

Table of 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](#)

[Security](#)

[Routing](#)

[Service endpoints](#)

[Peering](#)

[Virtual network for Azure services](#)

[Container networking](#)

[Business continuity](#)

[IP addressing](#)

[DDoS protection](#)

[Classic](#)

[IP addressing](#)

[Access control lists](#)

[How-to guides](#)

[Plan and design](#)

[Virtual networks](#)

[Network security groups](#)

[Deploy](#)

[Network security groups](#)

[Route tables](#)

[Service endpoints](#)

[Virtual network peering](#)

[Virtual machines](#)

[Connectivity scenarios](#)

[Security scenarios](#)

[Configure](#)

[Virtual machines](#)

[Use dynamic DNS with your own DNS server](#)

[Optimize network throughput](#)

[View and modify hostnames](#)

[Manage](#)

[Virtual networks](#)

[Network security groups](#)

[Route tables](#)

[Network interfaces](#)

[Public IP addresses](#)

[Virtual machines](#)

[Troubleshoot](#)

[Network security groups](#)

[Routes](#)

[Throughput testing](#)

[Cannot delete virtual networks](#)

[VM to VM connectivity problems](#)

[Configure PTR for SMTP Banner Check](#)

Classic

[Create and manage a virtual network](#)

[Create a network security group](#)

[Create a route table](#)

[Virtual machine networking](#)

[Cloud service and network security](#)

Reference

[Azure CLI](#)

[Azure PowerShell](#)

[.NET](#)

[Java](#)

[Node.js](#)

[Python](#)

[REST](#)

[Code samples](#)

Resources

[Azure roadmap](#)

[Networking blog](#)

[Networking forum](#)

[Pricing](#)

[Pricing calculator](#)

[Stack Overflow](#)

[FAQ](#)

What is Azure Virtual Network?

5/7/2018 • 4 min to read • [Edit Online](#)

Azure Virtual Network enables many types of Azure resources, such as Azure Virtual Machines (VM), to securely communicate with each other, the internet, and on-premises networks. Azure Virtual Network provides the following key capabilities:

Isolation and segmentation

You can implement multiple virtual networks within each Azure [subscription](#) and Azure [region](#). Each virtual network is isolated from other virtual networks. For each virtual network you can:

- 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.
- Segment the virtual network into one or more subnets and allocate a portion of the virtual network's address space to each subnet.
- Use Azure-provided name resolution, or specify your own DNS server, for use by resources in a virtual network.

Communicate with the internet

All resources in a virtual network can communicate outbound to the internet, by default. You can communicate inbound to a resource by assigning a public IP address to it. To learn more, see [Public IP addresses](#).

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](#).

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:

- **Network security groups:** A network security group 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](#).
- **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](#).

Connect virtual networks

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](#).

Next steps

You now have an overview of Azure Virtual Network. 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

4/9/2018 • 3 min to read • [Edit Online](#)

A virtual network enables Azure resources, such as virtual machines (VM), 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 one VM from the internet, and communicate privately between the two VMs.

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

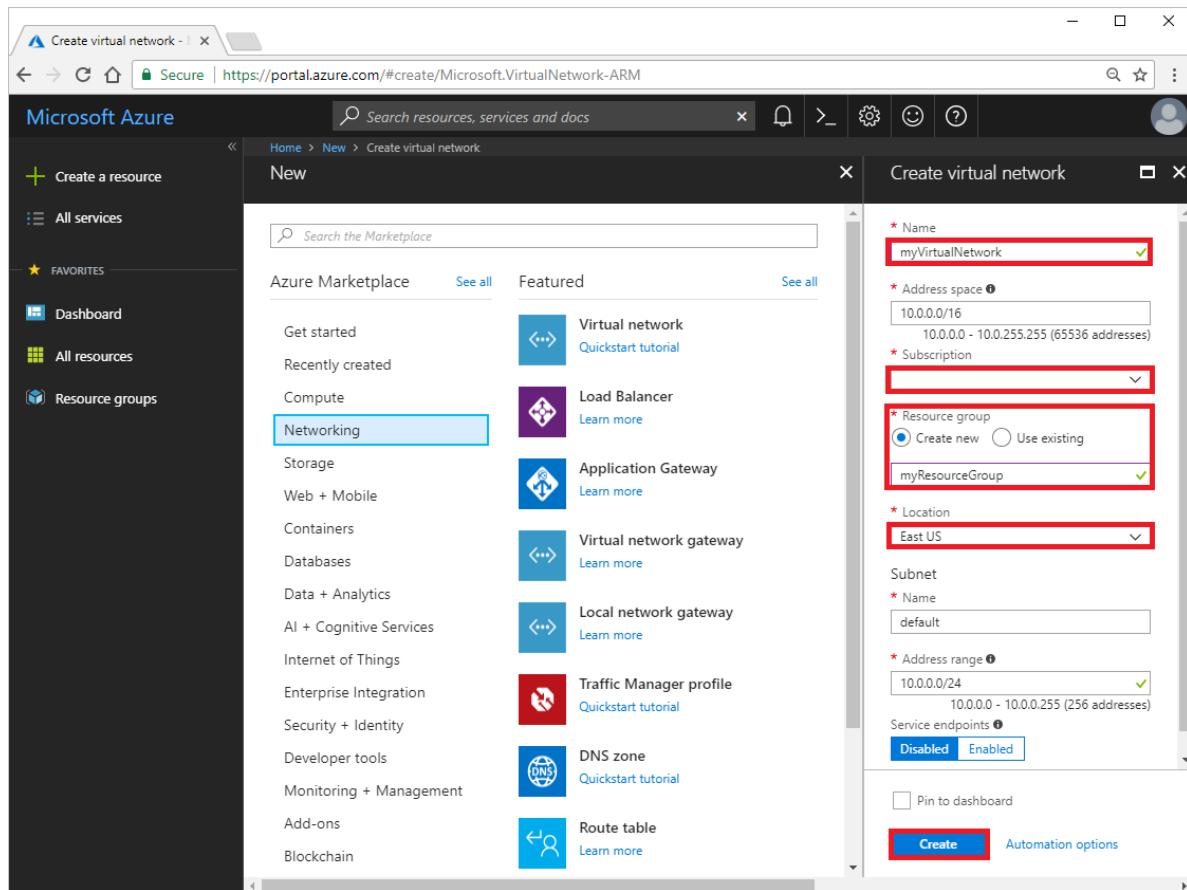
Log in to Azure

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, accept the defaults for the remaining settings, and then select **Create**:

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



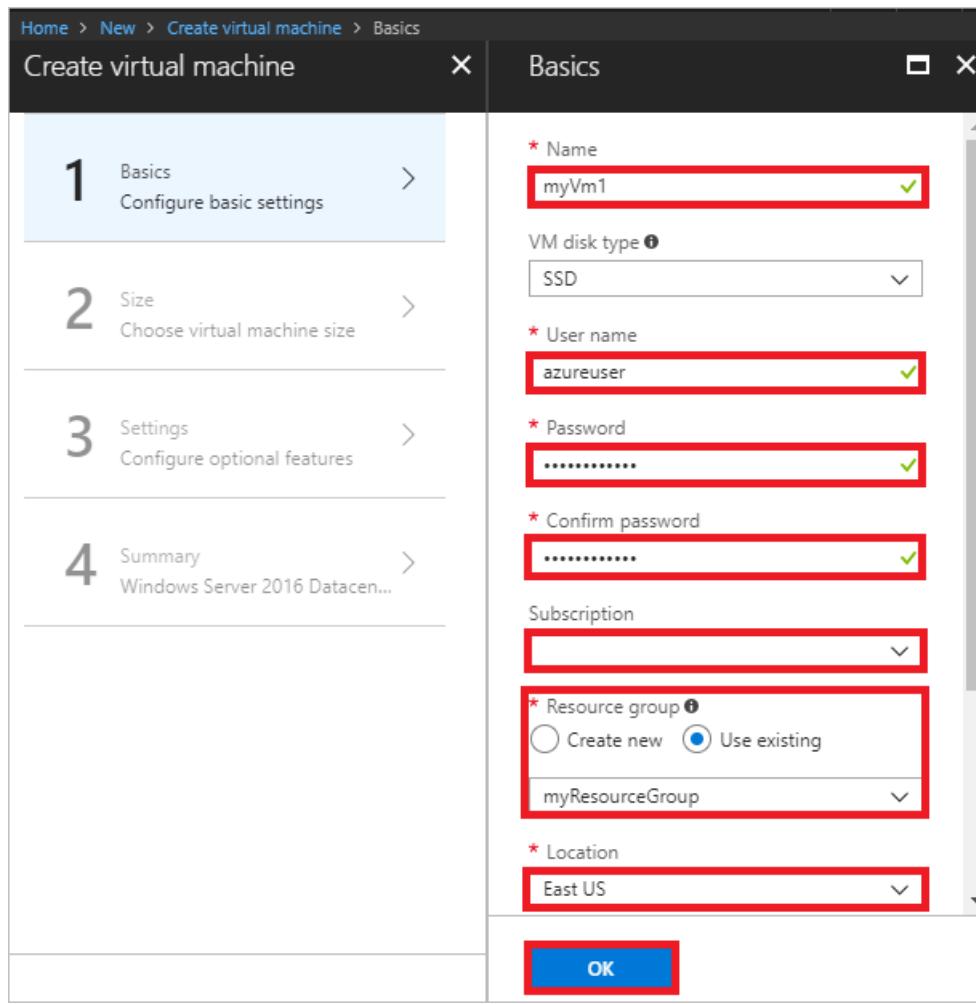
Create virtual machines

Create two VMs in the virtual network:

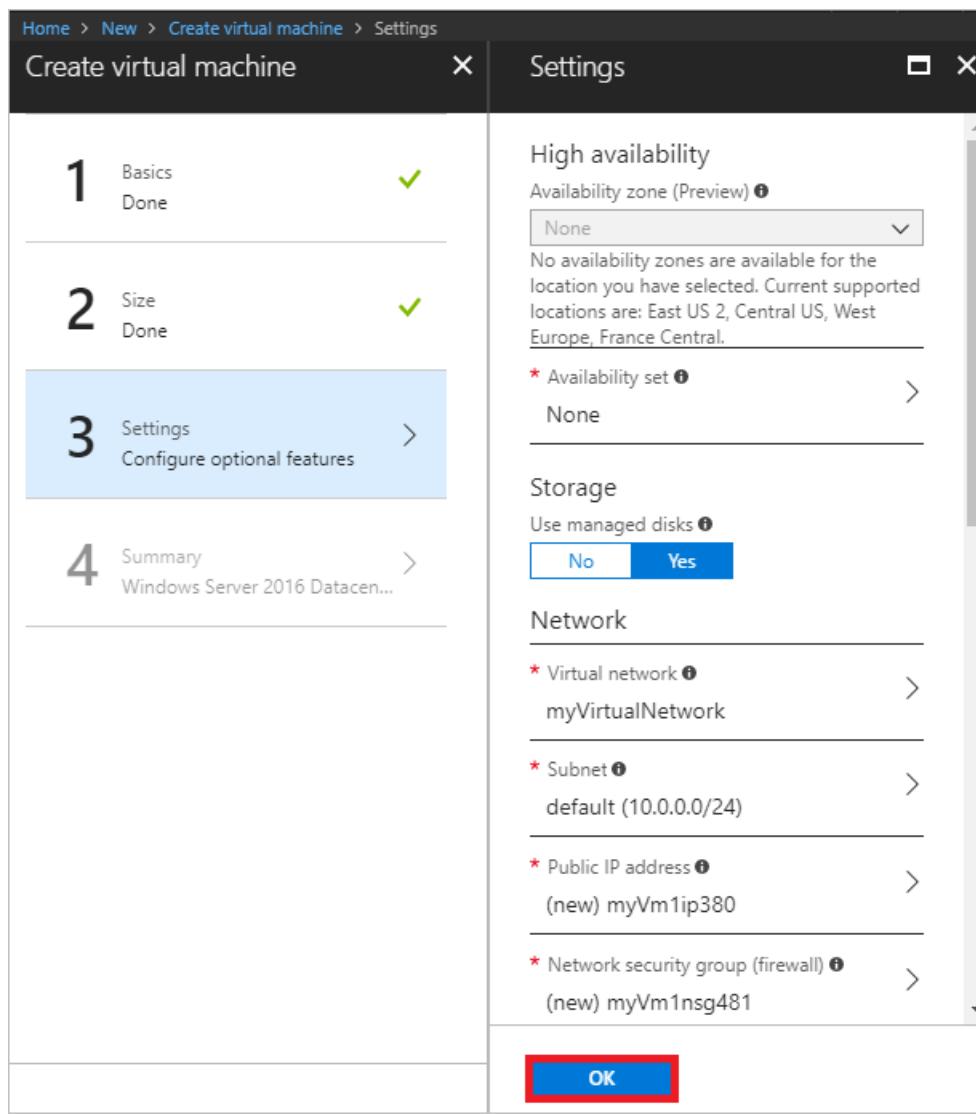
Create the first VM

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, accept the defaults for the remaining settings, and then select **OK**:

SETTING	VALUE
Name	myVm1
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 defined complexity requirements .
Subscription	Select your subscription.
Resource group	Select Use existing and select myResourceGroup .
Location	Select East US



4. Select a size for the VM and then select **Select**.
5. Under **Settings**, accept all the defaults and then select **OK**.



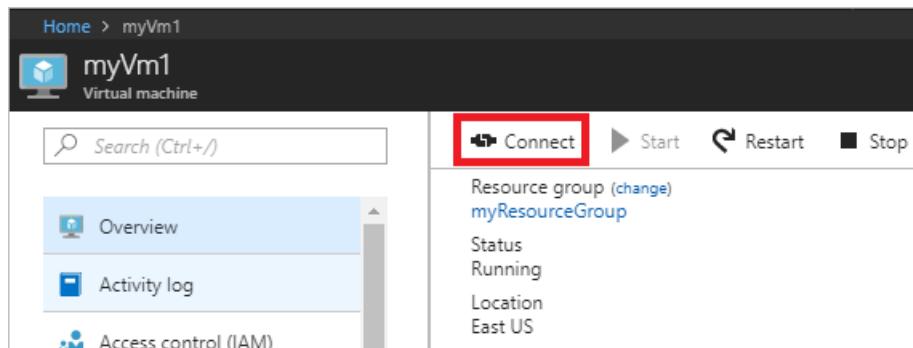
- Under **Create** of the **Summary**, select **Create** to start VM deployment. The VM takes a few minutes to deploy.

Create the second VM

Complete steps 1-6 again, but in step 3, name the VM *myVm2*.

Connect to a VM from the internet

- After *myVm1* is created, connect to it. At the top of the Azure portal, enter *myVm1*. When **myVm1** appears in the search results, select it. Select the **Connect** button.



- After selecting the **Connect** button, 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.

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.

Communicate between VMs

1. From PowerShell, enter `ping myvm2`. Ping fails, because ping uses the Internet Control Message Protocol (ICMP), and ICMP is not allowed through the Windows firewall, by default.
2. To allow *myVm2* to ping *myVm1* in a later step, enter the following command from PowerShell, which allows ICMP inbound through the Windows firewall:

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

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

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

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

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 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](#).

By default, Azure allows unrestricted private communication between virtual machines, but only allows inbound remote desktop connections to Windows VMs from the internet. To learn how to allow or restrict different types of network communication to and from VMs, advance to the [Filter network traffic](#) tutorial.

Quickstart: Create a virtual network using PowerShell

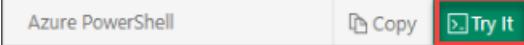
4/18/2018 • 4 min to read • [Edit Online](#)

A virtual network enables Azure resources, such as virtual machines (VM), 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 one VM from the internet, and communicate privately between the two VMs.

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

Launch Azure Cloud Shell

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. Just click the **Copy** to copy the code, paste it into the Cloud Shell, and then press enter to run it. There are a few ways to launch the Cloud Shell:

Click Try It in the upper right corner of a code block.	
Open Cloud Shell in your browser.	
Click the Cloud Shell button on the menu in the upper right of the Azure portal.	

If you choose to install and use PowerShell locally, this quickstart requires the AzureRM PowerShell module version 5.4.1 or later. To find the installed version, run `Get-Module -ListAvailable AzureRM`. If you need to upgrade, see [Install Azure PowerShell module](#). If you are running PowerShell locally, you also need to run `Connect-AzureRmAccount` to create a connection with Azure.

Create a virtual network

Before you can create a virtual network, you must create a resource group to contain the virtual network. Create a resource group with [New-AzureRmResourceGroup](#). The following example creates a resource group named *myResourceGroup* in the *EastUS* location.

```
New-AzureRmResourceGroup -Name myResourceGroup -Location EastUS
```

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

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

Azure resources are deployed to a subnet within a virtual network, so you need to create a subnet. Create a subnet

configuration with [New-AzureRmVirtualNetworkSubnetConfig](#).

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

Write the subnet configuration to the virtual network with [Set-AzureRmVirtualNetwork](#), which creates the subnet within the virtual network:

```
$virtualNetwork | Set-AzureRmVirtualNetwork
```

Create virtual machines

Create two VMs in the virtual network:

Create the first VM

Create a VM with [New-AzureRmVM](#). 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-AzureRmVm `  
-ResourceGroupName "myResourceGroup" `  
-Location "East US" `  
-VirtualNetworkName "myVirtualNetwork" `  
-SubnetName "default" `  
-Name "myVm1" `  
-AsJob
```

Output similar to the following example output is returned, and Azure starts creating the VM in the background.

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

Create the second VM

Enter the following command:

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

The VM takes a few minutes to create. Do not continue with the next step until the previous command executes and output is returned to PowerShell.

Connect to a VM from the internet

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

```
Get-AzureRmPublicIpAddress  
  -Name myVm1  
  -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.

Communicate between VMs

From PowerShell on the *myVm1* VM, enter `ping myvm2`. Ping fails, because ping uses the Internet Control Message Protocol (ICMP), and ICMP is not allowed through the Windows firewall, by default.

To allow *myVm2* to ping *myVm1* in a later step, enter the following command from PowerShell, which allows ICMP inbound through the Windows firewall:

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

Close the remote desktop connection to *myVm1*.

Complete the steps in [Connect to a VM from the internet again](#), but connect to *myVm2*.

From a command prompt on the *myVm2* VM, enter `ping myvm1`.

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

Close the remote desktop connection to *myVm2*.

Clean up resources

When no longer needed, you can use [Remove-AzureRmResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzureRmResourceGroup -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](#).

By default, Azure allows unrestricted private communication between virtual machines, but only allows inbound remote desktop connections to Windows VMs from the internet. To learn how to allow or restrict different types of network communication to and from VMs, continue to the [Filter network traffic](#) tutorial.

Quickstart: Create a virtual network using the Azure CLI

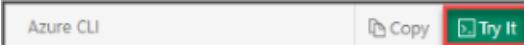
4/9/2018 • 3 min to read • [Edit Online](#)

A virtual network enables Azure resources, such as virtual machines (VM), 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 one VM from the internet, and communicate privately with the other VM.

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

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

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 installed version, run `az --version`. If you need to install or upgrade, see [Install Azure CLI 2.0](#).

Create a virtual network

Before you can create a virtual network, you must create a resource group to contain the virtual network. 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 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 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 that you can continue to the next step. The following 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

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

The VM takes a few minutes to create. After the VM is created, the Azure CLI returns output similar to the following example:

```
{
  "fqdns": "",
  "id": "/subscriptions/00000000-0000-0000-0000-
000000000000/resourceGroups/myResourceGroup/providers/Microsoft.Compute/virtualMachines/myVm1",
  "location": "eastus",
  "macAddress": "00-0D-3A-23-9A-49",
  "powerState": "VM running",
  "privateIpAddress": "10.0.0.5",
  "publicIpAddress": "40.68.254.142",
  "resourceGroup": "myResourceGroup"
}
```

Take note of the **publicIpAddress**. This address is used to connect to the VM from the internet in the next step.

Connect to a VM from the internet

Replace `<publicIpAddress>` with the public IP address of your `myVm2` VM in the command the follows, and then enter the following command:

```
ssh <publicIpAddress>
```

Communicate between VMs

To confirm private communication between the `myVm2` and `myVm1` VMs, enter the following command:

```
ping myVm1 -c 4
```

You 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 of the resources it contains:

```
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 VM and another VM. To learn more about virtual network settings, see [Manage a virtual network](#).

By default, Azure allows unrestricted private communication between virtual machines, but only allows inbound remote desktop connections to Windows VMs from the internet. To learn how to allow or restrict different types of network communication to and from VMs, see [Filter network traffic](#).

Tutorial: Filter network traffic with a network security group using PowerShell

4/18/2018 • 7 min 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:

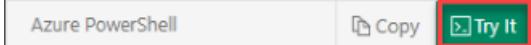
- Create a network security group and security rules
- Create a virtual network and associate a network security group to a subnet
- Deploy virtual machines (VM) into a subnet
- Test traffic filters

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

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

Launch Azure Cloud Shell

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. Just click the **Copy** to copy the code, paste it into the Cloud Shell, and then press enter to run it. There are a few ways to launch the Cloud Shell:

Click Try It in the upper right corner of a code block.	
Open Cloud Shell in your browser.	
Click the Cloud Shell button on the menu in the upper right of the Azure portal.	

If you choose to install and use PowerShell locally, this tutorial requires the Azure PowerShell module version 5.4.1 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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 tutorial with [New-AzureRmResourceGroup](#). The following example creates a resource group in the *eastus* location:

```
New-AzureRmResourceGroup -ResourceGroupName myResourceGroup -Location EastUS
```

Create an application security group with [New-AzureRmApplicationSecurityGroup](#). An application security group enables you to group servers with similar port filtering requirements. The following example creates two application security groups.

```
$webAsg = New-AzureRmApplicationSecurityGroup `  
-ResourceGroupName myResourceGroup `  
-Name myAsgWebServers `  
-Location eastus  
  
$mgmtAsg = New-AzureRmApplicationSecurityGroup `  
-ResourceGroupName myResourceGroup `  
-Name myAsgMgmtServers `  
-Location eastus
```

Create security rules

Create a security rule with [New-AzureRmNetworkSecurityRuleConfig](#). 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-AzureRmNetworkSecurityRuleConfig `  
-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-AzureRmNetworkSecurityRuleConfig `  
-Name "Allow-RDP-All" `  
-Access Allow `  
-Protocol Tcp `  
-Direction Inbound `  
-Priority 110 `  
-SourceAddressPrefix Internet `  
-SourcePortRange * `  
-DestinationApplicationSecurityGroupId $mgmtAsg.id `  
-DestinationPortRange 3389
```

In this tutorial, 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-AzureRmNetworkSecurityGroup](#). The following example creates a network security group named *myNsg*:

```
$nsg = New-AzureRmNetworkSecurityGroup `  
-ResourceGroupName myResourceGroup `  
-Location eastus `  
-Name myNsg `  
-SecurityRules $webRule,$mgmtRule
```

Create a virtual network

Create a virtual network with [New-AzureRmVirtualNetwork](#). The following example creates a virtual named *myVirtualNetwork*:

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

Create a subnet configuration with [New-AzureRmVirtualNetworkSubnetConfig](#), and then write the subnet configuration to the virtual network with [Set-AzureRmVirtualNetwork](#). The following example adds a subnet named *mySubnet* to the virtual network and associates the *myNsg* network security group to it:

```
Add-AzureRmVirtualNetworkSubnetConfig `  
    -Name mySubnet `  
    -VirtualNetwork $virtualNetwork `  
    -AddressPrefix "10.0.2.0/24" `  
    -NetworkSecurityGroup $nsg  
$virtualNetwork | Set-AzureRmVirtualNetwork
```

Create virtual machines

Before creating the VMs, retrieve the virtual network object with the subnet with [Get-AzureRmVirtualNetwork](#):

```
$virtualNetwork = Get-AzureRmVirtualNetwork `  
    -Name myVirtualNetwork `  
    -ResourceGroupName myResourceGroup
```

Create a public IP address for each VM with [New-AzureRmPublicIpAddress](#):

```
$publicIpWeb = New-AzureRmPublicIpAddress [-AllocationMethod Dynamic] -ResourceGroupName  
myResourceGroup [-Location eastus] -Name myVmWeb
```

```
$publicIpMgmt = New-AzureRmPublicIpAddress [-AllocationMethod Dynamic] -ResourceGroupName  
myResourceGroup [-Location eastus] -Name myVmMgmt
```

Create two network interfaces with [New-AzureRmNetworkInterface](#), 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-AzureRmNetworkInterface `  
    -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-AzureRmNetworkInterface ` 
    -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-AzureRmVMConfig](#), then create the VM with [New-AzureRmVM](#). 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-AzureRmVMConfig ` 
    -VMName myVmWeb ` 
    -VMSize Standard_DS1_V2 | ` 
Set-AzureRmVMOperatingSystem -Windows ` 
    -ComputerName myVmWeb ` 
    -Credential $cred | ` 
Set-AzureRmVMSourceImage ` 
    -PublisherName MicrosoftWindowsServer ` 
    -Offer WindowsServer ` 
    -Skus 2016-Datacenter ` 
    -Version latest | ` 
Add-AzureRmVMNetworkInterface ` 
    -Id $webNic.Id
New-AzureRmVM ` 
    -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-AzureRmVMConfig ` 
    -VMName myVmMgmt ` 
    -VMSize Standard_DS1_V2 | ` 
Set-AzureRmVMOperatingSystem -Windows ` 
    -ComputerName myVmMgmt ` 
    -Credential $cred | ` 
Set-AzureRmVMSourceImage ` 
    -PublisherName MicrosoftWindowsServer ` 
    -Offer WindowsServer ` 
    -Skus 2016-Datacenter ` 
    -Version latest | ` 
Add-AzureRmVMNetworkInterface ` 
    -Id $mgmtNic.Id
New-AzureRmVM ` 
    -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-AzureRmPublicIpAddress](#) to return the public IP address of a VM. The following example returns the public IP address of the *myVmMgmt* VM:

```
Get-AzureRmPublicIpAddress ` 
-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-AzureRmPublicIpAddress ` 
-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-AzureRmResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzureRmResourceGroup -Name myResourceGroup -Force
```

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

4/9/2018 • 8 min 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 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 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.

