **192.168.1.101**.

## NOTE

The following sample Azure CLI commands expect an existing simple environment. If you want to run the commands as they are displayed in this document, first build the test environment described in [create a vnet](#).

## Specify a static private IP address when creating a VM

To create a VM named **DNS01** in the **FrontEnd** subnet of a VNet named **TestVNet** with a static private IP of **192.168.1.101**, complete the following steps:

1. If you haven't yet, install and configure the latest [Azure CLI](#) and log in to an Azure account using [az login](#).
2. Create a public IP for the VM with the [az network public-ip create](#) command. The list shown after the output explains the parameters used.

**NOTE**

You may want or need to use different values for your arguments in this and subsequent steps, depending upon your environment.

```
az network public-ip create \
--name TestPIP \
--resource-group TestRG \
--location centralus \
--allocation-method Static
```

Expected output:

```
{
    "publicIp": {
        "idleTimeoutInMinutes": 4,
        "ipAddress": "52.176.43.167",
        "provisioningState": "Succeeded",
        "publicIPAllocationMethod": "Static",
        "resourceGuid": "79e8baa3-33ce-466a-846c-37af3c721ce1"
    }
}
```

- `--resource-group` : Name of the resource group in which to create the public IP.
- `--name` : Name of the public IP.
- `--location` : Azure region in which to create the public IP.

3. Run the [az network nic create](#) command to create a NIC with a static private IP. The list shown after the output explains the parameters used.

```
az network nic create \
--resource-group TestRG \
--name TestNIC \
--location centralus \
--subnet FrontEnd \
--private-ip-address 192.168.1.101 \
--vnet-name TestVNet
```

Expected output:

```
{
  "newNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": []
    },
    "enableIPForwarding": false,
    "ipConfigurations": [
      {
        "etag": "W/\"<guid>\",
        "id": "/subscriptions/<guid>/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC/ipConfigurations/ipconfig1",
        "name": "ipconfig1",
        "properties": {
          "primary": true,
          "privateIPAddress": "192.168.1.101",
          "privateIPAllocationMethod": "Static",
          "provisioningState": "Succeeded",
          "subnet": {
            "id": "/subscriptions/<guid>/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd",
            "resourceGroup": "TestRG"
          }
        },
        "resourceGroup": "TestRG"
      }
    ],
    "provisioningState": "Succeeded",
    "resourceGuid": "<guid>"
  }
}
```

#### Parameters:

- `--private-ip-address` : Static private IP address for the NIC.
- `--vnet-name` : Name of the VNet in which to create the NIC.
- `--subnet` : Name of the subnet in which to create the NIC.

4. Run the `azure vm create` command to create the VM using the public IP and NIC created previously. The list shown after the output explains the parameters used.

```
az vm create \
--resource-group TestRG \
--name DNS01 \
--location centralus \
--image Debian \
--admin-username adminuser \
--ssh-key-value ~/.ssh/id_rsa.pub \
--nics TestNIC
```

#### Expected output:

```
{  
    "fqdns": "",  
    "id":  
        "/subscriptions/<guid>/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/DNS01",  
    "location": "centralus",  
    "macAddress": "00-0D-3A-92-C1-66",  
    "powerState": "VM running",  
    "privateIpAddress": "192.168.1.101",  
    "publicIpAddress": "",  
    "resourceGroup": "TestRG"  
}
```

Parameters other than the basic `az vm create` parameters.

- `--nics`: Name of the NIC to which the VM is attached.

It's recommended that you do not statically assign the private IP assigned to the Azure virtual machine within the operating system of a VM, unless necessary, such as when [assigning multiple IP addresses to a Windows VM](#). If you do manually set the private IP address within the operating system, ensure that it is the same address as the private IP address assigned to the Azure [network interface](#), or you can lose connectivity to the virtual machine. Learn more about [private IP address](#) settings.

## Retrieve static private IP address information for a VM

Run the following Azure CLI command to observe the values for *Private IP alloc-method* and *Private IP address*:

```
az vm show -g TestRG -n DNS01 --show-details --query 'privateIps'
```

Expected output:

```
"192.168.1.101"
```

To display the specific IP information of the NIC for that VM, query the NIC specifically:

```
az network nic show \  
-g testrg \  
-n testnic \  
--query 'ipConfigurations[0].{PrivateAddress:privateIpAddress,IPVer:privateIpAddressVersion,IpAllocMethod:p  
rivateIpAllocationMethod,PublicAddress:publicIpAddress}'
```

The output is something like:

```
{  
    "IPVer": "IPv4",  
    "IpAllocMethod": "Static",  
    "PrivateAddress": "192.168.1.101",  
    "PublicAddress": null  
}
```

## Remove a static private IP address from a VM

You cannot remove a static private IP address from a NIC in Azure CLI for Azure Resource Manager deployments. You must:

- Create a new NIC that uses a dynamic IP

- Set the NIC on the VM do the newly created NIC.

To change the NIC for the VM used in the previous commands, complete the following steps:

1. Run the **azure network nic create** command to create a new NIC using dynamic IP allocation with a new IP address. Because no IP address is specified, the allocation method is **Dynamic**.

```
az network nic create    \
--resource-group TestRG    \
--name TestNIC2    \
--location centralus    \
--subnet FrontEnd    \
--vnet-name TestVNet
```

Expected output:

```
{
  "newNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": []
    },
    "enableIPForwarding": false,
    "ipConfigurations": [
      {
        "etag": "W/\\"<guid>\\"", 
        "id": "/subscriptions/<guid>/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC2/ipConfigurations/ipconfig1",
        "name": "ipconfig1",
        "properties": {
          "primary": true,
          "privateIPAddress": "192.168.1.4",
          "privateIPAllocationMethod": "Dynamic",
          "provisioningState": "Succeeded",
          "subnet": {
            "id": "/subscriptions/<guid>/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd",
            "resourceGroup": "TestRG"
          }
        },
        "resourceGroup": "TestRG"
      }
    ],
    "provisioningState": "Succeeded",
    "resourceGuid": "0808a61c-476f-4d08-98ee-0fa83671b010"
  }
}
```

2. Run the **azure vm set** command to change the NIC used by the VM.

```
az vm nic set --resource-group TestRG --vm-name DNS01 --nics TestNIC2
```

Expected output:

```
[  
  {  
    "id": "/subscriptions/0e220bf6-5caa-4e9f-8383-  
51f16b6c109f/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC3",  
    "primary": true,  
    "resourceGroup": "TestRG"  
  }  
]
```

**NOTE**

If the VM is large enough to have more than one NIC, run the **azure network nic delete** command to delete the old NIC.

## Next steps

Learn about managing [IP address settings](#).

# Assign multiple IP addresses to virtual machines using the Azure portal

1/3/2020 • 14 minutes to read • [Edit Online](#)

An Azure Virtual Machine (VM) has one or more network interfaces (NIC) attached to it. Any NIC can have one or more static or dynamic public and private IP addresses assigned to it. Assigning multiple IP addresses to a VM enables the following capabilities:

- Hosting multiple websites or services with different IP addresses and SSL certificates on a single server.
- Serve as a network virtual appliance, such as a firewall or load balancer.
- The ability to add any of the private IP addresses for any of the NICs to an Azure Load Balancer back-end pool. In the past, only the primary IP address for the primary NIC could be added to a back-end pool. To learn more about how to load balance multiple IP configurations, read the [Load balancing multiple IP configurations](#) article.

Every NIC attached to a VM has one or more IP configurations associated to it. Each configuration is assigned one static or dynamic private IP address. Each configuration may also have one public IP address resource associated to it. A public IP address resource has either a dynamic or static public IP address assigned to it. To learn more about IP addresses in Azure, read the [IP addresses in Azure](#) article.

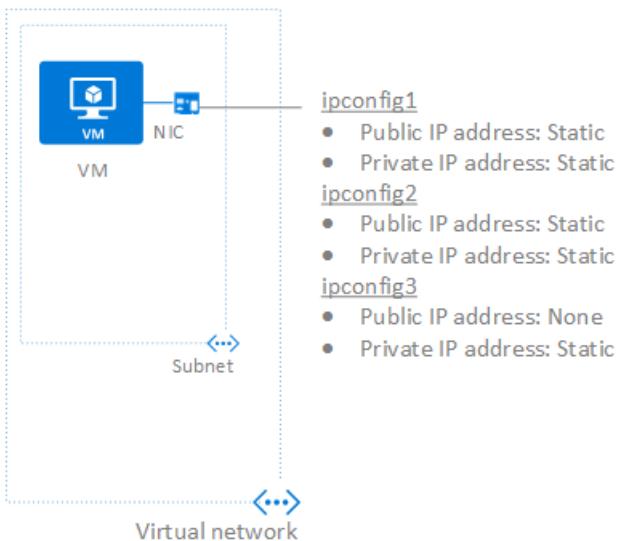
There is a limit to how many private IP addresses can be assigned to a NIC. There is also a limit to how many public IP addresses that can be used in an Azure subscription. See the [Azure limits](#) article for details.

This article explains how to create a virtual machine (VM) through the Azure Resource Manager deployment model using the Azure portal. Multiple IP addresses cannot be assigned to resources created through the classic deployment model. To learn more about Azure deployment models, read the [Understand deployment models](#) article.

## Scenario

A VM with a single NIC is created and connected to a virtual network. The VM requires three different *private* IP addresses and two *public* IP addresses. The IP addresses are assigned to the following IP configurations:

- **IPConfig-1:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-2:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-3:** Assigns a *static* private IP address and no public IP address.



The IP configurations are associated to the NIC when the NIC is created and the NIC is attached to the VM when the VM is created. The types of IP addresses used for the scenario are for illustration. You can assign whatever IP address and assignment types you require.

#### NOTE

Though the steps in this article assigns all IP configurations to a single NIC, you can also assign multiple IP configurations to any NIC in a multi-NIC VM. To learn how to create a VM with multiple NICs, read the [Create a VM with multiple NICs](#) article.

## Create a VM with multiple IP addresses

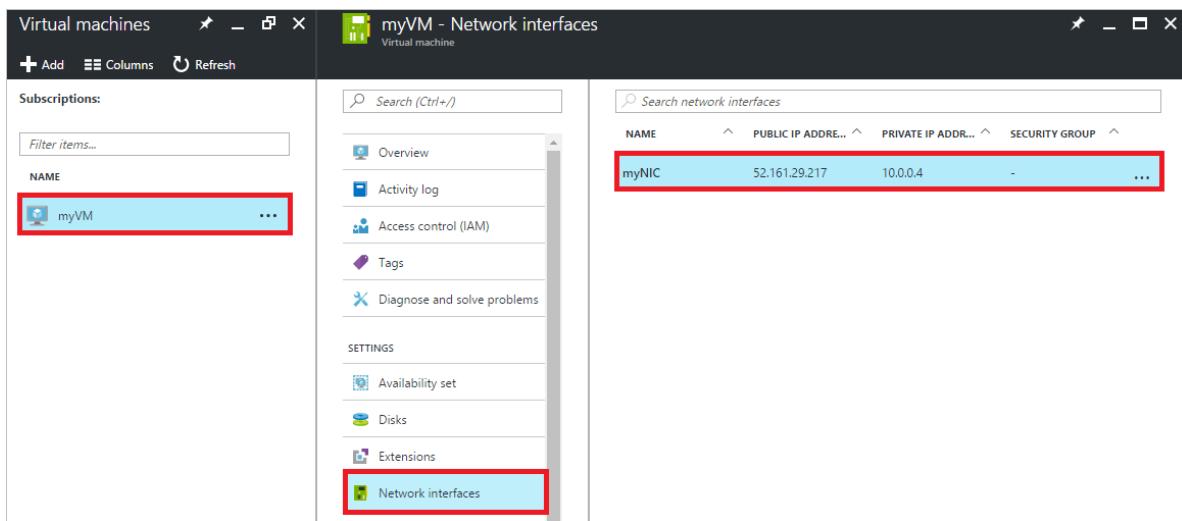
If you want to create a VM with multiple IP addresses, or a static private IP address, you must create it using PowerShell or the Azure CLI. To learn how, click the PowerShell or CLI options at the top of this article. You can create a VM with a single dynamic private IP address and (optionally) a single public IP address. Use the portal by following the steps in the [Create a Windows VM](#) or [Create a Linux VM](#) articles. After you create the VM, you can change the IP address type from dynamic to static and add additional IP addresses using the portal by following steps in the [Add IP addresses to a VM](#) section of this article.

## Add IP addresses to a VM

You can add private and public IP addresses to an Azure network interface by completing the steps that follow. The examples in the following sections assume that you already have a VM with the three IP configurations described in the [scenario](#), but it's not required.

### Core steps

1. Browse to the Azure portal at <https://portal.azure.com> and sign into it, if necessary.
2. In the portal, click **More services** > type *virtual machines* in the filter box, and then click **Virtual machines**.
3. In the **Virtual machines** pane, click the VM you want to add IP addresses to. Click **Network interfaces** in the virtual machine pane that appears, and then select the network interface you want to add the IP addresses to. In the example shown in the following picture, the NIC named *myNIC* from the VM named *myVM* is selected:



4. In the pane that appears for the NIC you selected, click **IP configurations**.

Complete the steps in one of the sections that follow, based on the type of IP address you want to add.

### Add a private IP address

Complete the following steps to add a new private IP address:

1. Complete the steps in the [Core steps](#) section of this article.
2. Click **Add**. In the **Add IP configuration** pane that appears, create an IP configuration named *IPConfig-4* with *10.0.0.7* as a *Static* private IP address, then click **OK**.

#### NOTE

When adding a static IP address, you must specify an unused, valid address on the subnet the NIC is connected to. If the address you select is not available, the portal displays an X for the IP address and you must select a different one.

3. Once you click OK, the pane closes and you see the new IP configuration listed. Click **OK** to close the **Add IP configuration** pane.
4. You can click **Add** to add additional IP configurations, or close all open blades to finish adding IP addresses.
5. Add the private IP addresses to the VM operating system by completing the steps in the [Add IP addresses to a VM operating system](#) section of this article.

### Add a public IP address

A public IP address is added by associating a public IP address resource to either a new IP configuration or an existing IP configuration.

#### NOTE

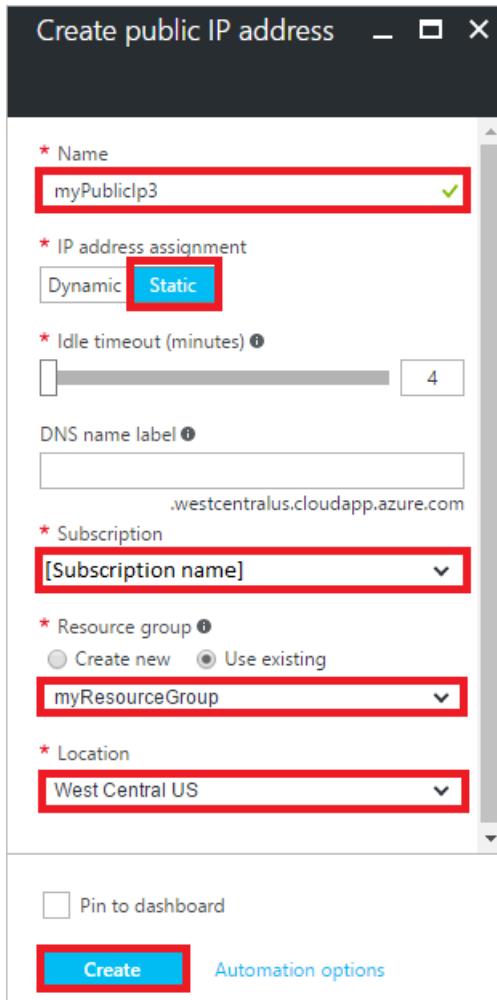
Public IP addresses have a nominal fee. To learn more about IP address pricing, read the [IP address pricing](#) page. There is a limit to the number of public IP addresses that can be used in a subscription. To learn more about the limits, read the [Azure limits](#) article.

### Create a public IP address resource

A public IP address is one setting for a public IP address resource. If you have a public IP address resource that is not currently associated to an IP configuration that you want to associate to an IP configuration, skip the following steps and complete the steps in one of the sections that follow, as you require. If you don't have an

available public IP address resource, complete the following steps to create one:

1. Browse to the Azure portal at <https://portal.azure.com> and sign into it, if necessary.
2. In the portal, click **Create a resource > Networking > Public IP address**.
3. In the **Create public IP address** pane that appears, enter a **Name**, select an **IP address assignment** type, a **Subscription**, a **Resource group**, and a **Location**, then click **Create**, as shown in the following picture:



4. Complete the steps in one of the sections that follow to associate the public IP address resource to an IP configuration.

#### Associate the public IP address resource to a new IP configuration

1. Complete the steps in the [Core steps](#) section of this article.
2. Click **Add**. In the **Add IP configuration** pane that appears, create an IP configuration named *IPConfig-4*. Enable the **Public IP address** and select an existing, available public IP address resource from the **Choose public IP address** pane that appears.

Once you've selected the public IP address resource, click **OK** and the pane closes. If you don't have an existing public IP address, you can create one by completing the steps in the [Create a public IP address resource](#) section of this article.

3. Review the new IP configuration. Even though a private IP address wasn't explicitly assigned, one was automatically assigned to the IP configuration, because all IP configurations must have a private IP address.
4. You can click **Add** to add additional IP configurations, or close all open blades to finish adding IP addresses.

5. Add the private IP address to the VM operating system by completing the steps for your operating system in the [Add IP addresses to a VM operating system](#) section of this article. Do not add the public IP address to the operating system.

#### Associate the public IP address resource to an existing IP configuration

1. Complete the steps in the [Core steps](#) section of this article.
2. Click the IP configuration you want to add the public IP address resource to.
3. In the IPConfig pane that appears, click **IP address**.
4. In the **Choose public IP address** pane that appears, select a public IP address.
5. Click **Save** and the panes close. If you don't have an existing public IP address, you can create one by completing the steps in the [Create a public IP address resource](#) section of this article.
6. Review the new IP configuration.
7. You can click **Add** to add additional IP configurations, or close all open blades to finish adding IP addresses. Do not add the public IP address to the operating system.

## Add IP addresses to a VM operating system

Connect and sign in to a VM you created with multiple private IP addresses. You must manually add all the private IP addresses (including the primary) that you added to the VM. Complete the steps that follow for your VM operating system.

### Windows

1. From a command prompt, type `ipconfig /all`. You only see the *Primary* private IP address (through DHCP).
2. Type `ncpa.cpl` in the command prompt to open the **Network connections** window.
3. Open the properties for the appropriate adapter: **Local Area Connection**.
4. Double-click Internet Protocol version 4 (IPv4).
5. Select **Use the following IP address** and enter the following values:

- **IP address:** Enter the *Primary* private IP address
- **Subnet mask:** Set based on your subnet. For example, if the subnet is a /24 subnet then the subnet mask is 255.255.255.0.
- **Default gateway:** The first IP address in the subnet. If your subnet is 10.0.0.0/24, then the gateway IP address is 10.0.0.1.
- Select **Use the following DNS server addresses** and enter the following values:
  - **Preferred DNS server:** If you are not using your own DNS server, enter 168.63.129.16. If you are using your own DNS server, enter the IP address for your server.
  - Select the **Advanced** button and add additional IP addresses. Add each of the secondary private IP addresses, that you added to the Azure network interface in a previous step, to the Windows network interface that is assigned the primary IP address assigned to the Azure network interface.

You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's operating system. When you manually set the IP address within the operating system, ensure that it is the same address as the private IP address assigned to the Azure [network interface](#), or you can lose connectivity to the virtual machine. Learn more about [private IP address](#) settings. You should never assign an Azure public IP address within the operating system.

- Click **OK** to close out the TCP/IP settings and then **OK** again to close the adapter settings. Your RDP connection is re-established.

6. From a command prompt, type `ipconfig /all`. All IP addresses you added are shown and DHCP is turned off.
7. Configure Windows to use the private IP address of the primary IP configuration in Azure as the primary IP address for Windows. See [No Internet access from Azure Windows VM that has multiple IP addresses](#) for details.

### Validation (Windows)

To ensure you are able to connect to the internet from your secondary IP configuration via the public IP associated with it, once you have added it correctly using steps above, use the following command:

```
ping -S 10.0.0.5 hotmail.com
```

#### NOTE

For secondary IP configurations, you can only ping to the Internet if the configuration has a public IP address associated with it. For primary IP configurations, a public IP address is not required to ping to the Internet.

### Linux (Ubuntu 14/16)

We recommend looking at the latest documentation for your Linux distribution.

1. Open a terminal window.
2. Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

3. Update the configuration file of the network interface (assuming 'eth0').

- Keep the existing line item for dhcp. The primary IP address remains configured as it was previously.
- Add a configuration for an additional static IP address with the following commands:

```
cd /etc/network/interfaces.d/  
ls
```

You should see a .cfg file.

4. Open the file. You should see the following lines at the end of the file:

```
auto eth0  
iface eth0 inet dhcp
```

5. Add the following lines after the lines that exist in this file:

```
iface eth0 inet static  
address <your private IP address here>  
netmask <your subnet mask>
```

6. Save the file by using the following command:

```
:wq
```

7. Reset the network interface with the following command:

```
sudo ifdown eth0 && sudo ifup eth0
```

**IMPORTANT**

Run both ifdown and ifup in the same line if using a remote connection.

8. Verify the IP address is added to the network interface with the following command:

```
ip addr list eth0
```

You should see the IP address you added as part of the list.

**Linux (Ubuntu 18.04+)**

Ubuntu 18.04 and above have changed to `netplan` for OS network management. We recommend looking at the latest documentation for your Linux distribution.

1. Open a terminal window.
2. Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

3. Create a file for the second interface and open it in a text editor:

```
vi /etc/netplan/60-static.yaml
```

4. Add the following lines to the file, replacing `10.0.0.6/24` with your IP/netmask:

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 10.0.0.6/24
```

5. Save the file by using the following command:

```
:wq
```

6. Test the changes using `netplan try` to confirm syntax:

```
netplan try
```

**NOTE**

`netplan try` will apply the changes temporarily and roll the changes back after 120 seconds. If there is a loss of connectivity, please wait 120 seconds, and then reconnect. At that time, the changes will have been rolled back.

7. Assuming no issues with `netplan try`, apply the configuration changes:

```
netplan apply
```

8. Verify the IP address is added to the network interface with the following command:

```
ip addr list eth0
```

You should see the IP address you added as part of the list. Example:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0d:3a:8c:14:a5 brd ff:ff:ff:ff:ff:ff
        inet 10.0.0.6/24 brd 10.0.0.255 scope global eth0
            valid_lft forever preferred_lft forever
        inet 10.0.0.4/24 brd 10.0.0.255 scope global secondary eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::20d:3aff:fe8c:14a5/64 scope link
            valid_lft forever preferred_lft forever
```

### Linux (Red Hat, CentOS, and others)

1. Open a terminal window.

2. Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

3. Enter your password and follow instructions as prompted. Once you are the root user, navigate to the network scripts folder with the following command:

```
cd /etc/sysconfig/network-scripts
```

4. List the related ifcfg files using the following command:

```
ls ifcfg-*
```

You should see `ifcfg-eth0` as one of the files.

5. To add an IP address, create a configuration file for it as shown below. Note that one file must be created for each IP configuration.

```
touch ifcfg-eth0:0
```

6. Open the `ifcfg-eth0:0` file with the following command:

```
vi ifcfg-eth0:0
```

7. Add content to the file, *eth0:0* in this case, with the following command. Be sure to update information based on your IP address.

```
DEVICE=eth0:0
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.101.101
NETMASK=255.255.255.0
```

8. Save the file with the following command:

```
:wq
```

9. Restart the network services and make sure the changes are successful by running the following commands:

```
/etc/init.d/network restart
ifconfig
```

You should see the IP address you added, *eth0:0*, in the list returned.

### Validation (Linux)

To ensure you are able to connect to the internet from your secondary IP configuration via the public IP associated with it, use the following command:

```
ping -I 10.0.0.5 hotmail.com
```

#### NOTE

For secondary IP configurations, you can only ping to the Internet if the configuration has a public IP address associated with it. For primary IP configurations, a public IP address is not required to ping to the Internet.

For Linux VMs, when trying to validate outbound connectivity from a secondary NIC, you may need to add appropriate routes. There are many ways to do this. Please see appropriate documentation for your Linux distribution. The following is one method to accomplish this:

```
echo 150 custom >> /etc/iproute2/rt_tables
ip rule add from 10.0.0.5 lookup custom
ip route add default via 10.0.0.1 dev eth2 table custom
```

- Be sure to replace:
  - **10.0.0.5** with the private IP address that has a public IP address associated with it
  - **10.0.0.1** to your default gateway
  - **eth2** to the name of your secondary NIC

# Assign multiple IP addresses to virtual machines using PowerShell

1/3/2020 • 18 minutes to read • [Edit Online](#)

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

An Azure Virtual Machine (VM) has one or more network interfaces (NIC) attached to it. Any NIC can have one or more static or dynamic public and private IP addresses assigned to it. Assigning multiple IP addresses to a VM enables the following capabilities:

- Hosting multiple websites or services with different IP addresses and SSL certificates on a single server.
- Serve as a network virtual appliance, such as a firewall or load balancer.
- The ability to add any of the private IP addresses for any of the NICs to an Azure Load Balancer back-end pool. In the past, only the primary IP address for the primary NIC could be added to a back-end pool. To learn more about how to load balance multiple IP configurations, read the [Load balancing multiple IP configurations](#) article.

Every NIC attached to a VM has one or more IP configurations associated to it. Each configuration is assigned one static or dynamic private IP address. Each configuration may also have one public IP address resource associated to it. A public IP address resource has either a dynamic or static public IP address assigned to it. To learn more about IP addresses in Azure, read the [IP addresses in Azure](#) article.

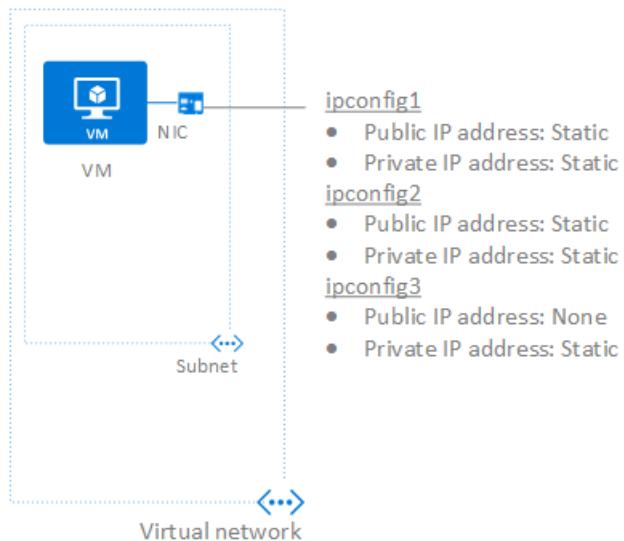
There is a limit to how many private IP addresses can be assigned to a NIC. There is also a limit to how many public IP addresses that can be used in an Azure subscription. See the [Azure limits](#) article for details.

This article explains how to create a virtual machine (VM) through the Azure Resource Manager deployment model using PowerShell. Multiple IP addresses cannot be assigned to resources created through the classic deployment model. To learn more about Azure deployment models, read the [Understand deployment models](#) article.

## Scenario

A VM with a single NIC is created and connected to a virtual network. The VM requires three different *private* IP addresses and two *public* IP addresses. The IP addresses are assigned to the following IP configurations:

- **IPConfig-1:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-2:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-3:** Assigns a *static* private IP address and no public IP address.



The IP configurations are associated to the NIC when the NIC is created and the NIC is attached to the VM when the VM is created. The types of IP addresses used for the scenario are for illustration. You can assign whatever IP address and assignment types you require.

#### NOTE

Though the steps in this article assigns all IP configurations to a single NIC, you can also assign multiple IP configurations to any NIC in a multi-NIC VM. To learn how to create a VM with multiple NICs, read the [Create a VM with multiple NICs](#) article.

## Create a VM with multiple IP addresses

The steps that follow explain how to create an example VM with multiple IP addresses, as described in the scenario. Change variable values as required for your implementation.

1. Open a PowerShell command prompt and complete the remaining steps in this section within a single PowerShell session. If you don't already have PowerShell installed and configured, complete the steps in the [How to install and configure Azure PowerShell](#) article.
2. Login to your account with the `Connect-AzAccount` command.
3. Replace *myResourceGroup* and *westus* with a name and location of your choosing. Create a resource group. A resource group is a logical container into which Azure resources are deployed and managed.

```
$RgName    = "MyResourceGroup"
$Location = "westus"

New-AzResourceGroup ` 
-Name $RgName ` 
-Location $Location
```

4. Create a virtual network (VNet) and subnet in the same location as the resource group:

```

# Create a subnet configuration
$SubnetConfig = New-AzVirtualNetworkSubnetConfig ` 
-Name MySubnet ` 
-AddressPrefix 10.0.0.0/24

# Create a virtual network
$VNet = New-AzVirtualNetwork ` 
-ResourceGroupName $RgName ` 
-Location $Location ` 
-Name MyVNet ` 
-AddressPrefix 10.0.0.0/16 ` 
-Subnet $subnetConfig

# Get the subnet object
$Subnet = Get-AzVirtualNetworkSubnetConfig -Name $SubnetConfig.Name -VirtualNetwork $VNet

```

5. Create a network security group (NSG) and a rule. The NSG secures the VM using inbound and outbound rules. In this case, an inbound rule is created for port 3389, which allows incoming remote desktop connections.

```

# Create an inbound network security group rule for port 3389

$NSGRule = New-AzNetworkSecurityRuleConfig ` 
-Name MyNsgRuleRDP ` 
-Protocol Tcp ` 
-Direction Inbound ` 
-Priority 1000 ` 
-SourceAddressPrefix * ` 
-SourcePortRange * ` 
-DestinationAddressPrefix * ` 
-DestinationPortRange 3389 -Access Allow

# Create a network security group
$NSG = New-AzNetworkSecurityGroup ` 
-ResourceGroupName $RgName ` 
-Location $Location ` 
-Name MyNetworkSecurityGroup ` 
-SecurityRules $NSGRule

```

6. Define the primary IP configuration for the NIC. Change 10.0.0.4 to a valid address in the subnet you created, if you didn't use the value defined previously. Before assigning a static IP address, it's recommended that you first confirm it's not already in use. Enter the command

`Test-AzPrivateIPAddressAvailability -IPAddress 10.0.0.4 -VirtualNetwork $VNet`. If the address is available, the output returns *True*. If it's not available, the output returns *False* and a list of addresses that are available.

In the following commands, **Replace <replace-with-your-unique-name> with the unique DNS name to use**. The name must be unique across all public IP addresses within an Azure region. This is an optional parameter. It can be removed if you only want to connect to the VM using the public IP address.

```

# Create a public IP address
$PublicIP1 = New-AzPublicIpAddress ` 
-Name "MyPublicIP1" ` 
-ResourceGroupName $RgName ` 
-Location $Location ` 
-DomainNameLabel <replace-with-your-unique-name> ` 
-AllocationMethod Static

#Create an IP configuration with a static private IP address and assign the public IP address to it
$IpConfigName1 = "IPConfig-1"
$IpConfig1      = New-AzNetworkInterfaceIpConfig ` 
-Name $IpConfigName1 ` 
-Subnet $Subnet ` 
-PrivateIpAddress 10.0.0.4 ` 
-PublicIpAddress $PublicIP1 ` 
-Primary

```

When you assign multiple IP configurations to a NIC, one configuration must be assigned as the *-Primary*.

#### **NOTE**

Public IP addresses have a nominal fee. To learn more about IP address pricing, read the [IP address pricing](#) page. There is a limit to the number of public IP addresses that can be used in a subscription. To learn more about the limits, read the [Azure limits](#) article.

7. Define the secondary IP configurations for the NIC. You can add or remove configurations as necessary. Each IP configuration must have a private IP address assigned. Each configuration can optionally have one public IP address assigned.

```

# Create a public IP address
$PublicIP2 = New-AzPublicIpAddress ` 
-Name "MyPublicIP2" ` 
-ResourceGroupName $RgName ` 
-Location $Location ` 
-AllocationMethod Static

#Create an IP configuration with a static private IP address and assign the public IP address to it
$IpConfigName2 = "IPConfig-2"
$IpConfig2      = New-AzNetworkInterfaceIpConfig ` 
-Name $IpConfigName2 ` 
-Subnet $Subnet ` 
-PrivateIpAddress 10.0.0.5 ` 
-PublicIpAddress $PublicIP2

$IpConfigName3 = "IpConfig-3"
$IpConfig3      = New-AzNetworkInterfaceIpConfig ` 
-Name $IPConfigName3 ` 
-Subnet $Subnet ` 
-PrivateIpAddress 10.0.0.6

```

8. Create the NIC and associate the three IP configurations to it:

```

$NIC = New-AzNetworkInterface ` 
-Name MyNIC ` 
-ResourceGroupName $RgName ` 
-Location $Location ` 
-NetworkSecurityGroupId $NSG.Id ` 
-IpConfiguration $IpConfig1,$IpConfig2,$IpConfig3

```

**NOTE**

Though all configurations are assigned to one NIC in this article, you can assign multiple IP configurations to every NIC attached to the VM. To learn how to create a VM with multiple NICs, read the [Create a VM with multiple NICs](#) article.

9. Create the VM by entering the following commands:

```
# Define a credential object. When you run these commands, you're prompted to enter a username and password for the VM you're creating.  
$cred = Get-Credential  
  
# Create a virtual machine configuration  
$VmConfig = New-AzVMConfig `  
-VMName MyVM `  
-VMSize Standard_DS1_v2 | `  
Set-AzVMOperatingSystem -Windows `  
-ComputerName MyVM `  
-Credential $cred | `  
Set-AzVMSourceImage `  
-PublisherName MicrosoftWindowsServer `  
-Offer WindowsServer `  
-Skus 2016-Datacenter `  
-Version latest | `  
Add-AzVMNetworkInterface `  
-Id $NIC.Id  
  
# Create the VM  
New-AzVM `  
-ResourceGroupName $RgName `  
-Location $Location `  
-VM $VmConfig
```

10. Add the private IP addresses to the VM operating system by completing the steps for your operating system in the [Add IP addresses to a VM operating system](#) section of this article. Do not add the public IP addresses to the operating system.

## Add IP addresses to a VM

You can add private and public IP addresses to the Azure network interface by completing the steps that follow. The examples in the following sections assume that you already have a VM with the three IP configurations described in the [scenario](#) in this article, but it's not required that you do.

1. Open a PowerShell command prompt and complete the remaining steps in this section within a single PowerShell session. If you don't already have PowerShell installed and configured, complete the steps in the [How to install and configure Azure PowerShell](#) article.
2. Change the "values" of the following \$Variables to the name of the NIC you want to add IP address to and the resource group and location the NIC exists in:

```
$NicName = "MyNIC"  
$RgName = "MyResourceGroup"  
$Location = "westus"
```

If you don't know the name of the NIC you want to change, enter the following commands, then change the values of the previous variables:

```
Get-AzNetworkInterface | Format-Table Name, ResourceGroupName, Location
```

3. Create a variable and set it to the existing NIC by typing the following command:

```
$MyNIC = Get-AzNetworkInterface -Name $NicName -ResourceGroupName $RgName
```

4. In the following commands, change *MyVNet* and *MySubnet* to the names of the VNet and subnet the NIC is connected to. Enter the commands to retrieve the VNet and subnet objects the NIC is connected to:

```
$MyVNet = Get-AzVirtualnetwork -Name MyVNet -ResourceGroupName $RgName  
$Subnet = $MyVnet.Subnets | Where-Object { $_.Name -eq "MySubnet" }
```

If you don't know the VNet or subnet name the NIC is connected to, enter the following command:

```
$MyNIC.IpConfigurations
```

In the output, look for text similar to the following example output:

```
"Id":  
"/subscriptions/[Id]/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetworks/MyVNet/  
subnets/MySubnet"
```

In this output, *MyVnet* is the VNet and *MySubnet* is the subnet the NIC is connected to.

5. Complete the steps in one of the following sections, based on your requirements:

### Add a private IP address

To add a private IP address to a NIC, you must create an IP configuration. The following command creates a configuration with a static IP address of 10.0.0.7. When specifying a static IP address, it must be an unused address for the subnet. It's recommended that you first test the address to ensure it's available by entering the `Test-AzPrivateIPAddressAvailability -IPAddress 10.0.0.7 -VirtualNetwork $myVnet` command. If the IP address is available, the output returns *True*. If it's not available, the output returns *False*, and a list of addresses that are available.

```
Add-AzNetworkInterfaceIpConfig -Name IPConfig-4 -NetworkInterface `  
$MyNIC -Subnet $Subnet -PrivateIpAddress 10.0.0.7
```

Create as many configurations as you require, using unique configuration names and private IP addresses (for configurations with static IP addresses).

Add the private IP address to the VM operating system by completing the steps for your operating system in the [Add IP addresses to a VM operating system](#) section of this article.

### Add a public IP address

A public IP address is added by associating a public IP address resource to either a new IP configuration or an existing IP configuration. Complete the steps in one of the sections that follow, as you require.

## NOTE

Public IP addresses have a nominal fee. To learn more about IP address pricing, read the [IP address pricing](#) page. There is a limit to the number of public IP addresses that can be used in a subscription. To learn more about the limits, read the [Azure limits](#) article.

## Associate the public IP address resource to a new IP configuration

Whenever you add a public IP address in a new IP configuration, you must also add a private IP address, because all IP configurations must have a private IP address. You can either add an existing public IP address resource, or create a new one. To create a new one, enter the following command:

```
$myPublicIp3 = New-AzPublicIpAddress  
-Name "myPublicIp3"  
-ResourceGroupName $RgName  
-Location $Location  
-AllocationMethod Static
```

To create a new IP configuration with a static private IP address and the associated *myPublicIp3* public IP address resource, enter the following command:

```
Add-AzNetworkInterfaceIpConfig  
-Name IPConfig-4  
-NetworkInterface $myNIC  
-Subnet $Subnet  
-PrivateIpAddress 10.0.0.7  
-PublicIpAddress $myPublicIp3
```

## Associate the public IP address resource to an existing IP configuration

A public IP address resource can only be associated to an IP configuration that doesn't already have one associated. You can determine whether an IP configuration has an associated public IP address by entering the following command:

```
$MyNIC.IpConfigurations | Format-Table Name, PrivateIPAddress, PublicIPAddress, Primary
```

You see output similar to the following:

Name	PrivateIpAddress	PublicIpAddress	Primary
IPConfig-1	10.0.0.4	Microsoft.Azure.Commands.Network.Models.PSPublicIpAddress	True
IPConfig-2	10.0.0.5	Microsoft.Azure.Commands.Network.Models.PSPublicIpAddress	False
IPConfig-3	10.0.0.6		False

Since the **PublicIpAddress** column for *IPConfig-3* is blank, no public IP address resource is currently associated to it. You can add an existing public IP address resource to *IPConfig-3*, or enter the following command to create one:

```
$MyPublicIp3 = New-AzPublicIpAddress  
-Name "MyPublicIp3"  
-ResourceGroupName $RgName  
-Location $Location -AllocationMethod Static
```

Enter the following command to associate the public IP address resource to the existing IP configuration

named *IpConfig-3*:

```
Set-AzNetworkInterfaceIpConfig ` 
-Name IpConfig-3 ` 
-NetworkInterface $myNIC ` 
-Subnet $Subnet ` 
-PublicIpAddress $myPublicIp3
```

6. Set the NIC with the new IP configuration by entering the following command:

```
Set-AzNetworkInterface -NetworkInterface $MyNIC
```

7. View the private IP addresses and the public IP address resources assigned to the NIC by entering the following command:

```
$MyNIC.IpConfigurations | Format-Table Name, PrivateIPAddress, PublicIPAddress, Primary
```

8. Add the private IP address to the VM operating system by completing the steps for your operating system in the [Add IP addresses to a VM operating system](#) section of this article. Do not add the public IP address to the operating system.

## Add IP addresses to a VM operating system

Connect and sign in to a VM you created with multiple private IP addresses. You must manually add all the private IP addresses (including the primary) that you added to the VM. Complete the steps that follow for your VM operating system.

### Windows

1. From a command prompt, type *ipconfig /all*. You only see the *Primary* private IP address (through DHCP).
2. Type *ncpa.cpl* in the command prompt to open the **Network connections** window.
3. Open the properties for the appropriate adapter: **Local Area Connection**.
4. Double-click Internet Protocol version 4 (IPv4).
5. Select **Use the following IP address** and enter the following values:
  - **IP address:** Enter the *Primary* private IP address
  - **Subnet mask:** Set based on your subnet. For example, if the subnet is a /24 subnet then the subnet mask is 255.255.255.0.
  - **Default gateway:** The first IP address in the subnet. If your subnet is 10.0.0.0/24, then the gateway IP address is 10.0.0.1.
  - Select **Use the following DNS server addresses** and enter the following values:
    - **Preferred DNS server:** If you are not using your own DNS server, enter 168.63.129.16. If you are using your own DNS server, enter the IP address for your server.
    - Select the **Advanced** button and add additional IP addresses. Add each of the secondary private IP addresses, that you added to the Azure network interface in a previous step, to the Windows network interface that is assigned the primary IP address assigned to the Azure network interface.

You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's operating system. When you manually set the IP address within the operating

system, ensure that it is the same address as the private IP address assigned to the Azure [network interface](#), or you can lose connectivity to the virtual machine. Learn more about [private IP address settings](#). You should never assign an Azure public IP address within the operating system.

- Click **OK** to close out the TCP/IP settings and then **OK** again to close the adapter settings. Your RDP connection is re-established.
6. From a command prompt, type `ipconfig /all`. All IP addresses you added are shown and DHCP is turned off.
  7. Configure Windows to use the private IP address of the primary IP configuration in Azure as the primary IP address for Windows. See [No Internet access from Azure Windows VM that has multiple IP addresses](#) for details.

### Validation (Windows)

To ensure you are able to connect to the internet from your secondary IP configuration via the public IP associated with it, once you have added it correctly using steps above, use the following command:

```
ping -S 10.0.0.5 hotmail.com
```

#### NOTE

For secondary IP configurations, you can only ping to the Internet if the configuration has a public IP address associated with it. For primary IP configurations, a public IP address is not required to ping to the Internet.

### Linux (Ubuntu 14/16)

We recommend looking at the latest documentation for your Linux distribution.

1. Open a terminal window.
2. Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

3. Update the configuration file of the network interface (assuming 'eth0').
  - Keep the existing line item for dhcp. The primary IP address remains configured as it was previously.
  - Add a configuration for an additional static IP address with the following commands:

```
cd /etc/network/interfaces.d/  
ls
```

You should see a .cfg file.

4. Open the file. You should see the following lines at the end of the file:

```
auto eth0  
iface eth0 inet dhcp
```

5. Add the following lines after the lines that exist in this file:

```
iface eth0 inet static  
address <your private IP address here>  
netmask <your subnet mask>
```

- Save the file by using the following command:

```
:wq
```

- Reset the network interface with the following command:

```
sudo ifdown eth0 && sudo ifup eth0
```

**IMPORTANT**

Run both ifdown and ifup in the same line if using a remote connection.

- Verify the IP address is added to the network interface with the following command:

```
ip addr list eth0
```

You should see the IP address you added as part of the list.

### Linux (Ubuntu 18.04+)

Ubuntu 18.04 and above have changed to `netplan` for OS network management. We recommend looking at the latest documentation for your Linux distribution.

- Open a terminal window.
- Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

- Create a file for the second interface and open it in a text editor:

```
vi /etc/netplan/60-static.yaml
```

- Add the following lines to the file, replacing `10.0.0.6/24` with your IP/netmask:

```
network:  
  version: 2  
  ethernets:  
    eth0:  
      addresses:  
        - 10.0.0.6/24
```

- Save the file by using the following command:

```
:wq
```

- Test the changes using `netplan try` to confirm syntax:

```
netplan try
```

#### NOTE

`netplan try` will apply the changes temporarily and roll the changes back after 120 seconds. If there is a loss of connectivity, please wait 120 seconds, and then reconnect. At that time, the changes will have been rolled back.

- Assuming no issues with `netplan try`, apply the configuration changes:

```
netplan apply
```

- Verify the IP address is added to the network interface with the following command:

```
ip addr list eth0
```

You should see the IP address you added as part of the list. Example:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0d:3a:8c:14:a5 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.6/24 brd 10.0.0.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet 10.0.0.4/24 brd 10.0.0.255 scope global secondary eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::20d:3aff:fe8c:14a5/64 scope link
        valid_lft forever preferred_lft forever
```

#### Linux (Red Hat, CentOS, and others)

- Open a terminal window.
- Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

- Enter your password and follow instructions as prompted. Once you are the root user, navigate to the network scripts folder with the following command:

```
cd /etc/sysconfig/network-scripts
```

- List the related ifcfg files using the following command:

```
ls ifcfg-*
```

You should see `ifcfg-eth0` as one of the files.

- To add an IP address, create a configuration file for it as shown below. Note that one file must be created for each IP configuration.

```
touch ifcfg-eth0:0
```

6. Open the *ifcfg-eth0:0* file with the following command:

```
vi ifcfg-eth0:0
```

7. Add content to the file, *eth0:0* in this case, with the following command. Be sure to update information based on your IP address.

```
DEVICE=eth0:0
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.101.101
NETMASK=255.255.255.0
```

8. Save the file with the following command:

```
:wq
```

9. Restart the network services and make sure the changes are successful by running the following commands:

```
/etc/init.d/network restart
ifconfig
```

You should see the IP address you added, *eth0:0*, in the list returned.

### Validation (Linux)

To ensure you are able to connect to the internet from your secondary IP configuration via the public IP associated with it, use the following command:

```
ping -I 10.0.0.5 hotmail.com
```

#### NOTE

For secondary IP configurations, you can only ping to the Internet if the configuration has a public IP address associated with it. For primary IP configurations, a public IP address is not required to ping to the Internet.

For Linux VMs, when trying to validate outbound connectivity from a secondary NIC, you may need to add appropriate routes. There are many ways to do this. Please see appropriate documentation for your Linux distribution. The following is one method to accomplish this:

```
echo 150 custom >> /etc/iproute2/rt_tables
ip rule add from 10.0.0.5 lookup custom
ip route add default via 10.0.0.1 dev eth2 table custom
```

- Be sure to replace:
  - **10.0.0.5** with the private IP address that has a public IP address associated to it

- o **10.0.0.1** to your default gateway
- o **eth2** to the name of your secondary NIC

# Assign multiple IP addresses to virtual machines using the Azure CLI

1/3/2020 • 16 minutes to read • [Edit Online](#)

An Azure Virtual Machine (VM) has one or more network interfaces (NIC) attached to it. Any NIC can have one or more static or dynamic public and private IP addresses assigned to it. Assigning multiple IP addresses to a VM enables the following capabilities:

- Hosting multiple websites or services with different IP addresses and SSL certificates on a single server.
- Serve as a network virtual appliance, such as a firewall or load balancer.
- The ability to add any of the private IP addresses for any of the NICs to an Azure Load Balancer back-end pool. In the past, only the primary IP address for the primary NIC could be added to a back-end pool. To learn more about how to load balance multiple IP configurations, read the [Load balancing multiple IP configurations](#) article.

Every NIC attached to a VM has one or more IP configurations associated to it. Each configuration is assigned one static or dynamic private IP address. Each configuration may also have one public IP address resource associated to it. A public IP address resource has either a dynamic or static public IP address assigned to it. To learn more about IP addresses in Azure, read the [IP addresses in Azure](#) article.

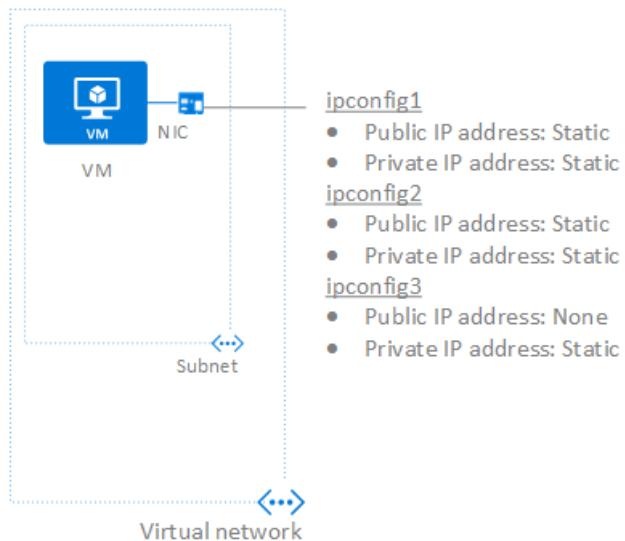
There is a limit to how many private IP addresses can be assigned to a NIC. There is also a limit to how many public IP addresses that can be used in an Azure subscription. See the [Azure limits](#) article for details.

This article explains how to create a virtual machine (VM) through the Azure Resource Manager deployment model using the Azure CLI. Multiple IP addresses cannot be assigned to resources created through the classic deployment model. To learn more about Azure deployment models, read the [Understand deployment models](#) article.

## Scenario

A VM with a single NIC is created and connected to a virtual network. The VM requires three different *private* IP addresses and two *public* IP addresses. The IP addresses are assigned to the following IP configurations:

- **IPConfig-1:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-2:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-3:** Assigns a *static* private IP address and no public IP address.



The IP configurations are associated to the NIC when the NIC is created and the NIC is attached to the VM when the VM is created. The types of IP addresses used for the scenario are for illustration. You can assign whatever IP address and assignment types you require.

#### NOTE

Though the steps in this article assigns all IP configurations to a single NIC, you can also assign multiple IP configurations to any NIC in a multi-NIC VM. To learn how to create a VM with multiple NICs, read the [Create a VM with multiple NICs](#) article.

## Create a VM with multiple IP addresses

The steps that follow explain how to create an example virtual machine with multiple IP addresses, as described in the scenario. Change variable values in "" and IP address types, as required, for your implementation.

1. Install the [Azure CLI](#) if you don't already have it installed.
2. Create an SSH public and private key pair for Linux VMs by completing the steps in the [Create an SSH public and private key pair for Linux VMs](#).
3. From a command shell, login with the command `az login` and select the subscription you're using.
4. Create the VM by executing the script that follows on a Linux or Mac computer. The script creates a resource group, one virtual network (VNet), one NIC with three IP configurations, and a VM with the two NICs attached to it. The NIC, public IP address, virtual network, and VM resources must all exist in the same location and subscription. Though the resources don't all have to exist in the same resource group, in the following script they do.

```
#!/bin/sh

RgName="myResourceGroup"
Location="westcentralus"
az group create --name $RgName --location $Location

# Create a public IP address resource with a static IP address using the `--allocation-method Static` option.
If you
# do not specify this option, the address is allocated dynamically. The address is assigned to the resource
from a pool
# of IP addresses unique to each Azure region. Download and view the file from
# https://www.microsoft.com/en-us/download/details.aspx?id=41653 that lists the ranges for each region.

PipName="myPublicIP"
```

```

# This name must be unique within an Azure location.
$dnsName="myDNSName"

az network public-ip create \
--name $PipName \
--resource-group $RgName \
--location $Location \
--dns-name $dnsName \
--allocation-method Static

# Create a virtual network with one subnet

$VnetName="myVnet"
$VnetPrefix="10.0.0.0/16"
$VnetSubnetName="mySubnet"
$VnetSubnetPrefix="10.0.0.0/24"

az network vnet create \
--name $VnetName \
--resource-group $RgName \
--location $Location \
--address-prefix $VnetPrefix \
--subnet-name $VnetSubnetName \
--subnet-prefix $VnetSubnetPrefix

# Create a network interface connected to the subnet and associate the public IP address to it. Azure will
# create the
# first IP configuration with a static private IP address and will associate the public IP address resource
# to it.

$nicName="MyNic1"
az network nic create \
--name $nicName \
--resource-group $RgName \
--location $Location \
--subnet $VnetSubnet1Name \
--private-ip-address 10.0.0.4
--vnet-name $VnetName \
--public-ip-address $PipName

# Create a second public IP address, a second IP configuration, and associate it to the NIC. This
# configuration has a
# static public IP address and a static private IP address.

az network public-ip create \
--resource-group $RgName \
--location $Location \
--name myPublicIP2 \
--dns-name mypublicdns2 \
--allocation-method Static

az network nic ip-config create \
--resource-group $RgName \
--nic-name $nicName \
--name IPConfig-2 \
--private-ip-address 10.0.0.5 \
--public-ip-name myPublicIP2

# Create a third IP configuration, and associate it to the NIC. This configuration has static private IP
# address and # no public IP address.

az network nic ip-config create \
--resource-group $RgName \
--nic-name $nicName \
--private-ip-address 10.0.0.6 \
--name IPConfig-3

# Note: Though this article assigns all IP configurations to a single NIC, you can also assign multiple IP
# configurations

```

```

# to any NIC in a VM. To learn how to create a VM with multiple NICs, read the Create a VM with multiple NICs
# article: https://docs.microsoft.com/azure/virtual-network/virtual-network-deploy-multinic-arm-cli.

# Create a VM and attach the NIC.

VmName="myVm"

# Replace the value for the following **VmSize** variable with a value from the
# https://docs.microsoft.com/azure/virtual-machines/virtual-machines-linux-sizes article. The script fails if
the VM size
# is not supported in the location you select. Run the `az vm sizes --location eastcentralus` command to
get a full list
# of VMs in US West Central, for example.

VmSize="Standard_DS1"

# Replace the value for the OsImage variable value with a value for *urn* from the output returned by
entering the
# `az vm image list` command.

OsImage="credativ:Debian:8:latest"

Username="adminuser"

# Replace the following value with the path to your public key file. If you're creating a Windows VM, remove
the following
# line and you'll be prompted for the password you want to configure for the VM.

SshKeyValue="~/.ssh/id_rsa.pub"

az vm create \
--name $VmName \
--resource-group $RgName \
--image $OsImage \
--location $Location \
--size $VmSize \
--nics $NicName \
--admin-username $Username \
--ssh-key-value $SshKeyValue

```

In addition to creating a VM with a NIC with 3 IP configurations, the script creates:

- A single premium managed disk by default, but you have other options for the disk type you can create. Read the [Create a Linux VM using the Azure CLI](#) article for details.
- A virtual network with one subnet and two public IP addresses. Alternatively, you can use *existing* virtual network, subnet, NIC, or public IP address resources. To learn how to use existing network resources rather than creating additional resources, enter `az vm create -h`.

Public IP addresses have a nominal fee. To learn more about IP address pricing, read the [IP address pricing](#) page. There is a limit to the number of public IP addresses that can be used in a subscription. To learn more about the limits, read the [Azure limits](#) article.

After the VM is created, enter the `az network nic show --name MyNic1 --resource-group myResourceGroup` command to view the NIC configuration. Enter the

`az network nic ip-config list --nic-name MyNic1 --resource-group myResourceGroup --output table` to view a list of the IP configurations associated to the NIC.

Add the private IP addresses to the VM operating system by completing the steps for your operating system in the [Add IP addresses to a VM operating system](#) section of this article.

## Add IP addresses to a VM

You can add additional private and public IP addresses to an existing Azure network interface by completing the steps that follow. The examples build upon the [scenario](#) described in this article.

1. Open a command shell and complete the remaining steps in this section within a single session. If you don't already have Azure CLI installed and configured, complete the steps in the [Azure CLI installation](#) article and login to your Azure account with the `az-login` command.
2. Complete the steps in one of the following sections, based on your requirements:

### Add a private IP address

To add a private IP address to a NIC, you must create an IP configuration using the command that follows. The static IP address must be an unused address for the subnet.

```
az network nic ip-config create \
--resource-group myResourceGroup \
--nic-name myNic1 \
--private-ip-address 10.0.0.7 \
--name IPConfig-4
```

Create as many configurations as you require, using unique configuration names and private IP addresses (for configurations with static IP addresses).

### Add a public IP address

A public IP address is added by associating it to either a new IP configuration or an existing IP configuration. Complete the steps in one of the sections that follow, as you require.

Public IP addresses have a nominal fee. To learn more about IP address pricing, read the [IP address pricing](#) page. There is a limit to the number of public IP addresses that can be used in a subscription. To learn more about the limits, read the [Azure limits](#) article.

- **Associate the resource to a new IP configuration**

Whenever you add a public IP address in a new IP configuration, you must also add a private IP address, because all IP configurations must have a private IP address. You can either add an existing public IP address resource, or create a new one. To create a new one, enter the following command:

```
az network public-ip create \
--resource-group myResourceGroup \
--location westcentralus \
--name myPublicIP3 \
--dns-name mypublicdns3
```

To create a new IP configuration with a static private IP address and the associated *myPublicIP3* public IP address resource, enter the following command:

```
az network nic ip-config create \
--resource-group myResourceGroup \
--nic-name myNic1 \
--name IPConfig-5 \
--private-ip-address 10.0.0.8 \
--public-ip-address myPublicIP3
```

- **Associate the resource to an existing IP configuration** A public IP address resource can only be associated to an IP configuration that doesn't already have one associated. You can determine whether an IP configuration has an associated public IP address by entering the following command:

```
az network nic ip-config list \
--resource-group myResourceGroup \
--nic-name myNic1 \
--query "[?provisioningState=='Succeeded'].{ Name: name, PublicIpAddressId: publicIpAddress.id }" --output table
```

Returned output:

Name	PublicIpAddressId
ipconfig1	/subscriptions/[Id]/resourceGroups/myResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIP1
IPConfig-2	/subscriptions/[Id]/resourceGroups/myResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIP2
IPConfig-3	

Since the **PublicIpAddressId** column for *IPConfig-3* is blank in the output, no public IP address resource is currently associated to it. You can add an existing public IP address resource to *IPConfig-3*, or enter the following command to create one:

```
az network public-ip create \
--resource-group myResourceGroup \
--location westcentralus \
--name myPublicIP3 \
--dns-name mypublicdns3 \
--allocation-method Static
```

Enter the following command to associate the public IP address resource to the existing IP configuration named *IPConfig-3*:

```
az network nic ip-config update \
--resource-group myResourceGroup \
--nic-name myNic1 \
--name IPConfig-3 \
--public-ip myPublicIP3
```

3. View the private IP addresses and the public IP address resource IDs assigned to the NIC by entering the following command:

```
az network nic ip-config list \
--resource-group myResourceGroup \
--nic-name myNic1 \
--query "[?provisioningState=='Succeeded'].{ Name: name, PrivateIpAddress: privateIpAddress, PrivateIpAllocationMethod: privateIpAllocationMethod, PublicIpAddressId: publicIpAddress.id }" --output table
```

Returned output:

Name	PrivateIpAddress	PrivateIpAllocationMethod	PublicIpAddressId
ipconfig1	10.0.0.4	Static	/subscriptions/[Id]/resourceGroups/myResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIP1
IPConfig-2	10.0.0.5	Static	/subscriptions/[Id]/resourceGroups/myResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIP2
IPConfig-3	10.0.0.6	Static	/subscriptions/[Id]/resourceGroups/myResourceGroup/providers/Microsoft.Network/publicIPAddresses/myPublicIP3

- Add the private IP addresses you added to the NIC to the VM operating system by following the instructions in the [Add IP addresses to a VM operating system](#) section of this article. Do not add the public IP addresses to the operating system.

## Add IP addresses to a VM operating system

Connect and sign in to a VM you created with multiple private IP addresses. You must manually add all the private IP addresses (including the primary) that you added to the VM. Complete the steps that follow for your VM operating system.

### Windows

- From a command prompt, type `ipconfig /all`. You only see the *Primary* private IP address (through DHCP).
- Type `ncpa.cpl` in the command prompt to open the **Network connections** window.
- Open the properties for the appropriate adapter: **Local Area Connection**.
- Double-click Internet Protocol version 4 (IPv4).
- Select **Use the following IP address** and enter the following values:
  - IP address:** Enter the *Primary* private IP address
  - Subnet mask:** Set based on your subnet. For example, if the subnet is a /24 subnet then the subnet mask is 255.255.255.0.
  - Default gateway:** The first IP address in the subnet. If your subnet is 10.0.0.0/24, then the gateway IP address is 10.0.0.1.
- Select **Use the following DNS server addresses** and enter the following values:
  - Preferred DNS server:** If you are not using your own DNS server, enter 168.63.129.16. If you are using your own DNS server, enter the IP address for your server.
- Select the **Advanced** button and add additional IP addresses. Add each of the secondary private IP addresses, that you added to the Azure network interface in a previous step, to the Windows network interface that is assigned the primary IP address assigned to the Azure network interface.

You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's operating system. When you manually set the IP address within the operating system, ensure that it is the same address as the private IP address assigned to the Azure network interface, or you can lose connectivity to the virtual machine. Learn more about [private IP address](#) settings. You should never assign an Azure public IP address within the operating system.

- Click **OK** to close out the TCP/IP settings and then **OK** again to close the adapter settings. Your RDP connection is re-established.
- From a command prompt, type `ipconfig /all`. All IP addresses you added are shown and DHCP is turned

off.

7. Configure Windows to use the private IP address of the primary IP configuration in Azure as the primary IP address for Windows. See [No Internet access from Azure Windows VM that has multiple IP addresses](#) for details.

### Validation (Windows)

To ensure you are able to connect to the internet from your secondary IP configuration via the public IP associated with it, once you have added it correctly using steps above, use the following command:

```
ping -S 10.0.0.5 hotmail.com
```

#### NOTE

For secondary IP configurations, you can only ping to the Internet if the configuration has a public IP address associated with it. For primary IP configurations, a public IP address is not required to ping to the Internet.

### Linux (Ubuntu 14/16)

We recommend looking at the latest documentation for your Linux distribution.

1. Open a terminal window.
2. Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

3. Update the configuration file of the network interface (assuming 'eth0').

- Keep the existing line item for dhcp. The primary IP address remains configured as it was previously.
- Add a configuration for an additional static IP address with the following commands:

```
cd /etc/network/interfaces.d/  
ls
```

You should see a .cfg file.

4. Open the file. You should see the following lines at the end of the file:

```
auto eth0  
iface eth0 inet dhcp
```

5. Add the following lines after the lines that exist in this file:

```
iface eth0 inet static  
address <your private IP address here>  
netmask <your subnet mask>
```

6. Save the file by using the following command:

```
:wq
```

7. Reset the network interface with the following command:

```
sudo ifdown eth0 && sudo ifup eth0
```

**IMPORTANT**

Run both ifdown and ifup in the same line if using a remote connection.

8. Verify the IP address is added to the network interface with the following command:

```
ip addr list eth0
```

You should see the IP address you added as part of the list.

**Linux (Ubuntu 18.04+)**

Ubuntu 18.04 and above have changed to `netplan` for OS network management. We recommend looking at the latest documentation for your Linux distribution.

1. Open a terminal window.

2. Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

3. Create a file for the second interface and open it in a text editor:

```
vi /etc/netplan/60-static.yaml
```

4. Add the following lines to the file, replacing `10.0.0.6/24` with your IP/netmask:

```
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 10.0.0.6/24
```

5. Save the file by using the following command:

```
:wq
```

6. Test the changes using `netplan try` to confirm syntax:

```
netplan try
```

**NOTE**

`netplan try` will apply the changes temporarily and roll the changes back after 120 seconds. If there is a loss of connectivity, please wait 120 seconds, and then reconnect. At that time, the changes will have been rolled back.

7. Assuming no issues with `netplan try`, apply the configuration changes:

```
netplan apply
```

8. Verify the IP address is added to the network interface with the following command:

```
ip addr list eth0
```

You should see the IP address you added as part of the list. Example:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:0d:3a:8c:14:a5 brd ff:ff:ff:ff:ff:ff
        inet 10.0.0.6/24 brd 10.0.0.255 scope global eth0
            valid_lft forever preferred_lft forever
        inet 10.0.0.4/24 brd 10.0.0.255 scope global secondary eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::20d:3aff:fe8c:14a5/64 scope link
            valid_lft forever preferred_lft forever
```

### Linux (Red Hat, CentOS, and others)

1. Open a terminal window.
2. Make sure you are the root user. If you are not, enter the following command:

```
sudo -i
```

3. Enter your password and follow instructions as prompted. Once you are the root user, navigate to the network scripts folder with the following command:

```
cd /etc/sysconfig/network-scripts
```

4. List the related ifcfg files using the following command:

```
ls ifcfg-*
```

You should see `ifcfg-eth0` as one of the files.