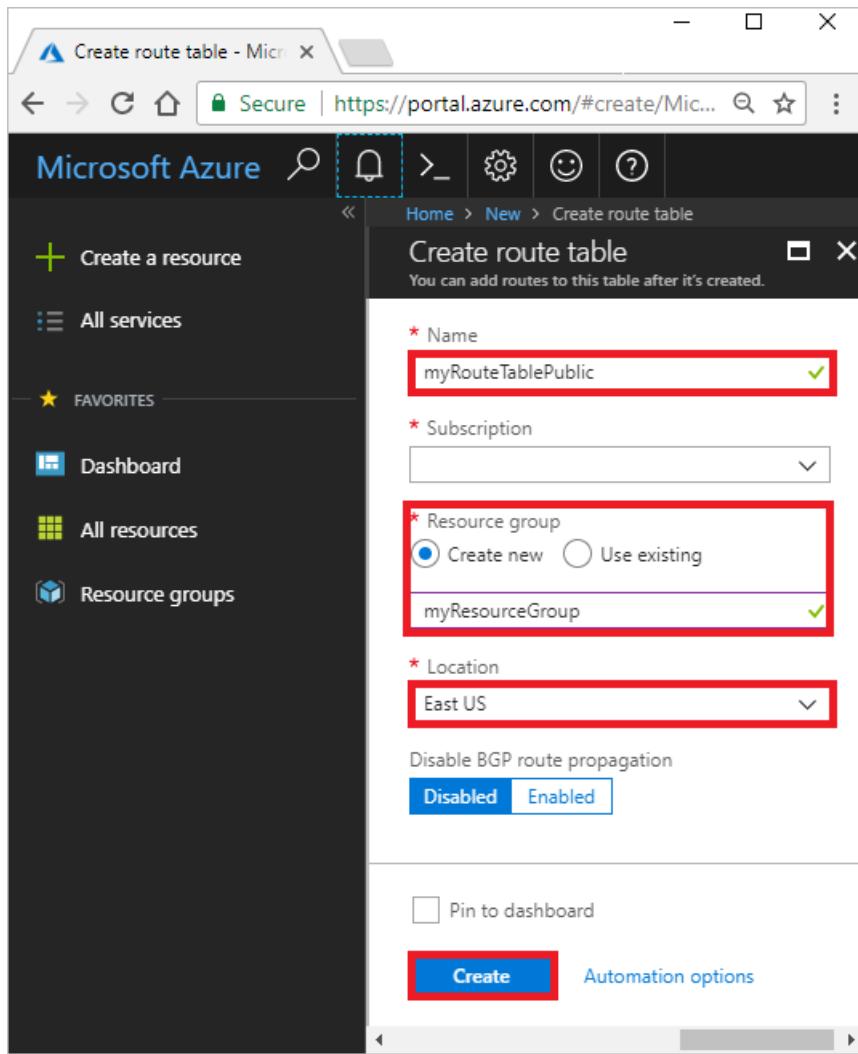
Log in to Azure

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

Create a route table

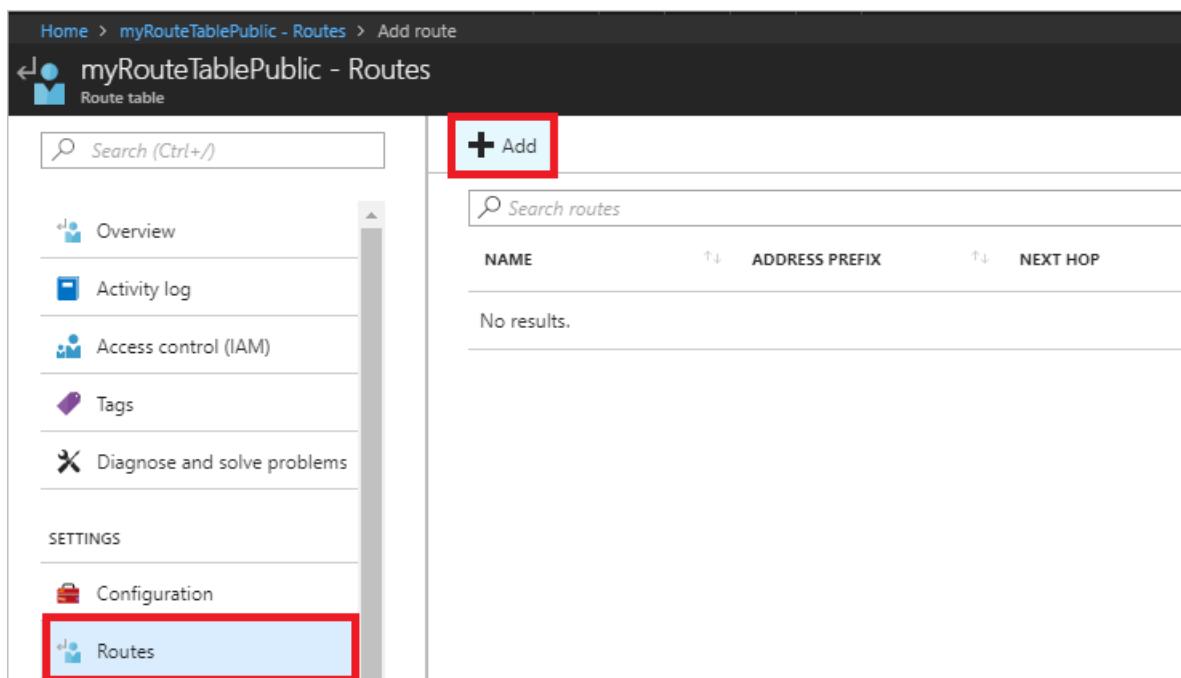
1. Select **+ Create a resource** on the upper, left corner of the Azure portal.
2. Select **Networking**, and then select **Route table**.
3. Enter, or select, the following information, accept the default for the remaining setting, and then select **Create**:

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



Create a route

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



3. Under **Add route**, enter, or select, the following information, accept the default for the remaining settings,

and then select **Create**:

SETTING	VALUE
Route name	ToPrivateSubnet
Address prefix	10.0.1.0/24
Next hop type	Select Virtual appliance .
Next hop 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, then you can associate the route table to a subnet:

1. Select + **Create a resource** on the upper, left corner of the Azure portal.
2. Select **Networking**, and then select **Virtual network**.
3. Under **Create virtual network**, Enter, or select, the following information, accept the default 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 Use existing and then select myResourceGroup .
Location	Select <i>East US</i>
Subnet name	Public
Address range	10.0.0.0/24

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

The screenshot shows the Azure portal interface for managing subnets. On the left, there's a sidebar with various navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Address space, Connected devices, and Subnets. The 'Subnets' option is highlighted with a red box. The main content area is titled 'myVirtualNetwork - Subnets' and shows a table of subnets. The table has columns for NAME, ADDRESS RANGE, and AVAILABLE ADDR... The single entry is 'Public' with the address range '10.0.0.0/24' and available addresses '251'. There are also '+' buttons for creating a new subnet or a gateway subnet.

6. Select or enter the following information, then select **OK**:

SETTING	VALUE
Name	Private
Address space	10.0.1.0/24

7. Complete steps 5 and 6 again, providing the following information:

SETTING	VALUE
Name	DMZ
Address space	10.0.2.0/24

8. The **myVirtualNetwork - Subnets** box is displayed after completing the previous step. Under **SETTINGS**, select **Subnets** and then select **Public**.

9. As shown in the following picture, select **Route table**, select **MyRouteTablePublic**, and then select **Save**:

The screenshot shows the configuration page for the 'Public' subnet. At the top, there are buttons for Save, Discard, Delete, and Refresh. Below that, there's a section for the address range '10.0.0.0/24' and a note about available addresses. Further down, there's a 'Network security group' field set to 'None'. The most prominent part is a 'Route table' dropdown menu, which is currently set to 'None' and has a red box around it. To the right of the dropdown, there's a list of route tables: 'None' and 'myRouteTablePublic (eastus)'. The 'myRouteTablePublic' option is also highlighted with a red box. A message at the top right says 'These are the route tables in the selected subscription and location 'East US''. The overall title of the page is 'myVirtualNetwork - Subnets > Public > Resource'.

Create an NVA

An NVA is a VM that performs a network function, such as routing, firewalling, or WAN optimization.

1. Select + **Create a resource** on the upper, left corner of the Azure portal.
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. Select or enter the following information for **Basics**, then select **OK**:

SETTING	VALUE
Name	myVmNva
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 defined complexity requirements .
Resource group	Select Use existing and then select <i>myResourceGroup</i> .
Location	Select East US .

4. Select a VM size under **Choose a size**.
5. Select or enter the following information for **Settings**, then select **OK**:

SETTING	VALUE
Virtual network	myVirtualNetwork - If it's not selected, select Virtual network , then select myVirtualNetwork under Choose virtual network .
Subnet	Select Subnet and then select DMZ under Choose subnet .
Public IP address	Select Public IP address and select None under Choose public IP address . No public IP address is assigned to this VM since it won't be connected to from the internet.

6. Under **Create** in the **Summary**, select **Create** to start the VM deployment.

The VM takes a few minutes to create. Do not continue to the next step until Azure finishes creating the VM and opens a box with information about the VM.

7. In the box that opened for the VM after it was created, under **SETTINGS**, select **Networking**, and then select **myvmnva158** (the network interface Azure created for your VM has a different number after **myvmnva**), as shown in the following picture:

8. 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. Under **SETTINGS**, select **IP configurations**, select **Enabled** for **IP forwarding**, and then select **Save**, as shown in the following picture:

Create virtual machines

Create two VMs in the virtual network, so that you can validate that traffic from the *Public* subnet is routed to the *Private* subnet through the NVA in a later step. Complete steps 1-6 of [Create a NVA](#). Use the same settings in steps 3 and 5, except for the following changes:

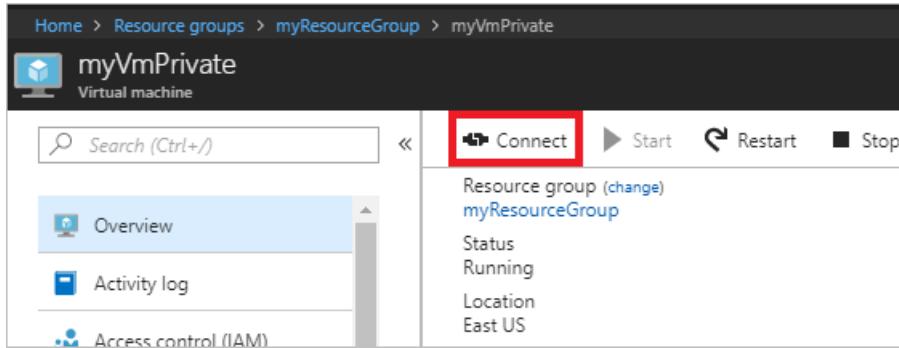
VIRTUAL MACHINE NAME	SUBNET	PUBLIC IP ADDRESS
myVmPublic	Public	Accept portal default
myVmPrivate	Private	Accept portal default

You can create the *myVmPrivate* VM while Azure creates the *myVmPublic* VM. Do not continue with the following steps until Azure finishes creating both VMs.

Route traffic through an NVA

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

following picture:



3. To connect to the VM, open the downloaded RDP file. If prompted, select **Connect**.
4. 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**.
5. You may receive a certificate warning during the sign-in process. Select **Yes** to proceed with the connection.
6. In a later step, the trace route tool is used to test routing. Trace route 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 tutorial, allowing ICMP through the Windows Firewall for production deployments is not recommended.

7. 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* VM:

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

```
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.

8. 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
```

9. 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.

10. Close the remote desktop session to the *myVmPublic* VM, which leaves you still connected to the *myVmPrivate* VM.
11. 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.

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

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 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 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

4/9/2018 • 9 min 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 <http://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 Create new and enter <i>myResourceGroup</i> .
Location	Select East US
Subnet Name	Public
Subnet Address range	10.0.0.0/24

SETTING	VALUE
Service endpoints	Disabled

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with options like 'Create a resource', 'All services', 'Dashboard', 'All resources', and 'Resource groups'. The main area has a search bar at the top. Below it, a 'New' section lists various Azure services. Under 'Networking', 'Virtual network' is highlighted with a blue box. To the right, a 'Create virtual network' configuration pane is open. It includes fields for 'Name' (set to 'myVirtualNetwork'), 'Address space' (set to '10.0.0.0/16'), 'Subscription' (a dropdown menu), 'Resource group' (radio buttons for 'Create new' or 'Use existing', with 'Create new' selected and 'myResourceGroup' chosen), 'Location' (set to 'East US'), 'Subnet' (with 'Name' set to 'Public'), 'Address range' (set to '10.0.0.0/24'), and 'Service endpoints' (radio buttons for 'Disabled' or 'Enabled', with 'Enabled' selected). At the bottom of the pane are 'Create' and 'Automation options' buttons.

Enable a service endpoint

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 Microsoft.Storage under Services

Restrict network access for a 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 Use existing and select <i>myResourceGroup</i> .
Location	Select East US

- 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.
- Under **SETTINGS**, select **Outbound security rules**.
- Select **+ Add**.
- Create a rule that allows outbound access to the public IP addresses assigned to the Azure Storage service. Enter, or select, the following information, and then select **OK**:

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

7. Create a rule that overrides a default security rule that allows outbound access to all public IP addresses.

Complete steps 6 and 7 again, using the following values:

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

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

9. Select **+ Add**.

10. Create a rule that allows Remote Desktop Protocol (RDP) traffic inbound 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. Complete steps 6 and 7 again, using the following values:

SETTING	VALUE
Source	Any
Source port ranges	*
Destination	Select Service Tag
Destination service tag	Select VirtualNetwork
Destination port ranges	3389
Protocol	Any
Action	Allow
Priority	120
Name	Allow-RDP-All

11. Under **SETTINGS**, select **Subnets**.
12. Select + **Associate**
13. Under **Associate subnet**, select **Virtual network** and then select **myVirtualNetwork** under **Choose a virtual network**.
14. 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.
Account kind	StorageV2 (general purpose v2)
Replication	Locally-redundant storage (LRS)
Subscription	Select your subscription
Resource group	Select Use existing and select <i>myResourceGroup</i> .

SETTING	VALUE
Location	Select East US

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:

The screenshot shows the Azure Storage Account blade for a resource group named 'myResourceGroup'. The left sidebar lists 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', and 'Diagnose and solve problems'. Under 'SETTINGS', there are links for 'Access keys', 'Configuration', 'Encryption', and 'Event Grid'. The main content area shows general account details: Resource group (myResourceGroup), Status (Primary: Available), Location (East US), Subscription (StorageV2 (general purpose v2)), and a 'Subscription ID'. Below this is the 'Services' section, which includes 'Blobs' (Object storage for unstructured data) and 'Files' (File shares that use SMB 3.0 protocol). The 'Files' link is highlighted with a red box.

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

Enable network access from a subnet

By default, storage accounts accept network connections from clients in any network. To allow access from only a specific subnet, and deny network access from all other networks, complete the following steps:

1. Under **SETTINGS** for the storage account, select **Firewalls and virtual networks**.
2. Under **Virtual networks**, 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 myVirtualNetwork , under Virtual networks
Subnets	Select Private , under Subnets

The screenshot shows the 'vnt - Firewalls and virtual networks' blade in the Azure portal. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and SETTINGS (which is expanded). Under SETTINGS, 'Access keys' is highlighted with a red box. The main content area shows 'Allow access from' settings with 'Selected networks' selected. Below that is a 'Virtual networks' section with a table for adding existing virtual networks. A red box highlights the 'Virtual networks' dropdown set to 'myVirtualNetwork' and the 'Subnets' dropdown set to 'Private'. At the bottom right is a large blue 'Add' button.

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 'vnt - Access keys' blade in the Azure portal. The left sidebar has 'Access keys' highlighted with a red box under the SETTINGS section. The main pane contains instructions about using access keys for authentication, a 'Storage account name' input field with 'vnt', and a 'key1' section with a key value starting with 'xxyyxy...'. A red box highlights both the 'key1' section and the long key value.

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.

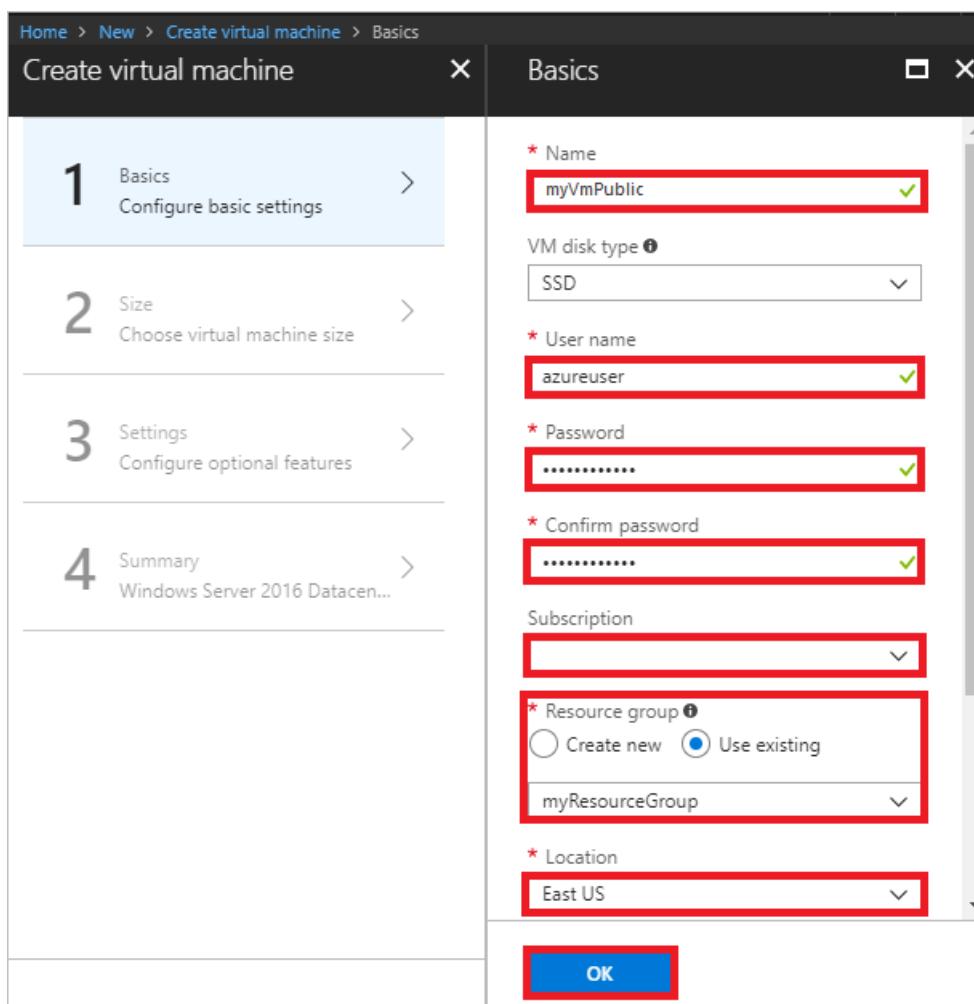
Create virtual machines

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

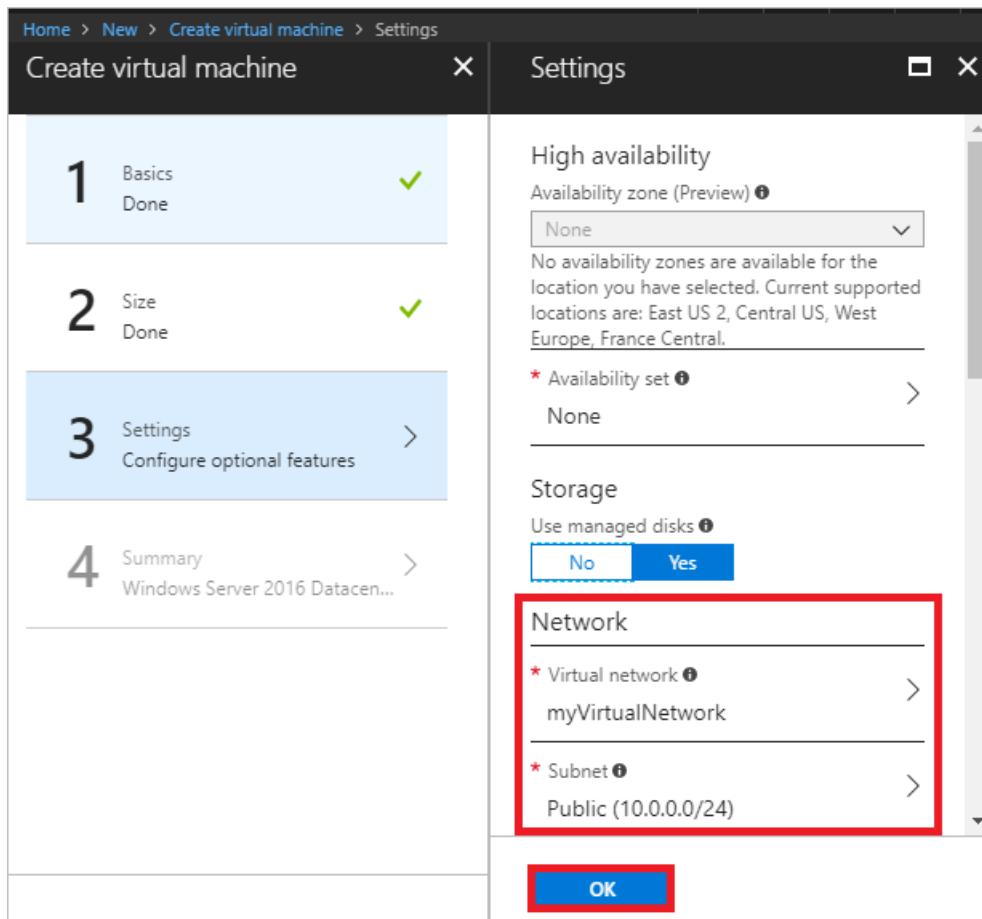
Create the first virtual machine

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 defined complexity requirements .
Subscription	Select your subscription.
Resource group	Select Use existing and select myResourceGroup .
Location	Select East US .



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:



- 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.

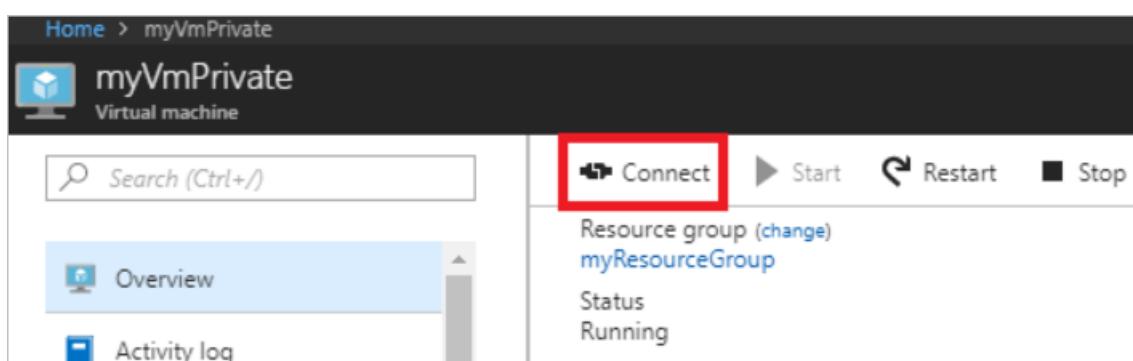
Create the second virtual machine

Complete steps 1-6 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

- 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:



- After selecting the **Connect** button, 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.

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 any other public IP addresses 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 public IP addresses other than the addresses assigned to the Azure Storage service.

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.

Access 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.

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:

Home > File service

File service

vnt

Access denied

You do not have access

This storage account's 'Firewalls and virtual networks' settings may be blocking access to storage services. Try adding your client IP address to the firewall exceptions, or by allowing access from 'all networks' instead of 'selected networks'.

[Learn more](#)



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 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 to connect virtual networks, advance to the next tutorial.

[Connect virtual networks](#)

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

4/9/2018 • 5 min 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 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. 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, accept the defaults for the remaining settings, and then select **Create**:

SETTING	VALUE
Name	myVirtualNetwork1
Address space	10.0.0.0/16
Subscription	Select your subscription.
Resource group	Select Create new and enter <i>myResourceGroup</i> .
Location	Select East US .
Subnet Name	Subnet1
Subnet Address range	10.0.0.0/24

4. Complete steps 1-3 again, with the following changes:

SETTING	VALUE
Name	myVirtualNetwork2
Address space	10.1.0.0/16
Resource group	Select Use existing and then select myResourceGroup .
Subnet Address range	10.1.0.0/24

Peer virtual networks

- In the Search box at the top of the Azure portal, begin typing *MyVirtualNetwork1*. When **myVirtualNetwork1** appears in the search results, select it.
- 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	myVirtualNetwork1-myVirtualNetwork2
Subscription	Select your subscription.
Virtual network	myVirtualNetwork2 - To select the <i>myVirtualNetwork2</i> virtual network, select Virtual network , then select myVirtualNetwork2 .

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

The screenshot shows the Azure portal interface for managing virtual network peerings. The left sidebar has links for Overview, Activity log, and Access control (IAM). The main area is titled 'myVirtualNetwork1 - Peerings' under 'Virtual network'. A search bar at the top says 'Search (Ctrl+ /)'. Below it, there's a table with one row. The first column is 'NAME' with the value 'myVirtualNetwork1-myVirtualNetwork2'. The second column is 'PEERING STATUS' with the value 'Initiated' (which is highlighted with a red box). The third column is 'PEER' with the value 'myVirtualNetwork2'. The fourth column is 'GATEWAY TRANSIT' with the value 'Disabled'. There's also a '...' button.

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

4. In the **Search** box at the top of the Azure portal, begin typing *MyVirtualNetwork2*. When **myVirtualNetwork2** appears in the search results, select it.
5. Complete steps 2-3 again, with the following changes, and then select **OK**:

SETTING	VALUE
Name	myVirtualNetwork2-myVirtualNetwork1
Virtual network	myVirtualNetwork1

The **PEERING STATUS** is *Connected*. Azure also changed the peering status for the *myVirtualNetwork2-myVirtualNetwork1* peering from *Initiated* to *Connected*. Virtual network peering is not fully established until the peering status for 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.

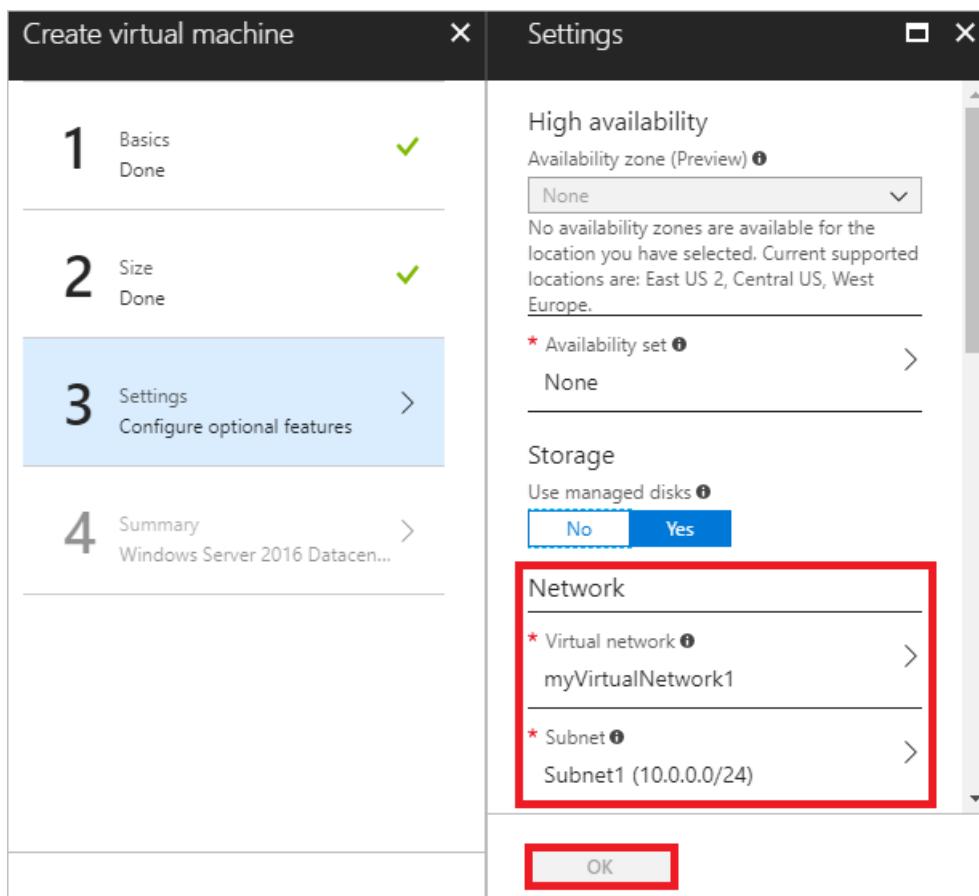
Create the first VM

1. Select **+ Create a resource** on the upper, left corner of the Azure portal.
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
Name	myVm1
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 defined complexity requirements .
Resource group	Select Use existing and then select myResourceGroup .
Location	Select East US .

4. Select a VM size under **Choose a size**.
5. Select the following values for **Settings**, then select **OK**:

SETTING	VALUE
Virtual network	myVirtualNetwork1 - If it's not already selected, select Virtual network and then select myVirtualNetwork1 under Choose virtual network .
Subnet	Subnet1 - If it's not already selected, select Subnet and then select Subnet1 under Choose subnet .



6. Under **Create** in the **Summary**, select **Create** 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:

Resource group (change)
myResourceGroup

Status
Running

Location
East US

Subscription (change)

Subscription ID

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

DNS name
myvm1-efb1f0.eastus.cloudapp.azure.com

3. To connect to the VM, open the downloaded RDP file. If prompted, select **Connect**.
4. 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**.
5. You may receive a certificate warning during the sign-in process. Select **Yes** to proceed with the connection.
6. 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

4/9/2018 • 1 min to read • [Edit Online](#)

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

Create a virtual network for multi-tier applications	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.
Peer two virtual networks	Creates and connects two virtual networks in the same region.
Route traffic through a network virtual appliance	Creates a virtual network with front-end and back-end subnets and a VM that is able to route traffic between the two subnets.
Filter inbound and outbound VM network traffic	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.

Azure PowerShell samples for virtual network

4/9/2018 • 1 min to read • [Edit Online](#)

The following table includes links to Azure Powershell scripts:

Create a virtual network for multi-tier applications	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.
Peer two virtual networks	Creates and connects two virtual networks in the same region.
Route traffic through a network virtual appliance	Creates a virtual network with front-end and back-end subnets and a VM that is able to route traffic between the two subnets.
Filter inbound and outbound VM network traffic	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.

Azure Resource Manager template samples for virtual network

4/17/2018 • 1 min 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](#).

Create a virtual network with two subnets	Creates a virtual network with two subnets.
Route traffic through a network virtual appliance	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.
Create a virtual network service endpoint for Azure Storage	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.
Connect two virtual networks	Creates two virtual networks and a virtual network peering between them.
Create a virtual machine with multiple IP addresses	Creates a Windows or Linux VM with multiple IP addresses.

Azure policy sample templates for virtual network

5/2/2018 • 1 min to read • [Edit Online](#)

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

Network	
NSG X on every NIC	Requires that a specific network security group is used with every virtual network interface. You specify the ID of the network security group to use.
NSG X on every subnet	Requires that a specific network security group is used with every virtual subnet. You specify the ID of the network security group to use.
No route table	Prohibits virtual networks from being deployed with a route table.
Use approved subnet for VM network interfaces	Requires that network interfaces use an approved subnet. You specify the ID of the approved subnet.
Use approved vNet for VM network interfaces	Requires that network interfaces use an approved virtual network. You specify the ID of the approved virtual network.
Monitoring	
Audit diagnostic setting	Audits if diagnostic settings are not enabled for specified resource types. You specify an array of resource types to check whether diagnostic settings are enabled.
Name and text conventions	
Allow multiple name patterns	Allow one of many name patterns to be used for resources.
Require like pattern	Ensure resource names meet the <i>like</i> condition for a pattern.
Require match pattern	Ensure resource names match a specified naming pattern.
Require tag match pattern	Ensure that a tag value matches a text pattern.
Tags	
Billing tags policy initiative	Requires specified tag values for cost center and product name. Uses built-in policies to apply and enforce required tags. You specify the required values for the tags.
Enforce tag and its value on resource groups	Requires a tag and value on a resource group. You specify the required tag name and value.

Enforce tag and its value	Requires a specified tag name and value. You specify the tag name and value to enforce.
Apply tag and its default value	Appends a specified tag name and value, if that tag is not provided. You specify the tag name and value to apply.
General	
Allowed locations	Requires that all resources are deployed to the approved locations. You specify an array of approved locations.
Allowed resource types	Ensures only approved resource types are deployed. You specify an array of resource types that are permitted.
Not allowed resource types	Prohibits the deployment of specified resource types. You specify an array of the resource types to block.

Network security

5/3/2018 • 15 min to read • [Edit Online](#)

You can limit network traffic to resources in a virtual network using a network security group. A network security group contains a list of security rules that allow or deny inbound or outbound network traffic based on source or destination IP address, port, and protocol.

Network security groups

Each network interface has zero, or one, associated network security group. Each network interface exists in a [virtual network](#) subnet. A subnet can also have zero, or one, associated network security group.

When applied to a subnet, security rules are applied to all resources in the subnet. In addition to network interfaces, you may have instances of other Azure services such as HDInsight, Virtual Machine Scale Sets, and Application Service Environments deployed in the subnet.

How network security groups are applied when a network security group is associated to both a network interface, and the subnet the network interface is in, is as follows:

- **Inbound traffic:** The network security group associated to the subnet the network interface is in is evaluated first. Any traffic allowed through the network security group associated to the subnet is then evaluated by the network security group associated to the network interface. For example, you might require inbound access to a virtual machine over port 80 from the Internet. If you associate a network security group to both the network interface, and the subnet the network interface is in, the network security group associated to the subnet and the network interface must allow port 80. If you only allowed port 80 through the network security group associated to the subnet or the network interface the subnet is in, communication fails due to default security rules. See [default security rules](#) for details. If you only applied a network security group to either the subnet or the network interface, and the network security group contained a rule that allowed inbound port 80 traffic, for example, the communication succeeds.
- **Outbound traffic:** The network security group associated to the network interface is evaluated first. Any traffic allowed through the network security group associated to the network interface is then evaluated by the network security group associated to the subnet.

You may not always be aware when network security groups are applied to both a network interface and a subnet. 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. The tool tells you whether communication is allowed, and which network security rule allows or denies traffic.

NOTE

Network security groups are associated to subnets or to virtual machines and cloud services deployed the classic deployment model, rather than to network interfaces in the Resource Manager deployment model. To learn more about Azure deployment models, see [Understand Azure deployment models](#).

The same network security group can be applied to as many individual network interfaces and subnets as you choose.

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, CIDR block (example 10.0.0.0/24, for example), service tag, or application security group. Learn more about service tags and application security groups . 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 augmented security rules. Learn more about augmented security rules . 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.
Protocol	TCP, UDP, or Any, which includes TCP, UDP, and ICMP. You cannot specify ICMP alone, so if you require ICMP, you must use Any.
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

NSG 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 records, this allows NSG 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. An 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 on either direction for at least a few minutes.

To learn about limits when creating security rules, see [Azure limits](#).

Augmented security rules

Augmented 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, multiple explicit IP addresses, Service tags, and Application security groups into a single, easily understood security rule. Use augmented rules in the source, destination, and port fields of a rule. When creating a rule, you can specify multiple explicit IP addresses, CIDR ranges, and ports. To simplify maintenance of your security rule definition, combine augmented security rules with service tags or application security groups.

To learn about limits when creating augmented security rules, see [Azure limits](#).

Default security rules

If a network security group is not associated to a subnet or network interface, all traffic is allowed inbound to, or outbound from, the subnet, or network interface. As soon as a network security group is created, Azure creates the following default rules within the network security group:

Inbound

AllowVNetInBound

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65000	VirtualNetwork	0-65535	VirtualNetwork	0-65535	All	Allow

AllowAzureLoadBalancerInBound

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65001	AzureLoadBalancer	0-65535	0.0.0.0/0	0-65535	All	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	All	Deny

Outbound

AllowVnetOutBound

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65000	VirtualNetwork	0-65535	VirtualNetwork	0-65535	All	Allow

AllowInternetOutBound

PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
65001	0.0.0.0/0	0-65535	Internet	0-65535	All	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	All	Deny

In the **Source** and **Destination** columns, *VirtualNetwork*, *AzureLoadBalancer*, and *Internet* are [service tags](#), rather than IP addresses. In the protocol column, **All** encompasses TCP, UDP, and ICMP. When creating a rule, you can specify TCP, UDP, or All, but you cannot specify ICMP alone. Therefore, if your rule requires ICMP, you must select *All* for protocol. *0.0.0.0/0* in the **Source** and **Destination** columns represents all addresses.

You cannot remove the default rules, but you can override them by creating rules with higher priorities.

Service tags

A service tag represents a group of IP address prefixes to help minimize complexity for security rule creation. You cannot create your own service tag, nor specify which IP addresses are included within a tag. Microsoft manages the address prefixes encompassed by the service tag, and automatically updates the service tag as addresses change. You can use service tags in place of specific IP addresses when creating security rules. The following service tags are available for use in security rule definition. Their names vary slightly between [Azure deployment models](#).

- **VirtualNetwork** (Resource Manager) (**VIRTUAL_NETWORK** for classic): This tag includes the virtual network address space (all CIDR ranges defined for the virtual network), all connected on-premises address spaces, and [peered](#) virtual networks or virtual network connected to a [virtual network gateway](#).
- **AzureLoadBalancer** (Resource Manager) (**AZURE_LOADBALANCER** for classic): This tag denotes Azure's infrastructure load balancer. The tag translates to an [Azure datacenter IP address](#) where Azure's health probes originate. If you are not using the Azure load balancer, you can override this rule.
- **Internet** (Resource Manager) (**INTERNET** for classic): This tag denotes the IP address space that is outside the virtual network and reachable by the public Internet. The address range includes the [Azure owned public IP address space](#).
- **AzureTrafficManager** (Resource Manager only): This tag denotes the IP address space for the Azure Traffic Manager probe IPs. More information on Traffic Manager probe IPs can be found in the [Azure Traffic Manager FAQ](#).
- **Storage** (Resource Manager only): This tag denotes the IP address space for the Azure Storage service. If you specify *Storage* for the value, traffic is allowed or denied to storage. If you only want to allow access to storage in a specific [region](#), you can specify the region. For example, if you want to allow access only to Azure Storage in the East US region, you could specify *Storage.EastUS* as a service tag. The 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. All address prefixes represented by this tag are also represented by the **Internet** tag.
- **Sql** (Resource Manager only): This tag denotes the address prefixes of the Azure SQL Database and Azure SQL Data Warehouse services. If you specify *Sql* for the value, traffic is allowed or denied to Sql. If you only want to allow access to Sql in a specific [region](#), you can specify the region. For example, if you want to allow access only to Azure SQL Database in the East US region, you could specify *Sql.EastUS* as a service tag. The 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. All address prefixes represented by this tag are also represented by the **Internet** tag.

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 the corresponding service tag.

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. This feature allows you to 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.

You can specify an application security group as the source and destination in a security rule. Once your security policy is defined, you can create virtual machines and assign the network interfaces in the virtual machine to an application security group. The policy is applied based on the application security group membership of each network interface within a virtual machine. The following example illustrates how you might use an application security group for all web servers in your subscription:

1. Create an application security group named *WebServers*.
2. Create a network security group named *MyNSG*.
3. Create an inbound security rule in the network security group, specifying the *Internet* service tag for source address and the *WebServers* application security group as the destination address, and allow ports 80 and 443.
4. Deploy a virtual machine running a web server application. Specify that the network interface in the virtual machine is a member of the *WebServers* application security group. Ports 80 and 443 are then allowed to the virtual machine. The ports are also allowed to any subsequent web servers you create that you make members of the *WebServers* application security group.

If you create other rules, specifying other application security groups as the destination, those rules aren't applied to the web servers, in the previous example. Rules specifying an application security group are only applied to network interfaces that are members of the application security group. Application security groups, combined with augmented security rules and service tags, make it possible to create a minimal number of network security groups to manage network security within your subscription.

To learn about limits when creating application security groups and specifying them in security rules, see [Azure limits](#).

Application security groups have the following constraints:

- All network interfaces within an application security group must exist in the same virtual network. You cannot add network interfaces from different virtual networks to the same application security group. The virtual network the first network interface assigned to the application security group is in defines the virtual network all subsequently assigned network interfaces must exist in.
- If you specify application security groups 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 ASG1 contained network interfaces from VNet1, and ASG2 contained network interfaces from VNet2, you could not assign ASG1 as the source and ASG2 as the destination in a rule, all network interfaces need to exist in VNet1.

Azure platform considerations

- **Virtual IP of the host node:** Basic infrastructure services such as DHCP, DNS, and health monitoring are provided through the virtualized host IP addresses 168.63.129.16 and 169.254.169.254. These public IP

addresses belong to Microsoft and are the only virtualized IP addresses used in all regions for this purpose. The addresses map to the physical IP address of the server machine (host node) hosting the virtual machine. The host node acts as the DHCP relay, the DNS recursive resolver, and the probe source for the load balancer health probe and the machine health probe. Communication to these IP addresses is not an attack. If you block traffic to or from these IP addresses, a virtual machine may not function properly.

- **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.
- **Cloud service provider (CSP), 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 must send email from your virtual machine, you must use an SMTP relay service.

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, you must 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 traffic routing

4/24/2018 • 24 min 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	172.16.0.0/12	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, 172.16.0.0/12, 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, 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 2.0 AND POWERSHELL (RESOURCE MANAGER)	AZURE CLI 1.0 AND POWERSHELL (CLASSIC)
Virtual network gateway	VirtualNetworkGateway	VPNGateway
Virtual network	VNetLocal	VNETLocal (not available in the CLI 1.0 in asm mode)
Internet	Internet	Internet (not available in the CLI 1.0 in asm mode)
Virtual appliance	VirtualAppliance	VirtualAppliance
None	None	Null (not available in the CLI 1.0 in asm mode)

Next hop type	Azure CLI 2.0 and PowerShell (Resource Manager)	Azure CLI 1.0 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.

BGP 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 BGP propagation disabled. Connectivity with VPN connections is achieved using custom routes](#custom-routes) with a next hop type of VPN. For details, see [How to disable BGP 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](#).

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

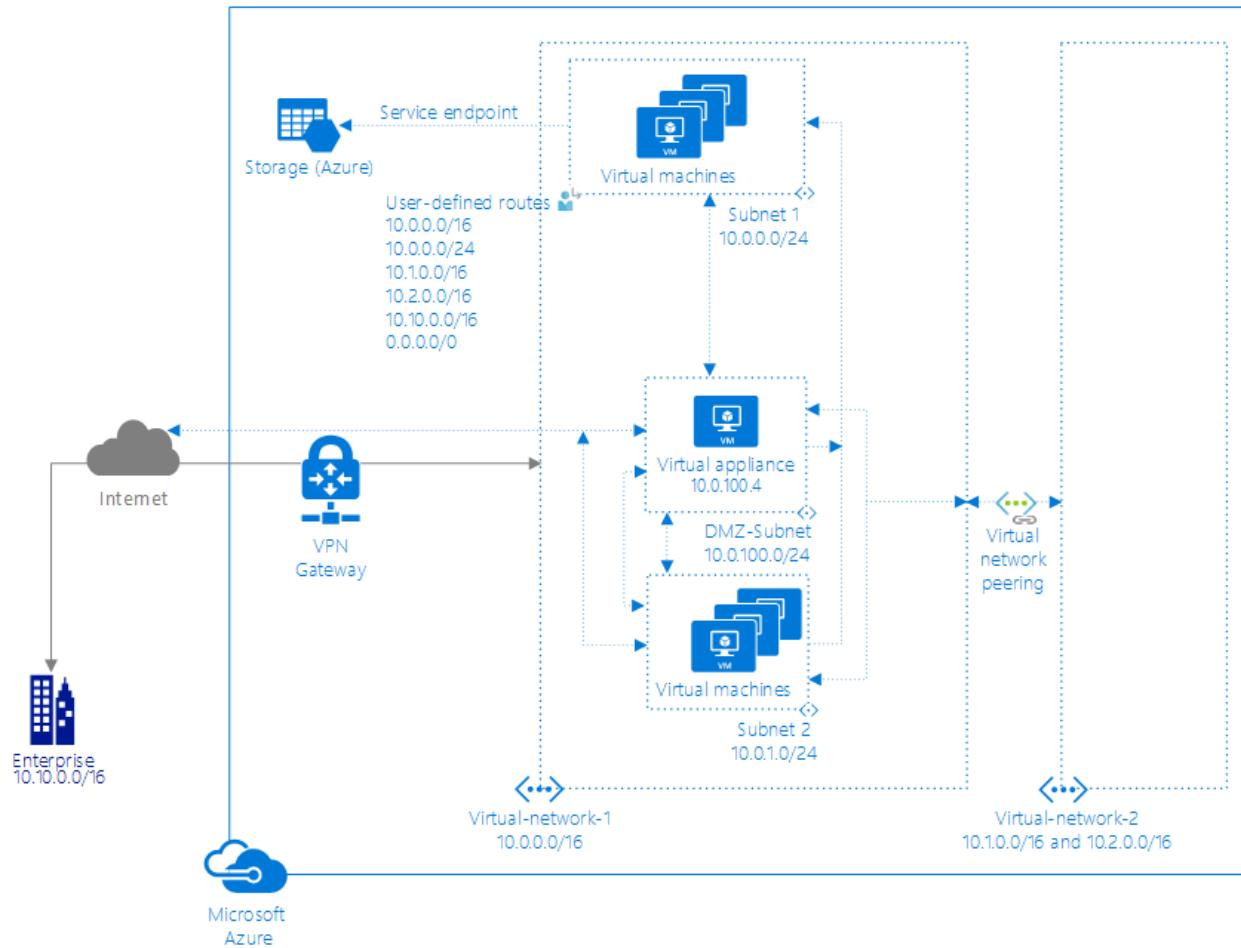
This example is not intended to be a recommended or best practice implementation. Rather, it is provided only to illustrate concepts in this article.

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 10.0.0.0.254 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 4 and 5 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 [requirement 3](#).

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	172.16.0.0/12	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, 172.16.0.0/12, 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.

Virtual Network Service Endpoints

5/7/2018 • 8 min to read • [Edit Online](#)

Virtual Network (VNet) service endpoints extend your virtual network private address space and 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:

- **Azure Storage:** Generally Available in all Azure regions
- **Azure SQL Database:** Generally Available in all Azure regions
- **Azure Cosmos DB:** Generally Available in all Azure public cloud regions
- **Azure SQL Data Warehouse:** Preview in all Azure public cloud regions

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:** With service endpoints, Azure service resources can be secured to your virtual network. Securing service resources to a virtual network provide 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 premises and/or virtual appliances, known as forced-tunneling, 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. Learn more about [user-defined routes and forced-tunneling](#).

- **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 NAT or gateway devices required to set up the service endpoints. Service endpoints are configured through a simple click on a subnet. There is no additional overhead to maintaining the endpoints.

Limitations

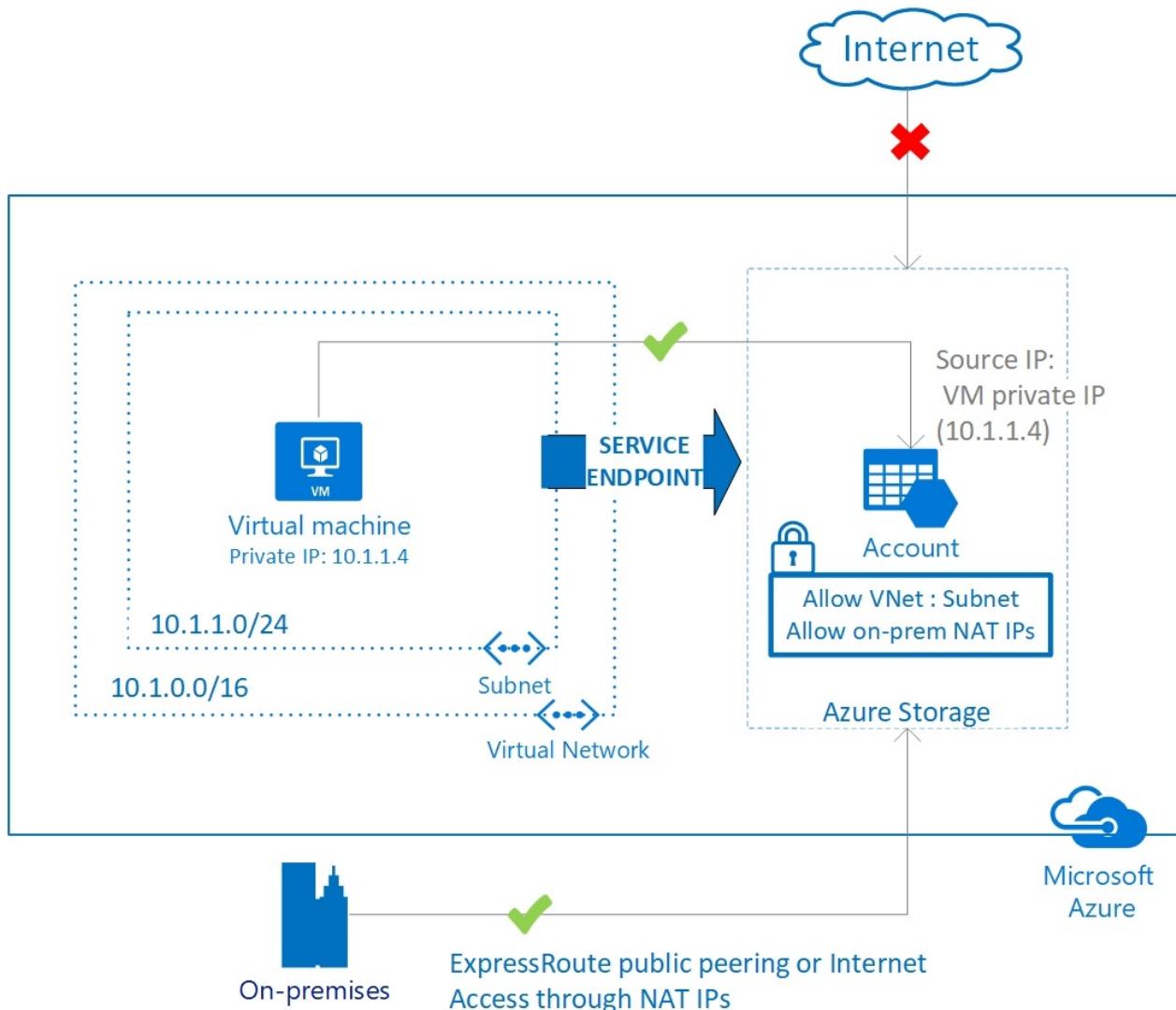
- The feature is available only to virtual networks deployed through the Azure Resource Manager deployment model.
- Endpoints are enabled on subnets configured in Azure virtual networks. Endpoints cannot be used for traffic from your premises to Azure services. For more information, see [Securing Azure 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, to support RA-GRS and GRS traffic, endpoints also extend to include paired regions where the virtual network is deployed. Learn more about [Azure paired regions..](#)

Securing Azure services to virtual networks

- A virtual network service endpoint provides the identity of your virtual network to the Azure service. Once service endpoints are enabled in your virtual network, you can secure Azure service resources to your virtual network by adding a virtual network rule to the resources.
- 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.
- **Securing Azure service access from on-premises:**

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 Azure service resources.