5. To add an IP address, create a configuration file for it as shown below. Note that one file must be created for each IP configuration.

```
touch ifcfg-eth0:0
```

6. Open the `ifcfg-eth0:0` file with the following command:

```
vi ifcfg-eth0:0
```

7. Add content to the file, *eth0:0* in this case, with the following command. Be sure to update information based on your IP address.

```
DEVICE=eth0:0
BOOTPROTO=static
ONBOOT=yes
IPADDR=192.168.101.101
NETMASK=255.255.255.0
```

8. Save the file with the following command:

```
:wq
```

9. Restart the network services and make sure the changes are successful by running the following commands:

```
/etc/init.d/network restart
ifconfig
```

You should see the IP address you added, *eth0:0*, in the list returned.

### Validation (Linux)

To ensure you are able to connect to the internet from your secondary IP configuration via the public IP associated with it, use the following command:

```
ping -I 10.0.0.5 hotmail.com
```

#### NOTE

For secondary IP configurations, you can only ping to the Internet if the configuration has a public IP address associated with it. For primary IP configurations, a public IP address is not required to ping to the Internet.

For Linux VMs, when trying to validate outbound connectivity from a secondary NIC, you may need to add appropriate routes. There are many ways to do this. Please see appropriate documentation for your Linux distribution. The following is one method to accomplish this:

```
echo 150 custom >> /etc/iproute2/rt_tables
ip rule add from 10.0.0.5 lookup custom
ip route add default via 10.0.0.1 dev eth2 table custom
```

- Be sure to replace:
  - **10.0.0.5** with the private IP address that has a public IP address associated with it
  - **10.0.0.1** to your default gateway
  - **eth2** to the name of your secondary NIC

# Add network interfaces to or remove network interfaces from virtual machines

11/19/2019 • 8 minutes to read • [Edit Online](#)

Learn how to add an existing network interface when you create an Azure virtual machine (VM), or to add or remove network interfaces from an existing VM in the stopped (deallocated) state. A network interface enables an Azure virtual machine to communicate with internet, Azure, and on-premises resources. A VM can have one or more network interfaces.

If you need to add, change, or remove IP addresses for a network interface, see [Manage network interface IP addresses](#). If you need to create, change, or delete network interfaces, see [Manage network interfaces](#).

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Complete the following tasks before completing steps in any section of this article:

- If you don't already have an Azure account, sign up for a [free trial account](#).
- If using the portal, open <https://portal.azure.com>, and log in with your Azure account.
- If using PowerShell commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running PowerShell from your computer. The Azure Cloud Shell is a free interactive shell that you can use to run the steps in this article. It has common Azure tools preinstalled and configured to use with your account. This tutorial requires the Azure PowerShell module version 1.0.0 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.
- If using Azure Command-line interface (CLI) commands to complete tasks in this article, either run the commands in the [Azure Cloud Shell](#), or by running the CLI from your computer. This tutorial requires the Azure CLI version 2.0.26 or later. Run `az --version` to find the installed version. If you need to install or upgrade, see [Install Azure CLI](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

## Add existing network interfaces to a new VM

When you create a virtual machine through the portal, the portal creates a network interface with default settings and attaches it to the VM for you. You cannot add existing network interfaces to a new VM, nor create a VM with multiple network interfaces, by using the Azure portal. You can do both by using the CLI or PowerShell, but be sure to familiarize yourself with the [constraints](#). If you create a VM with multiple network interfaces, you must also configure the operating system to use them properly after you create the VM. Learn how to configure [Linux](#) or [Windows](#) for multiple network interfaces.

### Commands

Before you create the VM, create a network interface by using the steps in [Create a network interface](#).

TOOL	COMMAND
CLI	<code>az vm create</code>
PowerShell	<code>New-AzVM</code>

## Add a network interface to an existing VM

1. Sign in to the Azure portal.
2. In the search box at the top of the portal, type the name of the VM to which you want to add the network interface, or browse for the VM by selecting **All services**, and then **Virtual machines**. After you've found the VM, select it. The VM must support the number of network interfaces you want to add. To find out how many network interfaces each VM size supports, see [Sizes for Linux virtual machines in Azure](#) or [Sizes for Windows virtual machines in Azure](#).
3. Select **Overview**, under **SETTINGS**. Select **Stop**, and then wait until the **Status** of the VM changes to **Stopped (deallocated)**.
4. Select **Networking**, under **SETTINGS**.
5. Select **Attach network interface**. From the list of network interfaces that aren't currently attached to another VM, select the one you'd like to attach.

### NOTE

The network interface you select cannot have accelerated networking enabled, cannot have an IPv6 address assigned to it, and must exist in the same virtual network as the one that contains the network interface currently attached to the VM.

If you don't have an existing network interface, you must first create one. To do so, select **Create network interface**. To learn more about how to create a network interface, see [Create a network interface](#). To learn more about additional constraints when adding network interfaces to virtual machines, see [Constraints](#).

6. Select **OK**.
7. Select **Overview**, under **SETTINGS**, and then **Start** to start the virtual machine.
8. Configure the VM operating system to use multiple network interfaces properly. Learn how to configure [Linux](#) or [Windows](#) for multiple network interfaces.

### Commands

TOOL	COMMAND
CLI	<a href="#">az vm nic add</a> (reference) or <a href="#">detailed steps</a>
PowerShell	<a href="#">Add-AzVMNetworkInterface</a> (reference) or <a href="#">detailed steps</a>

## View network interfaces for a VM

You can view the network interfaces currently attached to a VM to learn about each network interface's configuration, and the IP addresses assigned to each network interface.

1. Sign in to the [Azure portal](#) with an account that is assigned the Owner, Contributor, or Network Contributor role for your subscription. To learn more about how to assign roles to accounts, see [Built-in roles for Azure role-based access control](#).
2. In the box that contains the text **Search resources** at the top of the Azure portal, type **virtual machines**. When **virtual machines** appears in the search results, select it.
3. Select the name of the VM for which you want to view network interfaces.
4. In the **SETTINGS** section for the VM you selected, select **Networking**. To learn about network interface settings and how to change them, see [Manage network interfaces](#). To learn about how to add, change, or remove IP addresses assigned to a network interface, see [Manage network interface IP addresses](#).

## Commands

TOOL	COMMAND
CLI	<a href="#">az vm show</a>
PowerShell	<a href="#">Get-AzVM</a>

## Remove a network interface from a VM

1. Sign in to the Azure portal.
2. In the search box at the top of the portal, search for the name of the VM you want to remove (detach) the network interface from, or browse for the VM by selecting **All services**, and then **Virtual machines**. After you've found the VM, select it.
3. Select **Overview**, under **SETTINGS**, and then **Stop**. Wait until the **Status** of the VM changes to **Stopped (deallocated)**.
4. Select **Networking**, under **SETTINGS**.
5. Select **Detach network interface**. From the list of network interfaces currently attached to the virtual machine, select the network interface you'd like to detach.

### NOTE

If only one network interface is listed, you cannot detach it, because a virtual machine must always have at least one network interface attached to it.

6. Select **OK**.

## Commands

TOOL	COMMAND
CLI	<a href="#">az vm nic remove (reference) or <a href="#">detailed steps</a></a>
PowerShell	<a href="#">Remove-AzVMNetworkInterface (reference) or <a href="#">detailed steps</a></a>

## Constraints

- A VM must have at least one network interface attached to it.
- A VM can only have as many network interfaces attached to it as the VM size supports. To learn more about how many network interfaces each VM size supports, see [Sizes for Linux virtual machines in Azure](#) or [Sizes for](#)

[Windows virtual machines in Azure](#). All sizes support at least two network interfaces.

- The network interfaces you add to a VM cannot currently be attached to another VM. To learn more about how to create network interfaces, see [Create a network interface](#).
- In the past, network interfaces could only be added to VMs that supported multiple network interfaces and were created with at least two network interfaces. You could not add a network interface to a VM that was created with one network interface, even if the VM size supported multiple network interfaces. Conversely, you could only remove network interfaces from a VM with at least three network interfaces, because VMs created with at least two network interfaces always had to have at least two network interfaces. Neither of these constraints apply anymore. You can now create a VM with any number of network interfaces (up to the number supported by the VM size).
- By default, the first network interface attached to a VM is defined as the *primary* network interface. All other network interfaces in the VM are *secondary* network interfaces.
- Though you can control which network interface you sent outbound traffic to, by default, all outbound traffic from the VM is sent out the IP address assigned to the primary IP configuration of the primary network interface.
- In the past, all VMs within the same availability set were required to have a single, or multiple, network interfaces. VMs with any number of network interfaces can now exist in the same availability set, up to the number supported by the VM size. You can only add a VM to an availability set when it's created. To learn more about availability sets, see [Manage the availability of VMs in Azure](#).
- While network interfaces in the same VM can be connected to different subnets within a virtual network, the network interfaces must all be connected to the same virtual network.
- You can add any IP address for any IP configuration of any primary or secondary network interface to an Azure Load Balancer back-end pool. In the past, only the primary IP address for the primary network interface could be added to a back-end pool. To learn more about IP addresses and configurations, see [Add, change, or remove IP addresses](#).
- Deleting a VM does not delete the network interfaces that are attached to it. When you delete a VM, the network interfaces are detached from the VM. You can add the network interfaces to different VMs or delete them.
- As with IPv6, you cannot attach a network interface with accelerated networking enabled to a VM after you create it. Further, to take advantage of accelerated networking, you must also complete steps within the VM operating system. Learn more about accelerated networking, and other constraints when using it, for [Windows](#) or [Linux](#) virtual machines.

## Next steps

To create a VM with multiple network interfaces or IP addresses, see the following articles:

TASK	TOOL
Create a VM with multiple NICs	<a href="#">CLI</a> , <a href="#">PowerShell</a>
Create a single NIC VM with multiple IPv4 addresses	<a href="#">CLI</a> , <a href="#">PowerShell</a>
Create a single NIC VM with a private IPv6 address (behind an Azure Load Balancer)	<a href="#">CLI</a> , <a href="#">PowerShell</a> , <a href="#">Azure Resource Manager template</a>

# Create and manage a Windows virtual machine that has multiple NICs

12/23/2019 • 8 minutes to read • [Edit Online](#)

Virtual machines (VMs) in Azure can have multiple virtual network interface cards (NICs) attached to them. A common scenario is to have different subnets for front-end and back-end connectivity. You can associate multiple NICs on a VM to multiple subnets, but those subnets must all reside in the same virtual network (vNet). This article details how to create a VM that has multiple NICs attached to it. You also learn how to add or remove NICs from an existing VM. Different [VM sizes](#) support a varying number of NICs, so size your VM accordingly.

## Prerequisites

In the following examples, replace example parameter names with your own values. Example parameter names include *myResourceGroup*, *myVnet*, and *myVM*.

## Create a VM with multiple NICs

First, create a resource group. The following example creates a resource group named *myResourceGroup* in the *EastUs* location:

```
New-AzResourceGroup -Name "myResourceGroup" -Location "EastUS"
```

### Create virtual network and subnets

A common scenario is for a virtual network to have two or more subnets. One subnet may be for front-end traffic, the other for back-end traffic. To connect to both subnets, you then use multiple NICs on your VM.

1. Define two virtual network subnets with [New-AzVirtualNetworkSubnetConfig](#). The following example defines the subnets for *mySubnetFrontEnd* and *mySubnetBackEnd*:

```
$mySubnetFrontEnd = New-AzVirtualNetworkSubnetConfig -Name "mySubnetFrontEnd" `  
-AddressPrefix "192.168.1.0/24"  
$mySubnetBackEnd = New-AzVirtualNetworkSubnetConfig -Name "mySubnetBackEnd" `  
-AddressPrefix "192.168.2.0/24"
```

2. Create your virtual network and subnets with [New-AzVirtualNetwork](#). The following example creates a virtual network named *myVnet*:

```
$myVnet = New-AzVirtualNetwork -ResourceGroupName "myResourceGroup" `  
-Location "EastUs" `  
-Name "myVnet" `  
-AddressPrefix "192.168.0.0/16" `  
-Subnet $mySubnetFrontEnd,$mySubnetBackEnd
```

### Create multiple NICs

Create two NICs with [New-AzNetworkInterface](#). Attach one NIC to the front-end subnet and one NIC to the back-end subnet. The following example creates NICs named *myNic1* and *myNic2*:

```

$frontEnd = $myVnet.Subnets | ?{$_ .Name -eq 'mySubnetFrontEnd'}
$myNic1 = New-AzNetworkInterface -ResourceGroupName "myResourceGroup" ` 
    -Name "myNic1" ` 
    -Location "EastUs" ` 
    -SubnetId $frontEnd.Id

$backEnd = $myVnet.Subnets | ?{$_ .Name -eq 'mySubnetBackEnd'}
$myNic2 = New-AzNetworkInterface -ResourceGroupName "myResourceGroup" ` 
    -Name "myNic2" ` 
    -Location "EastUs" ` 
    -SubnetId $backEnd.Id

```

Typically you also create a [network security group](#) to filter network traffic to the VM and a [load balancer](#) to distribute traffic across multiple VMs.

### Create the virtual machine

Now start to build your VM configuration. Each VM size has a limit for the total number of NICs that you can add to a VM. For more information, see [Windows VM sizes](#).

1. Set your VM credentials to the `$cred` variable as follows:

```
$cred = Get-Credential
```

2. Define your VM with [New-AzVMConfig](#). The following example defines a VM named *myVM* and uses a VM size that supports more than two NICs (*Standard\_DS3\_v2*):

```
$vmConfig = New-AzVMConfig -VMName "myVM" -VMSize "Standard_DS3_v2"
```

3. Create the rest of your VM configuration with [Set-AzVMOperatingSystem](#) and [Set-AzVMSourceImage](#). The following example creates a Windows Server 2016 VM:

```

$vmConfig = Set-AzVMOperatingSystem -VM $vmConfig ` 
    -Windows ` 
    -ComputerName "myVM" ` 
    -Credential $cred ` 
    -ProvisionVMAgent ` 
    -EnableAutoUpdate
$vmConfig = Set-AzVMSourceImage -VM $vmConfig ` 
    -PublisherName "MicrosoftWindowsServer" ` 
    -Offer "WindowsServer" ` 
    -Skus "2016-Datacenter" ` 
    -Version "latest"

```

4. Attach the two NICs that you previously created with [Add-AzVMNetworkInterface](#):

```

$vmConfig = Add-AzVMNetworkInterface -VM $vmConfig -Id $myNic1.Id -Primary
$vmConfig = Add-AzVMNetworkInterface -VM $vmConfig -Id $myNic2.Id

```

5. Create your VM with [New-AzVM](#):

```
New-AzVM -VM $vmConfig -ResourceGroupName "myResourceGroup" -Location "EastUs"
```

6. Add routes for secondary NICs to the OS by completing the steps in [Configure the operating system for multiple NICs](#).

# Add a NIC to an existing VM

To add a virtual NIC to an existing VM, you deallocate the VM, add the virtual NIC, then start the VM. Different [VM sizes](#) support a varying number of NICs, so size your VM accordingly. If needed, you can [resize a VM](#).

1. Deallocate the VM with [Stop-AzVM](#). The following example deallocates the VM named *myVM* in *myResourceGroup*:

```
Stop-AzVM -Name "myVM" -ResourceGroupName "myResourceGroup"
```

2. Get the existing configuration of the VM with [Get-AzVm](#). The following example gets information for the VM named *myVM* in *myResourceGroup*:

```
$vm = Get-AzVm -Name "myVM" -ResourceGroupName "myResourceGroup"
```

3. The following example creates a virtual NIC with [New-AzNetworkInterface](#) named *myNic3* that is attached to *mySubnetBackEnd*. The virtual NIC is then attached to the VM named *myVM* in *myResourceGroup* with [Add-AzVMNetworkInterface](#):

```
# Get info for the back end subnet
$myVnet = Get-AzVirtualNetwork -Name "myVnet" -ResourceGroupName "myResourceGroup"
$backEnd = $myVnet.Subnets | ?{$_ . Name -eq 'mySubnetBackEnd'}`

# Create a virtual NIC
$myNic3 = New-AzNetworkInterface -ResourceGroupName "myResourceGroup" ` 
    -Name "myNic3" ` 
    -Location "EastUs" ` 
    -SubnetId $backEnd.Id

# Get the ID of the new virtual NIC and add to VM
$nicId = (Get-AzNetworkInterface -ResourceGroupName "myResourceGroup" -Name "MyNic3").Id
Add-AzVMNetworkInterface -VM $vm -Id $nicId | Update-AzVm -ResourceGroupName "myResourceGroup"
```

## Primary virtual NICs

One of the NICs on a multi-NIC VM needs to be primary. If one of the existing virtual NICs on the VM is already set as primary, you can skip this step. The following example assumes that two virtual NICs are now present on a VM and you wish to add the first NIC ( [0] ) as the primary:

```
# List existing NICs on the VM and find which one is primary
$vm.NetworkProfile.NetworkInterfaces

# Set NIC 0 to be primary
$vm.NetworkProfile.NetworkInterfaces[0].Primary = $true
$vm.NetworkProfile.NetworkInterfaces[1].Primary = $false

# Update the VM state in Azure
Update-AzVM -VM $vm -ResourceGroupName "myResourceGroup"
```

4. Start the VM with [Start-AzVm](#):

```
Start-AzVM -ResourceGroupName "myResourceGroup" -Name "myVM"
```

5. Add routes for secondary NICs to the OS by completing the steps in [Configure the operating system for multiple NICs](#).

## Remove a NIC from an existing VM

To remove a virtual NIC from an existing VM, you deallocate the VM, remove the virtual NIC, then start the VM.

1. Deallocate the VM with [Stop-AzVM](#). The following example deallocates the VM named *myVM* in *myResourceGroup*:

```
Stop-AzVM -Name "myVM" -ResourceGroupName "myResourceGroup"
```

2. Get the existing configuration of the VM with [Get-AzVm](#). The following example gets information for the VM named *myVM* in *myResourceGroup*:

```
$vm = Get-AzVm -Name "myVM" -ResourceGroupName "myResourceGroup"
```

3. Get information about the NIC remove with [Get-AzNetworkInterface](#). The following example gets information about *myNic3*:

```
# List existing NICs on the VM if you need to determine NIC name
$vm.NetworkProfile.NetworkInterfaces

$nicId = (Get-AzNetworkInterface -ResourceGroupName "myResourceGroup" -Name "myNic3").Id
```

4. Remove the NIC with [Remove-AzVMNetworkInterface](#) and then update the VM with [Update-AzVm](#). The following example removes *myNic3* as obtained by `$nicId` in the preceding step:

```
Remove-AzVMNetworkInterface -VM $vm -NetworkInterfaceIDs $nicId | ` 
    Update-AzVm -ResourceGroupName "myResourceGroup"
```

5. Start the VM with [Start-AzVm](#):

```
Start-AzVM -Name "myVM" -ResourceGroupName "myResourceGroup"
```

## Create multiple NICs with templates

Azure Resource Manager templates provide a way to create multiple instances of a resource during deployment, such as creating multiple NICs. Resource Manager templates use declarative JSON files to define your environment. For more information, see [overview of Azure Resource Manager](#). You can use `copy` to specify the number of instances to create:

```
"copy": {
    "name": "multiplenics",
    "count": "[parameters('count')]"
}
```

For more information, see [creating multiple instances by using copy](#).

You can also use `copyIndex()` to append a number to a resource name. You can then create *myNic1*, *MyNic2* and so on. The following code shows an example of appending the index value:

```
"name": "[concat('myNic', copyIndex())]",
```

You can read a complete example of [creating multiple NICs by using Resource Manager templates](#).

Add routes for secondary NICs to the OS by completing the steps in [Configure the operating system for multiple NICs](#).

## Configure guest OS for multiple NICs

Azure assigns a default gateway to the first (primary) network interface attached to the virtual machine. Azure does not assign a default gateway to additional (secondary) network interfaces attached to a virtual machine. Therefore, you are unable to communicate with resources outside the subnet that a secondary network interface is in, by default. Secondary network interfaces can, however, communicate with resources outside their subnet, though the steps to enable communication are different for different operating systems.

1. From a Windows command prompt, run the `route print` command, which returns output similar to the following output for a virtual machine with two attached network interfaces:

```
=====
Interface List
3...00 0d 3a 10 92 ce ....Microsoft Hyper-V Network Adapter #3
7...00 0d 3a 10 9b 2a ....Microsoft Hyper-V Network Adapter #4
=====
```

In this example, **Microsoft Hyper-V Network Adapter #4** (interface 7) is the secondary network interface that doesn't have a default gateway assigned to it.

2. From a command prompt, run the `ipconfig` command to see which IP address is assigned to the secondary network interface. In this example, 192.168.2.4 is assigned to interface 7. No default gateway address is returned for the secondary network interface.
3. To route all traffic destined for addresses outside the subnet of the secondary network interface to the gateway for the subnet, run the following command:

```
route add -p 0.0.0.0 MASK 0.0.0.0 192.168.2.1 METRIC 5015 IF 7
```

The gateway address for the subnet is the first IP address (ending in .1) in the address range defined for the subnet. If you don't want to route all traffic outside the subnet, you could add individual routes to specific destinations, instead. For example, if you only wanted to route traffic from the secondary network interface to the 192.168.3.0 network, you enter the command:

```
route add -p 192.168.3.0 MASK 255.255.255.0 192.168.2.1 METRIC 5015 IF 7
```

4. To confirm successful communication with a resource on the 192.168.3.0 network, for example, enter the following command to ping 192.168.3.4 using interface 7 (192.168.2.4):

```
ping 192.168.3.4 -S 192.168.2.4
```

You may need to open ICMP through the Windows firewall of the device you're pinging with the following command:

```
netsh advfirewall firewall add rule name=Allow-ping protocol=icmpv4 dir=in action=allow
```

5. To confirm the added route is in the route table, enter the `route print` command, which returns output similar to the following text:

=====					
Active Routes:					
Network Destination	Netmask	Gateway	Interface	Metric	
0.0.0.0	0.0.0.0	192.168.1.1	192.168.1.4	15	
0.0.0.0	0.0.0.0	192.168.2.1	192.168.2.4	5015	

The route listed with **192.168.1.1** under **Gateway**, is the route that is there by default for the primary network interface. The route with **192.168.2.1** under **Gateway**, is the route you added.

## Next steps

Review [Windows VM sizes](#) when you're trying to create a VM that has multiple NICs. Pay attention to the maximum number of NICs that each VM size supports.

# How to create a Linux virtual machine in Azure with multiple network interface cards

12/23/2019 • 6 minutes to read • [Edit Online](#)

This article details how to create a VM with multiple NICs with the Azure CLI.

## Create supporting resources

Install the latest [Azure CLI](#) and log in to an Azure account using [az login](#).

In the following examples, replace example parameter names with your own values. Example parameter names included *myResourceGroup*, *mystorageaccount*, and *myVM*.

First, create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location:

```
az group create --name myResourceGroup --location eastus
```

Create the virtual network with [az network vnet create](#). The following example creates a virtual network named *myVnet* and subnet named *mySubnetFrontEnd*:

```
az network vnet create \
--resource-group myResourceGroup \
--name myVnet \
--address-prefix 10.0.0.0/16 \
--subnet-name mySubnetFrontEnd \
--subnet-prefix 10.0.1.0/24
```

Create a subnet for the back-end traffic with [az network vnet subnet create](#). The following example creates a subnet named *mySubnetBackEnd*:

```
az network vnet subnet create \
--resource-group myResourceGroup \
--vnet-name myVnet \
--name mySubnetBackEnd \
--address-prefix 10.0.2.0/24
```

Create a network security group with [az network nsg create](#). The following example creates a network security group named *myNetworkSecurityGroup*:

```
az network nsg create \
--resource-group myResourceGroup \
--name myNetworkSecurityGroup
```

## Create and configure multiple NICs

Create two NICs with [az network nic create](#). The following example creates two NICs, named *myNic1* and *myNic2*, connected the network security group, with one NIC connecting to each subnet:

```
az network nic create \
--resource-group myResourceGroup \
--name myNic1 \
--vnet-name myVnet \
--subnet mySubnetFrontEnd \
--network-security-group myNetworkSecurityGroup
az network nic create \
--resource-group myResourceGroup \
--name myNic2 \
--vnet-name myVnet \
--subnet mySubnetBackEnd \
--network-security-group myNetworkSecurityGroup
```

## Create a VM and attach the NICs

When you create the VM, specify the NICs you created with `--nics`. You also need to take care when you select the VM size. There are limits for the total number of NICs that you can add to a VM. Read more about [Linux VM sizes](#).

Create a VM with [az vm create](#). The following example creates a VM named *myVM*:

```
az vm create \
--resource-group myResourceGroup \
--name myVM \
--image UbuntuLTS \
--size Standard_DS3_v2 \
--admin-username azureuser \
--generate-ssh-keys \
--nics myNic1 myNic2
```

Add routing tables to the guest OS by completing the steps in [Configure the guest OS for multiple NICs](#).

## Add a NIC to a VM

The previous steps created a VM with multiple NICs. You can also add NICs to an existing VM with the Azure CLI. Different [VM sizes](#) support a varying number of NICs, so size your VM accordingly. If needed, you can [resize a VM](#).

Create another NIC with [az network nic create](#). The following example creates a NIC named *myNic3* connected to the back-end subnet and network security group created in the previous steps:

```
az network nic create \
--resource-group myResourceGroup \
--name myNic3 \
--vnet-name myVnet \
--subnet mySubnetBackEnd \
--network-security-group myNetworkSecurityGroup
```

To add a NIC to an existing VM, first deallocate the VM with [az vm deallocate](#). The following example deallocates the VM named *myVM*:

```
az vm deallocate --resource-group myResourceGroup --name myVM
```

Add the NIC with [az vm nic add](#). The following example adds *myNic3* to *myVM*:

```
az vm nic add \
--resource-group myResourceGroup \
--vm-name myVM \
--nics myNic3
```

Start the VM with [az vm start](#):

```
az vm start --resource-group myResourceGroup --name myVM
```

Add routing tables to the guest OS by completing the steps in [Configure the guest OS for multiple NICs](#).

## Remove a NIC from a VM

To remove a NIC from an existing VM, first deallocate the VM with [az vm deallocate](#). The following example deallocates the VM named *myVM*:

```
az vm deallocate --resource-group myResourceGroup --name myVM
```

Remove the NIC with [az vm nic remove](#). The following example removes *myNic3* from *myVM*:

```
az vm nic remove \
--resource-group myResourceGroup \
--vm-name myVM \
--nics myNic3
```

Start the VM with [az vm start](#):

```
az vm start --resource-group myResourceGroup --name myVM
```

## Create multiple NICs using Resource Manager templates

Azure Resource Manager templates use declarative JSON files to define your environment. You can read an [overview of Azure Resource Manager](#). Resource Manager templates provide a way to create multiple instances of a resource during deployment, such as creating multiple NICs. You use *copy* to specify the number of instances to create:

```
"copy": {
  "name": "multiplenics"
  "count": "[parameters('count')]"
}
```

Read more about [creating multiple instances using copy](#).

You can also use a `copyIndex()` to then append a number to a resource name, which allows you to create `myNic1`, `myNic2`, etc. The following shows an example of appending the index value:

```
"name": "[concat('myNic', copyIndex())]",
```

You can read a complete example of [creating multiple NICs using Resource Manager templates](#).

Add routing tables to the guest OS by completing the steps in [Configure the guest OS for multiple NICs](#).

## Configure guest OS for multiple NICs

The previous steps created a virtual network and subnet, attached NICs, then created a VM. A public IP address and network security group rules that allow SSH traffic were not created. To configure the guest OS for multiple NICs, you need to allow remote connections and run commands locally on the VM.

To allow SSH traffic, create a network security group rule with [az network nsg rule create](#) as follows:

```
az network nsg rule create \
--resource-group myResourceGroup \
--nsg-name myNetworkSecurityGroup \
--name allow_ssh \
--priority 101 \
--destination-port-ranges 22
```

Create a public IP address with [az network public-ip create](#) and assign it to the first NIC with [az network nic ip-config update](#):

```
az network public-ip create --resource-group myResourceGroup --name myPublicIP

az network nic ip-config update \
--resource-group myResourceGroup \
--nic-name myNic1 \
--name ipconfig1 \
--public-ip myPublicIP
```

To view the public IP address of the VM, use [az vm show](#) as follows::

```
az vm show --resource-group myResourceGroup --name myVM -d --query publicIps -o tsv
```

Now SSH to the public IP address of your VM. The default username provided in a previous step was *azureuser*. Provide your own username and public IP address:

```
ssh azureuser@137.117.58.232
```

To send to or from a secondary network interface, you have to manually add persistent routes to the operating system for each secondary network interface. In this article, *eth1* is the secondary interface. Instructions for adding persistent routes to the operating system vary by distro. See documentation for your distro for instructions.

When adding the route to the operating system, the gateway address is *.1* for whichever subnet the network interface is in. For example, if the network interface is assigned the address *10.0.2.4*, the gateway you specify for the route is *10.0.2.1*. You can define a specific network for the route's destination, or specify a destination of *0.0.0.0*, if you want all traffic for the interface to go through the specified gateway. The gateway for each subnet is managed by the virtual network.

Once you've added the route for a secondary interface, verify that the route is in your route table with `route -n`. The following example output is for the route table that has the two network interfaces added to the VM in this article:

Kernel IP routing table						
Destination	Gateway	Genmask	Flags	Metric	Ref	Use Iface
0.0.0.0	10.0.1.1	0.0.0.0	UG	0	0	0 eth0
0.0.0.0	10.0.2.1	0.0.0.0	UG	0	0	0 eth1
10.0.1.0	0.0.0.0	255.255.255.0	U	0	0	0 eth0
10.0.2.0	0.0.0.0	255.255.255.0	U	0	0	0 eth1
168.63.129.16	10.0.1.1	255.255.255.255	UGH	0	0	0 eth0
169.254.169.254	10.0.1.1	255.255.255.255	UGH	0	0	0 eth0

Confirm that the route you added persists across reboots by checking your route table again after a reboot. To test connectivity, you can enter the following command, for example, where *eth1* is the name of a secondary network interface:

```
ping bing.com -c 4 -I eth1
```

## Next steps

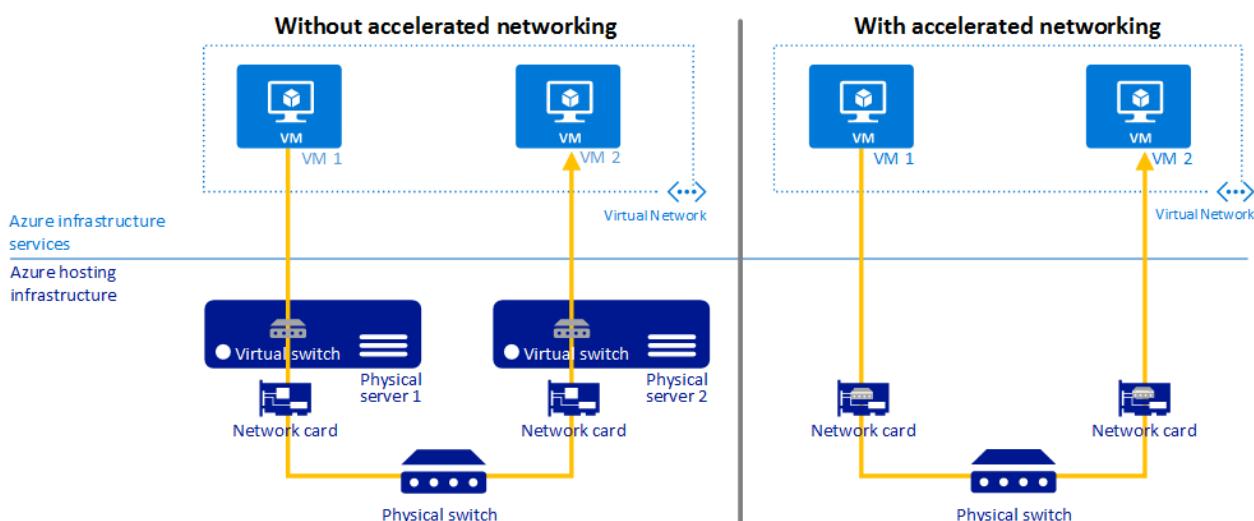
Review [Linux VM sizes](#) when trying to creating a VM with multiple NICs. Pay attention to the maximum number of NICs each VM size supports.

To further secure your VMs, use just in time VM access. This feature opens network security group rules for SSH traffic when needed, and for a defined period of time. For more information, see [Manage virtual machine access using just in time](#).

# Create a Windows virtual machine with Accelerated Networking using Azure PowerShell

11/14/2019 • 10 minutes to read • [Edit Online](#)

In this tutorial, you learn how to create a Windows virtual machine (VM) with Accelerated Networking. To create a Linux VM with Accelerated Networking, see [Create a Linux VM with Accelerated Networking](#). Accelerated networking enables single root I/O virtualization (SR-IOV) to a VM, greatly improving its networking performance. This high-performance path bypasses the host from the datapath, reducing latency, jitter, and CPU utilization, for use with the most demanding network workloads on supported VM types. The following picture shows communication between two VMs with and without accelerated networking:



Without accelerated networking, all networking traffic in and out of the VM must traverse the host and the virtual switch. The virtual switch provides all policy enforcement, such as network security groups, access control lists, isolation, and other network virtualized services to network traffic. To learn more about virtual switches, see [Hyper-V network virtualization and virtual switch](#).

With accelerated networking, network traffic arrives at the VM's network interface (NIC), and is then forwarded to the VM. All network policies that the virtual switch applies are now offloaded and applied in hardware. Applying policy in hardware enables the NIC to forward network traffic directly to the VM, bypassing the host and the virtual switch, while maintaining all the policy it applied in the host.

The benefits of accelerated networking only apply to the VM that it is enabled on. For the best results, it is ideal to enable this feature on at least two VMs connected to the same Azure Virtual Network (VNet). When communicating across VNets or connecting on-premises, this feature has minimal impact to overall latency.

## Benefits

- **Lower Latency / Higher packets per second (pps):** Removing the virtual switch from the datapath removes the time packets spend in the host for policy processing and increases the number of packets that can be processed inside the VM.
- **Reduced jitter:** Virtual switch processing depends on the amount of policy that needs to be applied and the workload of the CPU that is doing the processing. Offloading the policy enforcement to the hardware removes that variability by delivering packets directly to the VM, removing the host to VM communication and all software interrupts and context switches.
- **Decreased CPU utilization:** Bypassing the virtual switch in the host leads to less CPU utilization for

processing network traffic.

## Limitations and Constraints

### Supported operating systems

The following distributions are supported out of the box from the Azure Gallery:

- **Windows Server 2016 Datacenter**
- **Windows Server 2012 R2 Datacenter**
- **Windows Server 2019 Datacenter**

### Supported VM instances

Accelerated Networking is supported on most general purpose and compute-optimized instance sizes with 2 or more vCPUs. These supported series are: D/DSv2 and F/Fs

On instances that support hyperthreading, Accelerated Networking is supported on VM instances with 4 or more vCPUs. Supported series are: D/Dsv3, E/Esv3, Fsv2, Lsv2, Ms/Mms and Ms/Mmsv2.

For more information on VM instances, see [Windows VM sizes](#).

### Regions

Available in all public Azure regions and Azure Government Cloud.

### Enabling Accelerated Networking on a running VM

A supported VM size without accelerated networking enabled can only have the feature enabled when it is stopped and deallocated.

### Deployment through Azure Resource Manager

Virtual machines (classic) cannot be deployed with Accelerated Networking.

## Create a Windows VM with Azure Accelerated Networking

### Portal creation

Though this article provides steps to create a virtual machine with accelerated networking using Azure Powershell, you can also [create a virtual machine with accelerated networking using the Azure portal](#). When creating a virtual machine in the portal, in the **Create a virtual machine** blade, choose the **Networking** tab. In this tab, there is an option for **Accelerated networking**. If you have chosen a [supported operating system](#) and [VM size](#), this option will automatically populate to "On." If not, it will populate the "Off" option for Accelerated Networking and give the user a reason why it is not be enabled.

- *Note:* Only supported operating systems can be enabled through the portal. If you are using a custom image, and your image supports Accelerated Networking, please create your VM using CLI or Powershell.

After the virtual machine is created, you can confirm Accelerated Networking is enabled by following the instructions in the Confirm that accelerated networking is enabled.

### Powershell creation

### Create a virtual network

## 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Install [Azure PowerShell](#) version 1.0.0 or later. To find your currently installed version, run

```
Get-Module -ListAvailable Az
```

If you need to install or upgrade, install the latest version of the Az module from the [PowerShell Gallery](#). In a PowerShell session, log in to an Azure account using [Connect-AzAccount](#).

In the following examples, replace example parameter names with your own values. Example parameter names included *myResourceGroup*, *myNic*, and *myVM*.

Create a resource group with [New-AzResourceGroup](#). The following example creates a resource group named *myResourceGroup* in the *centralus* location:

```
New-AzResourceGroup -Name "myResourceGroup" -Location "centralus"
```

First, create a subnet configuration with [New-AzVirtualNetworkSubnetConfig](#). The following example creates a subnet named *mySubnet*:

```
$subnet = New-AzVirtualNetworkSubnetConfig `  
-Name "mySubnet" `  
-AddressPrefix "192.168.1.0/24"
```

Create a virtual network with [New-AzVirtualNetwork](#), with the *mySubnet* subnet.

```
$vnet = New-AzVirtualNetwork -ResourceGroupName "myResourceGroup" `  
-Location "centralus" `  
-Name "myVnet" `  
-AddressPrefix "192.168.0.0/16" `  
-Subnet $subnet
```

## Create a network security group

First, create a network security group rule with [New-AzNetworkSecurityRuleConfig](#).

```
$rdp = New-AzNetworkSecurityRuleConfig `  
-Name 'Allow-RDP-All' `  
-Description 'Allow RDP' `  
-Access Allow `  
-Protocol Tcp `  
-Direction Inbound `  
-Priority 100 `  
-SourceAddressPrefix * `  
-SourcePortRange * `  
-DestinationAddressPrefix * `  
-DestinationPortRange 3389
```

Create a network security group with [New-AzNetworkSecurityGroup](#) and assign the *Allow-RDP-All* security rule to it. In addition to the *Allow-RDP-All* rule, the network security group contains several default rules. One default rule disables all inbound access from the Internet, which is why the *Allow-RDP-All* rule is assigned to the network security group so that you can remotely connect to the virtual machine, once it's created.

```
$nsg = New-AzNetworkSecurityGroup `  
    -ResourceGroupName myResourceGroup `  
    -Location centralus `  
    -Name "myNsg" `  
    -SecurityRules $rdp
```

Associate the network security group to the *mySubnet* subnet with [Set-AzVirtualNetworkSubnetConfig](#). The rule in the network security group is effective for all resources deployed in the subnet.

```
Set-AzVirtualNetworkSubnetConfig `  
    -VirtualNetwork $vnet `  
    -Name 'mySubnet' `  
    -AddressPrefix "192.168.1.0/24" `  
    -NetworkSecurityGroup $nsg
```

## Create a network interface with accelerated networking

Create a public IP address with [New-AzPublicIpAddress](#). A public IP address isn't required if you don't plan to access the virtual machine from the Internet, but to complete the steps in this article, it is required.

```
$publicIp = New-AzPublicIpAddress `  
    -ResourceGroupName myResourceGroup `  
    -Name 'myPublicIp' `  
    -location centralus `  
    -AllocationMethod Dynamic
```

Create a network interface with [New-AzNetworkInterface](#) with accelerated networking enabled and assign the public IP address to the network interface. The following example creates a network interface named *myNic* in the *mySubnet* subnet of the *myVnet* virtual network and assigns the *myPublicIp* public IP address to it:

```
$nic = New-AzNetworkInterface `  
    -ResourceGroupName "myResourceGroup" `  
    -Name "myNic" `  
    -Location "centralus" `  
    -SubnetId $vnet.Subnets[0].Id `  
    -PublicIpAddressId $publicIp.Id `  
    -EnableAcceleratedNetworking
```

## Create the virtual machine

Set your VM credentials to the `$cred` variable using [Get-Credential](#):

```
$cred = Get-Credential
```

First, define your VM with [New-AzVMConfig](#). The following example defines a VM named *myVM* with a VM size that supports Accelerated Networking (*Standard\_DS4\_v2*):

```
$vmConfig = New-AzVMConfig -VMName "myVm" -VMSize "Standard_DS4_v2"
```

For a list of all VM sizes and characteristics, see [Windows VM sizes](#).

Create the rest of your VM configuration with [Set-AzVMOperatingSystem](#) and [Set-AzVMSourceImage](#). The following example creates a Windows Server 2016 VM:

```
$vmConfig = Set-AzVMOperatingSystem -VM $vmConfig `  
    -Windows `  
    -ComputerName "myVM" `  
    -Credential $cred `  
    -ProvisionVMAgent `  
    -EnableAutoUpdate  
  
$vmConfig = Set-AzVMSourceImage -VM $vmConfig `  
    -PublisherName "MicrosoftWindowsServer" `  
    -Offer "WindowsServer" `  
    -Skus "2016-Datacenter" `  
    -Version "latest"
```

Attach the network interface that you previously created with [Add-AzVMNetworkInterface](#):

```
$vmConfig = Add-AzVMNetworkInterface -VM $vmConfig -Id $nic.Id
```

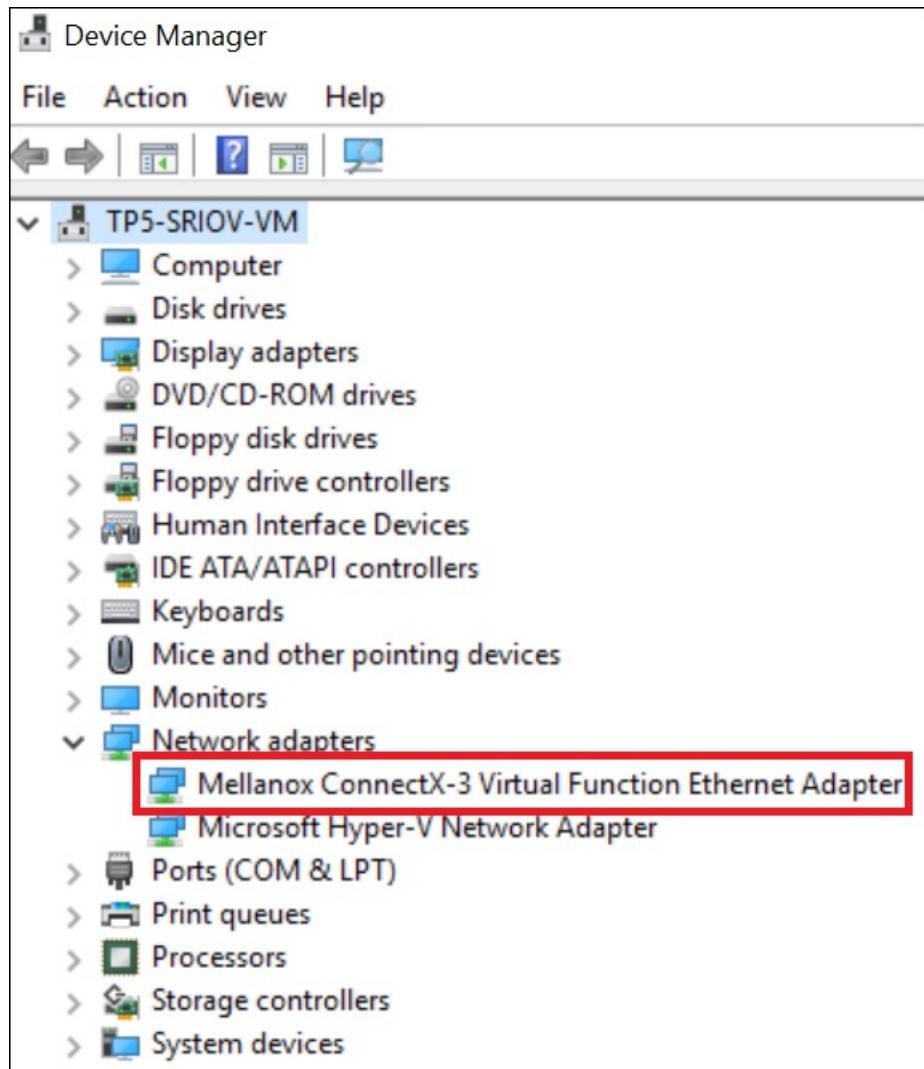
Finally, create your VM with [New-AzVM](#):

```
New-AzVM -VM $vmConfig -ResourceGroupName "myResourceGroup" -Location "centralus"
```

## Confirm the driver is installed in the operating system

Once you create the VM in Azure, connect to the VM and confirm that the driver is installed in Windows.

1. From an Internet browser, open the Azure [portal](#) and sign in with your Azure account.
2. In the box that contains the text *Search resources* at the top of the Azure portal, type *myVm*. When **myVm** appears in the search results, click it. If **Creating** is visible under the **Connect** button, Azure has not yet finished creating the VM. Click **Connect** in the top left corner of the overview only after you no longer see **Creating** under the **Connect** button.
3. Enter the username and password you entered in [Create the virtual machine](#). If you've never connected to a Windows VM in Azure, see [Connect to virtual machine](#).
4. Right-click the Windows Start button and click **Device Manager**. Expand the **Network adapters** node. Confirm that the **Mellanox ConnectX-3 Virtual Function Ethernet Adapter** appears, as shown in the following picture:



Accelerated Networking is now enabled for your VM.

## Enable Accelerated Networking on existing VMs

If you have created a VM without Accelerated Networking, it is possible to enable this feature on an existing VM. The VM must support Accelerated Networking by meeting the following prerequisites that are also outlined above:

- The VM must be a supported size for Accelerated Networking
- The VM must be a supported Azure Gallery image (and kernel version for Linux)
- All VMs in an availability set or VMSS must be stopped/deallocated before enabling Accelerated Networking on any NIC

### Individual VMs & VMs in an availability set

First stop/deallocate the VM or, if an Availability Set, all the VMs in the Set:

```
Stop-AzVM -ResourceGroup "myResourceGroup"  
-Name "myVM"
```

Important, please note, if your VM was created individually, without an availability set, you only need to stop/deallocate the individual VM to enable Accelerated Networking. If your VM was created with an availability set, all VMs contained in the availability set will need to be stopped/deallocated before enabling Accelerated Networking on any of the NICs.

Once stopped, enable Accelerated Networking on the NIC of your VM:

```
$nic = Get-AzNetworkInterface -ResourceGroupName "myResourceGroup" ` 
-Name "myNic"

$nic.EnableAcceleratedNetworking = $true

$nic | Set-AzNetworkInterface
```

Restart your VM or, if in an availability set, all the VMs in the set, and confirm that Accelerated Networking is enabled:

```
Start-AzVM -ResourceGroupName "myResourceGroup" ` 
-Name "myVM"
```

## VMSS

VMSS is slightly different but follows the same workflow. First, stop the VMs:

```
Stop-AzVmss -ResourceGroupName "myResourceGroup" ` 
-VMScaleSetName "myScaleSet"
```

Once the VMs are stopped, update the Accelerated Networking property under the network interface:

```
$vmss = Get-AzVmss -ResourceGroupName "myResourceGroup" ` 
-VMScaleSetName "myScaleSet"

$vmss.VirtualMachineProfile.NetworkProfile.NetworkInterfaceConfigurations[0].EnableAcceleratedNetworking = 
$true

Update-AzVmss -ResourceGroupName "myResourceGroup" ` 
-VMScaleSetName "myScaleSet" ` 
-VirtualMachineScaleSet $vmss
```

Please note, a VMSS has VM upgrades that apply updates using three different settings, automatic, rolling and manual. In these instructions the policy is set to automatic so that the VMSS will pick up the changes immediately after restarting. To set it to automatic so that the changes are immediately picked up:

```
$vmss.UpgradePolicy.AutomaticOSUpgrade = $true

Update-AzVmss -ResourceGroupName "myResourceGroup" ` 
-VMScaleSetName "myScaleSet" ` 
-VirtualMachineScaleSet $vmss
```

Finally, restart the VMSS:

```
Start-AzVmss -ResourceGroupName "myResourceGroup" ` 
-VMScaleSetName "myScaleSet"
```

Once you restart, wait for the upgrades to finish but once completed, the VF will appear inside the VM. (Please make sure you are using a supported OS and VM size)

## Resizing existing VMs with Accelerated Networking

VMs with Accelerated Networking enabled can only be resized to VMs that support Accelerated Networking.

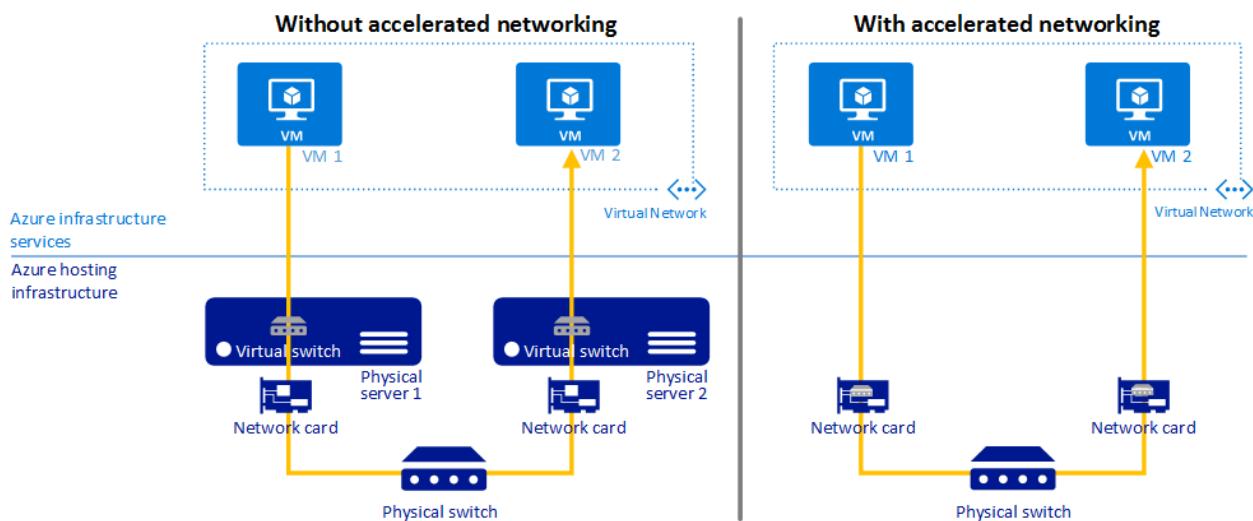
A VM with Accelerated Networking enabled cannot be resized to a VM instance that does not support Accelerated Networking using the resize operation. Instead, to resize one of these VMs:

- Stop/Deallocate the VM or if in an availability set/VMSS, stop/deallocate all the VMs in the set/VMSS.
- Accelerated Networking must be disabled on the NIC of the VM or if in an availability set/VMSS, all VMs in the set/VMSS.
- Once Accelerated Networking is disabled, the VM/availability set/VMSS can be moved to a new size that does not support Accelerated Networking and restarted.

# Create a Linux virtual machine with Accelerated Networking using Azure CLI

12/3/2019 • 10 minutes to read • [Edit Online](#)

In this tutorial, you learn how to create a Linux virtual machine (VM) with Accelerated Networking. To create a Windows VM with Accelerated Networking, see [Create a Windows VM with Accelerated Networking](#). Accelerated networking enables single root I/O virtualization (SR-IOV) to a VM, greatly improving its networking performance. This high-performance path bypasses the host from the datapath, reducing latency, jitter, and CPU utilization, for use with the most demanding network workloads on supported VM types. The following picture shows communication between two VMs with and without accelerated networking:



Without accelerated networking, all networking traffic in and out of the VM must traverse the host and the virtual switch. The virtual switch provides all policy enforcement, such as network security groups, access control lists, isolation, and other network virtualized services to network traffic. To learn more about virtual switches, read the [Hyper-V network virtualization and virtual switch](#) article.

With accelerated networking, network traffic arrives at the virtual machine's network interface (NIC), and is then forwarded to the VM. All network policies that the virtual switch applies are now offloaded and applied in hardware. Applying policy in hardware enables the NIC to forward network traffic directly to the VM, bypassing the host and the virtual switch, while maintaining all the policy it applied in the host.

The benefits of accelerated networking only apply to the VM that it is enabled on. For the best results, it is ideal to enable this feature on at least two VMs connected to the same Azure virtual network (VNet). When communicating across VNets or connecting on-premises, this feature has minimal impact to overall latency.

## Benefits

- **Lower Latency / Higher packets per second (pps):** Removing the virtual switch from the datapath removes the time packets spend in the host for policy processing and increases the number of packets that can be processed inside the VM.
- **Reduced jitter:** Virtual switch processing depends on the amount of policy that needs to be applied and the workload of the CPU that is doing the processing. Offloading the policy enforcement to the hardware removes that variability by delivering packets directly to the VM, removing the host to VM communication and all software interrupts and context switches.
- **Decreased CPU utilization:** Bypassing the virtual switch in the host leads to less CPU utilization for

processing network traffic.

## Supported operating systems

The following distributions are supported out of the box from the Azure Gallery:

- **Ubuntu 14.04 with the linux-azure kernel**
- **Ubuntu 16.04 or later**
- **SLES12 SP3 or later**
- **RHEL 7.4 or later**
- **CentOS 7.4 or later**
- **CoreOS Linux**
- **Debian "Stretch" with backports kernel**
- **Oracle Linux 7.4 and later with Red Hat Compatible Kernel (RHCK)**
- **Oracle Linux 7.5 and later with UEK version 5**
- **FreeBSD 10.4, 11.1 & 12.0**

## Limitations and Constraints

### Supported VM instances

Accelerated Networking is supported on most general purpose and compute-optimized instance sizes with 2 or more vCPUs. These supported series are: D/DSv2 and F/Fs

On instances that support hyperthreading, Accelerated Networking is supported on VM instances with 4 or more vCPUs. Supported series are: D/Dsv3, E/Esv3, Fsv2, Lsv2, Ms/Mms and Ms/Mmsv2.

For more information on VM instances, see [Linux VM sizes](#).

### Custom Images

If you are using a custom image, and your image supports Accelerated Networking, please make sure to have the required drivers to work with Mellanox ConnectX-3 and ConnectX-4 Lx NICs on Azure.

### Regions

Available in all public Azure regions as well as Azure Government Clouds.

### Enabling Accelerated Networking on a running VM

A supported VM size without accelerated networking enabled can only have the feature enabled when it is stopped and deallocated.

### Deployment through Azure Resource Manager

Virtual machines (classic) cannot be deployed with Accelerated Networking.

## Create a Linux VM with Azure Accelerated Networking

### Portal creation

Though this article provides steps to create a virtual machine with accelerated networking using the Azure CLI, you can also [create a virtual machine with accelerated networking using the Azure portal](#). When creating a virtual machine in the portal, in the **Create a virtual machine** blade, choose the **Networking** tab. In this tab, there is an option for **Accelerated networking**. If you have chosen a [supported operating system](#) and [VM size](#), this option will automatically populate to "On." If not, it will populate the "Off" option for Accelerated Networking and give the user a reason why it is not be enabled.

- Note: Only supported operating systems can be enabled through the portal. If you are using a custom image,

and your image supports Accelerated Networking, please create your VM using CLI or Powershell.

After the virtual machine is created, you can confirm Accelerated Networking is enabled by following the instructions in the [Confirm that accelerated networking is enabled](#).

## CLI creation

### Create a virtual network

Install the latest [Azure CLI](#) and log in to an Azure account using [az login](#). In the following examples, replace example parameter names with your own values. Example parameter names included *myResourceGroup*, *myNic*, and *myVm*.

Create a resource group with [az group create](#). The following example creates a resource group named *myResourceGroup* in the *centralus* location:

```
az group create --name myResourceGroup --location centralus
```

Select a supported Linux region listed in [Linux accelerated networking](#).