ExpressRoute: If you are using [ExpressRoute](#) from your premises, for public peering or Microsoft peering, you will need to identify the NAT IP addresses that are used. For public peering, each ExpressRoute circuit by default uses two NAT IP addresses applied to Azure service traffic when the traffic enters the Microsoft Azure network backbone. For Microsoft peering, the NAT IP address(es) that are used are either customer provided or are 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. Learn more about [NAT for ExpressRoute public and Microsoft peering](#).



Configuration

- Service endpoints are configured 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, 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, Azure service resources can be secured 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 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 is 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 so, also allow traffic from your VNet to Azure services. This continues to work as is, with service endpoints.
 - If you want to deny all outbound Internet traffic and allow only traffic to specific Azure services, you can do so using “**Azure service tags**” in your NSGs. You can specify supported Azure services as destination in your NSG rules and the maintenance of IP addresses underlying each tag is provided by Azure. 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 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. This scenario might 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](#).
- **Securing Azure resources to services deployed directly into virtual networks:** Various Azure services can be directly deployed 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 (including mount and unmount, diskIO), for managed/unmanaged disks, is not affected by service endpoints routing changes for Azure Storage. You can limit REST access to page blobs to select networks, through service endpoints and [Azure Storage network rules](#).

Logging and troubleshooting

Once service endpoints are configured 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. Learn more about [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/JoinServiceToaSubnet* for the subnets being added. This permission is included in the built-in service administrator roles, by default and can be modified by creating custom roles.

Learn more about [built-in roles](#) and assigning specific permissions to [custom roles](#).

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 is 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 is no limit on the total number of 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. Refer to the documentation for various services in [Next steps](#) for details.

Next steps

- Learn how to [configure virtual network service endpoints](#)
- Learn how to [secure an Azure Storage account to a virtual network](#)
- Learn how to [secure an Azure SQL Database to a virtual network](#)
- Learn about [Azure service integration in virtual networks](#)
- Quick start: [Azure resource manager template](#) to set up service endpoint on a VNet's Subnet and secure Azure Storage account to that subnet.

Virtual network peering

4/25/2018 • 5 min to read • [Edit Online](#)

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

- VNet peering - connecting VNets within the same Azure region
- Global VNet peering - connecting VNets across Azure regions

The benefits of using virtual network peering, whether local or global, include:

- 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.
- A low-latency, high-bandwidth connection between resources in different virtual networks.
- The ability for resources in one virtual network to communicate with resources in a different virtual network, once the virtual networks are peered.
- The ability to transfer data across Azure subscriptions, deployment models, and across Azure regions.
- The ability to peer virtual networks created through the Azure Resource Manager or to peer one virtual network created through Resource Manager to a virtual network 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.

Connectivity

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

The network latency between virtual machines in peered virtual networks 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.

Network security groups can be applied in either virtual network to block access to other virtual networks or subnets, if desired. When configuring virtual network peering, you can either open or close the network security group rules between the virtual networks. If you open full connectivity between peered virtual networks (which is the default option), you can apply network security groups to specific subnets or virtual machines to block or deny specific access. To learn more about network security groups, see [Network security groups overview](#).

Service chaining

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

You can deploy hub-and-spoke networks, where the hub virtual network can host infrastructure components

such as a network virtual appliance or VPN gateway. All the spoke virtual networks can then peer with the hub virtual network. Traffic can flow through network virtual appliances or VPN gateways in the hub virtual network.

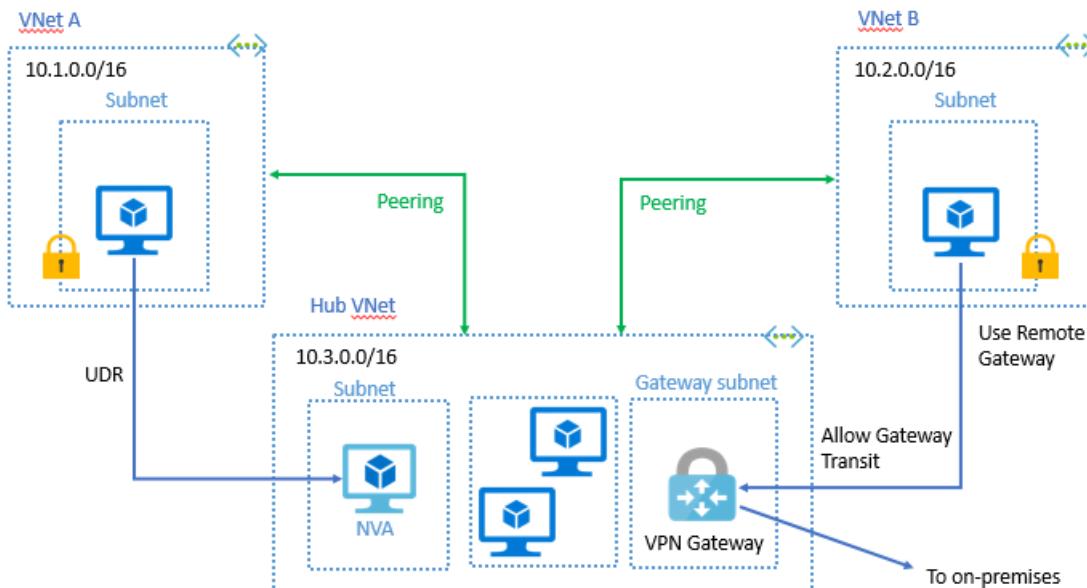
Virtual network peering enables the next hop in a user-defined route to be the IP address of a virtual machine in the peered virtual network, or a VPN gateway. You cannot however, route between virtual networks with a user-defined route specifying an ExpressRoute gateway as the next hop type. To learn more about user-defined routes, see [User-defined routes overview](#). To learn how to create a hub and spoke network topology, see [hub and spoke network topology](#).

Gateways and on-premises connectivity

Each virtual network, regardless of whether it is peered with another virtual network, can still have its own gateway and use it to connect to an on-premises network. You can also configure [virtual network-to-virtual network connections](#) by using gateways, even though the virtual networks are peered.

When both options for virtual network interconnectivity are configured, the traffic between the virtual networks flows through the peering configuration (that is, through the Azure backbone).

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



Gateway transit is not supported in the peering relationship between virtual networks created in different regions. Both virtual networks in the peering relationship must exist in the same region for gateway transit to work. Gateway transit between virtual networks created through different deployment models (Resource Manager and classic), is supported only if the gateway is in the virtual network (Resource Manager). To learn more about using a gateway for transit, see [Configure a VPN gateway for transit in a virtual network peering](#).

When the virtual networks that are sharing a single Azure ExpressRoute connection are peered, the traffic between them goes through the peering relationship (that is, through the Azure backbone network). You can still use local gateways in each virtual network to connect to the on-premises circuit. Alternatively, you can use a shared gateway and configure transit for on-premises connectivity.

Troubleshoot

To confirm a virtual network peering, you can [check effective 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.

You can also troubleshoot connectivity to a virtual machine in a peered virtual network using Network Watcher's [connectivity check](#). 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.

Requirements and constraints

To learn about requirements and constraints, see [Virtual network peering requirements and constraints](#). To learn about the limits for the number of peerings you can create for a virtual network, see [Azure networking limits](#).

Permissions

To learn about permissions required to create a virtual network peering, see [Virtual network peering permissions](#).

Pricing

There is a nominal charge for ingress and egress traffic that utilizes a virtual network peering connection. For more information, see the [pricing page](#).

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](#).
- Learn about all [virtual network peering settings](#) and [how to change them](#).

Virtual network integration for Azure services

5/7/2018 • 2 min to read • [Edit Online](#)

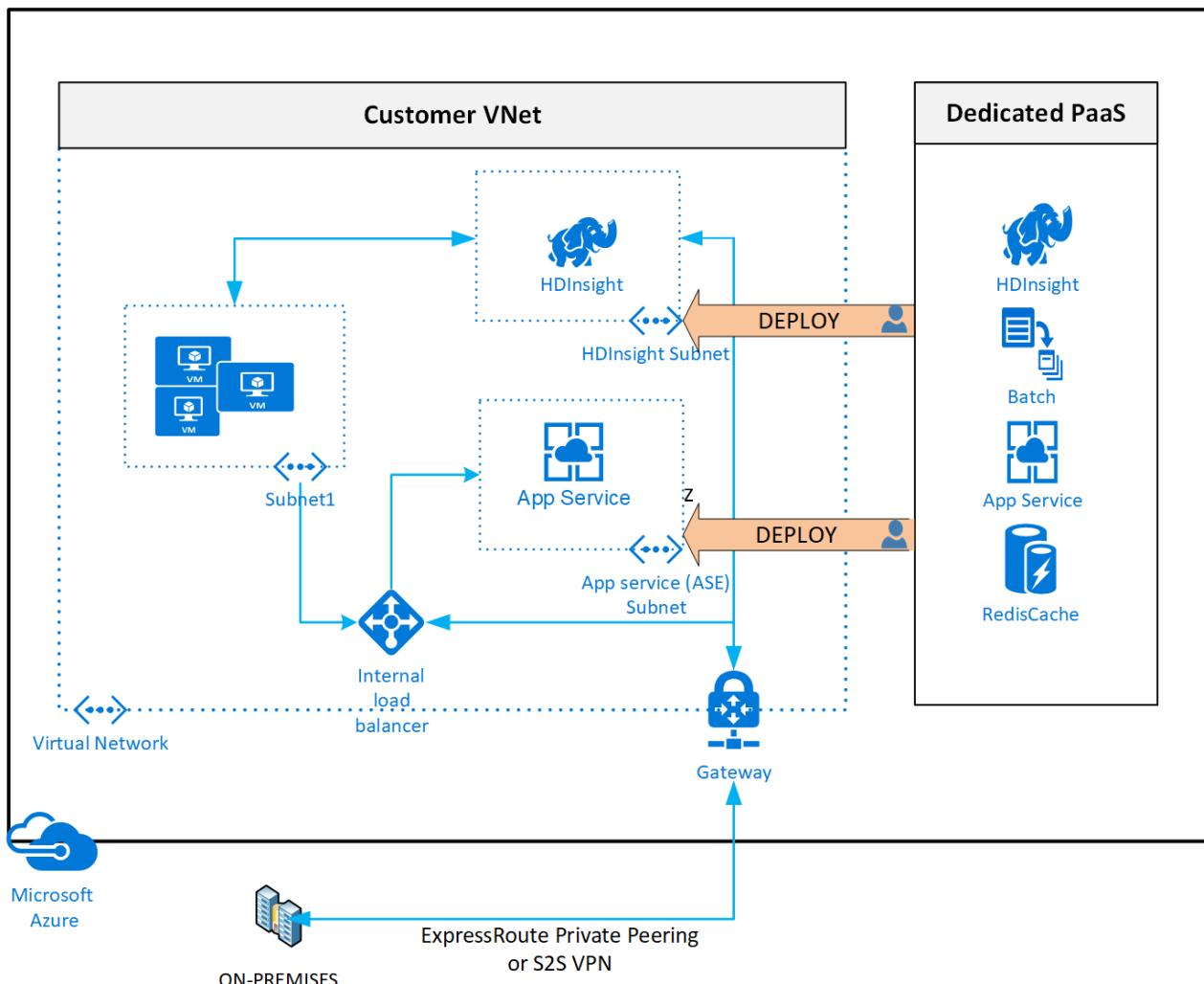
Integrating Azure services to an Azure virtual network allows private access from instances of a service deployed in the virtual network.

You can integrate Azure services with your virtual network with the following options:

- Directly deploying dedicated instances of the service into a virtual network. The dedicated instances of these services can be privately accessed within the virtual network and from on-premises networks.
- By extending a virtual network to the service, through service endpoints. Service endpoints allow individual service resources to be secured to the virtual network.

Deploy Azure services into virtual networks

You can communicate with most Azure resources over the Internet through public IP addresses. When you deploy 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 fully managed by the Azure service, to monitor health of the instances, and provide required scale, based on load.
- Service instances are deployed into a dedicated subnet in a virtual network. Inbound and outbound network access must be opened through [network security groups](#) for the subnet, per guidance provided by the services.

Services that can be deployed into a virtual network

Each service directly deployed into virtual network has specific requirements for routing and the types of traffic that must be allowed into and out of subnets. For more information, see:

- Virtual machines: [Linux](#) or [Windows](#)
- [Service fabric](#)
- [Virtual machine scale sets](#)
- [HDInsight](#)
- [App Service Environment](#)
- [RedisCache](#)
- [API Management](#)
- [VPN Gateway](#)
- [Application Gateway \(internal\)](#)
- [Azure Kubernetes Service \(AKS\)](#)
- [Azure Container Service Engine](#) with the Azure Virtual Network CNI [plug-in](#)
- [Azure Active Directory Domain Services](#): Virtual network (classic) only
- [Azure Batch](#)
- [Cloud services](#): Virtual network (classic) only

You can deploy an [internal Azure load balancer](#) to load balance many of the resources in the previous list. In some cases, the service automatically creates and deploys a load balancer, when you create a resource.

Service endpoints for Azure services

Some Azure services can't be deployed in virtual networks. You can restrict access to some of the service resources to only specific virtual network subnets, if you choose, by enabling a virtual network service endpoint. Learn more about [virtual network service endpoints](#).

Currently, service endpoints are supported for the following services:

- [Azure Storage](#): [Securing Azure Storage accounts to Virtual Networks](#)
- [Azure SQL Database](#): [Securing Azure SQL Database to Virtual networks](#)

Virtual network integration across multiple Azure services

You can deploy an Azure service into a subnet in a virtual network and secure critical service resources to that subnet. For example, you can deploy HDInsight into your virtual network and secure a storage account to the HDInsight subnet.

Network configuration in Azure Kubernetes Service (AKS)

5/7/2018 • 5 min to read • [Edit Online](#)

When you create an Azure Kubernetes Service (AKS) cluster, you can select from two networking options: **Basic** or **Advanced**.

Basic networking

The **Basic** networking option is the default configuration for AKS cluster creation. The network configuration of the cluster and its pods are managed completely by Azure, and is appropriate for deployments that do not require custom VNet configuration. You do not have control over network configuration such as subnets or the IP address ranges assigned to the cluster when you select Basic networking.

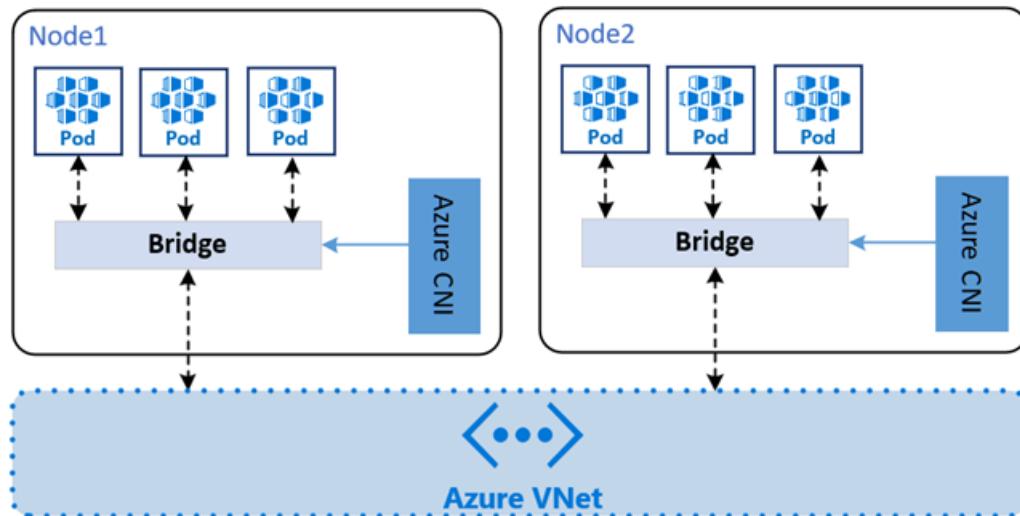
Nodes in an AKS cluster configured for Basic networking use the [kubenet](#) Kubernetes plugin.

Advanced networking

Advanced networking places your pods in an Azure Virtual Network (VNet) that you configure, providing them automatic connectivity to VNet resources and integration with the rich set of capabilities that VNets offer.

Advanced networking is currently available only when deploying AKS clusters in the [Azure portal](#) or with a Resource Manager template.

Nodes in an AKS cluster configured for Advanced networking use the [Azure Container Networking Interface \(CNI\)](#) Kubernetes plugin.



Advanced networking features

Advanced networking provides the following benefits:

- Deploy your AKS cluster into an existing VNet, or create a new VNet and subnet for your cluster.
- Every pod in the cluster is assigned an IP address in the VNet, and can directly communicate with other pods in the cluster, and other VMs in the VNet.
- A pod can connect to other services in a peered VNet, and to on-premises networks over ExpressRoute and site-to-site (S2S) VPN connections. Pods are also reachable from on-premises.

- Expose a Kubernetes service externally or internally through the Azure Load Balancer. Also a feature of Basic networking.
- Pods in a subnet that have service endpoints enabled can securely connect to Azure services, for example Azure Storage and SQL DB.
- Use user-defined routes (UDR) to route traffic from pods to a Network Virtual Appliance.
- Pods can access resources on the public Internet. Also a feature of Basic networking.

IMPORTANT

Each node in an AKS cluster configured for Advanced networking can host a maximum of **30 pods**. Each VNet provisioned for use with the Azure CNI plugin is limited to **4096 IP addresses** (/20).

Configure advanced networking

When you [create an AKS cluster](#) in the Azure portal, the following parameters are configurable for advanced networking:

Virtual network: The VNet into which you want to deploy the Kubernetes cluster. If you want to create a new VNet for your cluster, select *Create new* and follow the steps in the *Create virtual network* section.

Subnet: The subnet within the VNet where you want to deploy the cluster. If you want to create a new subnet in the VNet for your cluster, select *Create new* and follow the steps in the *Create subnet* section.

Kubernetes service address range: The IP address range for the Kubernetes cluster's service IPs. This range must not be within the VNet IP address range of your cluster.

Kubernetes DNS service IP address: The IP address for the cluster's DNS service. This address must be within the *Kubernetes service address range*.

Docker Bridge address: The IP address and netmask to assign to the Docker bridge. This IP address must not be within the VNet IP address range of your cluster.

The following screenshot from the Azure portal shows an example of configuring these settings during AKS cluster creation:

Create Kubernetes cluster

[Basics](#) [Networking](#) [Monitoring](#) [Tags](#) [Review + create](#)

You can enable Http ingress routing and choose between two networking options for Azure Kubernetes Services - "Basic" and "Advanced".

- "Basic" networking sets up a simple default config with a VNet and internal IP addresses.
- "Advanced" networking provides you the ability to configure your own VNet, providing pods automatic connectivity to VNet resources and full integration with VNet features.

[Learn more about networking in Azure Kubernetes Service](#)

Http application routing i	<input checked="" type="radio"/> No <input type="radio"/> Yes
Network configuration i	<input type="radio"/> Basic <input checked="" type="radio"/> Advanced
* Virtual network i	aks-vnet-16033614
Create new	
* Subnet i	aks-subnet
Create new	
* Kubernetes service address range i	10.0.0.0/16 ✓
* Kubernetes DNS service IP address i	10.0.0.10
* Docker Bridge address i	172.17.0.1/16 ✓

Plan IP addressing for your cluster

Clusters configured with Advanced networking require additional planning. The size of your VNet and its subnet must accommodate the number of pods you plan to run simultaneously in the cluster, as well as your scaling requirements.

IP addresses for the pods and the cluster's nodes are assigned from the specified subnet within the VNet. Each node is configured with a primary IP, which is the IP of the node itself, and 30 additional IP addresses pre-configured by Azure CNI that are assigned to pods scheduled to the node. When you scale out your cluster, each node is similarly configured with IP addresses from the subnet.

As mentioned previously, each VNet provisioned for use with the Azure CNI plugin is limited to **4096 IP addresses** (/20). Each node in a cluster configured for Advanced networking can host a maximum of **30 pods**.

Frequently asked questions

The following questions and answers apply to the **Advanced** networking configuration.

- *Can I configure Advanced networking with the Azure CLI?*

No. Advanced networking is currently available only when deploying AKS clusters in the Azure portal or with a Resource Manager template.

- *Can I deploy VMs in my cluster subnet?*

No. Deploying VMs in the subnet used by your Kubernetes cluster is not supported. VMs may be deployed in the same VNet, but in a different subnet.

- **Can I configure per-pod network policies?*

No. Per-pod network policies are currently unsupported.

- *Is the maximum number of pods deployable to a node configurable?*

By default, each node can host a maximum of 30 pods. You can currently change the maximum value only by modifying the `maxPods` property when deploying a cluster with a Resource Manager template.

- *How do I configure additional properties for the subnet that I created during AKS cluster creation? For example, service endpoints.*

The complete list of properties for the VNet and subnets that you create during AKS cluster creation can be configured in the standard VNet configuration page in the Azure portal.

Next steps

Networking in AKS

Learn more about networking in AKS in the following articles:

[Use a static IP address with the Azure Kubernetes Service \(AKS\) load balancer](#)

[HTTPS ingress on Azure Container Service \(AKS\)](#)

[Use an internal load balancer with Azure Container Service \(AKS\)](#)

ACS Engine

[Azure Container Service Engine \(ACS Engine\)](#) is an open-source project that generates Azure Resource Manager templates you can use for deploying Docker-enabled clusters on Azure. Kubernetes, DC/OS, Swarm Mode, and Swarm orchestrators can be deployed with ACS Engine.

Kubernetes clusters created with ACS Engine support both the [kubenet](#) and [Azure CNI](#) plugins. As such, both basic and advanced networking scenarios are supported by ACS Engine.

Virtual Network – Business Continuity

3/8/2018 • 2 min 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.

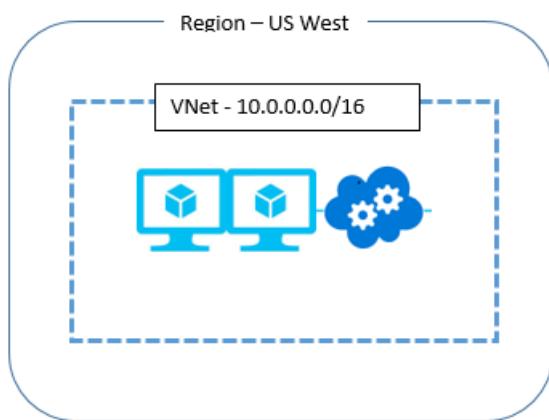
A VNet is created within the scope of a region. You can create VNets with same address space in two different regions (For example, US East and US West), but cannot connect them together.

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 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](#).

IP address types and allocation methods in Azure

5/2/2018 • 11 min 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.

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

IP address version

Public IP addresses are created with an IPv4 or IPv6 address. Public IPv6 addresses can only be assigned to Internet-facing load balancers.

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.
- 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.
- Can be assigned to a specific zone.
- Not zone redundant. To learn more about availability zones, see [Availability zones overview](#).

Standard

Standard SKU public IP addresses are:

- Assigned with the static allocation method only.
- 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 or public standard load balancers. For more information about Azure standard load balancers, see [Azure standard load balancer](#).
- Zone redundant by default. 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

Communication with a standard SKU resource fails until you create and associate a [network security group](#) and explicitly allow the desired inbound traffic.

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 Azure region. For details, see [Azure Datacenter IP ranges](#).

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. You can use the FQDN to create a custom domain CNAME record pointing to the public IP address in Azure. 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 [Use Azure DNS with an Azure public IP address](#).

IMPORTANT

Each domain name label created must be unique within its Azure location.

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 frontend 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

TOP-LEVEL RESOURCE	IP ADDRESS ASSOCIATION	DYNAMIC	STATIC
Internet-facing Load balancer	Front-end configuration	Yes	Yes
VPN gateway	Gateway IP configuration	Yes	No
Application gateway	Front-end configuration	Yes	No

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

IP address version

Private IP addresses are created with an IPv4 or IPv6 address. Private IPv6 addresses can only be assigned with the dynamic allocation method. You cannot communicate between private IPv6 addresses on a virtual network. You can communicate inbound to a private IPv6 address from the Internet, through an Internet-facing load balancer. See [Create an Internet-facing load balancer with IPv6](#) for details.

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 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](#)

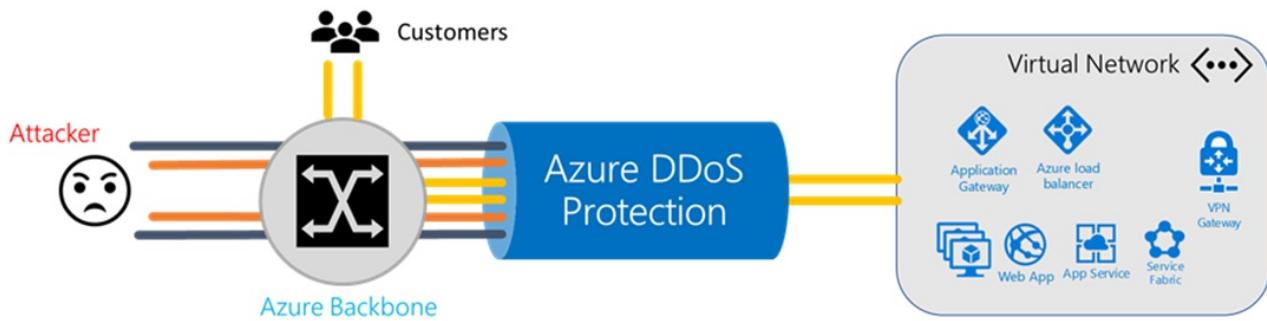
Azure DDoS Protection Standard overview

4/18/2018 • 4 min 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, at no additional charge. 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. Real-time telemetry is available through Azure Monitor views during an attack, and for history. Application layer protection can be added through the [Azure Application Gateway Web Application Firewall](#). Protection is provided for IPv4 Azure [public IP addresses](#).



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 the Azure [Application Gateway web application firewall](#), with 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, DDoS Protection Standard can provide full layer 3 to layer 7 mitigation capability.

DDoS Protection Standard features

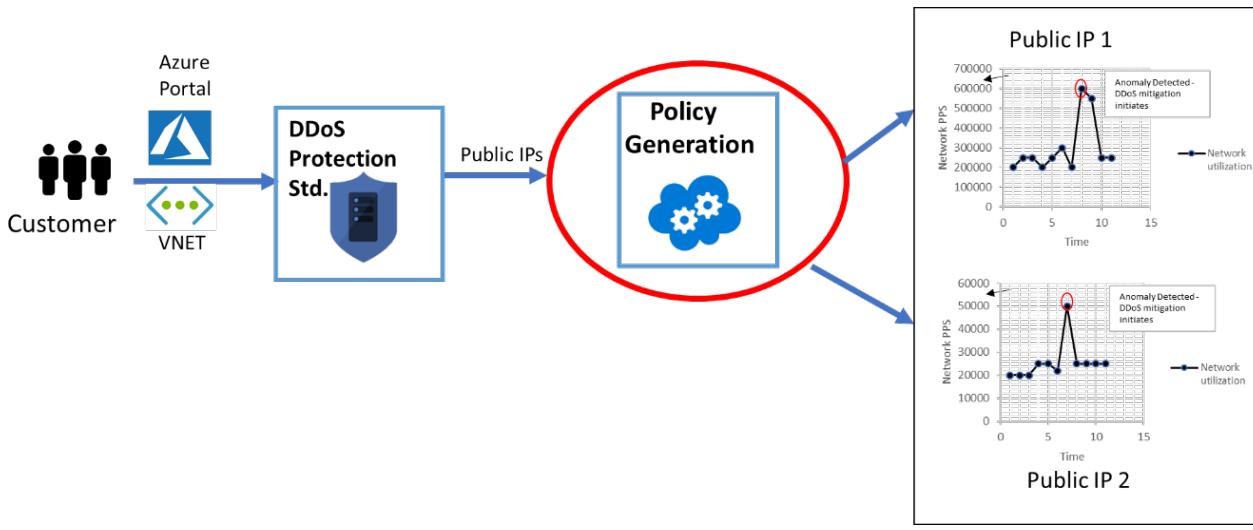
			
Reliable proven success	Scalable global capacity	Automatic simple, automated	Adaptive real time tuning

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.
- **Layer 3 to layer 7 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 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 Log Analytics, 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](#)

IP address types and allocation methods (classic) in Azure

4/19/2018 • 9 min 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 and private.

Public IP addresses are used for communication with the Internet, including Azure public-facing services.

Private IP addresses are used for communication within an Azure virtual network (VNet), a cloud service, and your on-premises network when you use a VPN gateway or ExpressRoute circuit to extend your network to Azure.

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the classic deployment model. Microsoft recommends that most new deployments use Resource Manager. Learn about IP addresses in Resource Manager by reading the [IP addresses](#) article.

Public IP addresses

Public IP addresses allow Azure resources to communicate with Internet and Azure public-facing services such as [Azure Redis Cache](#), [Azure Event Hubs](#), [SQL databases](#), and [Azure storage](#).

A public IP address is associated with the following resource types:

- Cloud services
- IaaS Virtual Machines (VMs)
- PaaS role instances
- VPN gateways
- Application gateways

Allocation method

When a public IP address needs to be assigned to an Azure resource, it is *dynamically* allocated from a pool of available public IP address within the location the resource is created. This IP address is released when the resource is stopped. In case of a cloud service, this happens when all the role instances are stopped, which can be avoided by using a *static* (reserved) IP address (see [Cloud Services](#)).

NOTE

The list of IP ranges from which public IP addresses are allocated to Azure resources is published at [Azure Datacenter IP ranges](#).

DNS hostname resolution

When you create a cloud service or an IaaS VM, you need to provide a cloud service DNS name which is unique across all resources in Azure. This creates a mapping in the Azure-managed DNS servers for `dnsname.cloudapp.net` to the public IP address of the resource. For instance, when you create a cloud service with a cloud service DNS name of **contoso**, the fully-qualified domain name (FQDN) **contoso.cloudapp.net** will resolve to a public IP address (VIP) of the cloud service. You can use this FQDN to create a custom domain CNAME record pointing to the public IP address in Azure.

Cloud services

A cloud service always has a public IP address referred to as a virtual IP address (VIP). You can create endpoints in a cloud service to associate different ports in the VIP to internal ports on VMs and role instances within the cloud service.

A cloud service can contain multiple IaaS VMs, or PaaS role instances, all exposed through the same cloud service VIP. You can also assign [multiple VIPs to a cloud service](#), which enables multi-VIP scenarios like multi-tenant environment with SSL-based websites.

You can ensure the public IP address of a cloud service remains the same, even when all the role instances are stopped, by using a *static* public IP address, referred to as [Reserved IP](#). You can create a static (reserved) IP resource in a specific location and assign it to any cloud service in that location. You cannot specify the actual IP address for the reserved IP; it is allocated from pool of available IP addresses in the location it is created. This IP address is not released until you explicitly delete it.

Static (reserved) public IP addresses are commonly used in the scenarios where a cloud service:

- requires firewall rules to be setup by end-users.
- depends on external DNS name resolution, and a dynamic IP would require updating A records.
- consumes external web services which use IP based security model.
- uses SSL certificates linked to an IP address.

NOTE

When you create a classic VM, a container *cloud service* is created by Azure, which has a virtual IP address (VIP). When the creation is done through portal, a default RDP or SSH *endpoint* is configured by the portal so you can connect to the VM through the cloud service VIP. This cloud service VIP can be reserved, which effectively provides a reserved IP address to connect to the VM. You can open additional ports by configuring more endpoints.

IaaS VMs and PaaS role instances

You can assign a public IP address directly to an IaaS [VM](#) or PaaS role instance within a cloud service. This is referred to as an instance-level public IP address ([ILPIP](#)). This public IP address can be dynamic only.

NOTE

This is different from the VIP of the cloud service, which is a container for IaaS VMs or PaaS role instances, since a cloud service can contain multiple IaaS VMs, or PaaS role instances, all exposed through the same cloud service VIP.

VPN gateways

A [VPN gateway](#) can be used to connect an Azure VNet to other Azure VNets or on-premises networks. A VPN gateway is assigned a public IP address *dynamically*, which enables communication with the remote network.

Application gateways

An Azure [Application gateway](#) can be used for Layer7 load-balancing to route network traffic based on HTTP. Application gateway is assigned a public IP address *dynamically*, which serves as the load-balanced VIP.

At a glance

The table below shows each resource type with the possible allocation methods (dynamic/static), and ability to assign multiple public IP addresses.

RESOURCE	DYNAMIC	STATIC	MULTIPLE IP ADDRESSES
Cloud service	Yes	Yes	Yes

RESOURCE	DYNAMIC	STATIC	MULTIPLE IP ADDRESSES
IaaS VM or PaaS role instance	Yes	No	No
VPN gateway	Yes	No	No
Application gateway	Yes	No	No

Private IP addresses

Private IP addresses allow Azure resources to communicate with other resources in a cloud service or a [virtual network](#)(VNet), or to on-premises network (through a VPN gateway or ExpressRoute circuit), without using an Internet-reachable IP address.

In Azure classic deployment model, a private IP address can be assigned to the following Azure resources:

- IaaS VMs and PaaS role instances
- Internal load balancer
- Application gateway

IaaS VMs and PaaS role instances

Virtual machines (VMs) created with the classic deployment model are always placed in a cloud service, similar to PaaS role instances. The behavior of private IP addresses are thus similar for these resources.

It is important to note that a cloud service can be deployed in two ways:

- As a *standalone* cloud service, where it is not within a virtual network.
- As part of a virtual network.

Allocation method

In case of a *standalone* cloud service, resources get a private IP address allocated *dynamically* from the Azure datacenter private IP address range. It can be used only for communication with other VMs within the same cloud service. This IP address can change when the resource is stopped and started.

In case of a cloud service deployed within a virtual network, resources get private IP address(es) allocated from the address range of the associated subnet(s) (as specified in its network configuration). This private IP address(es) can be used for communication between all VMs within the VNet.

Additionally, in case of cloud services within a VNet, a private IP address is allocated *dynamically* (using DHCP) by default. It can change when the resource is stopped and started. To ensure the IP address remains the same, you need to set the allocation method to *static*, and provide a valid IP address within the corresponding address range.

Static private IP addresses are commonly used for:

- VMs that act as domain controllers or DNS servers.
- VMs that require firewall rules using IP addresses.
- VMs running services accessed by other apps through an IP address.

Internal DNS hostname resolution

All Azure VMs and PaaS role instances are configured with [Azure-managed DNS servers](#) by default, unless you explicitly configure custom DNS servers. These DNS servers provide internal name resolution for VMs and role instances that reside within the same VNet or cloud service.

When you create a VM, a mapping for the hostname to its private IP address is added to the Azure-managed DNS servers. In case of a multi-NIC VM, the hostname is mapped to the private IP address of the primary NIC.

However, this mapping information is restricted to resources within the same cloud service or VNet.

In case of a *standalone* cloud service, you will be able to resolve hostnames of all VMs/role instances within the same cloud service only. In case of a cloud service within a VNet, you will be able to resolve hostnames of all the VMs/role instances within the VNet.

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 (VNet) and the remote networks connected to the VNet. You can assign either a dynamic or static private IP address to the front end configuration. You can also assign multiple private IP addresses to enable multi-vip scenarios.

At a glance

The table below shows each resource type with the possible allocation methods (dynamic/static), and ability to assign multiple private IP addresses.

RESOURCE	DYNAMIC	STATIC	MULTIPLE IP ADDRESSES
VM (in a <i>standalone</i> cloud service or VNet)	Yes	Yes	Yes
PaaS role instance (in a <i>standalone</i> cloud service or VNet)	Yes	No	No
Internal load balancer front end	Yes	Yes	Yes
Application gateway front end	Yes	Yes	Yes

Limits

The table below shows the limits imposed on IP addressing in Azure per subscription. You can [contact support](#) to increase the default limits up to the maximum limits based on your business needs.

	DEFAULT LIMIT	MAXIMUM LIMIT
Public IP addresses (dynamic)	5	contact support
Reserved public IP addresses	20	contact support
Public VIP per deployment (cloud service)	5	contact support
Private VIP (ILB) per deployment (cloud service)	1	1

Make sure you read the full set of [limits for Networking](#) in Azure.

Pricing

In most cases, public IP addresses are free. There is a nominal charge to use additional and/or static public IP addresses. Make sure you understand the [pricing structure for public IPs](#).

Differences between Resource Manager and classic deployments

Below is a comparison of IP addressing features in Resource Manager and the classic deployment model.

	RESOURCE	CLASSIC	RESOURCE MANAGER
Public IP Address	VM	Referred to as an ILPIP (dynamic only)	Referred to as a public IP (dynamic or static)
		Assigned to an IaaS VM or a PaaS role instance	Associated to the VM's NIC
	Internet facing load balancer	Referred to as VIP (dynamic) or Reserved IP (static)	Referred to as a public IP (dynamic or static)
		Assigned to a cloud service	Associated to the load balancer's front end config
Private IP Address	VM	Referred to as a DIP	Referred to as a private IP address
		Assigned to an IaaS VM or a PaaS role instance	Assigned to the VM's NIC
	Internal load balancer (ILB)	Assigned to the ILB (dynamic or static)	Assigned to the ILB's front end configuration (dynamic or static)

Next steps

- [Deploy a VM with a static private IP address using the Azure portal.](#)

What is an endpoint access control list?

4/19/2018 • 5 min to read • [Edit Online](#)

IMPORTANT

Azure has two different [deployment models](#) for creating and working with resources: Resource Manager and classic. This article covers using the classic deployment model. Microsoft recommends that most new deployments use the Resource Manager deployment model.

An endpoint access control list (ACL) is a security enhancement available for your Azure deployment. An ACL provides the ability to selectively permit or deny traffic for a virtual machine endpoint. This packet filtering capability provides an additional layer of security. You can specify network ACLs for endpoints only. You can't specify an ACL for a virtual network or a specific subnet contained in a virtual network. It is recommended to use network security groups (NSGs) instead of ACLs, whenever possible. To learn more about NSGs, see [Network security group overview](#)

ACLs can be configured by using either PowerShell or the Azure portal. To configure a network ACL by using PowerShell, see [Managing access control lists for endpoints using PowerShell](#). To configure a network ACL by using the Azure portal, see [How to set up endpoints to a virtual machine](#).

Using Network ACLs, you can do the following:

- Selectively permit or deny incoming traffic based on remote subnet IPv4 address range to a virtual machine input endpoint.
- Blacklist IP addresses
- Create multiple rules per virtual machine endpoint
- Use rule ordering to ensure the correct set of rules are applied on a given virtual machine endpoint (lowest to highest)
- Specify an ACL for a specific remote subnet IPv4 address.

See the [Azure limits](#) article for ACL limits.

How ACLs work

An ACL is an object that contains a list of rules. When you create an ACL and apply it to a virtual machine endpoint, packet filtering takes place on the host node of your VM. This means the traffic from remote IP addresses is filtered by the host node for matching ACL rules instead of on your VM. This prevents your VM from spending the precious CPU cycles on packet filtering.

When a virtual machine is created, a default ACL is put in place to block all incoming traffic. However, if an endpoint is created for (port 3389), then the default ACL is modified to allow all inbound traffic for that endpoint. Inbound traffic from any remote subnet is then allowed to that endpoint and no firewall provisioning is required. All other ports are blocked for inbound traffic unless endpoints are created for those ports. Outbound traffic is allowed by default.

Example Default ACL table

RULE #	REMOTE SUBNET	ENDPOINT	PERMIT/DENY
100	0.0.0.0/0	3389	Permit

Permit and deny

You can selectively permit or deny network traffic for a virtual machine input endpoint by creating rules that specify "permit" or "deny". It's important to note that by default, when an endpoint is created, all traffic is permitted to the endpoint. For that reason, it's important to understand how to create permit/deny rules and place them in the proper order of precedence if you want granular control over the network traffic that you choose to allow to reach the virtual machine endpoint.

Points to consider:

1. **No ACL** – By default when an endpoint is created, we permit all for the endpoint.
2. **Permit** - When you add one or more "permit" ranges, you are denying all other ranges by default. Only packets from the permitted IP range will be able to communicate with the virtual machine endpoint.
3. **Deny** - When you add one or more "deny" ranges, you are permitting all other ranges of traffic by default.
4. **Combination of Permit and Deny** - You can use a combination of "permit" and "deny" when you want to carve out a specific IP range to be permitted or denied.

Rules and rule precedence

Network ACLs can be set up on specific virtual machine endpoints. For example, you can specify a network ACL for an RDP endpoint created on a virtual machine which locks down access for certain IP addresses. The table below shows a way to grant access to public virtual IPs (VIPs) of a certain range to permit access for RDP. All other remote IPs are denied. We follow a *lowest takes precedence* rule order.

Multiple rules

In the example below, if you want to allow access to the RDP endpoint only from two public IPv4 address ranges (65.0.0.0/8, and 159.0.0.0/8), you can achieve this by specifying two *Permit* rules. In this case, since RDP is created by default for a virtual machine, you may want to lock down access to the RDP port based on a remote subnet. The example below shows a way to grant access to public virtual IPs (VIPs) of a certain range to permit access for RDP. All other remote IPs are denied. This works because network ACLs can be set up for a specific virtual machine endpoint and access is denied by default.

Example – Multiple rules

RULE #	REMOTE SUBNET	ENDPOINT	PERMIT/DENY
100	65.0.0.0/8	3389	Permit
200	159.0.0.0/8	3389	Permit

Rule order

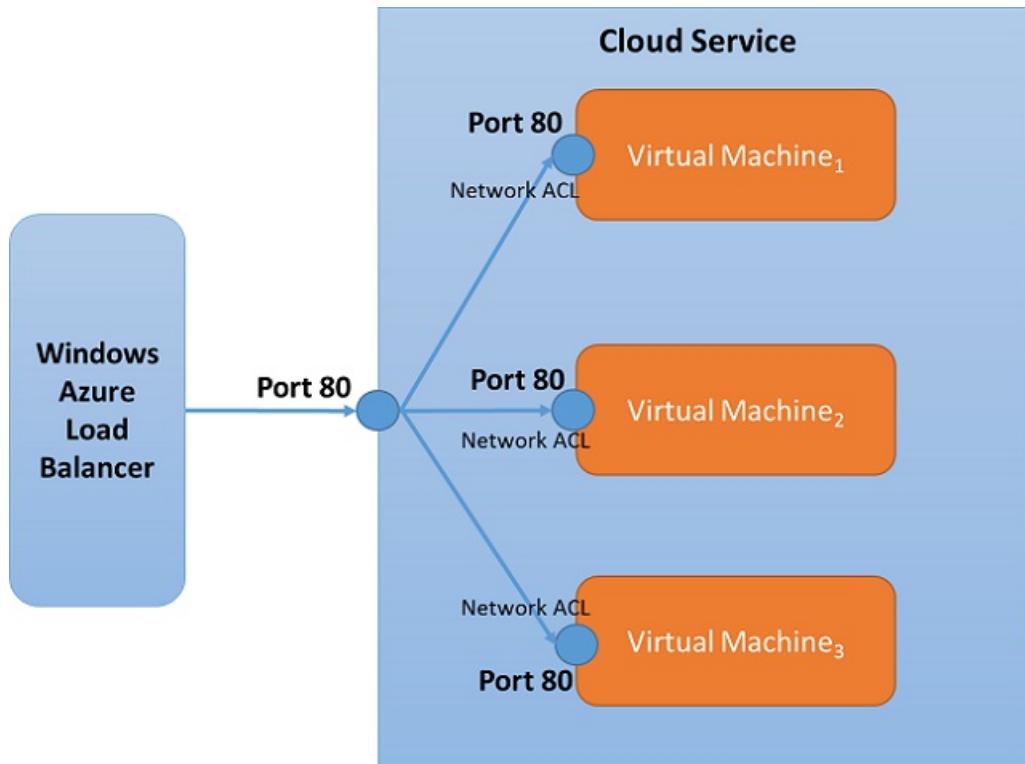
Because multiple rules can be specified for an endpoint, there must be a way to organize rules in order to determine which rule takes precedence. The rule order specifies precedence. Network ACLs follow a *lowest takes precedence* rule order. In the example below, the endpoint on port 80 is selectively granted access to only certain IP address ranges. To configure this, we have a deny rule (Rule # 100) for addresses in the 175.1.0.1/24 space. A second rule is then specified with precedence 200 that permits access to all other addresses under 175.0.0.0/8.

Example – Rule precedence

RULE #	REMOTE SUBNET	ENDPOINT	PERMIT/DENY
100	175.1.0.1/24	80	Deny
200	175.0.0.0/8	80	Permit

Network ACLs and load balanced sets

Network ACLs can be specified on a load balanced set endpoint. If an ACL is specified for a load balanced set, the network ACL is applied to all virtual machines in that load balanced set. For example, if a load balanced set is created with "Port 80" and the load balanced set contains 3 VMs, the network ACL created on endpoint "Port 80" of one VM will automatically apply to the other VMs.



Next Steps

[Manage access control lists for endpoints using PowerShell](#)

Plan and design Azure Virtual Networks

4/11/2018 • 14 min to read • [Edit Online](#)

Creating a VNet to experiment with is easy enough, but chances are, you will deploy multiple VNets over time to support the production needs of your organization. With some planning and design, you will be able to deploy VNets and connect the resources you need more effectively. If you are not familiar with VNets, it's recommended that you [learn about VNets](#) and [how to deploy](#) one before proceeding.

Plan

A thorough understanding of Azure subscriptions, regions, and network resources is critical for success. You can use the list of considerations below as a starting point. Once you understand those considerations, you can define the requirements for your network design.

Considerations

Before answering the planning questions below, consider the following:

- Everything you create in Azure is composed of one or more resources. A virtual machine (VM) is a resource, the network adapter interface (NIC) used by a VM is a resource, the public IP address used by a NIC is a resource, the VNet the NIC is connected to is a resource.
- You create resources within an [Azure region](#) and subscription. Resources can only be connected to a virtual network that exists in the same region and subscription the resource is in.
- You can connect virtual networks to each other by using:
 - **Virtual network peering:** The virtual networks must exist in the same Azure region. Bandwidth between resources in peered virtual networks is the same as if the resources were connected to the same virtual network.
 - **An Azure VPN Gateway:** The virtual networks can exist in the same, or different Azure regions. Bandwidth between resources in virtual networks connected through a VPN Gateway is limited by the bandwidth of the VPN Gateway.
- You can connect VNets to your on-premises network by using one of the [connectivity options](#) available in Azure.
- Different resources can be grouped together in [resource groups](#), making it easier to manage the resource as a unit. A resource group can contain resources from multiple regions, as long as the resources belong to the same subscription.

Define requirements

Use the questions below as a starting point for your Azure network design.

1. What Azure locations will you use to host VNets?
2. Do you need to provide communication between these Azure locations?
3. Do you need to provide communication between your Azure VNet(s) and your on-premises datacenter(s)?
4. How many Infrastructure as a Service (IaaS) VMs, cloud services roles, and web apps do you need for your solution?
5. Do you need to isolate traffic based on groups of VMs (i.e. front end web servers and back end database servers)?
6. Do you need to control traffic flow using virtual appliances?
7. Do users need different sets of permissions to different Azure resources?

Understand VNet and subnet properties

VNet and subnets resources help define a security boundary for workloads running in Azure. A VNet is characterized by a collection of address spaces, defined as CIDR blocks.

NOTE

Network administrators are familiar with CIDR notation. If you are not familiar with CIDR, [learn more about it](#).

VNets contain the following properties.

PROPERTY	DESCRIPTION	CONSTRAINTS
name	VNet name	String of up to 80 characters. May contain letters, numbers, underscore, periods, or hyphens. Must start with a letter or number. Must end with a letter, number, or underscore. Can contains upper or lower case letters.
location	Azure location (also referred to as region).	Must be one of the valid Azure locations.
addressSpace	Collection of address prefixes that make up the VNet in CIDR notation.	Must be an array of valid CIDR address blocks, including public IP address ranges.
subnets	Collection of subnets that make up the VNet	see the subnet properties table below.
dhcpOptions	Object that contains a single required property named dnsServers .	
dnsServers	Array of DNS servers used by the VNet. If no server is specified, Azure internal name resolution is used.	Must be an array of up to 10 DNS servers, by IP address.

A subnet is a child resource of a VNet, and helps define segments of address spaces within a CIDR block, using IP address prefixes. NICs can be added to subnets, and connected to VMs, providing connectivity for various workloads.

Subnets contain the following properties.

PROPERTY	DESCRIPTION	CONSTRAINTS
name	Subnet name	String of up to 80 characters. May contain letters, numbers, underscore, periods, or hyphens. Must start with a letter or number. Must end with a letter, number, or underscore. Can contains upper or lower case letters.
location	Azure location (also referred to as region).	Must be one of the valid Azure locations.
addressPrefix	Single address prefix that make up the subnet in CIDR notation	Must be a single CIDR block that is part of one of the VNet's address spaces.