Create a virtual network with [az network vnet create](#). The following example creates a virtual network named *myVnet* with one subnet:

```
az network vnet create \
  --resource-group myResourceGroup \
  --name myVnet \
  --address-prefix 192.168.0.0/16 \
  --subnet-name mySubnet \
  --subnet-prefix 192.168.1.0/24
```

### Create a network security group

Create a network security group with [az network nsg create](#). The following example creates a network security group named *myNetworkSecurityGroup*:

```
az network nsg create \
  --resource-group myResourceGroup \
  --name myNetworkSecurityGroup
```

The network security group contains several default rules, one of which disables all inbound access from the Internet. Open a port to allow SSH access to the virtual machine with [az network nsg rule create](#):

```
az network nsg rule create \
  --resource-group myResourceGroup \
  --nsg-name myNetworkSecurityGroup \
  --name Allow-SSH-Internet \
  --access Allow \
  --protocol Tcp \
  --direction Inbound \
  --priority 100 \
  --source-address-prefix Internet \
  --source-port-range "*" \
  --destination-address-prefix "*" \
  --destination-port-range 22
```

### Create a network interface with accelerated networking

Create a public IP address with [az network public-ip create](#). A public IP address isn't required if you don't plan to

access the virtual machine from the Internet, but to complete the steps in this article, it is required.

```
az network public-ip create \
--name myPublicIp \
--resource-group myResourceGroup
```

Create a network interface with [az network nic create](#) with accelerated networking enabled. The following example creates a network interface named *myNic* in the *mySubnet* subnet of the *myVnet* virtual network and associates the *myNetworkSecurityGroup* network security group to the network interface:

```
az network nic create \
--resource-group myResourceGroup \
--name myNic \
--vnet-name myVnet \
--subnet mySubnet \
--accelerated-networking true \
--public-ip-address myPublicIp \
--network-security-group myNetworkSecurityGroup
```

## Create a VM and attach the NIC

When you create the VM, specify the NIC you created with `--nics`. Select a size and distribution listed in [Linux accelerated networking](#).

Create a VM with [az vm create](#). The following example creates a VM named *myVM* with the UbuntuLTS image and a size that supports Accelerated Networking (*Standard\_DS4\_v2*):

```
az vm create \
--resource-group myResourceGroup \
--name myVM \
--image UbuntuLTS \
--size Standard_DS4_v2 \
--admin-username azureuser \
--generate-ssh-keys \
--nics myNic
```

For a list of all VM sizes and characteristics, see [Linux VM sizes](#).

Once the VM is created, output similar to the following example output is returned. Take note of the **publicIpAddress**. This address is used to access the VM in subsequent steps.

```
{
  "fqdns": "",
  "id": "/subscriptions/<ID>/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVM",
  "location": "centralus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "192.168.0.4",
  "publicIpAddress": "40.68.254.142",
  "resourceGroup": "myResourceGroup"
}
```

## Confirm that accelerated networking is enabled

Use the following command to create an SSH session with the VM. Replace `<your-public-ip-address>` with the public IP address assigned to the virtual machine you created, and replace *azureuser* if you used a different value for `--admin-username` when you created the VM.

```
ssh azureuser@<your-public-ip-address>
```

From the Bash shell, enter `uname -r` and confirm that the kernel version is one of the following versions, or greater:

- **Ubuntu 16.04:** 4.11.0-1013
- **SLES SP3:** 4.4.92-6.18
- **RHEL:** 7.4.2017120423
- **CentOS:** 7.4.20171206

Confirm the Mellanox VF device is exposed to the VM with the `lspci` command. The returned output is similar to the following output:

```
0000:00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX - 82443BX/ZX/DX Host bridge (AGP disabled) (rev 03)
0000:00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA (rev 01)
0000:00:07.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01)
0000:00:07.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 02)
0000:00:08.0 VGA compatible controller: Microsoft Corporation Hyper-V virtual VGA
0001:00:02.0 Ethernet controller: Mellanox Technologies MT27500/MT27520 Family [ConnectX-3/ConnectX-3 Pro
Virtual Function]
```

Check for activity on the VF (virtual function) with the `ethtool -S eth0 | grep vf_` command. If you receive output similar to the following sample output, accelerated networking is enabled and working.

```
vf_rx_packets: 992956
vf_rx_bytes: 2749784180
vf_tx_packets: 2656684
vf_tx_bytes: 1099443970
vf_tx_dropped: 0
```

Accelerated Networking is now enabled for your VM.

## Handle dynamic binding and revocation of virtual function

Applications must run over the synthetic NIC that is exposed in VM. If the application runs directly over the VF NIC, it doesn't receive **all** packets that are destined to the VM, since some packets show up over the synthetic interface. If you run an application over the synthetic NIC, it guarantees that the application receives **all** packets that are destined to it. It also makes sure that the application keeps running, even if the VF is revoked when the host is being serviced. Applications binding to the synthetic NIC is a **mandatory** requirement for all applications taking advantage of **Accelerated Networking**.

## Enable Accelerated Networking on existing VMs

If you have created a VM without Accelerated Networking, it is possible to enable this feature on an existing VM. The VM must support Accelerated Networking by meeting the following prerequisites that are also outlined above:

- The VM must be a supported size for Accelerated Networking
- The VM must be a supported Azure Gallery image (and kernel version for Linux)
- All VMs in an availability set or VMSS must be stopped/deallocated before enabling Accelerated Networking on any NIC

### Individual VMs & VMs in an availability set

First stop/deallocate the VM or, if an Availability Set, all the VMs in the Set:

```
az vm deallocate \
--resource-group myResourceGroup \
--name myVM
```

Important, please note, if your VM was created individually, without an availability set, you only need to stop/deallocate the individual VM to enable Accelerated Networking. If your VM was created with an availability set, all VMs contained in the availability set will need to be stopped/deallocated before enabling Accelerated Networking on any of the NICs.

Once stopped, enable Accelerated Networking on the NIC of your VM:

```
az network nic update \
--name myNic \
--resource-group myResourceGroup \
--accelerated-networking true
```

Restart your VM or, if in an Availability Set, all the VMs in the Set and confirm that Accelerated Networking is enabled:

```
az vm start --resource-group myResourceGroup \
--name myVM
```

## VMSS

VMSS is slightly different but follows the same workflow. First, stop the VMs:

```
az vmss deallocate \
--name myvmss \
--resource-group myrg
```

Once the VMs are stopped, update the Accelerated Networking property under the network interface:

```
az vmss update --name myvmss \
--resource-group myrg \
--set
virtualMachineProfile.networkProfile.networkInterfaceConfigurations[0].enableAcceleratedNetworking=true
```

Please note, a VMSS has VM upgrades that apply updates using three different settings, automatic, rolling and manual. In these instructions the policy is set to automatic so that the VMSS will pick up the changes immediately after restarting. To set it to automatic so that the changes are immediately picked up:

```
az vmss update \
--name myvmss \
--resource-group myrg \
--set upgradePolicy.mode="automatic"
```

Finally, restart the VMSS:

```
az vmss start \
--name myvmss \
--resource-group myrg
```

Once you restart, wait for the upgrades to finish but once completed, the VF will appear inside the VM. (Please make sure you are using a supported OS and VM size.)

## **Resizing existing VMs with Accelerated Networking**

VMs with Accelerated Networking enabled can only be resized to VMs that support Accelerated Networking.

A VM with Accelerated Networking enabled cannot be resized to a VM instance that does not support Accelerated Networking using the resize operation. Instead, to resize one of these VMs:

- Stop/Deallocate the VM or if in an availability set/VMSS, stop/deallocate all the VMs in the set/VMSS.
- Accelerated Networking must be disabled on the NIC of the VM or if in an availability set/VMSS, all VMs in the set/VMSS.
- Once Accelerated Networking is disabled, the VM/availability set/VMSS can be moved to a new size that does not support Accelerated Networking and restarted.

# Set up DPDK in a Linux virtual machine

12/18/2019 • 6 minutes to read • [Edit Online](#)

Data Plane Development Kit (DPDK) on Azure offers a faster user-space packet processing framework for performance-intensive applications. This framework bypasses the virtual machine's kernel network stack.

In typical packet processing that uses the kernel network stack, the process is interrupt-driven. When the network interface receives incoming packets, there is a kernel interrupt to process the packet and a context switch from the kernel space to the user space. DPDK eliminates context switching and the interrupt-driven method in favor of a user-space implementation that uses poll mode drivers for fast packet processing.

DPDK consists of sets of user-space libraries that provide access to lower-level resources. These resources can include hardware, logical cores, memory management, and poll mode drivers for network interface cards.

DPDK can run on Azure virtual machines that are supporting multiple operating system distributions. DPDK provides key performance differentiation in driving network function virtualization implementations. These implementations can take the form of network virtual appliances (NVAs), such as virtual routers, firewalls, VPNs, load balancers, evolved packet cores, and denial-of-service (DDoS) applications.

## Benefit

**Higher packets per second (PPS):** Bypassing the kernel and taking control of packets in the user space reduces the cycle count by eliminating context switches. It also improves the rate of packets that are processed per second in Azure Linux virtual machines.

## Supported operating systems

The following distributions from the Azure Gallery are supported:

LINUX OS	KERNEL VERSION
Ubuntu 16.04	4.15.0-1015-azure
Ubuntu 18.04	4.15.0-1015-azure
SLES 15	4.12.14-5.5-azure
RHEL 7.5	3.10.0-862.9.1.el7
CentOS 7.5	3.10.0-862.3.3.el7

### Custom kernel support

For any Linux kernel version that's not listed, see [Patches for building an Azure-tuned Linux kernel](#). For more information, you can also contact [azuredpdk@microsoft.com](mailto:azuredpdk@microsoft.com).

## Region support

All Azure regions support DPDK.

## Prerequisites

Accelerated networking must be enabled on a Linux virtual machine. The virtual machine should have at least two network interfaces, with one interface for management. Learn how to [create a Linux virtual machine with accelerated networking enabled](#).

## Install DPDK dependencies

### Ubuntu 16.04

```
sudo add-apt-repository ppa:canonical-server/dpdk-azure -y  
sudo apt-get update  
sudo apt-get install -y librddmacm-dev librddmacm1 build-essential libnuma-dev libmnl-dev
```

### Ubuntu 18.04

```
sudo add-apt-repository ppa:canonical-server/dpdk-azure -y  
sudo apt-get update  
sudo apt-get install -y librddmacm-dev librddmacm1 build-essential libnuma-dev libmnl-dev
```

### RHEL7.5/CentOS 7.5

```
yum -y groupinstall "Infiniband Support"  
sudo dracut --add-drivers "mlx4_en mlx4_ib mlx5_ib" -f  
yum install -y gcc kernel-devel-`uname -r` numactl-devel.x86_64 librddmacm-devel libmnl-devel
```

### SLES 15

#### Azure kernel

```
zypper \  
--no-gpg-checks \  
--non-interactive \  
--gpg-auto-import-keys install kernel-azure kernel-devel-azure gcc make libnuma-devel numactl librddmacm1  
rdma-core-devel
```

#### Default kernel

```
zypper \  
--no-gpg-checks \  
--non-interactive \  
--gpg-auto-import-keys install kernel-default-devel gcc make libnuma-devel numactl librddmacm1 rdma-core-devel
```

## Set up the virtual machine environment (once)

1. [Download the latest DPDK](#). Version 18.02 or higher is required for Azure.
2. Build the default config with `make config T=x86_64-native-linuxapp-gcc`.
3. Enable Mellanox PMDs in the generated config with `sed -ri 's,(MLX_.PMD=)n,\1y,' build/.config`.
4. Compile with `make`.
5. Install with `make install DESTDIR=<output folder>`.

## Configure the runtime environment

After restarting, run the following commands once:

1. Hugepages

- Configure hugepage by running the following command, once for all numanodes:

```
echo 1024 | sudo tee
/sys/devices/system/node/node*/hugepages/hugepages-2048kB/nr_hugepages
```

- Create a directory for mounting with `mkdir /mnt/huge`.
- Mount hugepages with `mount -t hugetlbfs nodev /mnt/huge`.
- Check that hugepages are reserved with `grep Huge /proc/meminfo`.

#### NOTE

There is a way to modify the grub file so that hugepages are reserved on boot by following the [instructions](#) for the DPDK. The instructions are at the bottom of the page. When you're using an Azure Linux virtual machine, modify files under `/etc/config/grub.d` instead, to reserve hugepages across reboots.

- MAC & IP addresses: Use `ifconfig -a` to view the MAC and IP address of the network interfaces. The *VF* network interface and *NETVSC* network interface have the same MAC address, but only the *NETVSC* network interface has an IP address. VF interfaces are running as subordinate interfaces of NETVSC interfaces.
- PCI addresses
- Use `ethtool -i <vf interface name>` to find out which PCI address to use for *VF*.  
If *eth0* has accelerated networking enabled, make sure that testpmd doesn't accidentally take over the VF pci device for *eth0*. If the DPDK application accidentally takes over the management network interface and causes you to lose your SSH connection, use the serial console to stop the DPDK application. You can also use the serial console to stop or start the virtual machine.
- Load *ibuverbs* on each reboot with `modprobe -a ib_uverbs`. For SLES 15 only, also load *mlx4\_ib* with `modprobe -a mlx4_ib`.

## Failsafe PMD

DPDK applications must run over the failsafe PMD that is exposed in Azure. If the application runs directly over the VF PMD, it doesn't receive **all** packets that are destined to the VM, since some packets show up over the synthetic interface.

If you run a DPDK application over the failsafe PMD, it guarantees that the application receives all packets that are destined to it. It also makes sure that the application keeps running in DPDK mode, even if the VF is revoked when the host is being serviced. For more information about failsafe PMD, see [Fail-safe poll mode driver library](#).

## Run testpmd

To run testpmd in root mode, use `sudo` before the *testpmd* command.

### Basic: Sanity check, failsafe adapter initialization

- Run the following commands to start a single port testpmd application:

```
testpmd -w <pci address from previous step> \
--vdev="net_vdev_netvsc0,iface=eth1" \
-- -i \
--port-topology=chained
```

- Run the following commands to start a dual port testpmd application:

```
testpmd -w <pci address nic1> \
-w <pci address nic2> \
--vdev="net_vdev_netvsc0,iface=eth1" \
--vdev="net_vdev_netvsc1,iface=eth2" \
-- -i
```

If you're running testpmd with more than two NICs, the `--vdev` argument follows this pattern:

```
net_vdev_netvsc<id>,iface=<vf's pairing eth>.
```

- After it's started, run `show port info all` to check port information. You should see one or two DPDK ports that are `net_failsafe` (not `net_mlx4`).

- Use `start <port> /stop <port>` to start traffic.

The previous commands start `testpmd` in interactive mode, which is recommended for trying out testpmd commands.

### Basic: Single sender/single receiver

The following commands periodically print the packets per second statistics:

- On the TX side, run the following command:

```
testpmd \
-l <core-list> \
-n <num of mem channels> \
-w <pci address of the device you plan to use> \
--vdev="net_vdev_netvsc<id>,iface=<the iface to attach to>" \
-- --port-topology=chained \
--nb-cores <number of cores to use for test pmd> \
--forward-mode=txonly \
--eth-peer=<port id>,<receiver peer MAC address> \
--stats-period <display interval in seconds>
```

- On the RX side, run the following command:

```
testpmd \
-l <core-list> \
-n <num of mem channels> \
-w <pci address of the device you plan to use> \
--vdev="net_vdev_netvsc<id>,iface=<the iface to attach to>" \
-- --port-topology=chained \
--nb-cores <number of cores to use for test pmd> \
--forward-mode=rxonly \
--eth-peer=<port id>,<sender peer MAC address> \
--stats-period <display interval in seconds>
```

When you're running the previous commands on a virtual machine, change `IP_SRC_ADDR` and `IP_DST_ADDR` in `app/test-pmd/txonly.c` to match the actual IP address of the virtual machines before you compile. Otherwise, the packets are dropped before reaching the receiver.

### Advanced: Single sender/single forwarder

The following commands periodically print the packets per second statistics:

- On the TX side, run the following command:

```
testpmd \
-l <core-list> \
-n <num of mem channels> \
-w <pci address of the device you plan to use> \
--vdev="net_vdev_netvsc<id>,iface=<the iface to attach to>" \
--port-topology=chained \
--nb-cores <number of cores to use for test pmd> \
--forward-mode=txonly \
--eth-peer=<port id>,<receiver peer MAC address> \
--stats-period <display interval in seconds>
```

2. On the FWD side, run the following command:

```
testpmd \
-l <core-list> \
-n <num of mem channels> \
-w <pci address NIC1> \
-w <pci address NIC2> \
--vdev="net_vdev_netvsc<id>,iface=<the iface to attach to>" \
--vdev="net_vdev_netvsc<2nd id>,iface=<2nd iface to attach to>" (you need as many --vdev arguments as
the number of devices used by testpmd, in this case) \
--nb-cores <number of cores to use for test pmd> \
--forward-mode=io \
--eth-peer=<recv port id>,<sender peer MAC address> \
--stats-period <display interval in seconds>
```

When you're running the previous commands on a virtual machine, change *IP\_SRC\_ADDR* and *IP\_DST\_ADDR* in `app/test-pmd/txonly.c` to match the actual IP address of the virtual machines before you compile. Otherwise, the packets are dropped before reaching the forwarder. You won't be able to have a third machine receive forwarded traffic, because the *testpmd* forwarder doesn't modify the layer-3 addresses, unless you make some code changes.

## References

- [EAL options](#)
- [Testpmd commands](#)

# TCP/IP performance tuning for Azure VMs

7/17/2019 • 23 minutes to read • [Edit Online](#)

This article discusses common TCP/IP performance tuning techniques and some things to consider when you use them for virtual machines running on Azure. It will provide a basic overview of the techniques and explore how they can be tuned.

## Common TCP/IP tuning techniques

### **MTU, fragmentation, and large send offload**

#### **MTU**

The maximum transmission unit (MTU) is the largest size frame (packet), specified in bytes, that can be sent over a network interface. The MTU is a configurable setting. The default MTU used on Azure VMs, and the default setting on most network devices globally, is 1,500 bytes.

#### **Fragmentation**

Fragmentation occurs when a packet is sent that exceeds the MTU of a network interface. The TCP/IP stack will break the packet into smaller pieces (fragments) that conform to the interface's MTU. Fragmentation occurs at the IP layer and is independent of the underlying protocol (such as TCP). When a 2,000-byte packet is sent over a network interface with an MTU of 1,500, the packet will be broken down into one 1,500-byte packet and one 500-byte packet.

Network devices in the path between a source and destination can either drop packets that exceed the MTU or fragment the packet into smaller pieces.

#### **The Don't Fragment bit in an IP packet**

The Don't Fragment (DF) bit is a flag in the IP protocol header. The DF bit indicates that network devices on the path between the sender and receiver must not fragment the packet. This bit could be set for many reasons. (See the "Path MTU Discovery" section of this article for one example.) When a network device receives a packet with the Don't Fragment bit set, and that packet exceeds the device's interface MTU, the standard behavior is for the device to drop the packet. The device sends an ICMP Fragmentation Needed message back to the original source of the packet.

#### **Performance implications of fragmentation**

Fragmentation can have negative performance implications. One of the main reasons for the effect on performance is the CPU/memory impact of the fragmentation and reassembly of packets. When a network device needs to fragment a packet, it will have to allocate CPU/memory resources to perform fragmentation.

The same thing happens when the packet is reassembled. The network device has to store all the fragments until they're received so it can reassemble them into the original packet. This process of fragmentation and reassembly can also cause latency.

The other possible negative performance implication of fragmentation is that fragmented packets might arrive out of order. When packets are received out of order, some types of network devices can drop them. When that happens, the whole packet has to be retransmitted.

Fragments are typically dropped by security devices like network firewalls or when a network device's receive buffers are exhausted. When a network device's receive buffers are exhausted, a network device is attempting to reassemble a fragmented packet but doesn't have the resources to store and reassemble the packet.

Fragmentation can be seen as a negative operation, but support for fragmentation is necessary when you're connecting diverse networks over the internet.

## **Benefits and consequences of modifying the MTU**

Generally speaking, you can create a more efficient network by increasing the MTU. Every packet that's transmitted has header information that's added to the original packet. When fragmentation creates more packets, there's more header overhead, and that makes the network less efficient.

Here's an example. The Ethernet header size is 14 bytes plus a 4-byte frame check sequence to ensure frame consistency. If one 2,000-byte packet is sent, 18 bytes of Ethernet overhead is added on the network. If the packet is fragmented into a 1,500-byte packet and a 500-byte packet, each packet will have 18 bytes of Ethernet header, a total of 36 bytes.

Keep in mind that increasing the MTU won't necessarily create a more efficient network. If an application sends only 500-byte packets, the same header overhead will exist whether the MTU is 1,500 bytes or 9,000 bytes. The network will become more efficient only if it uses larger packet sizes that are affected by the MTU.

### **Azure and VM MTU**

The default MTU for Azure VMs is 1,500 bytes. The Azure Virtual Network stack will attempt to fragment a packet at 1,400 bytes.

Note that the Virtual Network stack isn't inherently inefficient because it fragments packets at 1,400 bytes even though VMs have an MTU of 1,500. A large percentage of network packets are much smaller than 1,400 or 1,500 bytes.

### **Azure and fragmentation**

Virtual Network stack is set up to drop "out of order fragments," that is, fragmented packets that don't arrive in their original fragmented order. These packets are dropped mainly because of a network security vulnerability announced in November 2018 called FragmentSmack.

FragmentSmack is a defect in the way the Linux kernel handled reassembly of fragmented IPv4 and IPv6 packets. A remote attacker could use this flaw to trigger expensive fragment reassembly operations, which could lead to increased CPU and a denial of service on the target system.

### **Tune the MTU**

You can configure an Azure VM MTU, as you can in any other operating system. But you should consider the fragmentation that occurs in Azure, described above, when you're configuring an MTU.

We don't encourage customers to increase VM MTUs. This discussion is meant to explain the details of how Azure implements MTU and performs fragmentation.

#### **IMPORTANT**

Increasing MTU isn't known to improve performance and could have a negative effect on application performance.

### **Large send offload**

Large send offload (LSO) can improve network performance by offloading the segmentation of packets to the Ethernet adapter. When LSO is enabled, the TCP/IP stack creates a large TCP packet and sends it to the Ethernet adapter for segmentation before forwarding it. The benefit of LSO is that it can free the CPU from segmenting packets into sizes that conform to the MTU and offload that processing to the Ethernet interface where it's performed in hardware. To learn more about the benefits of LSO, see [Supporting large send offload](#).

When LSO is enabled, Azure customers might see large frame sizes when they perform packet captures. These large frame sizes might lead some customers to think fragmentation is occurring or that a large MTU is being used when it's not. With LSO, the Ethernet adapter can advertise a larger maximum segment size (MSS) to the TCP/IP stack to create a larger TCP packet. This entire non-segmented frame is then forwarded to the Ethernet adapter and would be visible in a packet capture performed on the VM. But the packet will be broken down into many smaller frames by the Ethernet adapter, according to the Ethernet adapter's MTU.

### **TCP MSS window scaling and PMTUD**

## TCP maximum segment size

TCP maximum segment size (MSS) is a setting that limits the size of TCP segments, which avoids fragmentation of TCP packets. Operating systems will typically use this formula to set MSS:

$$\text{MSS} = \text{MTU} - (\text{IP header size} + \text{TCP header size})$$

The IP header and the TCP header are 20 bytes each, or 40 bytes total. So an interface with an MTU of 1,500 will have an MSS of 1,460. But the MSS is configurable.

This setting is agreed to in the TCP three-way handshake when a TCP session is set up between a source and a destination. Both sides send an MSS value, and the lower of the two is used for the TCP connection.

Keep in mind that the MTUs of the source and destination aren't the only factors that determine the MSS value. Intermediary network devices, like VPN gateways, including Azure VPN Gateway, can adjust the MTU independently of the source and destination to ensure optimal network performance.

### Path MTU Discovery

MSS is negotiated, but it might not indicate the actual MSS that can be used. This is because other network devices in the path between the source and the destination might have a lower MTU value than the source and destination. In this case, the device whose MTU is smaller than the packet will drop the packet. The device will send back an ICMP Fragmentation Needed (Type 3, Code 4) message that contains its MTU. This ICMP message allows the source host to reduce its Path MTU appropriately. The process is called Path MTU Discovery (PMTUD).

The PMTUD process is inefficient and affects network performance. When packets are sent that exceed a network path's MTU, the packets need to be retransmitted with a lower MSS. If the sender doesn't receive the ICMP Fragmentation Needed message, maybe because of a network firewall in the path (commonly referred to as a *PMTUD blackhole*), the sender doesn't know it needs to lower the MSS and will continuously retransmit the packet. This is why we don't recommend increasing the Azure VM MTU.

### VPN and MTU

If you use VMs that perform encapsulation (like IPsec VPNs), there are some additional considerations regarding packet size and MTU. VPNs add more headers to packets, which increases the packet size and requires a smaller MSS.

For Azure, we recommend that you set TCP MSS clamping to 1,350 bytes and tunnel interface MTU to 1,400. For more information, see the [VPN devices and IPsec/IKE parameters page](#).

## Latency, round-trip time, and TCP window scaling

### Latency and round-trip time

Network latency is governed by the speed of light over a fiber optic network. Network throughput of TCP is also effectively governed by the round-trip time (RTT) between two network devices.

Route	Distance	One-way time	RTT
New York to San Francisco	4,148 km	21 ms	42 ms
New York to London	5,585 km	28 ms	56 ms
New York to Sydney	15,993 km	80 ms	160 ms

This table shows the straight-line distance between two locations. In networks, the distance is typically longer than the straight-line distance. Here's a simple formula to calculate minimum RTT as governed by the speed of light:

$$\text{minimum RTT} = 2 * (\text{Distance in kilometers} / \text{Speed of propagation})$$

You can use 200 for the speed of propagation. This is the distance, in meters, that light travels in 1 millisecond.

Let's take New York to San Francisco as an example. The straight-line distance is 4,148 km. Plugging that value into the equation, we get the following:

$$\text{Minimum RTT} = 2 * (4,148 / 200)$$

The output of the equation is in milliseconds.

If you want to get the best network performance, the logical option is to select destinations with the shortest distance between them. You should also design your virtual network to optimize the path of traffic and reduce latency. For more information, see the "Network design considerations" section of this article.

#### Latency and round-trip time effects on TCP

Round-trip time has a direct effect on maximum TCP throughput. In TCP protocol, *window size* is the maximum amount of traffic that can be sent over a TCP connection before the sender needs to receive acknowledgement from the receiver. If the TCP MSS is set to 1,460 and the TCP window size is set to 65,535, the sender can send 45 packets before it has to receive acknowledgement from the receiver. If the sender doesn't get acknowledgement, it will retransmit the data. Here's the formula:

$$\text{TCP window size} / \text{TCP MSS} = \text{packets sent}$$

In this example, 65,535 / 1,460 is rounded up to 45.

This "waiting for acknowledgement" state, a mechanism to ensure reliable delivery of data, is what causes RTT to affect TCP throughput. The longer the sender waits for acknowledgement, the longer it needs to wait before sending more data.

Here's the formula for calculating the maximum throughput of a single TCP connection:

$$\text{Window size} / (\text{RTT latency in milliseconds} / 1,000) = \text{maximum bytes/second}$$

This table shows the maximum megabytes/second throughput of a single TCP connection. (For readability, megabytes is used for the unit of measure.)

TCP window size (bytes)	RTT latency (ms)	Maximum megabyte/second throughput	Maximum megabit/second throughput
65,535	1	65.54	524.29
65,535	30	2.18	17.48
65,535	60	1.09	8.74
65,535	90	.73	5.83
65,535	120	.55	4.37

If packets are lost, the maximum throughput of a TCP connection will be reduced while the sender retransmits data it has already sent.

#### TCP window scaling

TCP window scaling is a technique that dynamically increases the TCP window size to allow more data to be sent before an acknowledgement is required. In the previous example, 45 packets would be sent before an acknowledgement was required. If you increase the number of packets that can be sent before an acknowledgement is needed, you're reducing the number of times a sender is waiting for acknowledgement, which increases the TCP maximum throughput.

This table illustrates those relationships:

TCP window size (bytes)	RTT latency (ms)	Maximum megabyte/second throughput	Maximum megabit/second throughput
65,535	30	2.18	17.48
131,070	30	4.37	34.95
262,140	30	8.74	69.91
524,280	30	17.48	139.81

But the TCP header value for TCP window size is only 2 bytes long, which means the maximum value for a receive window is 65,535. To increase the maximum window size, a TCP window scale factor was introduced.

The scale factor is also a setting that you can configure in an operating system. Here's the formula for calculating the TCP window size by using scale factors:

```
TCP window size = TCP window size in bytes \* (2^scale factor)
```

Here's the calculation for a window scale factor of 3 and a window size of 65,535:

```
65,535 \* (2^3) = 262,140 bytes
```

A scale factor of 14 results in a TCP window size of 14 (the maximum offset allowed). The TCP window size will be 1,073,725,440 bytes (8.5 gigabits).

#### Support for TCP window scaling

Windows can set different scaling factors for different connection types. (Classes of connections include datacenter, internet, and so on.) You use the `Get-NetTCPConnection` PowerShell command to view the window scaling connection type:

```
Get-NetTCPConnection
```

You can use the `Get-NetTCPSetting` PowerShell command to view the values of each class:

```
Get-NetTCPSetting
```

You can set the initial TCP window size and TCP scaling factor in Windows by using the `Set-NetTCPSetting` PowerShell command. For more information, see [Set-NetTCPSetting](#).

```
Set-NetTCPSetting
```

These are the effective TCP settings for `AutoTuningLevel`:

AutoTuningLevel	Scaling factor	Scaling multiplier	Formula to calculate maximum window size

Disabled	None	None	Window size
Restricted	4	$2^4$	Window size * ( $2^4$ )
Highly restricted	2	$2^2$	Window size * ( $2^2$ )
Normal	8	$2^8$	Window size * ( $2^8$ )
Experimental	14	$2^{14}$	Window size * ( $2^{14}$ )

These settings are the most likely to affect TCP performance, but keep in mind that many other factors across the internet, outside the control of Azure, can also affect TCP performance.

#### Increase MTU size

Because a larger MTU means a larger MSS, you might wonder whether increasing the MTU can increase TCP performance. Probably not. There are pros and cons to packet size beyond just TCP traffic. As discussed earlier, the most important factors affecting TCP throughput performance are TCP window size, packet loss, and RTT.

#### IMPORTANT

We don't recommend that Azure customers change the default MTU value on virtual machines.

## Accelerated networking and receive side scaling

### Accelerated networking

Virtual machine network functions have historically been CPU intensive on both the guest VM and the hypervisor/host. Every packet that transits through the host is processed in software by the host CPU, including all virtual network encapsulation and decapsulation. So the more traffic that goes through the host, the higher the CPU load. And if the host CPU is busy with other operations, that will also affect network throughput and latency. Azure addresses this issue with accelerated networking.