PROPERTY	DESCRIPTION	CONSTRAINTS
networkSecurityGroup	NSG applied to the subnet	
routeTable	Route table applied to the subnet	
ipConfigurations	Collection of IP configuration objects used by NICs connected to the subnet	

Name resolution

By default, your VNet uses [Azure-provided name resolution](#) to resolve names inside the VNet, and on the public Internet. However, if you connect your VNets to your on-premises data centers, you need to provide [your own DNS server](#) to resolve names between your networks.

Limits

Review the networking limits in the [Azure limits](#) article to ensure that your design doesn't conflict with any of the limits. Some limits can be increased by opening a support ticket.

Role-Based Access Control (RBAC)

You can use [Azure RBAC](#) to control the level of access different users may have to different resources in Azure. That way you can segregate the work done by your team based on their needs.

As far as virtual networks are concerned, users in the **Network Contributor** role have full control over Azure Resource Manager virtual network resources. Similarly, users in the **Classic Network Contributor** role have full control over classic virtual network resources.

NOTE

You can also [create your own roles](#) to separate your administrative needs.

Design

Once you know the answers to the questions in the [Plan](#) section, review the following before defining your VNets.

Number of subscriptions and VNets

You should consider creating multiple VNets in the following scenarios:

- **VMs that need to be placed in different Azure locations.** VNets in Azure are regional. They cannot span locations. Therefore you need at least one VNet for each Azure location you want to host VMs in.
- **Workloads that need to be completely isolated from one another.** You can create separate VNets, that even use the same IP address spaces, to isolate different workloads from one another.

Keep in mind that the limits you see above are per region, per subscription. That means you can use multiple subscriptions to increase the limit of resources you can maintain in Azure. You can use a site-to-site VPN, or an ExpressRoute circuit, to connect VNets in different subscriptions.

Subscription and VNet design patterns

The table below shows some common design patterns for using subscriptions and VNets.

SCENARIO	DIAGRAM	PROS	CONS
----------	---------	------	------

SCENARIO	DIAGRAM	PROS	CONS
Single subscription, two VNets per app	<pre>graph TD; Enterprise[Enterprise] --- Subscription[Subscription]; Subscription --- VNetApp1Prod[VNet app1 (production)]; Subscription --- VNetApp1Dev[VNet app1 (development)]; Subscription --- VNetApp2Prod[VNet app2 (production)]; Subscription --- VNetApp2Dev[VNet app2 (development)];</pre>	Only one subscription to manage.	Maximum number of VNets per Azure region. You need more subscriptions after that. Review the Azure limits article for details.
One subscription per app, two VNets per app	<pre>graph TD; Enterprise[Enterprise] --- Subscription1[Subscription 1 (app 1)]; Subscription1 --- VNet1Prod[VNet (production)]; Subscription1 --- VNet1Dev[VNet (development)]; Enterprise --- Subscription2[Subscription 2 (app 2)]; Subscription2 --- VNet2Prod[VNet (production)]; Subscription2 --- VNet2Dev[VNet (development)];</pre>	Uses only two VNets per subscription.	Harder to manage when there are too many apps.
One subscription per business unit, two VNets per app	<pre>graph TD; Enterprise[Enterprise] --- Subscription1[Subscription 1 (BU 1)]; Subscription1 --- VNetApp1Prod1[VNet app1 (production)]; Subscription1 --- VNetApp1Dev1[VNet app1 (development)]; Subscription1 --- VNetApp2Prod1[VNet app2 (development)]; Subscription1 --- VNetApp2Dev1[VNet app2 (production)]; Enterprise --- Subscription2[Subscription 2 (BU 2)]; Subscription2 --- VNetApp1Prod2[VNet app1 (production)]; Subscription2 --- VNetApp1Dev2[VNet app1 (development)]; Subscription2 --- VNetApp2Prod2[VNet app2 (development)]; Subscription2 --- VNetApp2Dev2[VNet app2 (production)];</pre>	Balance between number of subscriptions and VNets.	Maximum number of VNets per business unit (subscription). Review the Azure limits article for details.
One subscription per business unit, two VNets per group of apps	<pre>graph TD; Enterprise[Enterprise] --- Subscription1[Subscription 1 (BU 1)]; Subscription1 --- VNet1Prod1[VNet (production)]; Subscription1 --- VNet1Dev1[VNet (development)]; Enterprise --- Subscription2[Subscription 2 (BU 2)]; Subscription2 --- VNet2Prod1[VNet (production)]; Subscription2 --- VNet2Dev1[VNet (development)];</pre>	Balance between number of subscriptions and VNets.	Apps must be isolated by using subnets and NSGs.

Number of subnets

You should consider multiple subnets in a VNet in the following scenarios:

- **Not enough private IP addresses for all NICs in a subnet.** If your subnet address space does not contain enough IP addresses for the number of NICs in the subnet, you need to create multiple subnets. Keep in mind that Azure reserves 5 private IP addresses from each subnet that cannot be used: the first and last addresses of the address space (for the subnet address, and multicast) and 3 addresses to be used internally (for DHCP and DNS purposes).
- **Security.** You can use subnets to separate groups of VMs from one another for workloads that have a multi-layer structure, and apply different [network security groups \(NSGs\)](#) for those subnets.
- **Hybrid connectivity.** You can use VPN gateways and ExpressRoute circuits to [connect](#) your VNets to one another, and to your on-premises data center(s). VPN gateways and ExpressRoute circuits require a subnet of their own to be created.

- **Virtual appliances.** You can use a virtual appliance, such as a firewall, WAN accelerator, or VPN gateway in an Azure VNet. When you do so, you need to [route traffic](#) to those appliances and isolate them in their own subnet.

Subnet and NSG design patterns

The table below shows some common design patterns for using subnets.

SCENARIO	DIAGRAM	PROS	CONS
Single subnet, NSGs per application layer, per app	<pre> graph TD Subnet1[Subnet 1] --- NSG1[NSG 1 Front end (app1)] Subnet1 --- NSG2[NSG 2 Back end (app1)] Subnet1 --- NSG3[NSG 3 Front end (app2)] Subnet1 --- NSG4[NSG 4 Back end (app2)] </pre>	Only one subnet to manage.	Multiple NSGs necessary to isolate each application.
One subnet per app, NSGs per application layer	<pre> graph TD Subnet1[Subnet 1 (app1)] --- NSG1Front1[NSG 1 Front end] Subnet1 --- NSG2Back1[NSG 2 Back end] Subnet2[Subnet 2 (app2)] --- NSG1Front2[NSG 1 Front end] Subnet2 --- NSG2Back2[NSG 2 Back end] </pre>	Fewer NSGs to manage.	Multiple subnets to manage.
One subnet per application layer, NSGs per app.	<pre> graph TD Subnet1[Subnet 1 (Front end)] --- NSG1App1[NSG 1 (app1)] Subnet1 --- NSG2App2[NSG 2 (app2)] Subnet2[Subnet 2 (Back end)] --- NSG1App1[NSG 1 (app1)] Subnet2 --- NSG2App2[NSG 2 (app2)] </pre>	Balance between number of subnets and NSGs.	Maximum number of NSGs per subscription. Review the Azure limits article for details.
One subnet per application layer, per app, NSGs per subnet	<pre> graph TD Subnet1[Subnet 1 (app1 front end)] --- NSG1[NSG 1] Subnet2[Subnet 2 (app1 back end)] --- NSG2[NSG 2] Subnet3[Subnet 3 (app2 front end)] --- NSG3[NSG 3] Subnet4[Subnet 2 (app2 back end)] --- NSG4[NSG 4] </pre>	Possibly smaller number of NSGs.	Multiple subnets to manage.

Sample design

To illustrate the application of the information in this article, consider the following scenario.

You work for a company that has 2 data centers in North America, and two data centers Europe. You identified 6 different customer facing applications maintained by 2 different business units that you want to migrate to Azure as a pilot. The basic architecture for the applications are as follows:

- App1, App2, App3, and App4 are web applications hosted on Linux servers running Ubuntu. Each application connects to a separate application server that hosts RESTful services on Linux servers. The RESTful services connect to a back end MySQL database.
- App5 and App6 are web applications hosted on Windows servers running Windows Server 2012 R2. Each application connects to a back end SQL Server database.
- All apps are currently hosted in one of the company's data centers in North America.
- The on-premises data centers use the 10.0.0.0/8 address space.

You need to design a virtual network solution that meets the following requirements:

- Each business unit should not be affected by resource consumption of other business units.
- You should minimize the amount of VNets and subnets to make management easier.
- Each business unit should have a single test/development VNet used for all applications.
- Each application is hosted in 2 different Azure data centers per continent (North America and Europe).
- Each application is completely isolated from each other.
- Each application can be accessed by customers over the Internet using HTTP.
- Each application can be accessed by users connected to the on-premises data centers by using an encrypted tunnel.
- Connection to on-premises data centers should use existing VPN devices.
- The company's networking group should have full control over the VNet configuration.
- Developers in each business unit should only be able to deploy VMs to existing subnets.
- All applications will be migrated as they are to Azure (lift-and-shift).
- The databases in each location should replicate to other Azure locations once a day.
- Each application should use 5 front end web servers, 2 application servers (when necessary), and 2 database servers.

Plan

You should start your design planning by answering the question in the [Define requirements](#) section as shown below.

1. What Azure locations will you use to host VNets?

2 locations in North America, and 2 locations in Europe. You should pick those based on the physical location of your existing on-premises data centers. That way your connection from your physical locations to Azure will have a better latency.

2. Do you need to provide communication between these Azure locations?

Yes. Since the databases must be replicated to all locations.

3. Do you need to provide communication between your Azure VNet(s) and your on-premises data center(s)?

Yes. Since users connected to the on-premises data centers must be able to access the applications through an encrypted tunnel.

4. How many IaaS VMs do you need for your solution?

200 IaaS VMs. App1, App2, App3, and App4 require 5 web servers each, 2 applications servers each, and 2 database servers each. That's a total of 9 IaaS VMs per application, or 36 IaaS VMs. App5 and App6 require 5 web servers and 2 database servers each. That's a total of 7 IaaS VMs per application, or 14 IaaS VMs. Therefore, you need 50 IaaS VMs for all applications in each Azure region. Since we need to use 4 regions,

there will be 200 IaaS VMs.

You will also need to provide DNS servers in each VNet, or in your on-premises data centers to resolve name between your Azure IaaS VMs and your on-premises network.

5. Do you need to isolate traffic based on groups of VMs (i.e. front end web servers and back end database servers)?

Yes. Each application should be completely isolated from each other, and each application layer should also be isolated.

6. Do you need to control traffic flow using virtual appliances?

No. Virtual appliances can be used to provide more control over traffic flow, including more detailed data plane logging.

7. Do users need different sets of permissions to different Azure resources?

Yes. The networking team needs full control on the virtual networking settings, while developers should only be able to deploy their VMs to pre-existing subnets.

Design

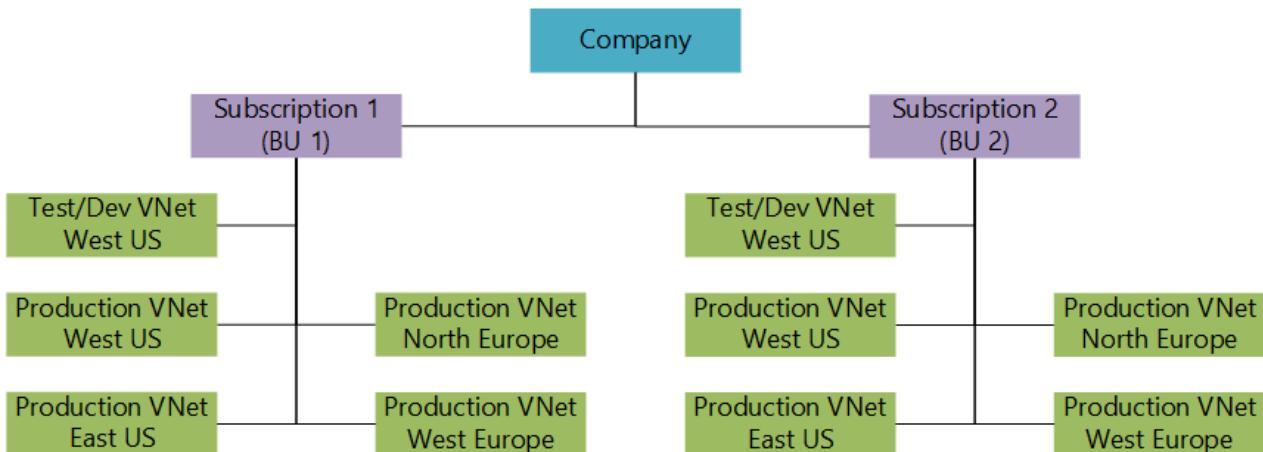
You should follow the design specifying subscriptions, VNets, subnets, and NSGs. We will discuss NSGs here, but you should learn more about [NSGs](#) before finishing your design.

Number of subscriptions and VNets

The following requirements are related to subscriptions and VNets:

- Each business unit should not be affected by resource consumption of other business units.
- You should minimize the amount of VNets and subnets.
- Each business unit should have a single test/development VNet used for all applications.
- Each application is hosted in 2 different Azure data centers per continent (North America and Europe).

Based on those requirements, you need a subscription for each business unit. That way, consumption of resources from a business unit will not count towards limits for other business units. And since you want to minimize the number of VNets, you should consider using the **one subscription per business unit, two VNets per group of apps** pattern as seen below.



You also need to specify the address space for each VNet. Since you need connectivity between the on-premises data centers and the Azure regions, the address space used for Azure VNets cannot clash with the on-premises network, and the address space used by each VNet should not clash with other existing VNets. You could use the address spaces in the table below to satisfy these requirements.

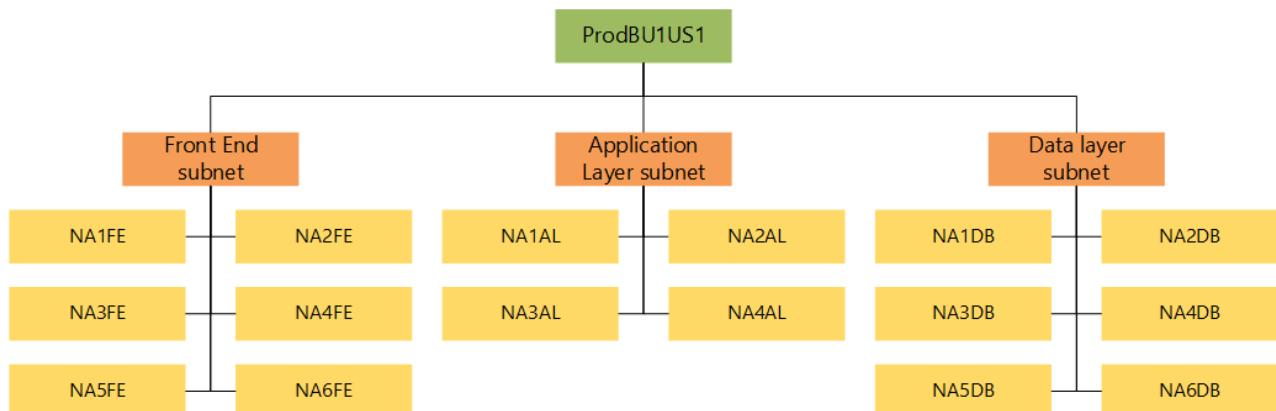
SUBSCRIPTION	VNET	AZURE REGION	ADDRESS SPACE
BU1	ProdBU1US1	West US	172.16.0.0/16
BU1	ProdBU1US2	East US	172.17.0.0/16
BU1	ProdBU1EU1	North Europe	172.18.0.0/16
BU1	ProdBU1EU2	West Europe	172.19.0.0/16
BU1	TestDevBU1	West US	172.20.0.0/16
BU2	TestDevBU2	West US	172.21.0.0/16
BU2	ProdBU2US1	West US	172.22.0.0/16
BU2	ProdBU2US2	East US	172.23.0.0/16
BU2	ProdBU2EU1	North Europe	172.24.0.0/16
BU2	ProdBU2EU2	West Europe	172.25.0.0/16

Number of subnets and NSGs

The following requirements are related to subnets and NSGs:

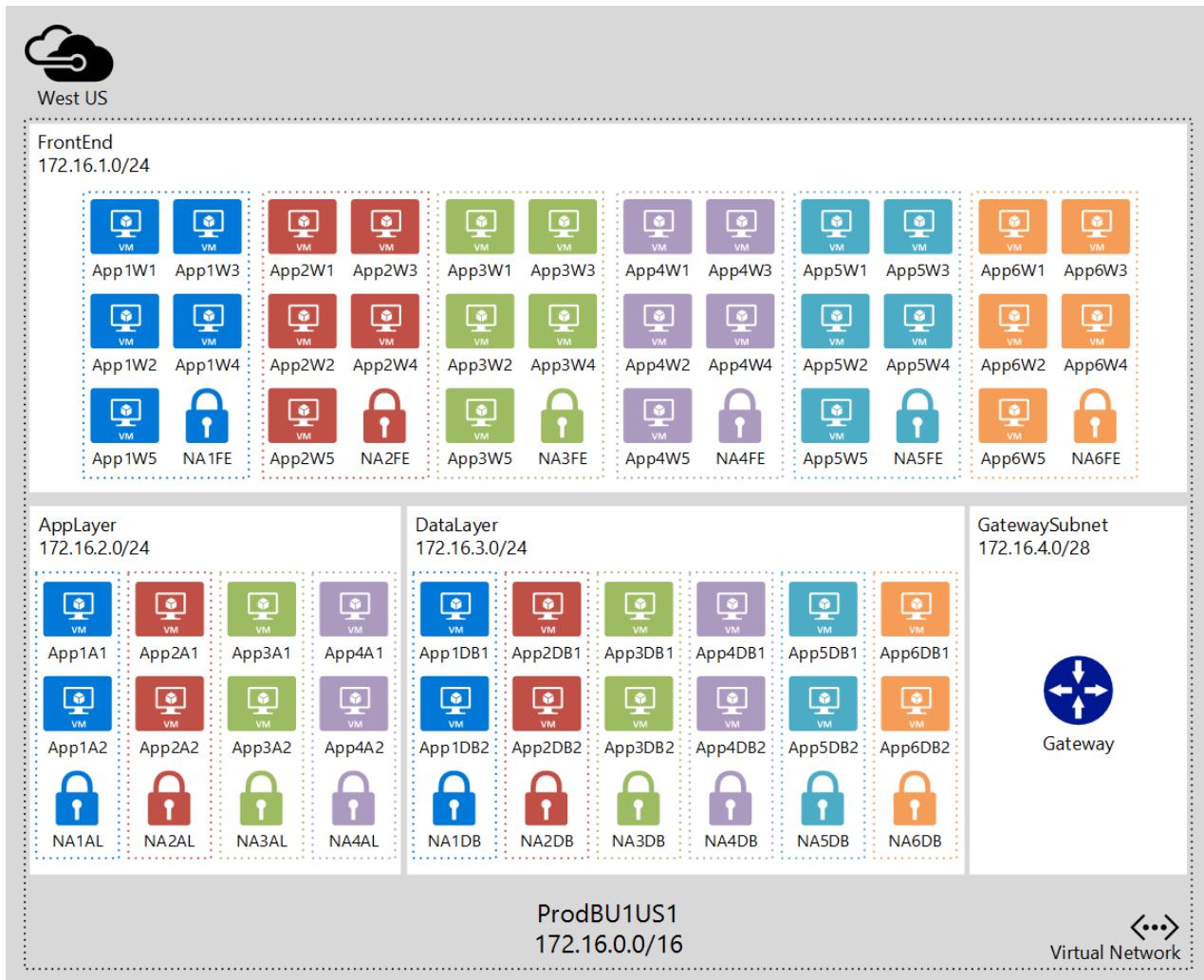
- You should minimize the amount of VNets and subnets.
- Each application is completely isolated from each other.
- Each application can be accessed by customers over the Internet using HTTP.
- Each application can be accessed by users connected to the on-premises data centers by using an encrypted tunnel.
- Connection to on-premises data centers should use existing VPN devices.
- The databases in each location should replicate to other Azure locations once a day.

Based on those requirements, you could use one subnet per application layer, and use NSGs to filter traffic per application. That way, you only have 3 subnets in each VNet (front end, application layer, and data layer) and one NSG per application per subnet. In this case, you should consider using the **one subnet per application layer, NSGs per app** design pattern. The figure below shows the use of the design pattern representing the **ProdBU1US1** VNet.



However, you also need to create an extra subnet for the VPN connectivity between the VNets, and your on-premises data centers. And you need to specify the address space for each subnet. The figure below shows a

sample solution for **ProdBU1US1** VNet. You would replicate this scenario for each VNet. Each color represents a different application.



Access Control

The following requirements are related to access control:

- The company's networking group should have full control over the VNet configuration.
- Developers in each business unit should only be able to deploy VMs to existing subnets.

Based on those requirements, you could add users from the networking team to the built-in **Network Contributor** role in each subscription; and create a custom role for the application developers in each subscription giving them rights to add VMs to existing subnets.

Next steps

- [Deploy a virtual network](#).
- Understand how to [load balance](#) IaaS VMs and [manage routing over multiple Azure regions](#).
- Learn more about [network security groups](#) an NSG solution.
- Learn more about your [cross-premises and VNet connectivity options](#).

Filter network traffic with network security groups

4/16/2018 • 13 min to read • [Edit Online](#)

A network security group (NSG) contains a list of security rules that allow or deny network traffic to resources connected to Azure Virtual Networks (VNet). NSGs can be associated to subnets, individual VMs (classic), or individual network interfaces (NIC) attached to VMs (Resource Manager). When an NSG is associated to a subnet, the rules apply to all resources connected to the subnet. Traffic can further be restricted by also associating an NSG to a VM or NIC.

NOTE

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using both models, but Microsoft recommends that most new deployments use the Resource Manager model.

NSG resource

NSGs contain the following properties:

PROPERTY	DESCRIPTION	CONSTRAINTS	CONSIDERATIONS
Name	Name for the NSG	Must be unique within the region. Can contain letters, numbers, underscores, periods, and hyphens. Must start with a letter or number. Must end with a letter, number, or underscore. Cannot exceed 80 characters.	Since you may need to create several NSGs, make sure you have a naming convention that makes it easy to identify the function of your NSGs.
Region	Azure region where the NSG is created.	NSGs can only be associated to resources within the same region as the NSG.	To learn about how many NSGs you can have per region, read the Azure limits article.
Resource group	The resource group the NSG exists in.	Although an NSG exists in a resource group, it can be associated to resources in any resource group, as long as the resource is part of the same Azure region as the NSG.	Resource groups are used to manage multiple resources together, as a deployment unit. You may consider grouping the NSG with resources it is associated to.
Rules	Inbound or outbound rules that define what traffic is allowed or denied.		See the NSG rules section of this article.

NOTE

Endpoint-based ACLs and network security groups are not supported on the same VM instance. If you want to use an NSG and have an endpoint ACL already in place, first remove the endpoint ACL. To learn how to remove an ACL, read the [Managing Access Control Lists \(ACLs\) for Endpoints by using PowerShell](#) article.

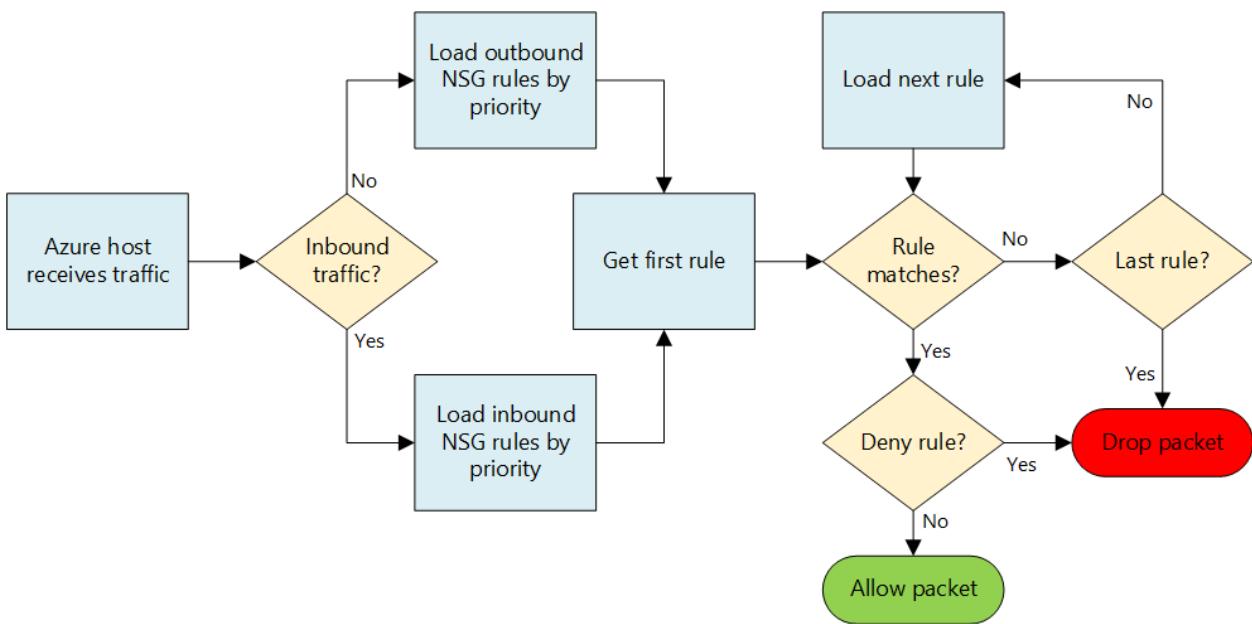
NSG rules

NSG rules contain the following properties:

PROPERTY	DESCRIPTION	CONSTRAINTS	CONSIDERATIONS
Name	Name for the rule.	Must be unique within the region. Can contain letters, numbers, underscores, periods, and hyphens. Must start with a letter or number. Must end with a letter, number, or underscore. Cannot exceed 80 characters.	You may have several rules within an NSG, so make sure you follow a naming convention that allows you to identify the function of your rule.
Protocol	Protocol to match for the rule.	TCP, UDP, or *	Using * as a protocol includes ICMP (East-West traffic only), as well as UDP and TCP, and may reduce the number of rules you need. At the same time, using * might be too broad an approach, so it's recommended that you use * only when necessary.
Source port range	Source port range to match for the rule.	Single port number from 1 to 65535, port range (example: 1-65535), or * (for all ports).	Source ports could be ephemeral. Unless your client program is using a specific port, use * in most cases. Try to use port ranges as much as possible to avoid the need for multiple rules. Multiple ports or port ranges cannot be grouped by a comma.
Destination port range	Destination port range to match for the rule.	Single port number from 1 to 65535, port range (example: 1-65535), or * (for all ports).	Try to use port ranges as much as possible to avoid the need for multiple rules. Multiple ports or port ranges cannot be grouped by a comma.
Source address prefix	Source address prefix or tag to match for the rule.	Single IP address (example: 10.10.10.10), IP subnet (example: 192.168.1.0/24), service tag , or * (for all addresses).	Consider using ranges, service tags, and * to reduce the number of rules.

PROPERTY	DESCRIPTION	CONSTRAINTS	CONSIDERATIONS
Destination address prefix	Destination address prefix or tag to match for the rule.	Single IP address (example: 10.10.10.10), IP subnet (example: 192.168.1.0/24), default tag , or * (for all addresses).	Consider using ranges, service tags, and * to reduce the number of rules.
Direction	Direction of traffic to match for the rule.	Inbound or outbound.	Inbound and outbound rules are processed separately, based on direction.
Priority	Rules are checked in the order of priority. Once a rule applies, no more rules are tested for matching.	Number between 100 and 4096.	Consider creating rules jumping priorities by 100 for each rule to leave space for new rules you might create in the future.
Access	Type of access to apply if the rule matches.	Allow or deny.	Keep in mind that if an allow rule is not found for a packet, the packet is dropped.

NSGs contain two sets of rules: Inbound and outbound. The priority for a rule must be unique within each set.



The previous picture shows how NSG rules are processed.

System tags

Service tags are system-provided identifiers to address a category of IP addresses. You can use service tags in the **source address prefix** and **destination address prefix** properties of any security rule. Learn more about [service tags](#).

Default security rules

All NSGs contain a set of default security rules. The default rules cannot be deleted, but because they are assigned the lowest priority, they can be overridden by the rules that you create. Learn more about [default security rules](#).

Associating NSGs

You can associate an NSG to VMs, NICs, and subnets, depending on the deployment model you are using, as follows:

- **VM (classic only):** Security rules are applied to all traffic to/from the VM.
- **NIC (Resource Manager only):** Security rules are applied to all traffic to/from the NIC the NSG is associated to. In a multi-NIC VM, you can apply different (or the same) NSG to each NIC individually.
- **Subnet (Resource Manager and classic):** Security rules are applied to any traffic to/from any resources connected to the Subnet.

You can associate different NSGs to a VM (or NIC, depending on the deployment model) and the subnet that a NIC or VM is connected to. Security rules are applied to the traffic, by priority, in each NSG, in the following order:

- **Inbound traffic**

1. **NSG applied to subnet:** If a subnet NSG has a matching rule to deny traffic, the packet is dropped.
2. **NSG applied to NIC** (Resource Manager) or VM (classic): If VM\NIC NSG has a matching rule that denies traffic, packets are dropped at the VM\NIC, even if a subnet NSG has a matching rule that allows traffic.

- **Outbound traffic**

1. **NSG applied to NIC** (Resource Manager) or VM (classic): If a VM\NIC NSG has a matching rule that denies traffic, packets are dropped.
2. **NSG applied to subnet:** If a subnet NSG has a matching rule that denies traffic, packets are dropped, even if a VM\NIC NSG has a matching rule that allows traffic.

NOTE

Although you can only associate a single NSG to a subnet, VM, or NIC; you can associate the same NSG to as many resources as you want.

Implementation

You can implement NSGs in the Resource Manager or classic deployment models using the following tools:

DEPLOYMENT TOOL	CLASSIC	RESOURCE MANAGER
Azure portal	Yes	Yes
PowerShell	Yes	Yes
Azure CLI V1	Yes	Yes
Azure CLI V2	No	Yes
Azure Resource Manager template	No	Yes

Planning

Before implementing NSGs, you need to answer the following questions:

1. What types of resources do you want to filter traffic to or from? You can connect resources such as NICs (Resource Manager), VMs (classic), Cloud Services, Application Service Environments, and VM Scale Sets.
2. Are the resources you want to filter traffic to/from connected to subnets in existing VNets?

For more information on planning for network security in Azure, read the [Cloud services and network security](#) article.

Design considerations

Once you know the answers to the questions in the [Planning](#) section, review the following sections before defining your NSGs:

Limits

There are limits to the number of NSGs you can have in a subscription and number of rules per NSG. To learn more about the limits, read the [Azure limits](#) article.

VNet and subnet design

Since NSGs can be applied to subnets, you can minimize the number of NSGs by grouping your resources by subnet, and applying NSGs to subnets. If you decide to apply NSGs to subnets, you may find that existing VNets and subnets you have were not defined with NSGs in mind. You may need to define new VNets and subnets to support your NSG design and deploy your new resources to your new subnets. You could then define a migration strategy to move existing resources to the new subnets.

Special rules

If you block traffic allowed by the following rules, your infrastructure can't communicate with essential Azure services:

- **Virtual IP of the host node:** Basic infrastructure services such as DHCP, DNS, and health monitoring are provided through the virtualized host IP address 168.63.129.16. This public IP address belongs to Microsoft and is the only virtualized IP address used in all regions for this purpose. This IP address maps to the physical IP address of the server machine (host node) hosting the VM. The host node acts as the DHCP relay, the DNS recursive resolver, and the probe source for the load balancer health probe and the machine health probe. Communication to this IP address is not an attack.
- **Licensing (Key Management Service):** Windows images running in VMs 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.

ICMP traffic

The current NSG rules only allow for protocols *TCP* or *UDP*. There is not a specific tag for *ICMP*. However, ICMP traffic is allowed within a VNet by the *AllowVNetInBound* default rule, that allows traffic to and from any port and protocol within the VNet.

Subnets

- Consider the number of tiers your workload requires. Each tier can be isolated by using a subnet, with an NSG applied to the subnet.
- If you need to implement a subnet for a VPN gateway, or ExpressRoute circuit, do **not** apply an NSG to that subnet. If you do so, cross-VNet or cross-premises connectivity may fail.
- If you need to implement a network virtual appliance (NVA), connect the NVA to its own subnet and create user-defined routes (UDR) to and from the NVA. You can implement a subnet level NSG to filter traffic in and out of this subnet. To learn more about UDRs, read the [User-defined routes](#) article.

Load balancers

- Consider the load balancing and network address translation (NAT) rules for each load balancer used by each of your workloads. NAT rules are bound to a back-end pool that contains NICs (Resource Manager) or

VMs/Cloud Services role instances (classic). Consider creating an NSG for each back-end pool, allowing only traffic mapped through the rules implemented in the load balancers. Creating an NSG for each back-end pool guarantees that traffic coming to the back-end pool directly (rather than through the load balancer), is also filtered.

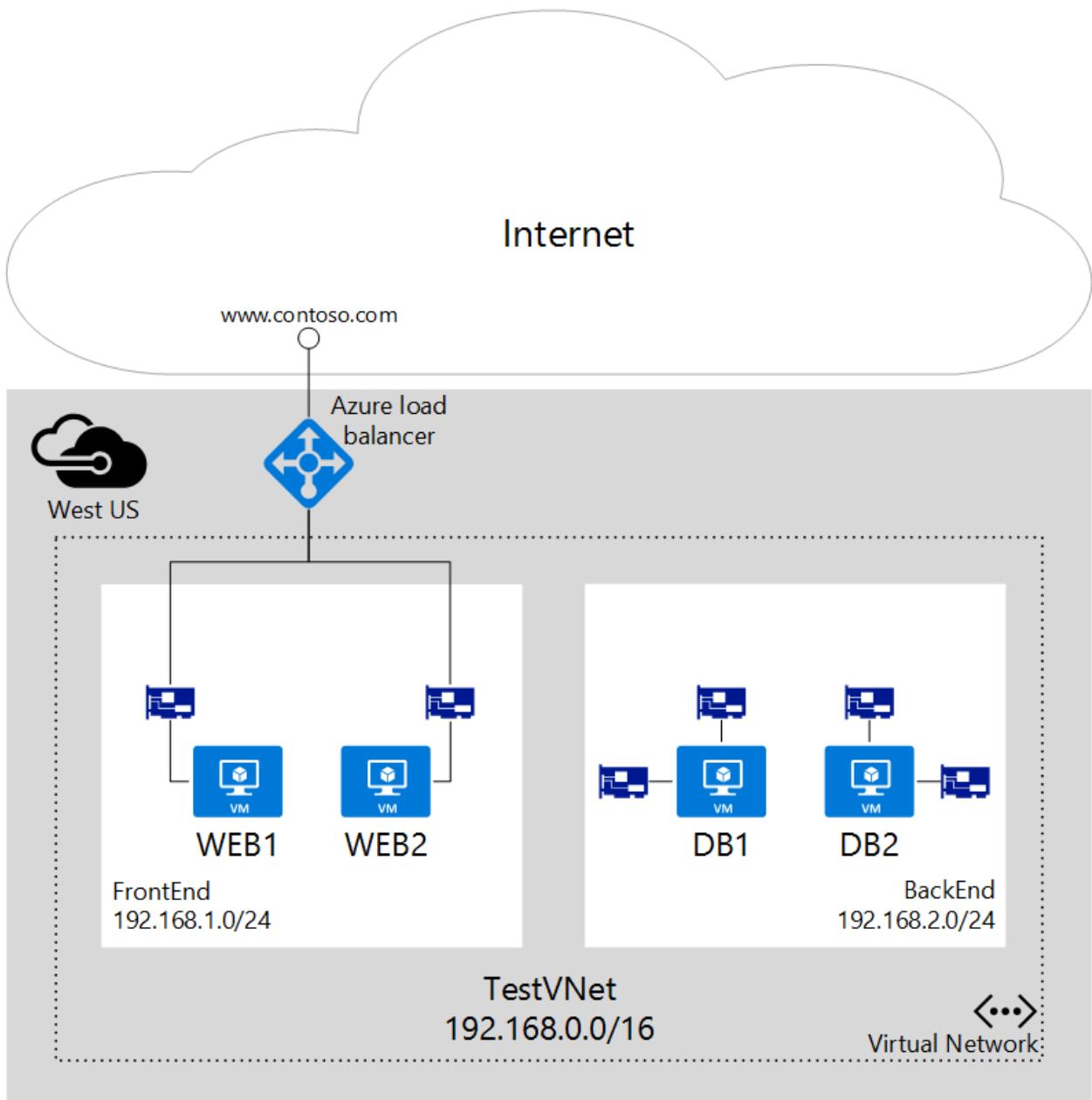
- In classic deployments, you create endpoints that map ports on a load balancer to ports on your VMs or role instances. You can also create your own individual public-facing load balancer through Resource Manager. The destination port for incoming traffic is the actual port in the VM or role instance, not the port exposed by a load balancer. The source port and address for the connection to the VM is a port and address on the remote computer in the Internet, not the port and address exposed by the load balancer.
- When you create NSGs to filter traffic coming through an Azure Load Balancer, the source port and address range applied are from the originating computer, not the load balancer frontend. The destination port and address range are those of the destination computer, not the load balancer frontend.
- If you block the `AzureLoadBalancer` tag, the health probes from Azure Load Balancer will fail and your service may be impacted.

Other

- Endpoint-based access control lists (ACL) and NSGs are not supported on the same VM instance. If you want to use an NSG and have an endpoint ACL already in place, first remove the endpoint ACL. For information about how to remove an endpoint ACL, see the [Manage endpoint ACLs](#) article.
- In Resource Manager, you can use an NSG associated to a NIC for VMs with multiple NICs to enable management (remote access) on a per NIC basis. Associating unique NSGs to each NIC enables separation of traffic types across NICs.
- Similar to the use of load balancers, when filtering traffic from other VNets, you must use the source address range of the remote computer, not the gateway connecting the VNets.
- Many Azure services cannot be connected to VNets. If an Azure resource is not connected to a VNet, you cannot use an NSG to filter traffic to the resource. Read the documentation for the services you use to determine whether the service can be connected to a VNet.

Sample deployment

To illustrate the application of the information in this article, consider a common scenario of a two tier application shown in the following picture:



As shown in the diagram, the *Web1* and *Web2* VMs are connected to the *FrontEnd* subnet, and the *DB1* and *DB2* VMs are connected to the *BackEnd* subnet. Both subnets are part of the *TestVNet* VNet. The application components each run within an Azure VM connected to a VNet. The scenario has the following requirements:

1. Separation of traffic between the WEB and DB servers.
2. Load balancing rules forward traffic from the load balancer to all web servers on port 80.
3. Load balancer NAT rules forward traffic coming into the load balancer on port 50001 to port 3389 on the *WEB1* VM.
4. No access to the front-end or back-end VMs from the Internet, except requirements 2 and 3.
5. No outbound Internet access from the WEB or DB servers.
6. Access from the *FrontEnd* subnet is allowed to port 3389 of any web server.
7. Access from the *FrontEnd* subnet is allowed to port 3389 of any DB server.
8. Access from the *FrontEnd* subnet is allowed to port 1433 of all DB servers.
9. Separation of management traffic (port 3389) and database traffic (1433) on different NICs in DB servers.

Requirements 1-6 (except requirements 3 and 4) are all confined to subnet spaces. The following NSGs meet the previous requirements, while minimizing the number of NSGs required:

FrontEnd

Inbound rules

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
Allow-Inbound-HTTP-Internet	Allow	100	Internet	*	*	80	TCP
Allow-Inbound-RDP-Internet	Allow	200	Internet	*	*	3389	TCP
Deny-Inbound-All	Deny	300	Internet	*	*	*	TCP

Outbound rules

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
Deny-Internet-All	Deny	100	*	*	Internet	*	*

BackEnd

Inbound rules

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
Deny-Internet-All	Deny	100	Internet	*	*	*	*

Outbound rules

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
Deny-Internet-All	Deny	100	*	*	Internet	*	*

The following NSGs are created and associated to NICs in the following VMs:

WEB1

Inbound rules

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
Allow-Inbound-RDP-Internet	Allow	100	Internet	*	*	3389	TCP

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
------	--------	----------	----------------------	-------------	---------------------------	------------------	----------

Allow-Inbound-HTTP-Internet	Allow	200	Internet	*	*	80	TCP
-----------------------------	-------	-----	----------	---	---	----	-----

NOTE

The source address range for the previous rules is **Internet**, not the virtual IP address of for the load balancer. The source port is , *not 500001. NAT rules for load balancers are not the same as NSG security rules. NSG security rules are always related to the original source and final destination of traffic, **not** the load balancer between the two.* Azure Load Balancer always preserves the source IP address and port.

WEB2

Inbound rules

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
Deny-Inbound-RDP-Internet	Deny	100	Internet	*	*	3389	TCP
Allow-Inbound-HTTP-Internet	Allow	200	Internet	*	*	80	TCP

DB servers (Management NIC)

Inbound rules

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
Allow-Inbound-RDP-Front-end	Allow	100	192.168.1.0/24	*	*	3389	TCP

DB servers (Database traffic NIC)

Inbound rules

Rule	Access	Priority	Source Address Range	Source Port	Destination Address Range	Destination Port	Protocol
Allow-Inbound-SQL-Front-end	Allow	100	192.168.1.0 /24	*	*	1433	TCP

Since some of the NSGs are associated to individual NICs, the rules are for resources deployed through Resource Manager. Rules are combined for subnet and NIC, depending on how they are associated.

Next steps

- [Deploy NSGs \(Resource Manager\)](#).
- [Deploy NSGs \(classic\)](#).
- [Manage NSG logs](#).
- [Troubleshoot NSGs](#)

Filter network traffic with a network security group using the Azure CLI

4/11/2018 • 6 min 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.

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

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 2.0](#).

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 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](#).

Create a network security group using the Azure portal

4/19/2018 • 3 min to read • [Edit Online](#)

You can use an NSG to control traffic to one or more virtual machines (VMs), role instances, network adapters (NICs), or subnets in your virtual network. An NSG contains access control rules that allow or deny traffic based on traffic direction, protocol, source address and port, and destination address and port. The rules of an NSG can be changed at any time, and changes are applied to all associated instances.

For more information about NSGs, visit [what is an NSG](#).

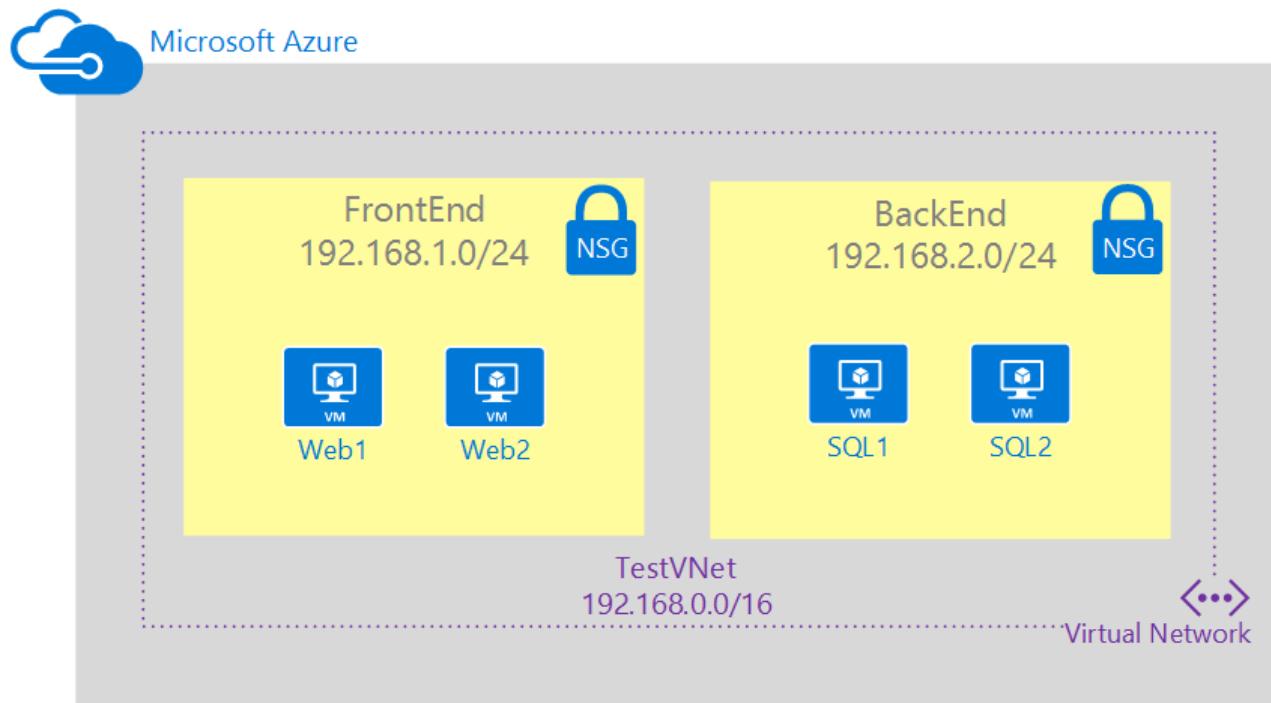
IMPORTANT

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 [create NSGs in the classic deployment model](#).

Scenario

To better illustrate how to create NSGs, this document uses the following scenario:



In this scenario, you create an NSG for each subnet in the **TestVNet** virtual network, as follows:

- **NSG-FrontEnd.** The front-end NSG is applied to the *FrontEnd* subnet, and contains two rules:
 - **rdp-rule.** Allows RDP traffic to the *FrontEnd* subnet.
 - **web-rule.** Allows HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd.** The back-end NSG is applied to the *BackEnd* subnet, and contains two rules:

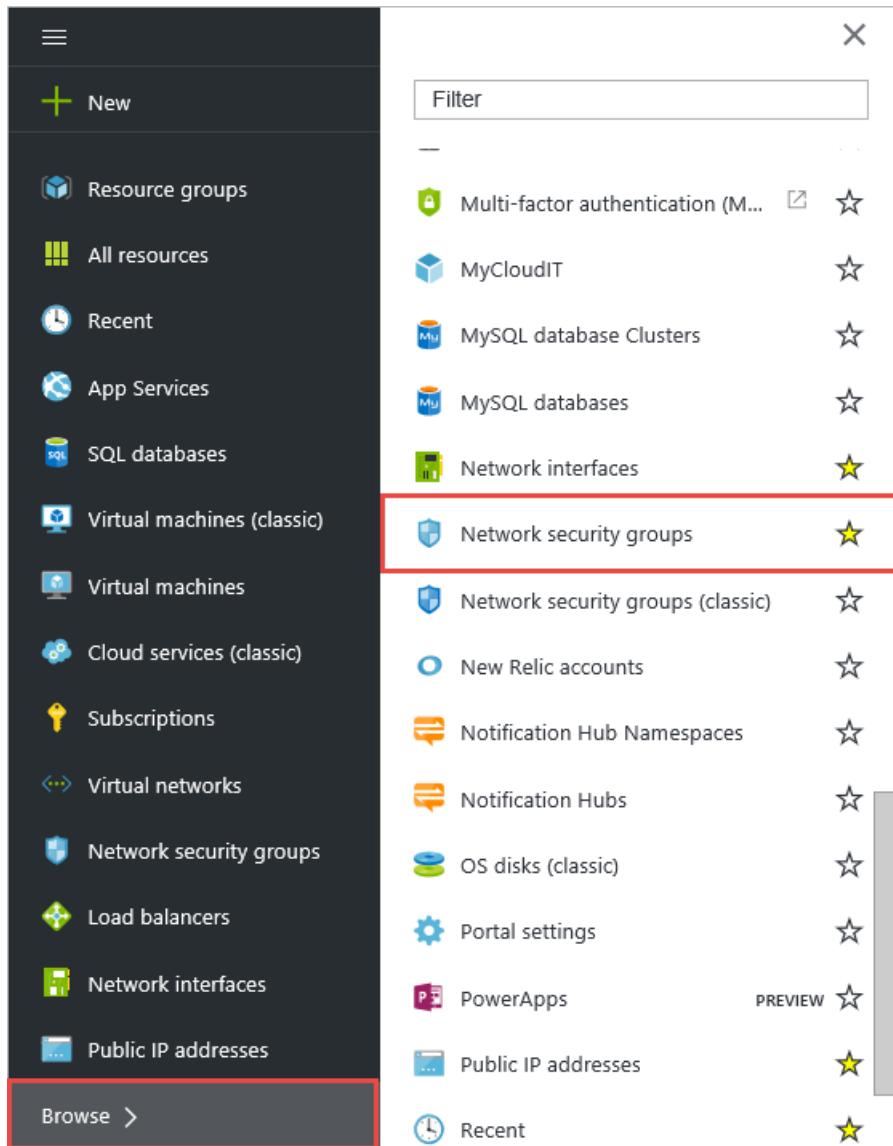
- **sql-rule.** Allows SQL traffic only from the *FrontEnd* subnet.
- **web-rule.** Denies all internet bound traffic from the *BackEnd* subnet.

The combination of these rules create a DMZ-like scenario, where the back-end subnet can only receive incoming traffic for SQL from the front-end subnet, and has no access to the Internet, while the front-end subnet can communicate with the Internet, and receive incoming HTTP requests only.

Create the NSG-FrontEnd NSG

To create the **NSG-FrontEnd** NSG as shown in the scenario, complete the following steps:

1. From a browser, navigate to <https://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Select + **Create a resource > > Network Security Groups.**



3. Under **Network security groups**, select **Add**.

The screenshot shows the 'Network security groups' blade in the Azure portal. At the top, there are three buttons: 'Add' (highlighted with a red box), 'Columns', and 'Refresh'. Below these is a search bar labeled 'Filter by name...'. The main area is titled 'NAME' and displays the message 'No network security groups to display'.

- Under **Create network security group**, create an NSG named *NSG-FrontEnd* in the *RG-NSG* resource group, and then select **Create**.

The screenshot shows the 'Create network security group' dialog box. It includes fields for 'Name' (set to 'NSG-FrontEnd'), 'Subscription' (set to 'Microsoft Azure Internal Consumption (628)'), 'Resource Group' (set to 'RG-NSG'), 'Location' (set to 'West US'), and a 'Pin to dashboard' checkbox (unchecked). A large blue 'Create' button is at the bottom.