Accelerated networking provides consistent ultralow network latency via the in-house programmable hardware of Azure and technologies like SR-IOV. Accelerated networking moves much of the Azure software-defined networking stack off the CPUs and into FPGA-based SmartNICs. This change enables end-user applications to reclaim compute cycles, which puts less load on the VM, decreasing jitter and inconsistency in latency. In other words, performance can be more deterministic.

Accelerated networking improves performance by allowing the guest VM to bypass the host and establish a datapath directly with a host's SmartNIC. Here are some benefits of accelerated networking:

- **Lower latency / higher packets per second (pps):** Removing the virtual switch from the datapath eliminates the time packets spend in the host for policy processing and increases the number of packets that can be processed in the VM.
- **Reduced jitter:** Virtual switch processing depends on the amount of policy that needs to be applied and the workload of the CPU that's doing the processing. Offloading the policy enforcement to the hardware removes that variability by delivering packets directly to the VM, eliminating the host-to-VM communication and all software interrupts and context switches.
- **Decreased CPU utilization:** Bypassing the virtual switch in the host leads to less CPU utilization for processing network traffic.

To use accelerated networking, you need to explicitly enable it on each applicable VM. See [Create a Linux virtual machine with Accelerated Networking](#) for instructions.

## **Receive side scaling**

Receive side scaling (RSS) is a network driver technology that distributes the receiving of network traffic more efficiently by distributing receive processing across multiple CPUs in a multiprocessor system. In simple terms, RSS allows a system to process more received traffic because it uses all available CPUs instead of just one. For a more technical discussion of RSS, see [Introduction to receive side scaling](#).

To get the best performance when accelerated networking is enabled on a VM, you need to enable RSS. RSS can also provide benefits on VMs that don't use accelerated networking. For an overview of how to determine if RSS is enabled and how to enable it, see [Optimize network throughput for Azure virtual machines](#).

## **TCP TIME\_WAIT and TIME\_WAIT assassination**

TCP TIME\_WAIT is another common setting that affects network and application performance. On busy VMs that are opening and closing many sockets, either as clients or as servers (Source IP:Source Port + Destination IP:Destination Port), during the normal operation of TCP, a given socket can end up in a TIME\_WAIT state for a long time. The TIME\_WAIT state is meant to allow any additional data to be delivered on a socket before closing it. So TCP/IP stacks generally prevent the reuse of a socket by silently dropping the client's TCP SYN packet.

The amount of time a socket is in TIME\_WAIT is configurable. It could range from 30 seconds to 240 seconds. Sockets are a finite resource, and the number of sockets that can be used at any given time is configurable. (The number of available sockets is typically about 30,000.) If the available sockets are consumed, or if clients and servers have mismatched TIME\_WAIT settings, and a VM tries to reuse a socket in a TIME\_WAIT state, new connections will fail as TCP SYN packets are silently dropped.

The value for port range for outbound sockets is usually configurable within the TCP/IP stack of an operating system. The same thing is true for TCP TIME\_WAIT settings and socket reuse. Changing these numbers can potentially improve scalability. But, depending on the situation, these changes could cause interoperability issues. You should be careful if you change these values.

You can use TIME\_WAIT assassination to address this scaling limitation. TIME\_WAIT assassination allows a socket to be reused in certain situations, like when the sequence number in the IP packet of the new connection exceeds the sequence number of the last packet from the previous connection. In this case, the operating system will allow the new connection to be established (it will accept the new SYN/ACK) and force close the previous connection that was in a TIME\_WAIT state. This capability is supported on Windows VMs in Azure. To learn about support in other VMs, check with the OS vendor.

To learn about configuring TCP TIME\_WAIT settings and source port range, see [Settings that can be modified to improve network performance](#).

# Virtual network factors that can affect performance

## **VM maximum outbound throughput**

Azure provides a variety of VM sizes and types, each with a different mix of performance capabilities. One of these capabilities is network throughput (or bandwidth), which is measured in megabits per second (Mbps). Because virtual machines are hosted on shared hardware, the network capacity needs to be shared fairly among the virtual machines using the same hardware. Larger virtual machines are allocated more bandwidth than smaller virtual machines.

The network bandwidth allocated to each virtual machine is metered on egress (outbound) traffic from the virtual machine. All network traffic leaving the virtual machine is counted toward the allocated limit, regardless of destination. For example, if a virtual machine has a 1,000-Mbps limit, that limit applies whether the outbound traffic is destined for another virtual machine in the same virtual network or one outside of Azure.

Ingress is not metered or limited directly. But there are other factors, like CPU and storage limits, that can affect a virtual machine's ability to process incoming data.

Accelerated networking is designed to improve network performance, including latency, throughput, and CPU

utilization. Accelerated networking can improve a virtual machine's throughput, but it can do that only up to the virtual machine's allocated bandwidth.

Azure virtual machines have at least one network interface attached to them. They might have several. The bandwidth allocated to a virtual machine is the sum of all outbound traffic across all network interfaces attached to the machine. In other words, the bandwidth is allocated on a per-virtual machine basis, regardless of how many network interfaces are attached to the machine.

Expected outbound throughput and the number of network interfaces supported by each VM size are detailed in [Sizes for Windows virtual machines in Azure](#). To see maximum throughput, select a type, like **General purpose**, and then find the section about the size series on the resulting page (for example, "Dv2-series"). For each series, there's a table that provides networking specifications in the last column, which is titled "Max NICs / Expected network bandwidth (Mbps)."

The throughput limit applies to the virtual machine. Throughput is not affected by these factors:

- **Number of network interfaces:** The bandwidth limit applies to the sum of all outbound traffic from the virtual machine.
- **Accelerated networking:** Though this feature can be helpful in achieving the published limit, it doesn't change the limit.
- **Traffic destination:** All destinations count toward the outbound limit.
- **Protocol:** All outbound traffic over all protocols counts towards the limit.

For more information, see [Virtual machine network bandwidth](#).

## Internet performance considerations

As discussed throughout this article, factors on the internet and outside the control of Azure can affect network performance. Here are some of those factors:

- **Latency:** The round-trip time between two destinations can be affected by issues on intermediate networks, by traffic that doesn't take the "shortest" distance path, and by suboptimal peering paths.
- **Packet loss:** Packet loss can be caused by network congestion, physical path issues, and underperforming network devices.
- **MTU size/Fragmentation:** Fragmentation along the path can lead to delays in data arrival or in packets arriving out of order, which can affect the delivery of packets.

Traceroute is a good tool for measuring network performance characteristics (like packet loss and latency) along every network path between a source device and a destination device.

## Network design considerations

Along with the considerations discussed earlier in this article, the topology of a virtual network can affect the network's performance. For example, a hub-and-spoke design that backhauls traffic globally to a single-hub virtual network will introduce network latency, which will affect overall network performance.

The number of network devices that network traffic passes through can also affect overall latency. For example, in a hub-and-spoke design, if traffic passes through a spoke network virtual appliance and a hub virtual appliance before transiting to the internet, the network virtual appliances can introduce latency.

## Azure regions, virtual networks, and latency

Azure regions are made up of multiple datacenters that exist within a general geographic area. These datacenters might not be physically next to each other. In some cases they're separated by as much as 10 kilometers. The virtual network is a logical overlay on top of the Azure physical datacenter network. A virtual network doesn't imply any specific network topology within the datacenter.

For example, two VMs that are in the same virtual network and subnet might be in different racks, rows, or even datacenters. They could be separated by feet of fiber optic cable or by kilometers of fiber optic cable. This variation could introduce variable latency (a few milliseconds difference) between different VMs.

The geographic placement of VMs, and the potential resulting latency between two VMs, can be influenced by the configuration of availability sets and Availability Zones. But the distance between datacenters in a region is region-specific and primarily influenced by datacenter topology in the region.

### Source NAT port exhaustion

A deployment in Azure can communicate with endpoints outside of Azure on the public internet and/or in the public IP space. When an instance initiates an outbound connection, Azure dynamically maps the private IP address to a public IP address. After Azure creates this mapping, return traffic for the outbound originated flow can also reach the private IP address where the flow originated.

For every outbound connection, the Azure Load Balancer needs to maintain this mapping for some period of time. With the multitenant nature of Azure, maintaining this mapping for every outbound flow for every VM can be resource intensive. So there are limits that are set and based on the configuration of the Azure Virtual Network. Or, to say that more precisely, an Azure VM can only make a certain number of outbound connections at a given time. When these limits are reached, the VM won't be able to make more outbound connections.

But this behavior is configurable. For more information about SNAT and SNAT port exhaustion, see [this article](#).

## Measure network performance on Azure

A number of the performance maximums in this article are related to the network latency / round-trip time (RTT) between two VMs. This section provides some suggestions for how to test latency/RTT and how to test TCP performance and VM network performance. You can tune and performance test the TCP/IP and network values discussed earlier by using the techniques described in this section. You can plug latency, MTU, MSS, and window size values into the calculations provided earlier and compare theoretical maximums to actual values that you observe during testing.

### Measure round-trip time and packet loss

TCP performance relies heavily on RTT and packet Loss. The PING utility available in Windows and Linux provides the easiest way to measure RTT and packet loss. The output of PING will show the minimum/maximum/average latency between a source and destination. It will also show packet loss. PING uses the ICMP protocol by default. You can use PsPing to test TCP RTT. For more information, see [PsPing](#).

### Measure actual throughput of a TCP connection

NTtcp is a tool for testing the TCP performance of a Linux or Windows VM. You can change various TCP settings and then test the benefits by using NTtcp. For more information, see these resources:

- [Bandwidth/Throughput testing \(NTtcp\)](#)
- [NTtcp Utility](#)

### Measure actual bandwidth of a virtual machine

You can test the performance of different VM types, accelerated networking, and so on, by using a tool called iPerf. iPerf is also available on Linux and Windows. iPerf can use TCP or UDP to test overall network throughput. iPerf TCP throughput tests are influenced by the factors discussed in this article (like latency and RTT). So UDP might yield better results if you just want to test maximum throughput.

For more information, see these articles:

- [Troubleshooting Expressroute network performance](#)
- [How to validate VPN throughput to a virtual network](#)

## **Detect inefficient TCP behaviors**

In packet captures, Azure customers might see TCP packets with TCP flags (SACK, DUP ACK, RETRANSMIT, and FAST RETRANSMIT) that could indicate network performance problems. These packets specifically indicate network inefficiencies that result from packet loss. But packet loss isn't necessarily caused by Azure performance problems. Performance problems could be the result of application problems, operating system problems, or other problems that might not be directly related to the Azure platform.

Also, keep in mind that some retransmission and duplicate ACKs are normal on a network. TCP protocols were built to be reliable. Evidence of these TCP packets in a packet capture doesn't necessarily indicate a systemic network problem, unless they're excessive.

Still, these packet types are indications that TCP throughput isn't achieving its maximum performance, for reasons discussed in other sections of this article.

## **Next steps**

Now that you've learned about TCP/IP performance tuning for Azure VMs, you might want to read about other considerations for [planning virtual networks](#) or [learn more about connecting and configuring virtual networks](#).

# Virtual machine network bandwidth

10/22/2019 • 3 minutes to read • [Edit Online](#)

Azure offers a variety of VM sizes and types, each with a different mix of performance capabilities. One capability is network throughput (or bandwidth), measured in megabits per second (Mbps). Because virtual machines are hosted on shared hardware, the network capacity must be shared fairly among the virtual machines sharing the same hardware. Larger virtual machines are allocated relatively more bandwidth than smaller virtual machines.

The network bandwidth allocated to each virtual machine is metered on egress (outbound) traffic from the virtual machine. All network traffic leaving the virtual machine is counted toward the allocated limit, regardless of destination. For example, if a virtual machine has a 1,000 Mbps limit, that limit applies whether the outbound traffic is destined for another virtual machine in the same virtual network, or outside of Azure.

Ingress is not metered or limited directly. However, there are other factors, such as CPU and storage limits, which can impact a virtual machine's ability to process incoming data.

Accelerated networking is a feature designed to improve network performance, including latency, throughput, and CPU utilization. While accelerated networking can improve a virtual machine's throughput, it can do so only up to the virtual machine's allocated bandwidth. To learn more about Accelerated networking, see Accelerated networking for [Windows](#) or [Linux](#) virtual machines.

Azure virtual machines must have one, but may have several, network interfaces attached to them. Bandwidth allocated to a virtual machine is the sum of all outbound traffic across all network interfaces attached to a virtual machine. In other words, the allocated bandwidth is per virtual machine, regardless of how many network interfaces are attached to the virtual machine. To learn how many network interfaces different Azure VM sizes support, see Azure [Windows](#) and [Linux](#) VM sizes.

## Expected network throughput

Expected outbound throughput and the number of network interfaces supported by each VM size is detailed in Azure [Windows](#) and [Linux](#) VM sizes. Select a type, such as General purpose, then select a size-series on the resulting page, such as the Dv2-series. Each series has a table with networking specifications in the last column titled, **Max NICs / Expected network performance (Mbps)**.

The throughput limit applies to the virtual machine. Throughput is unaffected by the following factors:

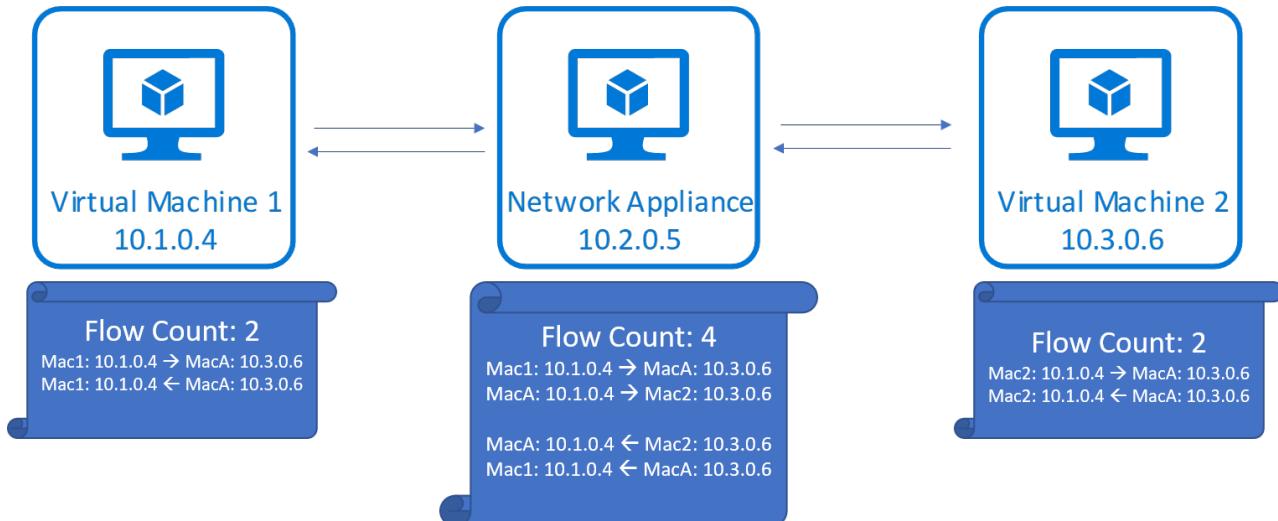
- **Number of network interfaces:** The bandwidth limit is cumulative of all outbound traffic from the virtual machine.
- **Accelerated networking:** Though the feature can be helpful in achieving the published limit, it does not change the limit.
- **Traffic destination:** All destinations count toward the outbound limit.
- **Protocol:** All outbound traffic over all protocols counts towards the limit.

## Network Flow Limits

In addition to bandwidth, the number of network connections present on a VM at any given time can affect its network performance. The Azure networking stack maintains state for each direction of a TCP/UDP connection in data structures called 'flows'. A typical TCP/UDP connection will have 2 flows created, one for the inbound and another for the outbound direction.

Data transfer between endpoints requires creation of several flows in addition to those that perform the data transfer. Some examples are flows created for DNS resolution and flows created for load balancer health probes.

Also note that network virtual appliances (NVAs) such as gateways, proxies, firewalls, will see flows being created for connections terminated at the appliance and originated by the appliance.

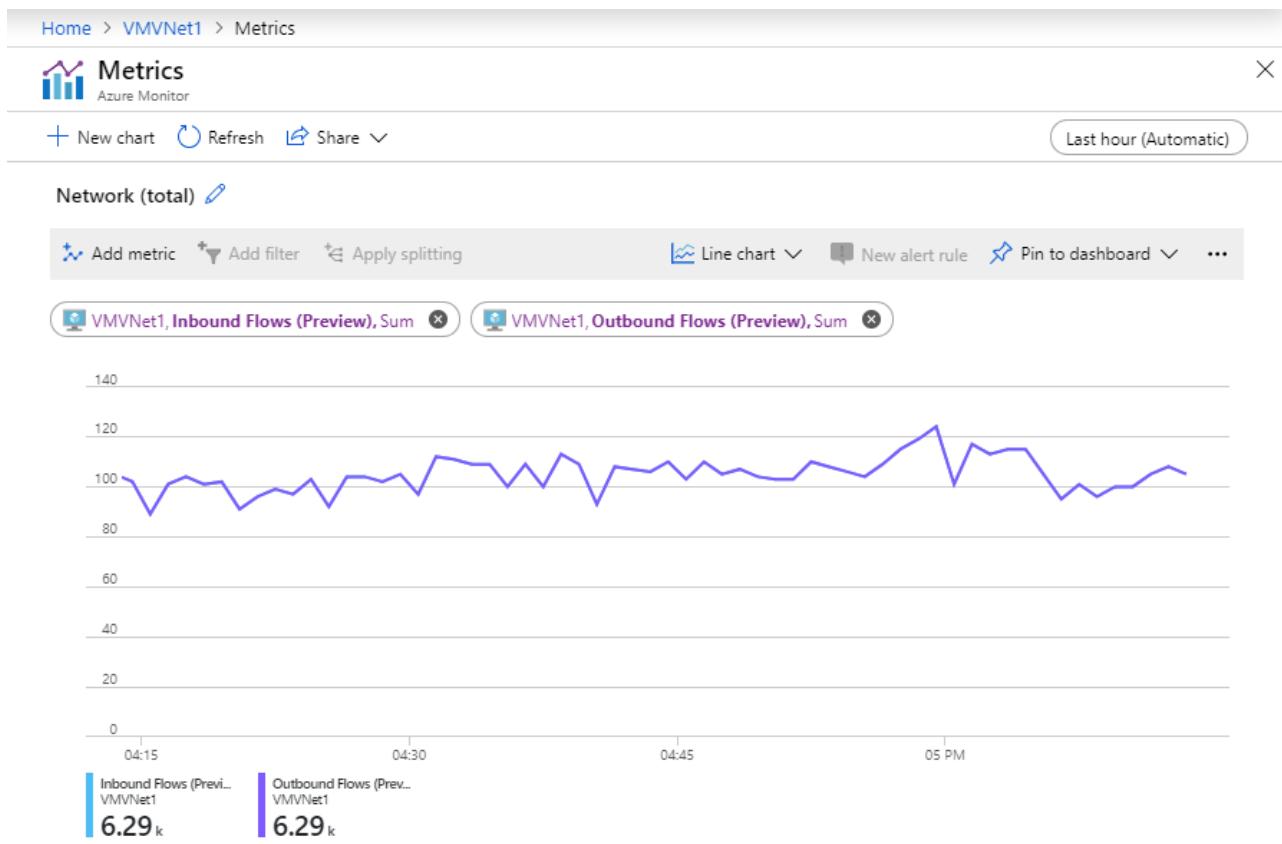


## Flow Limits and Recommendations

Today, the Azure networking stack supports 250K total network flows with good performance for VMs with greater than 8 CPU cores and 100k total flows with good performance for VMs with fewer than 8 CPU cores. Past this limit network performance degrades gracefully for additional flows up to a hard limit of 500K total flows, 250K inbound and 250K outbound, after which additional flows are dropped.

	VMS WITH <8 CPU CORES	VMS WITH 8+ CPU CORES
<b>Good Performance</b>	100K Flows	250K Flows
<b>Degraded Performance</b>	Above 100k Flows	Above 250K Flows
<b>Flow Limit</b>	500K Flows	500K Flows

Metrics are available in [Azure Monitor](#) to track the number of network flows and the flow creation rate on your VM or VMSS instances.



Connection establishment and termination rates can also affect network performance as connection establishment and termination shares CPU with packet processing routines. We recommend that you benchmark workloads against expected traffic patterns and scale out workloads appropriately to match your performance needs.

## Next steps

- Optimize network throughput for a virtual machine operating system
- Test network throughput for a virtual machine.

# Move Azure network security group (NSG) to another region using the Azure portal

1/3/2020 • 4 minutes to read • [Edit Online](#)

There are various scenarios in which you'd want to move your existing NSGs from one region to another. For example, you may want to create an NSG with the same configuration and security rules for testing. You may also want to move an NSG to another region as part of disaster recovery planning.

Azure security groups can't be moved from one region to another. You can however, use an Azure Resource Manager template to export the existing configuration and security rules of an NSG. You can then stage the resource in another region by exporting the NSG to a template, modifying the parameters to match the destination region, and then deploy the template to the new region. For more information on Resource Manager and templates, see [Quickstart: Create and deploy Azure Resource Manager templates by using the Azure portal](#).

## Prerequisites

- Make sure that the Azure network security group is in the Azure region from which you want to move.
- Azure network security groups can't be moved between regions. You'll have to associate the new NSG to resources in the target region.
- To export an NSG configuration and deploy a template to create an NSG in another region, you'll need the Network Contributor role or higher.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, public IPs, and virtual networks.
- Verify that your Azure subscription allows you to create NSGs in the target region that's used. Contact support to enable the required quota.
- Make sure that your subscription has enough resources to support the addition of NSGs for this process. See [Azure subscription and service limits, quotas, and constraints](#).

## Prepare and move

The following steps show how to prepare the network security group for the configuration and security rule move using a Resource Manager template, and move the NSG configuration and security rules to the target region using the portal.

### Export the template and deploy from the portal

1. Login to the [Azure portal](#) > **Resource Groups**.
2. Locate the Resource Group that contains the source NSG and click on it.
3. Select > **Settings** > **Export template**.
4. Choose **Deploy** in the **Export template** blade.
5. Click **TEMPLATE** > **Edit parameters** to open the **parameters.json** file in the online editor.
6. To edit the parameter of the NSG name, change the **value** property under **parameters**:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "networkSecurityGroups_myVM1_nsg_name": {
      "value": "<target-nsg-name>"
    }
  }
}
```

7. Change the source NSG value in the editor to a name of your choice for the target NSG. Ensure you enclose the name in quotes.
8. Click **Save** in the editor.
9. Click **TEMPLATE > Edit template** to open the **template.json** file in the online editor.
10. To edit the target region where the NSG configuration and security rules will be moved, change the **location** property under **resources** in the online editor:

```
"resources": [
  {
    "type": "Microsoft.Network/networkSecurityGroups",
    "apiVersion": "2019-06-01",
    "name": "[parameters('networkSecurityGroups_myVM1_nsg_name')]",
    "location": "<target-region>",
    "properties": {
      "provisioningState": "Succeeded",
      "resourceGuid": "2c846acf-58c8-416d-be97-ccd00a4cccd78",
    }
  }
]
```

11. To obtain region location codes, see [Azure Locations](#). The code for a region is the region name with no spaces, **Central US = centralus**.
12. You can also change other parameters in the template if you choose, and are optional depending on your requirements:
  - **Security rules** - You can edit which rules are deployed into the target NSG by adding or removing rules to the **securityRules** section in the **template.json** file:

```

"resources": [
    {
        "type": "Microsoft.Network/networkSecurityGroups",
        "apiVersion": "2019-06-01",
        "name": "[parameters('networkSecurityGroups_myVM1_nsg_name')]",
        "location": "<target-region>",
        "properties": {
            "provisioningState": "Succeeded",
            "resourceGuid": "2c846acf-58c8-416d-be97-ccd00a4cccd78",
            "securityRules": [
                {
                    "name": "RDP",
                    "etag": "W/\"c630c458-6b52-4202-8fd7-172b7ab49cf5\"",
                    "properties": {
                        "provisioningState": "Succeeded",
                        "protocol": "TCP",
                        "sourcePortRange": "*",
                        "destinationPortRange": "3389",
                        "sourceAddressPrefix": "*",
                        "destinationAddressPrefix": "*",
                        "access": "Allow",
                        "priority": 300,
                        "direction": "Inbound",
                        "sourcePortRanges": [],
                        "destinationPortRanges": [],
                        "sourceAddressPrefixes": [],
                        "destinationAddressPrefixes": []
                    }
                },
            ],
        }
    }
]
}

```

To complete the addition or the removal of the rules in the target NSG, you must also edit the custom rule types at the end of the **template.json** file in the format of the example below:

```

{
    "type": "Microsoft.Network/networkSecurityGroups/securityRules",
    "apiVersion": "2019-06-01",
    "name": "[concat(parameters('networkSecurityGroups_myVM1_nsg_name'), '/Port_80')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('networkSecurityGroups_myVM1_nsg_name'))]"
    ],
    "properties": {
        "provisioningState": "Succeeded",
        "protocol": "*",
        "sourcePortRange": "*",
        "destinationPortRange": "80",
        "sourceAddressPrefix": "*",
        "destinationAddressPrefix": "*",
        "access": "Allow",
        "priority": 310,
        "direction": "Inbound",
        "sourcePortRanges": [],
        "destinationPortRanges": [],
        "sourceAddressPrefixes": [],
        "destinationAddressPrefixes": []
    }
}

```

13. Click **Save** in the online editor.
14. Click **BASICS > Subscription** to choose the subscription where the target NSG will be deployed.
15. Click **BASICS > Resource group** to choose the resource group where the target NSG will be deployed. You

can click **Create new** to create a new resource group for the target NSG. Ensure the name isn't the same as the source resource group of the existing NSG.

16. Verify **BASICS > Location** is set to the target location where you wish for the NSG to be deployed.
17. Verify under **SETTINGS** that the name matches the name that you entered in the parameters editor above.
18. Check the box under **TERMS AND CONDITIONS**.
19. Click the **Purchase** button to deploy the target network security group.

## Discard

If you wish to discard the target NSG, delete the resource group that contains the target NSG. To do so, select the resource group from your dashboard in the portal and select **Delete** at the top of the overview page.

## Clean up

To commit the changes and complete the move of the NSG, delete the source NSG or resource group. To do so, select the network security group or resource group from your dashboard in the portal and select **Delete** at the top of each page.

## Next steps

In this tutorial, you moved an Azure network security group from one region to another and cleaned up the source resources. To learn more about moving resources between regions and disaster recovery in Azure, refer to:

- [Move resources to a new resource group or subscription](#)
- [Move Azure VMs to another region](#)

# Move Azure network security group (NSG) to another region using Azure PowerShell

1/3/2020 • 4 minutes to read • [Edit Online](#)

There are various scenarios in which you'd want to move your existing NSGs from one region to another. For example, you may want to create an NSG with the same configuration and security rules for testing. You may also want to move an NSG to another region as part of disaster recovery planning.

Azure security groups can't be moved from one region to another. You can however, use an Azure Resource Manager template to export the existing configuration and security rules of an NSG. You can then stage the resource in another region by exporting the NSG to a template, modifying the parameters to match the destination region, and then deploy the template to the new region. For more information on Resource Manager and templates, see [Export resource groups to templates](#).

## Prerequisites

- Make sure that the Azure network security group is in the Azure region from which you want to move.
- Azure network security groups can't be moved between regions. You'll have to associate the new NSG to resources in the target region.
- To export an NSG configuration and deploy a template to create an NSG in another region, you'll need the Network Contributor role or higher.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, public IPs, and virtual networks.
- Verify that your Azure subscription allows you to create NSGs in the target region that's used. Contact support to enable the required quota.
- Make sure that your subscription has enough resources to support the addition of NSGs for this process. See [Azure subscription and service limits, quotas, and constraints](#).

## Prepare and move

The following steps show how to prepare the network security group for the configuration and security rule move using a Resource Manager template, and move the NSG configuration and security rules to the target region using Azure PowerShell.

### 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

### Export the template and deploy from a script

1. Sign in to your Azure subscription with the [Connect-AzAccount](#) command and follow the on-screen directions:

```
Connect-AzAccount
```

2. Obtain the resource ID of the NSG you want to move to the target region and place it in a variable using [Get-AzNetworkSecurityGroup](#):

```
$sourceNSGID = (Get-AzNetworkSecurityGroup -Name <source-nsg-name> -ResourceGroupName <source-resource-group-name>).Id
```

3. Export the source NSG to a json file into the directory where you execute the command [Export-AzResourceGroup](#):

```
Export-AzResourceGroup -ResourceGroupName <source-resource-group-name> -Resource $sourceNSGID -  
IncludeParameterDefaultValue
```

4. The file downloaded will be named after the resource group the resource was exported from. Locate the file that was exported from the command named **<resource-group-name>.json** and open it in an editor of your choice:

```
notepad <source-resource-group-name>.json
```

5. To edit the parameter of the NSG name, change the property **defaultValue** of the source NSG name to the name of your target NSG, ensure the name is in quotes:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "networkSecurityGroups_myVM1_nsg_name": {
      "defaultValue": "<target-nsg-name>",
      "type": "String"
    }
  }
}
```

6. To edit the target region where the NSG configuration and security rules will be moved, change the **location** property under **resources**:

```
"resources": [
  {
    "type": "Microsoft.Network/networkSecurityGroups",
    "apiVersion": "2019-06-01",
    "name": "[parameters('networkSecurityGroups_myVM1_nsg_name')]",
    "location": "<target-region>",
    "properties": {
      "provisioningState": "Succeeded",
      "resourceGuid": "2c846acf-58c8-416d-be97-ccd00a4cccd78",
    }
  }
]
```

7. To obtain region location codes, you can use the Azure PowerShell cmdlet [Get-AzLocation](#) by running the following command:

```
Get-AzLocation | format-table
```

8. You can also change other parameters in the **<resource-group-name>.json** if you choose, and are optional depending on your requirements:

- **Security rules** - You can edit which rules are deployed into the target NSG by adding or removing rules to the **securityRules** section in the **<resource-group-name>.json** file:

```
"resources": [
    {
        "type": "Microsoft.Network/networkSecurityGroups",
        "apiVersion": "2019-06-01",
        "name": "[parameters('networkSecurityGroups_myVM1_nsg_name')]",
        "location": "TARGET REGION",
        "properties": {
            "provisioningState": "Succeeded",
            "resourceGuid": "2c846acf-58c8-416d-be97-ccd00a4cccd78",
            "securityRules": [
                {
                    "name": "RDP",
                    "etag": "W/\"c630c458-6b52-4202-8fd7-172b7ab49cf5\"",
                    "properties": {
                        "provisioningState": "Succeeded",
                        "protocol": "TCP",
                        "sourcePortRange": "*",
                        "destinationPortRange": "3389",
                        "sourceAddressPrefix": "*",
                        "destinationAddressPrefix": "*",
                        "access": "Allow",
                        "priority": 300,
                        "direction": "Inbound",
                        "sourcePortRanges": [],
                        "destinationPortRanges": [],
                        "sourceAddressPrefixes": [],
                        "destinationAddressPrefixes": []
                    }
                }
            ]
        }
    }
]
```

To complete the addition or the removal of the rules in the target NSG, you must also edit the custom rule types at the end of the **<resource-group-name>.json** file in the format of the example below:

```
{
  "type": "Microsoft.Network/networkSecurityGroups/securityRules",
  "apiVersion": "2019-06-01",
  "name": "[concat(parameters('networkSecurityGroups_myVM1_nsg_name'), '/Port_80')]",
  "dependsOn": [
    "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('networkSecurityGroups_myVM1_nsg_name'))]"
  ],
  "properties": {
    "provisioningState": "Succeeded",
    "protocol": "*",
    "sourcePortRange": "*",
    "destinationPortRange": "80",
    "sourceAddressPrefix": "*",
    "destinationAddressPrefix": "*",
    "access": "Allow",
    "priority": 310,
    "direction": "Inbound",
    "sourcePortRanges": [],
    "destinationPortRanges": [],
    "sourceAddressPrefixes": [],
    "destinationAddressPrefixes": []
  }
}
```

9. Save the **<resource-group-name>.json** file.

10. Create a resource group in the target region for the target NSG to be deployed using [New-AzResourceGroup](#):

```
New-AzResourceGroup -Name <target-resource-group-name> -location <target-region>
```

11. Deploy the edited **<resource-group-name>.json** file to the resource group created in the previous step using [New-AzResourceGroupDeployment](#):

```
New-AzResourceGroupDeployment -ResourceGroupName <target-resource-group-name> -TemplateFile <source-resource-group-name>.json
```

12. To verify the resources were created in the target region, use [Get-AzResourceGroup](#) and [Get-AzNetworkSecurityGroup](#):

```
Get-AzResourceGroup -Name <target-resource-group-name>
```

```
Get-AzNetworkSecurityGroup -Name <target-nsg-name> -ResourceGroupName <target-resource-group-name>
```

## Discard

After the deployment, if you wish to start over or discard the NSG in the target, delete the resource group that was created in the target and the moved NSG will be deleted. To remove the resource group, use [Remove-AzResourceGroup](#):

```
Remove-AzResourceGroup -Name <target-resource-group-name>
```

## Clean up

To commit the changes and complete the move of the NSG, delete the source NSG or resource group, use [Remove-AzResourceGroup](#) or [Remove-AzNetworkSecurityGroup](#):

```
Remove-AzResourceGroup -Name <source-resource-group-name>
```

```
Remove-AzNetworkSecurityGroup -Name <source-nsg-name> -ResourceGroupName <source-resource-group-name>
```

## Next steps

In this tutorial, you moved an Azure network security group from one region to another and cleaned up the source resources. To learn more about moving resources between regions and disaster recovery in Azure, refer to:

- [Move resources to a new resource group or subscription](#)
- [Move Azure VMs to another region](#)

# Move an Azure virtual network to another region by using the Azure portal

1/3/2020 • 5 minutes to read • [Edit Online](#)

There are various scenarios for moving an existing Azure virtual network from one region to another. For example, you might want to create a virtual network with the same configuration for testing and availability as your existing virtual network. Or you might want to move a production virtual network to another region as part of your disaster recovery planning.

You can use an Azure Resource Manager template to complete the move of the virtual network to another region. You do this by exporting the virtual network to a template, modifying the parameters to match the destination region, and then deploying the template to the new region. For more information about Resource Manager templates, see [Quickstart: Create and deploy Azure Resource Manager templates by using the Azure portal](#).

## Prerequisites

- Make sure that your virtual network is in the Azure region that you want to move from.
- To export a virtual network and deploy a template to create a virtual network in another region, you need to have the Network Contributor role or higher.
- Virtual network peerings won't be re-created, and they'll fail if they're still present in the template. Before you export the template, you have to remove any virtual network peers. You can then reestablish them after the virtual network move.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, network security groups (NSGs), and public IPs.
- Verify that your Azure subscription allows you to create virtual networks in the target region. To enable the required quota, contact support.
- Make sure that your subscription has enough resources to support the addition of virtual networks for this process. For more information, see [Azure subscription and service limits, quotas, and constraints](#).

## Prepare for the move

In this section, you prepare the virtual network for the move by using a Resource Manager template. You then move the virtual network to the target region by using the Azure portal.

To export the virtual network and deploy the target virtual network by using the Azure portal, do the following:

1. Sign in to the [Azure portal](#), and then select **Resource Groups**.
2. Locate the resource group that contains the source virtual network, and then select it.
3. Select **Settings > Export template**.
4. In the **Export template** pane, select **Deploy**.
5. To open the *parameters.json* file in your online editor, select **Template > Edit parameters**.
6. To edit the parameter of the virtual network name, change the **value** property under **parameters**:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "virtualNetworks_myVNET1_name": {
            "value": "<target-virtual-network-name>"
        }
    }
}
```

7. In the editor, change the source virtual network name value in the editor to a name that you want for the target virtual network. Be sure to enclose the name in quotation marks.
8. Select **Save** in the editor.
9. To open the *template.json* file in the online editor, select **Template > Edit template**.
10. In the online editor, to edit the target region where the virtual network will be moved, change the **location** property under **resources**:

```
"resources": [
    {
        "type": "Microsoft.Network/virtualNetworks",
        "apiVersion": "2019-06-01",
        "name": "[parameters('virtualNetworks_myVNET1_name')]",
        "location": "<target-region>",
        "properties": {
            "provisioningState": "Succeeded",
            "resourceGuid": "6e2652be-35ac-4e68-8c70-621b9ec87dcb",
            "addressSpace": {
                "addressPrefixes": [
                    "10.0.0.0/16"
                ]
            },
        }
    }
]
```

11. To obtain region location codes, see [Azure Locations](#). The code for a region is the region name, without spaces (for example, **Central US = centralus**).
12. (Optional) You can also change other parameters in the template, depending on your requirements:

- **Address Space:** Before you save the file, you can alter the address space of the virtual network by modifying the **resources > addressSpace** section and changing the **addressPrefixes** property:

```
"resources": [
    {
        "type": "Microsoft.Network/virtualNetworks",
        "apiVersion": "2019-06-01",
        "name": "[parameters('virtualNetworks_myVNET1_name')]",
        "location": "<target-region>",
        "properties": {
            "provisioningState": "Succeeded",
            "resourceGuid": "6e2652be-35ac-4e68-8c70-621b9ec87dcb",
            "addressSpace": {
                "addressPrefixes": [
                    "10.0.0.0/16"
                ]
            },
        }
    }
]
```

- **Subnet:** You can change or add to the subnet name and the subnet address space by changing the

template's **subnets** section. You can change the name of the subnet by changing the **name** property. And you can change the subnet address space by changing the **addressPrefix** property:

```
"subnets": [
  {
    "name": "subnet-1",
    "etag": "W/\"d9f6e6d6-2c15-4f7c-b01f-bed40f748dea\"",
    "properties": {
      "provisioningState": "Succeeded",
      "addressPrefix": "10.0.0.0/24",
      "delegations": [],
      "privateEndpointNetworkPolicies": "Enabled",
      "privateLinkServiceNetworkPolicies": "Enabled"
    }
  },
  {
    "name": "GatewaySubnet",
    "etag": "W/\"d9f6e6d6-2c15-4f7c-b01f-bed40f748dea\"",
    "properties": {
      "provisioningState": "Succeeded",
      "addressPrefix": "10.0.1.0/29",
      "serviceEndpoints": [],
      "delegations": [],
      "privateEndpointNetworkPolicies": "Enabled",
      "privateLinkServiceNetworkPolicies": "Enabled"
    }
  }
]
```

To change the address prefix in the *template.json* file, edit it in two places: in the code in the preceding section and in the **type** section of the following code. Change the **addressPrefix** property in the following code to match the **addressPrefix** property in the code in the preceding section.

```

"type": "Microsoft.Network/virtualNetworks/subnets",
"apiVersion": "2019-06-01",
"name": "[concat(parameters('virtualNetworks_myVNET1_name'), '/GatewaySubnet')]",
"dependsOn": [
    "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworks_myVNET1_name'))]"
],
"properties": {
    "provisioningState": "Succeeded",
    "addressPrefix": "10.0.1.0/29",
    "serviceEndpoints": [],
    "delegations": [],
    "privateEndpointNetworkPolicies": "Enabled",
    "privateLinkServiceNetworkPolicies": "Enabled"
}
},
{
    "type": "Microsoft.Network/virtualNetworks/subnets",
    "apiVersion": "2019-06-01",
    "name": "[concat(parameters('virtualNetworks_myVNET1_name'), '/subnet-1')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworks_myVNET1_name'))]"
],
    "properties": {
        "provisioningState": "Succeeded",
        "addressPrefix": "10.0.0.0/24",
        "delegations": [],
        "privateEndpointNetworkPolicies": "Enabled",
        "privateLinkServiceNetworkPolicies": "Enabled"
    }
}
]

```

13. In the online editor, select **Save**.
14. To choose the subscription where the target virtual network will be deployed, select **Basics > Subscription**.
15. To choose the resource group where the target virtual network will be deployed, select **Basics > Resource group**.  
If you need to create a new resource group for the target virtual network, select **Create new**. Make sure that the name isn't the same as the source resource group name in the existing virtual network.
16. Verify that **Basics > Location** is set to the target location where you want the virtual network to be deployed.
17. Under **Settings**, verify that the name matches the name that you entered previously in the parameters editor.
18. Select the **Terms and Conditions** check box.
19. To deploy the target virtual network, select **Purchase**.

## Delete the target virtual network

To discard the target virtual network, you delete the resource group that contains the target virtual network. To do so:

1. On the Azure portal dashboard, select the resource group.
2. At the top of the **Overview** pane, select **Delete**.

## Clean up

To commit the changes and complete the virtual network move, you delete the source virtual network or resource group. To do so:

1. On the Azure portal dashboard, select the virtual network or resource group.
2. At the top of each pane, select **Delete**.

## Next steps

In this tutorial, you moved an Azure virtual network from one region to another by using the Azure portal and then cleaned up the unneeded source resources. To learn more about moving resources between regions and disaster recovery in Azure, see:

- [Move resources to a new resource group or subscription](#)
- [Move Azure virtual machines to another region](#)

# Move an Azure virtual network to another region by using Azure PowerShell

1/3/2020 • 5 minutes to read • [Edit Online](#)

There are various scenarios for moving an existing Azure virtual network from one region to another. For example, you might want to create a virtual network with the same configuration for testing and availability as your existing virtual network. Or you might want to move a production virtual network to another region as part of your disaster recovery planning.

You can use an Azure Resource Manager template to complete the move of the virtual network to another region. You do this by exporting the virtual network to a template, modifying the parameters to match the destination region, and then deploying the template to the new region. For more information about Resource Manager templates, see [Export resource groups to templates](#).

## Prerequisites

- Make sure that your virtual network is in the Azure region that you want to move from.
- To export a virtual network and deploy a template to create a virtual network in another region, you need to have the Network Contributor role or higher.
- Virtual network peerings won't be re-created, and they'll fail if they're still present in the template. Before you export the template, you have to remove any virtual network peers. You can then reestablish them after the virtual network move.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, network security groups (NSGs), and public IPs.
- Verify that your Azure subscription allows you to create virtual networks in the target region. To enable the required quota, contact support.
- Make sure that your subscription has enough resources to support the addition of virtual networks for this process. For more information, see [Azure subscription and service limits, quotas, and constraints](#).

## Prepare for the move

In this section, you prepare the virtual network for the move by using a Resource Manager template. You then move the virtual network to the target region by using Azure PowerShell commands.

### 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

To export the virtual network and deploy the target virtual network by using PowerShell, do the following:

1. Sign in to your Azure subscription with the [Connect-AzAccount](#) command, and then follow the on-screen directions:

```
Connect-AzAccount
```

- Obtain the resource ID of the virtual network that you want to move to the target region, and then place it in a variable by using [Get-AzVirtualNetwork](#):

```
$sourceVNETID = (Get-AzVirtualNetwork -Name <source-virtual-network-name> -ResourceGroupName <source-resource-group-name>).Id
```

- Export the source virtual network to a json file in the directory where you execute the command [Export-AzResourceGroup](#):

```
Export-AzResourceGroup -ResourceGroupName <source-resource-group-name> -Resource $sourceVNETID -  
IncludeParameterDefaultValue
```

- The downloaded file has the same name as the resource group that the resource was exported from. Locate the `<resource-group-name>.json` file, which you exported with the command, and then open it in your editor:

```
notepad <source-resource-group-name>.json
```

- To edit the parameter of the virtual network name, change the **defaultValue** property of the source virtual network name to the name of your target virtual network. Be sure to enclose the name in quotation marks.

```
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentMyResourceGroupVNET.json#",  
"contentVersion": "1.0.0.0",  
"parameters": {  
    "virtualNetworks_myVNET1_name": {  
        "defaultValue": "<target-virtual-network-name>",  
        "type": "String"  
    }  
}
```

- To edit the target region where the virtual network will be moved, change the **location** property under resources:

```
"resources": [  
    {  
        "type": "Microsoft.Network/virtualNetworks",  
        "apiVersion": "2019-06-01",  
        "name": "[parameters('virtualNetworks_myVNET1_name')]",  
        "location": "<target-region>",  
        "properties": {  
            "provisioningState": "Succeeded",  
            "resourceGuid": "6e2652be-35ac-4e68-8c70-621b9ec87dcb",  
            "addressSpace": {  
                "addressPrefixes": [  
                    "10.0.0.0/16"  
                ]  
            },  
        },  
    },
```

- To obtain region location codes, you can use the Azure PowerShell cmdlet [Get-AzLocation](#) by running the following command:

```
Get-AzLocation | format-table
```

8. (Optional) You can also change other parameters in the *<resource-group-name>.json* file, depending on your requirements:

- **Address Space:** Before you save the file, you can alter the address space of the virtual network by modifying the **resources > addressSpace** section and changing the **addressPrefixes** property:

```
"resources": [
  {
    "type": "Microsoft.Network/virtualNetworks",
    "apiVersion": "2019-06-01",
    "name": "[parameters('virtualNetworks_myVNET1_name')]",
    "location": "<target-region",
    "properties": {
      "provisioningState": "Succeeded",
      "resourceGuid": "6e2652be-35ac-4e68-8c70-621b9ec87dcb",
      "addressSpace": {
        "addressPrefixes": [
          "10.0.0.0/16"
        ]
      },
    }
  ],
}
```

- **Subnet:** You can change or add to the subnet name and the subnet address space by changing the file's **subnets** section. You can change the name of the subnet by changing the **name** property. And you can change the subnet address space by changing the **addressPrefix** property:

```
"subnets": [
  {
    "name": "subnet-1",
    "etag": "W/\\"d9f6e6d6-2c15-4f7c-b01f-bed40f748dea\\\"",
    "properties": {
      "provisioningState": "Succeeded",
      "addressPrefix": "10.0.0.0/24",
      "delegations": [],
      "privateEndpointNetworkPolicies": "Enabled",
      "privateLinkServiceNetworkPolicies": "Enabled"
    }
  },
  {
    "name": "GatewaySubnet",
    "etag": "W/\\"d9f6e6d6-2c15-4f7c-b01f-bed40f748dea\\\"",
    "properties": {
      "provisioningState": "Succeeded",
      "addressPrefix": "10.0.1.0/29",
      "serviceEndpoints": [],
      "delegations": [],
      "privateEndpointNetworkPolicies": "Enabled",
      "privateLinkServiceNetworkPolicies": "Enabled"
    }
  }
]
```

To change the address prefix, edit the file in two places: in the code in the preceding section and in the **type** section of the following code. Change the **addressPrefix** property in the following code to match the **addressPrefix** property in the code in the preceding section.

```

"type": "Microsoft.Network/virtualNetworks/subnets",
"apiVersion": "2019-06-01",
"name": "[concat(parameters('virtualNetworks_myVNET1_name'), '/GatewaySubnet')]",
"dependsOn": [
    "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworks_myVNET1_name'))]"
],
"properties": {
    "provisioningState": "Succeeded",
    "addressPrefix": "10.0.1.0/29",
    "serviceEndpoints": [],
    "delegations": [],
    "privateEndpointNetworkPolicies": "Enabled",
    "privateLinkServiceNetworkPolicies": "Enabled"
}
},
{
    "type": "Microsoft.Network/virtualNetworks/subnets",
    "apiVersion": "2019-06-01",
    "name": "[concat(parameters('virtualNetworks_myVNET1_name'), '/subnet-1')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworks_myVNET1_name'))]"
],
    "properties": {
        "provisioningState": "Succeeded",
        "addressPrefix": "10.0.0.0/24",
        "delegations": [],
        "privateEndpointNetworkPolicies": "Enabled",
        "privateLinkServiceNetworkPolicies": "Enabled"
    }
}
]

```

9. Save the *<resource-group-name>.json* file.

10. Create a resource group in the target region for the target virtual network to be deployed by using [New-AzResourceGroup](#):

```
New-AzResourceGroup -Name <target-resource-group-name> -location <target-region>
```

11. Deploy the edited *<resource-group-name>.json* file to the resource group that you created in the previous step by using [New-AzResourceGroupDeployment](#):

```
New-AzResourceGroupDeployment -ResourceGroupName <target-resource-group-name> -TemplateFile <source-resource-group-name>.json
```

12. To verify that the resources were created in the target region, use [Get-AzResourceGroup](#) and [Get-AzVirtualNetwork](#):

```
Get-AzResourceGroup -Name <target-resource-group-name>
```

```
Get-AzVirtualNetwork -Name <target-virtual-network-name> -ResourceGroupName <target-resource-group-name>
```

## Delete the virtual network or resource group

After you've deployed the virtual network, to start over or discard the virtual network in the target region, delete the resource group that you created in the target region, and the moved virtual network will be deleted.

To remove the resource group, use [Remove-AzResourceGroup](#):

```
Remove-AzResourceGroup -Name <target-resource-group-name>
```

## Clean up

To commit your changes and complete the virtual network move, do either of the following:

- Delete the resource group by using [Remove-AzResourceGroup](#):

```
Remove-AzResourceGroup -Name <source-resource-group-name>
```

- Delete the source virtual network by using [Remove-AzVirtualNetwork](#):

```
Remove-AzVirtualNetwork -Name <source-virtual-network-name> -ResourceGroupName <source-resource-group-name>
```

## Next steps

In this tutorial, you moved a virtual network from one region to another by using PowerShell and then cleaned up the unneeded source resources. To learn more about moving resources between regions and disaster recovery in Azure, see:

- [Move resources to a new resource group or subscription](#)
- [Move Azure virtual machines to another region](#)

# Move Azure Public IP to another region using the Azure portal

1/3/2020 • 4 minutes to read • [Edit Online](#)

There are various scenarios in which you'd want to move your existing Azure Public IPs from one region to another. For example, you may want to create a public IP with the same configuration and sku for testing. You may also want to move a public IP to another region as part of disaster recovery planning.

Azure Public IPs are region specific and can't be moved from one region to another. You can however, use an Azure Resource Manager template to export the existing configuration of a public IP. You can then stage the resource in another region by exporting the public IP to a template, modifying the parameters to match the destination region, and then deploy the template to the new region. For more information on Resource Manager and templates, see [Quickstart: Create and deploy Azure Resource Manager templates by using the Azure portal](#).

## Prerequisites

- Make sure that the Azure Public IP is in the Azure region from which you want to move.
- Azure Public IPs can't be moved between regions. You'll have to associate the new public ip to resources in the target region.
- To export a public IP configuration and deploy a template to create a public IP in another region, you'll need the Network Contributor role or higher.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, network security groups (NSGs), and virtual networks.
- Verify that your Azure subscription allows you to create public IPs in the target region that's used. Contact support to enable the required quota.
- Make sure that your subscription has enough resources to support the addition of public IPs for this process. See [Azure subscription and service limits, quotas, and constraints](#).

## Prepare and move

The following steps show how to prepare the public IP for the configuration move using a Resource Manager template, and move the public IP configuration to the target region using the Azure portal.

### Export the template and deploy from a script

1. Login to the [Azure portal](#) > **Resource Groups**.
2. Locate the Resource Group that contains the source public IP and click on it.
3. Select > **Settings** > **Export template**.
4. Choose **Deploy** in the **Export template** blade.
5. Click **TEMPLATE** > **Edit parameters** to open the **parameters.json** file in the online editor.
6. To edit the parameter of the public IP name, change the property under **parameters** > **value** from the source public IP name to the name of your target public IP, ensure the name is in quotes:

```

{
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
"contentVersion": "1.0.0.0",
"parameters": {
    "publicIPAddresses_myVM1pubIP_name": {
        "value": "<target-publicip-name>"
    }
}

```

7. Click **Save** in the editor.
8. Click **TEMPLATE > Edit template** to open the **template.json** file in the online editor.
9. To edit the target region where the public IP will be moved, change the **location** property under **resources**:

```

"resources": [
{
    "type": "Microsoft.Network/publicIPAddresses",
    "apiVersion": "2019-06-01",
    "name": "[parameters('publicIPAddresses_myPubIP_name')]",
    "location": "<target-region>",
    "sku": {
        "name": "Basic",
        "tier": "Regional"
    },
    "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "7549a8f1-80c2-481a-a073-018f5b0b69be",
        "ipAddress": "52.177.6.204",
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Dynamic",
        "idleTimeoutInMinutes": 4,
        "ipTags": []
    }
}
]

```

10. To obtain region location codes, see [Azure Locations](#). The code for a region is the region name with no spaces, **Central US = centralus**.
11. You can also change other parameters in the template if you choose, and are optional depending on your requirements:
  - **Sku** - You can change the sku of the public IP in the configuration from standard to basic or basic to standard by altering the **sku > name** property in the **template.json** file:

```

"resources": [
{
    "type": "Microsoft.Network/publicIPAddresses",
    "apiVersion": "2019-06-01",
    "name": "[parameters('publicIPAddresses_myPubIP_name')]",
    "location": "<target-region>",
    "sku": {
        "name": "Basic",
        "tier": "Regional"
    },

```

For more information on the differences between basic and standard sku public ips, see [Create, change, or delete a public IP address](#):

- **Public IP allocation method** and **Idle timeout** - You can change both of these options in the template by altering the **publicIPAllocationMethod** property from **Dynamic** to **Static** or **Static** to **Dynamic**. The idle timeout can be changed by altering the **idleTimeoutInMinutes** property to your desired amount. The default is **4**:

```

"resources": [
{
  "type": "Microsoft.Network/publicIPAddresses",
  "apiVersion": "2019-06-01",
  "name": "[parameters('publicIPAddresses_myPubIP_name')]",
  "location": "<target-region>",
  "sku": {
    "name": "Basic",
    "tier": "Regional"
  },
  "properties": {
    "provisioningState": "Succeeded",
    "resourceGuid": "7549a8f1-80c2-481a-a073-018f5b0b69be",
    "ipAddress": "52.177.6.204",
    "publicIPAddressVersion": "IPv4",
    "publicIPAllocationMethod": "Dynamic",
    "idleTimeoutInMinutes": 4,
    "ipTags": []
  }
}
]
  
```

For more information on the allocation methods and the idle timeout values, see [Create, change, or delete a public IP address](#).

12. Click **Save** in the online editor.
13. Click **BASICS > Subscription** to choose the subscription where the target public IP will be deployed.
14. Click **BASICS > Resource group** to choose the resource group where the target public IP will be deployed.  
You can click **Create new** to create a new resource group for the target public IP. Ensure the name isn't the same as the source resource group of the existing source public IP.
15. Verify **BASICS > Location** is set to the target location where you wish for the public IP to be deployed.
16. Verify under **SETTINGS** that the name matches the name that you entered in the parameters editor above.
17. Check the box under **TERMS AND CONDITIONS**.
18. Click the **Purchase** button to deploy the target public IP.

## Discard

If you wish to discard the target public IP, delete the resource group that contains the target public IP. To do so, select the resource group from your dashboard in the portal and select **Delete** at the top of the overview page.

## Clean up

To commit the changes and complete the move of the public IP, delete the source public IP or resource group. To do so, select the public IP or resource group from your dashboard in the portal and select **Delete** at the top of each page.

## Next steps

In this tutorial, you moved an Azure Public IP from one region to another and cleaned up the source resources. To learn more about moving resources between regions and disaster recovery in Azure, refer to:

- Move resources to a new resource group or subscription
- Move Azure VMs to another region

# Move Azure Public IP to another region using Azure PowerShell

1/3/2020 • 5 minutes to read • [Edit Online](#)

There are various scenarios in which you'd want to move your existing Azure Public IPs from one region to another. For example, you may want to create a public IP with the same configuration and sku for testing. You may also want to move a public IP to another region as part of disaster recovery planning.

Azure Public IPs are region specific and can't be moved from one region to another. You can however, use an Azure Resource Manager template to export the existing configuration of a public IP. You can then stage the resource in another region by exporting the public IP to a template, modifying the parameters to match the destination region, and then deploy the template to the new region. For more information on Resource Manager and templates, see [Export resource groups to templates](#)

## Prerequisites

- Make sure that the Azure Public IP is in the Azure region from which you want to move.
- Azure Public IPs cannot be moved between regions. You'll have to associate the new public ip to resources in the target region.
- To export a public IP configuration and deploy a template to create a public IP in another region, you'll need the Network Contributor role or higher.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, network security groups (NSGs), and virtual networks.
- Verify that your Azure subscription allows you to create public IPs in the target region that's used. Contact support to enable the required quota.
- Make sure that your subscription has enough resources to support the addition of public IPs for this process. See [Azure subscription and service limits, quotas, and constraints](#).

## Prepare and move

The following steps show how to prepare the public IP for the configuration move using a Resource Manager template, and move the public IP configuration to the target region using Azure PowerShell.

### 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](#). For Az module installation instructions, see [Install Azure PowerShell](#).

### Export the template and deploy from a script

1. Sign in to your Azure subscription with the [Connect-AzAccount](#) command and follow the on-screen directions:

```
Connect-AzAccount
```

2. Obtain the resource ID of the public IP you want to move to the target region and place it in a variable using [Get-AzPublicIPAddress](#):

```
$sourcePubIPID = (Get-AzPublicIPAddress -Name <source-public-ip-name> -ResourceGroupName <source-resource-group-name>).Id
```

3. Export the source virtual network to a json file into the directory where you execute the command [Export-AzResourceGroup](#):

```
Export-AzResourceGroup -ResourceGroupName <source-resource-group-name> -Resource $sourceVNETID -  
IncludeParameterDefaultValue
```

4. The file downloaded will be named after the resource group the resource was exported from. Locate the file that was exported from the command named **<resource-group-name>.json** and open it in an editor of your choice:

```
notepad <source-resource-group-name>.json
```

5. To edit the parameter of the public IP name, change the property **defaultValue** of the source public IP name to the name of your target public IP, ensure the name is in quotes:

```
{
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
"contentVersion": "1.0.0.0",
"parameters": {
    "publicIPAddresses_myVM1pubIP_name": {
        "defaultValue": "<target-publicip-name>",
        "type": "String"
    }
}
```

6. To edit the target region where the public IP will be moved, change the **location** property under resources:

```
"resources": [
{
    "type": "Microsoft.Network/publicIPAddresses",
    "apiVersion": "2019-06-01",
    "name": "[parameters('publicIPAddresses_myPubIP_name')]",
    "location": "<target-region>",
    "sku": {
        "name": "Basic",
        "tier": "Regional"
    },
    "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "7549a8f1-80c2-481a-a073-018f5b0b69be",
        "ipAddress": "52.177.6.204",
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Dynamic",
        "idleTimeoutInMinutes": 4,
        "ipTags": []
    }
}
]
```

7. To obtain region location codes, you can use the Azure PowerShell cmdlet [Get-AzLocation](#) by running the

following command:

```
Get-AzLocation | format-table
```

8. You can also change other parameters in the template if you choose, and are optional depending on your requirements:

- **Sku** - You can change the sku of the public IP in the configuration from standard to basic or basic to standard by altering the **sku > name** property in the **<resource-group-name>.json** file:

```
"resources": [
    {
        "type": "Microsoft.Network/publicIPAddresses",
        "apiVersion": "2019-06-01",
        "name": "[parameters('publicIPAddresses_myPubIP_name')]",
        "location": "<target-region>",
        "sku": {
            "name": "Basic",
            "tier": "Regional"
        },
    },
```

For more information on the differences between basic and standard sku public ips, see [Create, change, or delete a public IP address](#).

- **Public IP allocation method** and **Idle timeout** - You can change both of these options in the template by altering the **publicIPAllocationMethod** property from **Dynamic** to **Static** or **Static** to **Dynamic**. The idle timeout can be changed by altering the **idleTimeoutInMinutes** property to your desired amount. The default is **4**:

```
"resources": [
    {
        "type": "Microsoft.Network/publicIPAddresses",
        "apiVersion": "2019-06-01",
        "name": "[parameters('publicIPAddresses_myPubIP_name')]",
        "location": "<target-region>",
        "sku": {
            "name": "Basic",
            "tier": "Regional"
        },
        "properties": {
            "provisioningState": "Succeeded",
            "resourceGuid": "7549a8f1-80c2-481a-a073-018f5b0b69be",
            "ipAddress": "52.177.6.204",
            "publicIPAddressVersion": "IPv4",
            "publicIPAllocationMethod": "Dynamic",
            "idleTimeoutInMinutes": 4,
            "ipTags": []
        }
    }
}
```

For more information on the allocation methods and the idle timeout values, see [Create, change, or delete a public IP address](#).

9. Save the **<resource-group-name>.json** file.
10. Create a resource group in the target region for the target public IP to be deployed using [New-AzResourceGroup](#).

```
New-AzResourceGroup -Name <target-resource-group-name> -location <target-region>
```

11. Deploy the edited **<resource-group-name>.json** file to the resource group created in the previous step using [New-AzResourceGroupDeployment](#):

```
New-AzResourceGroupDeployment -ResourceGroupName <target-resource-group-name> -TemplateFile <source-resource-group-name>.json
```

12. To verify the resources were created in the target region, use [Get-AzResourceGroup](#) and [Get-AzPublicIPAddress](#):

```
Get-AzResourceGroup -Name <target-resource-group-name>
```

```
Get-AzPublicIpAddress -Name <target-publicip-name> -ResourceGroupName <target-resource-group-name>
```

## Discard

After the deployment, if you wish to start over or discard the public ip in the target, delete the resource group that was created in the target and the moved public IP will be deleted. To remove the resource group, use [Remove-AzResourceGroup](#):

```
Remove-AzResourceGroup -Name <target-resource-group-name>
```

## Clean up

To commit the changes and complete the move of the virtual network, delete the source virtual network or resource group, use [Remove-AzResourceGroup](#) or [Remove-AzPublicIPAddress](#):

```
Remove-AzResourceGroup -Name <source-resource-group-name>
```

```
Remove-AzPublicIpAddress -Name <source-publicip-name> -ResourceGroupName <resource-group-name>
```

## Next steps

In this tutorial, you moved an Azure Public IP from one region to another and cleaned up the source resources. To learn more about moving resources between regions and disaster recovery in Azure, refer to:

- [Move resources to a new resource group or subscription](#)

- Move Azure VMs to another region

# Azure Virtual Network frequently asked questions (FAQ)

2/10/2020 • 28 minutes to read • [Edit Online](#)

## Virtual Network basics

### What is an Azure Virtual Network (VNet)?

An Azure Virtual Network (VNet) is a representation of your own network in the cloud. It is a logical isolation of the Azure cloud dedicated to your subscription. You can use VNets to provision and manage virtual private networks (VPNs) in Azure and, optionally, link the VNets with other VNets in Azure, or with your on-premises IT infrastructure to create hybrid or cross-premises solutions. Each VNet you create has its own CIDR block and can be linked to other VNets and on-premises networks as long as the CIDR blocks do not overlap. You also have control of DNS server settings for VNets, and segmentation of the VNet into subnets.

Use VNets to:

- Create a dedicated private cloud-only VNet. Sometimes you don't require a cross-premises configuration for your solution. When you create a VNet, your services and VMs within your VNet can communicate directly and securely with each other in the cloud. You can still configure endpoint connections for the VMs and services that require Internet communication, as part of your solution.
- Securely extend your data center. With VNets, you can build traditional site-to-site (S2S) VPNs to securely scale your datacenter capacity. S2S VPNs use IPSEC to provide a secure connection between your corporate VPN gateway and Azure.
- Enable hybrid cloud scenarios. VNets give you the flexibility to support a range of hybrid cloud scenarios. You can securely connect cloud-based applications to any type of on-premises system such as mainframes and Unix systems.

### How do I get started?

Visit the [Virtual network documentation](#) to get started. This content provides overview and deployment information for all of the VNet features.

### Can I use VNets without cross-premises connectivity?

Yes. You can use a VNet without connecting it to your premises. For example, you could run Microsoft Windows Server Active Directory domain controllers and SharePoint farms solely in an Azure VNet.

### Can I perform WAN optimization between VNets or a VNet and my on-premises data center?

Yes. You can deploy a [WAN optimization network virtual appliance](#) from several vendors through the Azure Marketplace.

## Configuration

### What tools do I use to create a VNet?

You can use the following tools to create or configure a VNet:

- Azure portal
- PowerShell
- Azure CLI
- A network configuration file (netcfg - for classic VNets only). See the [Configure a VNet using a network](#)

[configuration file](#) article.

## What address ranges can I use in my VNets?

Any IP address range defined in [RFC 1918](#). For example, 10.0.0.0/16. You cannot add the following address ranges:

- 224.0.0.0/4 (Multicast)
- 255.255.255.255/32 (Broadcast)
- 127.0.0.0/8 (Loopback)
- 169.254.0.0/16 (Link-local)
- 168.63.129.16/32 (Internal DNS)

## Can I have public IP addresses in my VNets?

Yes. For more information about public IP address ranges, see [Create a virtual network](#). Public IP addresses are not directly accessible from the internet.

## Is there a limit to the number of subnets in my VNet?

Yes. See [Azure limits](#) for details. Subnet address spaces cannot overlap one another.

## Are there any restrictions on using IP addresses within these subnets?

Yes. Azure reserves 5 IP addresses within each subnet. These are x.x.x.0-x.x.x.3 and the last address of the subnet. x.x.x.1-x.x.x.3 is reserved in each subnet for Azure services.

- x.x.x.0: Network address
- x.x.x.1: Reserved by Azure for the default gateway
- x.x.x.2, x.x.x.3: Reserved by Azure to map the Azure DNS IPs to the VNet space
- x.x.x.255: Network broadcast address

## How small and how large can VNets and subnets be?

The smallest supported IPv4 subnet is /29, and the largest is /8 (using CIDR subnet definitions). IPv6 subnets must be exactly /64 in size.

## Can I bring my VLANs to Azure using VNets?

No. VNets are Layer-3 overlays. Azure does not support any Layer-2 semantics.

## Can I specify custom routing policies on my VNets and subnets?

Yes. You can create a route table and associate it to a subnet. For more information about routing in Azure, see [Routing overview](#).

## Do VNets support multicast or broadcast?

No. Multicast and broadcast are not supported.

## What protocols can I use within VNets?

You can use TCP, UDP, and ICMP TCP/IP protocols within VNets. Unicast is supported within VNets, with the exception of Dynamic Host Configuration Protocol (DHCP) via Unicast (source port UDP/68 / destination port UDP/67). Multicast, broadcast, IP-in-IP encapsulated packets, and Generic Routing Encapsulation (GRE) packets are blocked within VNets.

## Can I ping my default routers within a VNet?

No.

## Can I use tracert to diagnose connectivity?

No.

## Can I add subnets after the VNet is created?

Yes. Subnets can be added to VNets at any time as long as the subnet address range is not part of another subnet

and there is available space left in the virtual network's address range.

### **Can I modify the size of my subnet after I create it?**

Yes. You can add, remove, expand, or shrink a subnet if there are no VMs or services deployed within it.

### **Can I modify subnets after I created them?**

Yes. You can add, remove, and modify the CIDR blocks used by a VNet.

### **If I am running my services in a VNet, can I connect to the internet?**

Yes. All services deployed within a VNet can connect outbound to the internet. To learn more about outbound internet connections in Azure, see [Outbound connections](#). If you want to connect inbound to a resource deployed through Resource Manager, the resource must have a public IP address assigned to it. To learn more about public IP addresses, see [Public IP addresses](#). Every Azure Cloud Service deployed in Azure has a publicly addressable VIP assigned to it. You define input endpoints for PaaS roles and endpoints for virtual machines to enable these services to accept connections from the internet.

### **Do VNets support IPv6?**

Yes, VNets can be IPv4-only or dual stack (IPv4+IPv6). For details, see [Overview of IPv6 for Azure Virtual Networks](#).

### **Can a VNet span regions?**

No. A VNet is limited to a single region. A virtual network does, however, span availability zones. To learn more about availability zones, see [Availability zones overview](#). You can connect virtual networks in different regions with virtual network peering. For details, see [Virtual network peering overview](#)

### **Can I connect a VNet to another VNet in Azure?**

Yes. You can connect one VNet to another VNet using either:

- **Virtual network peering:** For details, see [VNet peering overview](#)
- **An Azure VPN Gateway:** For details, see [Configure a VNet-to-VNet connection](#).

## Name Resolution (DNS)

### **What are my DNS options for VNets?**

Use the decision table on the [Name Resolution for VMs and Role Instances](#) page to guide you through all the DNS options available.

### **Can I specify DNS servers for a VNet?**

Yes. You can specify DNS server IP addresses in the VNet settings. The setting is applied as the default DNS server(s) for all VMs in the VNet.

### **How many DNS servers can I specify?**

Reference [Azure limits](#).

### **Can I modify my DNS servers after I have created the network?**

Yes. You can change the DNS server list for your VNet at any time. If you change your DNS server list, you need to perform a DHCP lease renewal on all affected VMs in the VNet, for the new DNS settings to take effect. For VMs running Windows OS you can do this by typing `ipconfig /renew` directly on the VM. For other OS types, refer to the DHCP lease renewal documentation for the specific OS type.

### **What is Azure-provided DNS and does it work with VNets?**

Azure-provided DNS is a multi-tenant DNS service offered by Microsoft. Azure registers all of your VMs and cloud service role instances in this service. This service provides name resolution by hostname for VMs and role instances contained within the same cloud service, and by FQDN for VMs and role instances in the same VNet. To learn more about DNS, see [Name Resolution for VMs and Cloud Services role instances](#).

There is a limitation to the first 100 cloud services in a VNet for cross-tenant name resolution using Azure-provided DNS. If you are using your own DNS server, this limitation does not apply.

### **Can I override my DNS settings on a per-VM or cloud service basis?**

Yes. You can set DNS servers per VM or cloud service to override the default network settings. However, it's recommended that you use network-wide DNS as much as possible.

### **Can I bring my own DNS suffix?**

No. You cannot specify a custom DNS suffix for your VNets.

## Connecting virtual machines

### **Can I deploy VMs to a VNet?**

Yes. All network interfaces (NIC) attached to a VM deployed through the Resource Manager deployment model must be connected to a VNet. VMs deployed through the classic deployment model can optionally be connected to a VNet.

### **What are the different types of IP addresses I can assign to VMs?**

- **Private:** Assigned to each NIC within each VM. The address is assigned using either the static or dynamic method. Private IP addresses are assigned from the range that you specified in the subnet settings of your VNet. Resources deployed through the classic deployment model are assigned private IP addresses, even if they're not connected to a VNet. The behavior of the allocation method is different depending on whether a resource was deployed with the Resource Manager or classic deployment model:
  - **Resource Manager:** A private IP address assigned with the dynamic or static method remains assigned to a virtual machine (Resource Manager) until the resource is deleted. The difference is that you select the address to assign when using static, and Azure chooses when using dynamic.
  - **Classic:** A private IP address assigned with the dynamic method may change when a virtual machine (classic) VM is restarted after having been in the stopped (deallocated) state. If you need to ensure that the private IP address for a resource deployed through the classic deployment model never changes, assign a private IP address with the static method.
- **Public:** Optionally assigned to NICs attached to VMs deployed through the Azure Resource Manager deployment model. The address can be assigned with the static or dynamic allocation method. All VMs and Cloud Services role instances deployed through the classic deployment model exist within a cloud service, which is assigned a *dynamic*, public virtual IP (VIP) address. A public *static* IP address, called a [Reserved IP address](#), can optionally be assigned as a VIP. You can assign public IP addresses to individual VMs or Cloud Services role instances deployed through the classic deployment model. These addresses are called [Instance level public IP \(ILPIP\)](#) addresses and can be assigned dynamically.

### **Can I reserve a private IP address for a VM that I will create at a later time?**

No. You cannot reserve a private IP address. If a private IP address is available, it is assigned to a VM or role instance by the DHCP server. The VM may or may not be the one that you want the private IP address assigned to. You can, however, change the private IP address of an already created VM, to any available private IP address.

### **Do private IP addresses change for VMs in a VNet?**

It depends. If the VM was deployed through Resource Manager, no, regardless of whether the IP address was assigned with the static or dynamic allocation method. If the VM was deployed through the classic deployment model, dynamic IP addresses can change when a VM is started after having been in the stopped (deallocated) state. The address is released from a VM deployed through either deployment model when the VM is deleted.

### **Can I manually assign IP addresses to NICs within the VM operating system?**

Yes, but it's not recommended unless necessary, such as when assigning multiple IP addresses to a virtual machine. For details, see [Adding multiple IP addresses to a virtual machine](#). If the IP address assigned to an Azure NIC

attached to a VM changes, and the IP address within the VM operating system is different, you lose connectivity to the VM.

**If I stop a Cloud Service deployment slot or shutdown a VM from within the operating system, what happens to my IP addresses?**

Nothing. The IP addresses (public VIP, public, and private) remain assigned to the cloud service deployment slot or VM.

**Can I move VMs from one subnet to another subnet in a VNet without redeploying?**

Yes. You can find more information in the [How to move a VM or role instance to a different subnet](#) article.

**Can I configure a static MAC address for my VM?**

No. A MAC address cannot be statically configured.

**Will the MAC address remain the same for my VM once it's created?**

Yes, the MAC address remains the same for a VM deployed through both the Resource Manager and classic deployment models until it's deleted. Previously, the MAC address was released if the VM was stopped (deallocated), but now the MAC address is retained even when the VM is in the deallocated state. The MAC address remains assigned to the network interface until the network interface is deleted or the private IP address assigned to the primary IP configuration of the primary network interface is changed.

**Can I connect to the internet from a VM in a VNet?**

Yes. All VMs and Cloud Services role instances deployed within a VNet can connect to the Internet.

## Azure services that connect to VNets

**Can I use Azure App Service Web Apps with a VNet?**

Yes. You can deploy Web Apps inside a VNet using an ASE (App Service Environment), connect the backend of your apps to your VNets with VNet Integration, and lock down inbound traffic to your app with service endpoints. For more information, see the following articles:

- [App Service networking features](#)
- [Creating Web Apps in an App Service Environment](#)
- [Integrate your app with an Azure Virtual Network](#)
- [App Service access restrictions](#)

**Can I deploy Cloud Services with web and worker roles (PaaS) in a VNet?**

Yes. You can (optionally) deploy Cloud Services role instances within VNets. To do so, you specify the VNet name and the role/subnet mappings in the network configuration section of your service configuration. You do not need to update any of your binaries.

**Can I connect a virtual machine scale set to a VNet?**

Yes. You must connect a virtual machine scale set to a VNet.

**Is there a complete list of Azure services that can I deploy resources from into a VNet?**

Yes, For details, see [Virtual network integration for Azure services](#).

**How can I restrict access to Azure PaaS resources from a VNet?**

Resources deployed through some Azure PaaS services (such as Azure Storage and Azure SQL Database), can restrict network access to VNet through the use of virtual network service endpoints or Azure Private Link. For details, see [Virtual network service endpoints overview](#), [Azure Private Link overview](#)

**Can I move my services in and out of VNets?**

No. You cannot move services in and out of VNets. To move a resource to another VNet, you have to delete and redeploy the resource.

# Security

## What is the security model for VNets?

VNets are isolated from one another, and other services hosted in the Azure infrastructure. A VNet is a trust boundary.

## Can I restrict inbound or outbound traffic flow to VNet-connected resources?

Yes. You can apply [Network Security Groups](#) to individual subnets within a VNet, NICs attached to a VNet, or both.

## Can I implement a firewall between VNet-connected resources?

Yes. You can deploy a [firewall network virtual appliance](#) from several vendors through the Azure Marketplace.

## Is there information available about securing VNets?

Yes. For details, see [Azure Network Security Overview](#).

# APIs, schemas, and tools

## Can I manage VNets from code?

Yes. You can use REST APIs for VNets in the [Azure Resource Manager](#) and [classic](#) deployment models.

## Is there tooling support for VNets?

Yes. Learn more about using:

- The Azure portal to deploy VNets through the [Azure Resource Manager](#) and [classic](#) deployment models.
- PowerShell to manage VNets deployed through the [Resource Manager](#) and [classic](#) deployment models.
- The Azure command-line interface (CLI) to deploy and manage VNets deployed through the [Resource Manager](#) and [classic](#) deployment models.

# VNet peering

## What is VNet peering?

VNet peering (or virtual network peering) enables you to connect virtual networks. A VNet peering connection between virtual networks enables you to route traffic between them privately through IPv4 addresses. Virtual machines in the peered VNets can communicate with each other as if they are within the same network. These virtual networks can be in the same region or in different regions (also known as Global VNet Peering). VNet peering connections can also be created across Azure subscriptions.

## Can I create a peering connection to a VNet in a different region?

Yes. Global VNet peering enables you to peer VNets in different regions. Global VNet peering is available in all Azure public regions, China cloud regions, and Government cloud regions. You cannot globally peer from Azure public regions to national cloud regions.

## What are the constraints related to Global VNet Peering and Load Balancers?

If the two virtual networks in two different regions are peered over Global VNet Peering, you cannot connect to resources that are behind a Basic Load Balancer through the Front End IP of the Load Balancer. This restriction does not exist for a Standard Load Balancer. The following resources can use Basic Load Balancers which means you cannot reach them through the Load Balancer's Front End IP over Global VNet Peering. You can however use Global VNet peering to reach the resources directly through their private VNet IPs, if permitted.

- VMs behind Basic Load Balancers
- Virtual machine scale sets with Basic Load Balancers
- Redis Cache
- Application Gateway (v1) SKU

- Service Fabric
- SQL MI
- API Management
- Active Directory Domain Service (ADDS)
- Logic Apps
- HDInsight
- Azure Batch
- App Service Environment

You can connect to these resources via ExpressRoute or VNet-to-VNet through VNet Gateways.

### **Can I enable VNet Peering if my virtual networks belong to subscriptions within different Azure Active Directory tenants?**

Yes. It is possible to establish VNet Peering (whether local or global) if your subscriptions belong to different Azure Active Directory tenants. You can do this via PowerShell or CLI. Portal is not yet supported.

### **My VNet peering connection is in *Initiated* state, why can't I connect?**

If your peering connection is in an *Initiated* state, this means you have created only one link. A bidirectional link must be created in order to establish a successful connection. For example, to peer VNet A to VNet B, a link must be created from VNetA to VNetB and from VNetB to VNetA. Creating both links will change the state to *Connected*.

### **My VNet peering connection is in *Disconnected* state, why can't I create a peering connection?**

If your VNet peering connection is in a *Disconnected* state, it means one of the links created was deleted. In order to re-establish a peering connection, you will need to delete the link and recreate it.

### **Can I peer my VNet with a VNet in a different subscription?**

Yes. You can peer VNets across subscriptions and across regions.

### **Can I peer two VNets with matching or overlapping address ranges?**

No. Address spaces must not overlap to enable VNet Peering.

### **How much do VNet peering links cost?**

There is no charge for creating a VNet peering connection. Data transfer across peering connections is charged. [See here.](#)

### **Is VNet peering traffic encrypted?**

No. Traffic between resources in peered VNets is private and isolated. It remains completely on the Microsoft Backbone.

### **Why is my peering connection in a *Disconnected* state?**

VNet peering connections go into *Disconnected* state when one VNet peering link is deleted. You must delete both links in order to reestablish a successful peering connection.

### **If I peer VNetA to VNetB and I peer VNetB to VNetC, does that mean VNetA and VNetC are peered?**

No. Transitive peering is not supported. You must peer VNetA and VNetC for this to take place.

### **Are there any bandwidth limitations for peering connections?**

No. VNet peering, whether local or global, does not impose any bandwidth restrictions. Bandwidth is only limited by the VM or the compute resource.

### **How can I troubleshoot VNet Peering issues?**

Here is a [troubleshooter guide](#) you can try.

# Virtual network TAP

## Which Azure regions are available for virtual network TAP?

Virtual network TAP preview is available in all Azure regions. The monitored network interfaces, the virtual network TAP resource, and the collector or analytics solution must be deployed in the same region.

## Does Virtual Network TAP support any filtering capabilities on the mirrored packets?

Filtering capabilities are not supported with the virtual network TAP preview. When a TAP configuration is added to a network interface a deep copy of all the ingress and egress traffic on the network interface is streamed to the TAP destination.

## Can multiple TAP configurations be added to a monitored network interface?

A monitored network interface can have only one TAP configuration. Check with the individual [partner solution](#) for the capability to stream multiple copies of the TAP traffic to the analytics tools of your choice.

## Can the same virtual network TAP resource aggregate traffic from monitored network interfaces in more than one virtual network?

Yes. The same virtual network TAP resource can be used to aggregate mirrored traffic from monitored network interfaces in peered virtual networks in the same subscription or a different subscription. The virtual network TAP resource and the destination load balancer or destination network interface must be in the same subscription. All subscriptions must be under the same Azure Active Directory tenant.

## Are there any performance considerations on production traffic if I enable a virtual network TAP configuration on a network interface?

Virtual network TAP is in preview. During preview, there is no service level agreement. The capability should not be used for production workloads. When a virtual machine network interface is enabled with a TAP configuration, the same resources on the Azure host allocated to the virtual machine to send the production traffic is used to perform the mirroring function and send the mirrored packets. Select the correct [Linux](#) or [Windows](#) virtual machine size to ensure that sufficient resources are available for the virtual machine to send the production traffic and the mirrored traffic.

## Is accelerated networking for [Linux](#) or [Windows](#) supported with virtual network TAP?

You will be able to add a TAP configuration on a network interface attached to a virtual machine that is enabled with accelerated networking. But the performance and latency on the virtual machine will be affected by adding TAP configuration since the offload for mirroring traffic is currently not supported by Azure accelerated networking.

# Virtual network service endpoints

## What is the right sequence of operations to set up service endpoints to an Azure service?

There are two steps to secure an Azure service resource through service endpoints:

1. Turn on service endpoints for the Azure service.
2. Set up VNet ACLs on the Azure service.

The first step is a network side operation and the second step is a service resource side operation. Both steps can be performed either by the same administrator or different administrators based on the RBAC permissions granted to the administrator role. We recommend that you first turn on service endpoints for your virtual network prior to setting up VNet ACLs on Azure service side. Hence, the steps must be performed in the sequence listed above to set up VNet service endpoints.

#### **NOTE**

Both the operations described above must be completed before you can limit the Azure service access to the allowed VNet and subnet. Only turning on service endpoints for the Azure service on the network side does not provide you the limited access. In addition, you must also set up VNet ACLs on the Azure service side.

Certain services (such as SQL and CosmosDB) allow exceptions to the above sequence through the **IgnoreMissingVnetServiceEndpoint** flag. Once the flag is set to **True**, VNet ACLs can be set on the Azure service side prior to setting up the service endpoints on the network side. Azure services provide this flag to help customers in cases where the specific IP firewalls are configured on Azure services and turning on the service endpoints on the network side can lead to a connectivity drop since the source IP changes from a public IPv4 address to a private address. Setting up VNet ACLs on the Azure service side before setting service endpoints on the network side can help avoid a connectivity drop.

#### **Do all Azure services reside in the Azure virtual network provided by the customer? How does VNet service endpoint work with Azure services?**

No, not all Azure services reside in the customer's virtual network. The majority of Azure data services such as Azure Storage, Azure SQL, and Azure Cosmos DB, are multi-tenant services that can be accessed over public IP addresses. You can learn more about virtual network integration for Azure services [here](#).

When you use the VNet service endpoints feature (turning on VNet service endpoint on the network side and setting up appropriate VNet ACLs on the Azure service side), access to an Azure service is restricted from an allowed VNet and subnet.

#### **How does VNet service endpoint provide security?**

The VNet service endpoint feature (turning on VNet service endpoint on the network side and setting up appropriate VNet ACLs on the Azure service side) limits the Azure service access to the allowed VNet and subnet, thus providing a network level security and isolation of the Azure service traffic. All traffic using VNet service endpoints flows over Microsoft backbone, thus providing another layer of isolation from the public internet. Moreover, customers can choose to fully remove public Internet access to the Azure service resources and allow traffic only from their virtual network through a combination of IP firewall and VNet ACLs, thus protecting the Azure service resources from unauthorized access.

#### **What does the VNet service endpoint protect - VNet resources or Azure service?**

VNet service endpoints help protect Azure service resources. VNet resources are protected through Network Security Groups (NSGs).

#### **Is there any cost for using VNet service endpoints?**

No, there is no additional cost for using VNet service endpoints.

#### **Can I turn on VNet service endpoints and set up VNet ACLs if the virtual network and the Azure service resources belong to different subscriptions?**

Yes, it is possible. Virtual networks and Azure service resources can be either in the same or different subscriptions. The only requirement is that both the virtual network and Azure service resources must be under the same Active Directory (AD) tenant.

#### **Can I turn on VNet service endpoints and set up VNet ACLs if the virtual network and the Azure service resources belong to different AD tenants?**

No, VNet service endpoints and VNet ACLs are not supported across AD tenants.

#### **Can an on-premises device's IP address that is connected through Azure Virtual Network gateway (VPN) or ExpressRoute gateway access Azure PaaS Service over VNet service endpoints?**

By default, Azure service resources secured to virtual networks are not reachable from on-premises networks. If you want to allow traffic from on-premises, you must also allow public (typically, NAT) IP addresses from your on-

premises or ExpressRoute. These IP addresses can be added through the IP firewall configuration for the Azure service resources.

### **Can I use VNet Service Endpoint feature to secure Azure service to multiple subnets within a virtual network or across multiple virtual networks?**

To secure Azure services to multiple subnets within a virtual network or across multiple virtual networks, enable service endpoints on the network side on each of the subnets independently and then secure Azure service resources to all of the subnets by setting up appropriate VNet ACLs on the Azure service side.

### **How can I filter outbound traffic from a virtual network to Azure services and still use service endpoints?**

If you want to inspect or filter the traffic destined to an Azure service from a virtual network, you can deploy a network virtual appliance within the virtual network. You can then apply service endpoints to the subnet where the network virtual appliance is deployed and secure Azure service resources only to this subnet through VNet ACLs. This scenario might also be helpful if you wish to restrict Azure service access from your virtual network only to specific Azure resources using network virtual appliance filtering. For more information, see [egress with network virtual appliances](#).

### **What happens when you access an Azure service account that has a virtual network access control list (ACL) enabled from outside the VNet?**

The HTTP 403 or HTTP 404 error is returned.

### **Are subnets of a virtual network created in different regions allowed to access an Azure service account in another region?**

Yes, for most of the Azure services, virtual networks created in different regions can access Azure services in another region through the VNet service endpoints. For example, if an Azure Cosmos DB account is in West US or East US and virtual networks are in multiple regions, the virtual network can access Azure Cosmos DB. Storage and SQL are exceptions and are regional in nature and both the virtual network and the Azure service need to be in the same region.

### **Can an Azure service have both a VNet ACL and an IP firewall?**

Yes, a VNet ACL and an IP firewall can co-exist. Both features complement each other to ensure isolation and security.

### **What happens if you delete a virtual network or subnet that has service endpoint turned on for Azure service?**

Deletion of VNets and subnets are independent operations and are supported even when service endpoints are turned on for Azure services. In cases where the Azure services have VNet ACLs set up, for those VNets and subnets, the VNet ACL information associated with that Azure service is disabled when a VNet or subnet that has VNet service endpoint turned on is deleted.

### **What happens if an Azure service account that has a VNet Service endpoint enabled is deleted?**

The deletion of an Azure service account is an independent operation and is supported even when the service endpoint is enabled on the network side and VNet ACLs are set up on Azure service side.

### **What happens to the source IP address of a resource (like a VM in a subnet) that has VNet service endpoint enabled?**

When virtual network service endpoints are enabled, the source IP addresses of the resources in your virtual network's subnet switches from using public IPV4 addresses to the Azure virtual network's private IP addresses for traffic to Azure service. Note that this can cause specific IP firewalls that are set to public IPV4 address earlier on the Azure services to fail.

### **Does the service endpoint route always take precedence?**

Service endpoints add a system route which takes precedence over BGP routes and provides optimum routing for the service endpoint traffic. Service endpoints always take service traffic directly from your virtual network to the service on the Microsoft Azure backbone network. For more information about how Azure selects a route, see [Azure Virtual network traffic routing](#).

## **How does NSG on a subnet work with service endpoints?**

To reach the Azure service, NSGs need to allow outbound connectivity. If your NSGs are opened to all Internet outbound traffic, then the service endpoint traffic should work. You can also limit the outbound traffic to service IPs only using the Service tags.

## **What permissions do I need to set up service endpoints?**

Service endpoints can be configured on a virtual network independently by a user with write access to the virtual network. To secure Azure service resources to a VNet, the user must have permission

**Microsoft.Network/virtualNetworks/subnets/joinViaServiceEndpoint/action** for the subnets being added.

This permission is included in the built-in service administrator role by default and can be modified by creating custom roles. Learn more about built-in roles and assigning specific permissions to [custom roles](#).

## **Can I filter virtual network traffic to Azure services, allowing only specific azure service resources, over VNet service endpoints?**

Virtual network (VNet) service endpoint policies allow you to filter virtual network traffic to Azure services, allowing only specific Azure service resources over the service endpoints. Endpoint policies provide granular access control from the virtual network traffic to the Azure services. You can learn more about the service endpoint policies [here](#).

## **Does Azure Active Directory (Azure AD) support VNet service endpoints?**

Azure Active Directory (Azure AD) doesn't support service endpoints natively. Complete list of Azure Services supporting VNet service endpoints can be seen [here](#). Note that the "Microsoft.AzureActiveDirectory" tag listed under services supporting service endpoints is used for supporting service endpoints to ADLS Gen 1. For ADLS Gen 1, virtual network integration for Azure Data Lake Storage Gen1 makes use of the virtual network service endpoint security between your virtual network and Azure Active Directory (Azure AD) to generate additional security claims in the access token. These claims are then used to authenticate your virtual network to your Data Lake Storage Gen1 account and allow access. Learn more about [Azure Data Lake Store Gen 1 VNet Integration](#)

## **Are there any limits on how many VNet service endpoints I can set up from my VNet?**

There is no limit on the total number of VNet service endpoints in a virtual network. For an Azure service resource (such as an Azure Storage account), services may enforce limits on the number of subnets used for securing the resource. The following table shows some example limits:

Azure service	Limits on VNet rules
Azure Storage	100
Azure SQL	128
Azure SQL Data Warehouse	128
Azure KeyVault	127
Azure Cosmos DB	64
Azure Event Hub	128
Azure Service Bus	128
Azure Data Lake Store V1	100

**NOTE**

The limits are subjected to changes at the discretion of the Azure service. Refer to the respective service documentation for services details.