Create rules in an existing NSG

To create rules in an existing NSG from the Azure portal, complete the following steps:

- Select **All Services**, then search for **Network security groups**. When **Network security groups** appear, select it.
- In the list of NSGs, select **NSG-FrontEnd > Inbound security rules**

The screenshot shows two windows side-by-side. On the left, the 'Network security groups' blade lists three groups: 'joamatearm', 'NSG-BackEnd', and 'NSG-FrontEnd'. The 'NSG-FrontEnd' row is selected and highlighted with a red box. On the right, the 'NSG-FrontEnd' settings blade is open. In the 'Essentials' section, it shows the resource group 'TestRG', location 'West US', subscription 'Microsoft Azure Internal Consumption', and a single inbound security rule. The 'Inbound security rules' link is highlighted with a red box. Below that, there's an 'Operations' section with a bar chart titled 'Events' for 'NSG-FRONTEND'.

3. In the list of **Inbound security rules**, select **Add**.

This screenshot shows the 'Settings' blade for 'NSG-FrontEnd'. The 'Inbound security rules' section is expanded, showing a list of existing rules. A red box highlights the '+ Add' button. The left sidebar contains links for Properties, Inbound security rules (which is currently selected), Outbound security rules, Network interfaces, Subnets, Users, and Tags.

4. Under **Add inbound security rule**, create a rule named *web-rule* with priority of *200* allowing access via *TCP* to port *80* to any VM from any source, and then select **OK**. Notice that most of these settings are default values already.

Add inbound security rule
NSG-FrontEnd

* Name: web-rule

* Priority: 200

Source: Any

Protocol: TCP

* Source port range: *

Destination: Any

* Destination port range: 80

Action: Allow

OK

5. After a few seconds, you see the new rule in the NSG.

Inbound security rules
NSG-FrontEnd

+ Add

Search inbound security rules

PRIORITY	NAME	SOURCE	DESTINATION	SERVICE	ACTION
100	sql-rule	192.168.1.0/24	Any	TCP/1433	Allow
200	web-rule	Any	Any	TCP/80	Allow ...

6. Repeat steps to 6 to create an inbound rule named *rdp-rule* with a priority of *250* allowing access via *TCP* to port *3389* to any VM from any source.

Associate the NSG to the FrontEnd subnet

- Select **All services** >, enter **Resource groups**, select **Resource groups** when it appears, then select **RG-NSG**.
- Under **RG-NSG**, select ... > **TestVNet**.

RG-NSG Resource group

Subscription name: Microsoft Azure Internal Consumption
Last deployment: 2/15/2016 (Succeeded)

Subscription ID: 628dad04-b5d1-4f10-b3a4-dc61d88cf97c
Location: West US

All settings →

Summary

Add tiles +

Resources

- sqlAvSet
- webAvSet
- SQL1
- SQL2
- Web1
- Web2
- TestNICSQL1
- TestNICSQL2
- TestNICWeb1

Add a group +

NAME	RESOURCE GROUP
sqlAvSet	RG-NSG
webAvSet	RG-NSG
SQL1	RG-NSG
SQL2	RG-NSG
Web1	RG-NSG
Web2	RG-NSG
TestNICSQL1	RG-NSG
TestNICSQL2	RG-NSG
TestNICWeb1	RG-NSG
TestNICWeb2	RG-NSG
NSG-FrontEnd	RG-NSG
TestPIPSQL1	RG-NSG
TestPIPSQL2	RG-NSG
TestPIPWeb1	RG-NSG
TestPIPWeb2	RG-NSG
TestVNet	RG-NSG

3. Under **Settings**, select **Subnets** > **FrontEnd** > **Network security group** > **NSG-FrontEnd**.

Settings

TestNet

Search settings

SUPPORT & TROUBLESHOOTING

- Audit logs

GENERAL

- Properties
- Address space
- Subnets**
- DNS servers

RESOURCE MANAGEMENT

- Users

Subnets

TestNet

+

Search subnets

NAME	ADDRESS RANGE	AVAILABLE ADDRESSES	SECURITY GROUP
FrontEnd	192.168.1.0/24	249	...
BackEnd	192.168.2.0/24	249	...

FrontEnd

TestNet

Save Discard Delete

* Address range (CIDR block) 192.168.1.0/24
192.168.1.0 - 192.168.1.255 (256 addresses)

Available addresses 249

Network security group None

Route table None

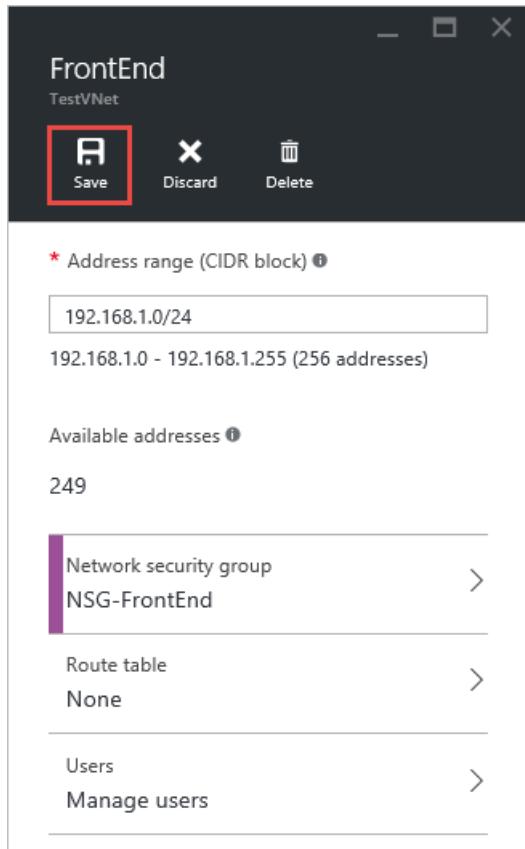
Users Manage users

Choose network security group

None

NSG-FrontEnd RG-NSG

4. In the **FrontEnd** blade, select **Save**.



Create the NSG-BackEnd NSG

To create the **NSG-BackEnd** NSG and associate it to the **BackEnd** subnet, complete the following steps:

1. To create an NSG named *NSG-BackEnd*, repeat the steps in [Create the NSG-FrontEnd NSG](#).
2. To create the **inbound** rules in the table that follows, repeat the steps in [Create rules in an existing NSG](#).

INBOUND RULE	OUTBOUND RULE
<p>* Name sql-rule ✓</p> <p>* Priority 100</p> <p>Source Any CIDR block Tag</p> <p>* Source IP address range 192.168.1.0/24 ✓</p> <p>Protocol Any TCP UDP</p> <p>* Source port range*</p> <p>Destination Any CIDR block Tag</p> <p>* Destination port range 1433 ✓</p> <p>Action Deny Allow</p>	<p>* Name web-rule ✓</p> <p>* Priority 100</p> <p>Destination Any CIDR block Tag</p> <p>Destination tag Internet</p> <p>* Destination port range*</p> <p>Source Any CIDR block Tag</p> <p>Protocol Any TCP UDP</p> <p>* Source port range*</p> <p>Action Deny Allow</p>

- To associate the **NSG-Backend** NSG to the **BackEnd** subnet, repeat the steps in [Associate the NSG to the FrontEnd subnet](#).

Next Steps

- Learn how to [manage existing NSGs](#)
- [Enable logging](#) for NSGs.

Route network traffic with a route table using PowerShell

4/18/2018 • 8 min 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.

Launch Azure Cloud Shell

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. Just click the **Copy** to copy the code, paste it into the Cloud Shell, and then press enter to run it. There are a few ways to launch the Cloud Shell:

Click Try It in the upper right corner of a code block.	
Open Cloud Shell in your browser.	
Click the Cloud Shell button on the menu in the upper right of the Azure portal.	

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 5.4.1 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` to create a connection with Azure.

Create a route table

Before you can create a route table, create a resource group with [New-AzureRmResourceGroup](#). The following example creates a resource group named *myResourceGroup* for all resources created in this article.

```
New-AzureRmResourceGroup -ResourceGroupName myResourceGroup -Location EastUS
```

Create a route table with [New-AzureRmRouteTable](#). The following example creates a route table named

`myRouteTablePublic.`

```
$routeTablePublic = New-AzureRmRouteTable `  
    -Name 'myRouteTablePublic' `  
    -ResourceGroupName myResourceGroup `  
    -location EastUS
```

Create a route

Create a route by retrieving the route table object with [Get-AzureRmRouteTable](#), create a route with [Add-AzureRmRouteConfig](#), then write the route configuration to the route table with [Set-AzureRmRouteTable](#).

```
Get-AzureRmRouteTable `  
    -ResourceGroupName "myResourceGroup" `  
    -Name "myRouteTablePublic" `  
    | Add-AzureRmRouteConfig `  
        -Name "ToPrivateSubnet" `  
        -AddressPrefix 10.0.1.0/24 `  
        -NextHopType "VirtualAppliance" `  
        -NextHopIpAddress 10.0.2.4 `  
    | Set-AzureRmRouteTable
```

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-AzureRmVirtualNetwork](#). The following example creates a virtual network named *myVirtualNetwork* with the address prefix *10.0.0.0/16*.

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

Create three subnets by creating three subnet configurations with [New-AzureRmVirtualNetworkSubnetConfig](#). The following example creates three subnet configurations for *Public*, *Private*, and *DMZ* subnets:

```
$subnetConfigPublic = Add-AzureRmVirtualNetworkSubnetConfig `  
    -Name Public `  
    -AddressPrefix 10.0.0.0/24 `  
    -VirtualNetwork $virtualNetwork  
  
$subnetConfigPrivate = Add-AzureRmVirtualNetworkSubnetConfig `  
    -Name Private `  
    -AddressPrefix 10.0.1.0/24 `  
    -VirtualNetwork $virtualNetwork  
  
$subnetConfigDmz = Add-AzureRmVirtualNetworkSubnetConfig `  
    -Name DMZ `  
    -AddressPrefix 10.0.2.0/24 `  
    -VirtualNetwork $virtualNetwork
```

Write the subnet configurations to the virtual network with [Set-AzureRmVirtualNetwork](#), which creates the subnets in the virtual network:

```
$virtualNetwork | Set-AzureRmVirtualNetwork
```

Associate the *myRouteTablePublic* route table to the *Public* subnet with [Set-AzureRmVirtualNetworkSubnetConfig](#) and then write the subnet configuration to the virtual network with [Set-AzureRmVirtualNetwork](#).

```
Set-AzureRmVirtualNetworkSubnetConfig `  
-VirtualNetwork $virtualNetwork `  
-Name 'Public' `  
-AddressPrefix 10.0.0.0/24 `  
-RouteTable $routeTablePublic | `  
Set-AzureRmVirtualNetwork
```

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-AzureRmVirtualNetwork](#), then the subnet Id with [Get-AzureRmVirtualNetworkSubnetConfig](#). Create a network interface with [New-AzureRmNetworkInterface](#) in the *DMZ* subnet with IP forwarding enabled:

```
# Retrieve the virtual network object into a variable.  
$virtualNetwork=Get-AzureRmVirtualNetwork `  
-Name myVirtualNetwork `  
-ResourceGroupName myResourceGroup  
  
# Retrieve the subnet configuration into a variable.  
$subnetConfigDmz = Get-AzureRmVirtualNetworkSubnetConfig `  
-Name DMZ `  
-VirtualNetwork $virtualNetwork  
  
# Create the network interface.  
$nic = New-AzureRmNetworkInterface `  
-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-AzureRmVMConfig](#). 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-AzureRmVMConfig `

    -VMName 'myVmNva' `

    -VMSize Standard_DS2 | `

    Set-AzureRmVMOperatingSystem -Windows `

        -ComputerName 'myVmNva' `

        -Credential $cred | `

    Set-AzureRmVMSourceImage `

        -PublisherName MicrosoftWindowsServer `

        -Offer WindowsServer `

        -Skus 2016-Datacenter `

        -Version latest | `

    Add-AzureRmVMNetworkInterface -Id $nic.Id

```

Create the VM using the VM configuration with [New-AzureRmVM](#). The following example creates a VM named *myVmNva*.

```

$vmNva = New-AzureRmVM `

    -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-AzureRmVM](#). The following example creates a VM named *myVmPublic* in the *Public* subnet of the *myVirtualNetwork* virtual network.

```

New-AzureRmVm `

    -ResourceGroupName "myResourceGroup" `

    -Location "East US" `

    -VirtualNetworkName "myVirtualNetwork" `

    -SubnetName "Public" `

    -ImageName "Win2016Datacenter" `

    -Name "myVmPublic" `

    -AsJob

```

Create a VM in the *Private* subnet.

```

New-AzureRmVm `

    -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-AzureRmPublicIpAddress](#) to return the public IP address of the *myVmPrivate* VM. The following example returns the public IP address of the *myVmPrivate* VM:

```
Get-AzureRmPublicIpAddress `  
-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-AzureRmResourcegroup](#) to remove the resource group and all of the resources it contains.

```
Remove-AzureRmResourceGroup -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

4/9/2018 • 7 min 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.

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

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 2.0](#).

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-
000000000000/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 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](#).

Restrict network access to PaaS resources with virtual network service endpoints using PowerShell

4/18/2018 • 10 min 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.

Launch Azure Cloud Shell

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. Just click the **Copy** to copy the code, paste it into the Cloud Shell, and then press enter to run it. There are a few ways to launch the Cloud Shell:

Click Try It in the upper right corner of a code block.	
Open Cloud Shell in your browser.	
Click the Cloud Shell button on the menu in the upper right of the Azure portal.	

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 5.4.1 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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-AzureRmResourceGroup](#). The following example creates a resource group named *myResourceGroup*:

```
New-AzureRmResourceGroup -ResourceGroupName myResourceGroup -Location EastUS
```

Create a virtual network with [New-AzureRmVirtualNetwork](#). The following example creates a virtual network named *myVirtualNetwork* with the address prefix *10.0.0.0/16*.

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

Create a subnet configuration with [New-AzureRmVirtualNetworkSubnetConfig](#). The following example creates a subnet configuration for a subnet named *Public*:

```
$subnetConfigPublic = Add-AzureRmVirtualNetworkSubnetConfig `  
    -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-AzureRmVirtualNetwork](#):

```
$virtualNetwork | Set-AzureRmVirtualNetwork
```

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-AzureRmVirtualNetworkAvailableEndpointService](#). 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-AzureRmVirtualNetworkAvailableEndpointService -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-AzureRmVirtualNetworkSubnetConfig `  
    -Name Private `  
    -AddressPrefix 10.0.1.0/24 `  
    -VirtualNetwork $virtualNetwork `  
    -ServiceEndpoint Microsoft.Storage  
  
$virtualNetwork | Set-AzureRmVirtualNetwork
```

Restrict network access for a subnet

Create network security group security rules with [New-AzureRmNetworkSecurityRuleConfig](#). The following rule allows outbound access to the public IP addresses assigned to the Azure Storage service:

```
$rule1 = New-AzureRmNetworkSecurityRuleConfig `  
    -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-AzureRmNetworkSecurityRuleConfig `  
    -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-AzureRmNetworkSecurityRuleConfig `  
    -Name Allow-RDP-All `  
    -Access Allow `  
    -DestinationAddressPrefix VirtualNetwork `  
    -DestinationPortRange 3389 `  
    -Direction Inbound `  
    -Priority 120 `  
    -Protocol * `  
    -SourceAddressPrefix * `  
    -SourcePortRange *
```

Create a network security group with [New-AzureRmNetworkSecurityGroup](#). The following example creates a network security group named *myNsgPrivate*.

```
$nsg = New-AzureRmNetworkSecurityGroup `  
    -ResourceGroupName myResourceGroup `  
    -Location EastUS `  
    -Name myNsgPrivate `  
    -SecurityRules $rule1,$rule2,$rule3
```

Associate the network security group to the *Private* subnet with [Set-AzureRmVirtualNetworkSubnetConfig](#) and then write the subnet configuration to the virtual network. The following example associates the *myNsgPrivate* network security group to the *Private* subnet:

```
Set-AzureRmVirtualNetworkSubnetConfig ` 
    -VirtualNetwork $VirtualNetwork ` 
    -Name Private ` 
    -AddressPrefix 10.0.1.0/24 ` 
    -ServiceEndpoint Microsoft.Storage ` 
    -NetworkSecurityGroup $nsg 

$virtualNetwork | Set-AzureRmVirtualNetwork
```

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-AzureRmStorageAccount](#). 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-AzureRmStorageAccount ` 
    -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-AzureRmStorageAccountKey](#):

```
$storageAcctKey = (Get-AzureRmStorageAccountKey ` 
    -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-AzureStorageContext](#). The context encapsulates the storage account name and account key:

```
$storageContext = New-AzureStorageContext $storageAcctName $storageAcctKey
```

Create a file share with [New-AzureStorageShare](#):

```
$share = New-AzureStorageShare 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-AzureRmStorageAccountNetworkRuleSet](#). Once network access is denied, the storage account is not accessible from any network.

```
Update-AzureRmStorageAccountNetworkRuleSet ` 
-ResourceGroupName "myresourcegroup" ` 
-Name $storageAcctName ` 
-DefaultAction Deny
```

Enable network access from a subnet

Retrieve the created virtual network with [Get-AzureRmVirtualNetwork](#) and then retrieve the private subnet object into a variable with [Get-AzureRmVirtualNetworkSubnetConfig](#):

```
$privateSubnet = Get-AzureRmVirtualNetwork ` 
-ResourceGroupName "myResourceGroup" ` 
-Name "myVirtualNetwork" ` 
| Get-AzureRmVirtualNetworkSubnetConfig ` 
-Name "Private"
```

Allow network access to the storage account from the *Private* subnet with [Add-AzureRmStorageAccountNetworkRule](#).

```
Add-AzureRmStorageAccountNetworkRule ` 
-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-AzureRmVM](#). 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-AzureRmVm ` 
-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-AzureRmVM

Create the second virtual machine

Create a virtual machine in the *Private* subnet:

```
New-AzureRmVm ` 
    -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-AzureRmPublicIpAddress](#) to return the public IP address of a VM. The following example returns the public IP address of the *myVmPrivate* VM:

```
Get-AzureRmPublicIpAddress ` 
    -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-AzureRmPublicIpAddress `  
-Name myVmPublic `  
-ResourceGroupName myResourceGroup `  
| SelectIpAddress
```

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 -PSProvider 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-AzureStorageFile `  
-ShareName my-file-share `  
-Context $storageContext
```

Access is denied, and you receive a `Get-AzureStorageFile : 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-AzureRmResourceGroup](#) to remove the resource group and all of the resources it contains:

```
Remove-AzureRmResourceGroup -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

4/9/2018 • 9 min 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.

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal .	

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 2.0](#).

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](security-overview.md#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 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](#).

Connect virtual networks with virtual network peering using PowerShell

4/18/2018 • 5 min 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.

Launch Azure Cloud Shell

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. Just click the **Copy** to copy the code, paste it into the Cloud Shell, and then press enter to run it. There are a few ways to launch the Cloud Shell:

Click Try It in the upper right corner of a code block.	
Open Cloud Shell in your browser.	
Click the Cloud Shell button on the menu in the upper right of the Azure portal.	

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 5.4.1 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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-AzureRmResourceGroup](#). The following example creates a resource group named *myResourceGroup* in the *eastus* location.

```
New-AzureRmResourceGroup -ResourceGroupName myResourceGroup -Location EastUS
```

Create a virtual network with [New-AzureRmVirtualNetwork](#). The following example creates a virtual network named *myVirtualNetwork1* with the address prefix *10.0.0.0/16*.

```
$virtualNetwork1 = New-AzureRmVirtualNetwork `  
    -ResourceGroupName myResourceGroup `  
    -Location EastUS `  
    -Name myVirtualNetwork1 `  
    -AddressPrefix 10.0.0.0/16
```

Create a subnet configuration with [New-AzureRmVirtualNetworkSubnetConfig](#). The following example creates a subnet configuration with a 10.0.0.0/24 address prefix:

```
$subnetConfig = Add-AzureRmVirtualNetworkSubnetConfig `  
    -Name Subnet1 `  
    -AddressPrefix 10.0.0.0/24 `  
    -VirtualNetwork $virtualNetwork1
```

Write the subnet configuration to the virtual network with [Set-AzureRmVirtualNetwork](#), which creates the subnet:

```
$virtualNetwork1 | Set-AzureRmVirtualNetwork
```

Create a virtual network with a 10.1.0.0/16 address prefix and one subnet:

```
# Create the virtual network.  
$virtualNetwork2 = New-AzureRmVirtualNetwork `  
    -ResourceGroupName myResourceGroup `  
    -Location EastUS `  
    -Name myVirtualNetwork2 `  
    -AddressPrefix 10.1.0.0/16  
  
# Create the subnet configuration.  
$subnetConfig = Add-AzureRmVirtualNetworkSubnetConfig `  
    -Name Subnet1 `  
    -AddressPrefix 10.1.0.0/24 `  
    -VirtualNetwork $virtualNetwork2  
  
# Write the subnet configuration to the virtual network.  
$virtualNetwork2 | Set-AzureRmVirtualNetwork
```

Peer virtual networks

Create a peering with [Add-AzureRmVirtualNetworkPeering](#). The following example peers *myVirtualNetwork1* to *myVirtualNetwork2*.

```
Add-AzureRmVirtualNetworkPeering `  
    -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-AzureRmVirtualNetworkPeering `  
    -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-AzureRmVirtualNetworkPeering](#).

```
Get-AzureRmVirtualNetworkPeering ` 
-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-AzureRmVm](#). 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-AzureRmVm ` 
-ResourceGroupName "myResourceGroup" ` 
-Location "East US" ` 
-VirtualNetworkName "myVirtualNetwork1" ` 
-SubnetName "Subnet1" ` 
-ImageName "Win2016Datacenter" ` 
-Name "myVm1" ` 
-AsJob
```

Create the second VM

```
New-AzureRmVm ` 
-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-AzureRmPublicIpAddress](#) to return the public IP address of a VM. The following example returns the public IP address of the *myVm1* VM:

```
Get-AzureRmPublicIpAddress ` 
-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-AzureRmResourcegroup](#) to remove the resource group and all of the resources it contains.

```
Remove-AzureRmResourceGroup -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

4/9/2018 • 5 min 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.

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the **Copy** button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select Try It in the upper-right corner of a code block.	
Open Cloud Shell in your browser.	
Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal.	

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 2.0](#).

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-id $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-id $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

4/18/2018 • 16 min 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](#).

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.

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. 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**.
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. The list of users shown is from the same Azure Active Directory tenant as the virtual network you're setting up the peering for. If you don't see UserB, it's likely because UserB is in a different Active Directory tenant than UserA. If you want to connect virtual networks in different Active Directory tenants, you can connect them with an [Azure VPN Gateway](#), rather than a virtual network peering.
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//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//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. Both of the virtual networks that you want to peer must be in subscriptions associated to the same Azure Active Directory tenant. If you want to connect virtual networks in different Active Directory tenants, you can connect them with an [Azure VPN Gateway](#), rather than a virtual network peering.

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 2.0](#).
- Works in a Bash shell. For options on running Azure CLI scripts on Windows client, see [Running the Azure CLI in 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`.

- 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.

- 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.
- 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

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. Both of the virtual networks that you want to peer must be in subscriptions associated to the same Azure Active Directory tenant. If you want to connect virtual networks in different Active Directory tenants, you can connect them with an [Azure VPN Gateway](#), rather than a virtual network peering.

- Install the latest version of the PowerShell `AzureRm` module. If you're new to Azure PowerShell, see [Azure PowerShell overview](#).
- Start a PowerShell session.
- In PowerShell, log in to Azure as UserA by entering the `Connect-AzureRmAccount` 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](#).
- 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-AzureRmSubscription` 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-AzureRmResourceGroup ` 
    -Name MyResourceGroupA ` 
    -Location eastus

# Create virtual network A.
$vNetA = New-AzureRmVirtualNetwork ` 
    -ResourceGroupName MyResourceGroupA ` 
    -Name 'myVnetA' ` 
    -AddressPrefix '10.0.0.0/16' ` 
    -Location eastus

# Assign UserB permissions to myVnetA.
New-AzureRmRoleAssignment ` 
    -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 it in to PowerShell, and then press **Enter**.

```

# Peer myVnetA to myVnetB.
$vNetA=Get-AzureRmVirtualNetwork -Name myVnetA -ResourceGroupName myResourceGroupA
Add-AzureRmVirtualNetworkPeering ` 
    -Name 'myVnetAToMyVnetB' ` 
    -VirtualNetwork $vNetA ` 
    -RemoteVirtualNetworkId "/subscriptions/<SubscriptionB-Id>/resourceGroups/myResourceGroupB/providers/Microsoft.Network/virtualNetworks/myVnetB"

```

9. View the peering state of myVnetA.

```

Get-AzureRmVirtualNetworkPeering ` 
    -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-AzureRmResourceGroup -Name myResourceGroupA -force
```

2. Log out of Azure as UserA and log in as UserB.
3. Execute the following command:

```
Remove-AzureRmResourceGroup -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

4/18/2018 • 10 min 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 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](#).

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. Log in to the [Azure portal](#). 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:** *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 contains the two virtual networks created in previous steps.
 8. Click **myVNet1**.
 9. In the **myVnet1** blade that appears, click **Peering** from the vertical list of options on the left side of the blade.
 10. In the **myVnet1 - Peering** 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 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 the [Delete resources](#) section of this article.

Create peering - Azure CLI

1. [Install](#) the Azure CLI 1.0 to create the virtual network (classic).
2. Open a command session and log in to Azure using the `azure login` command.
3. Run the CLI in Service Management mode by entering the `azure config mode asm` command.
4. 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"
```

5. Create a resource group and a virtual network (Resource Manager). You can use either the CLI 1.0 or 2.0 ([install](#)). In this tutorial, the CLI 2.0 is used to create the virtual network (Resource Manager), since 2.0 must be used to create the peering. Execute the following bash CLI script from your local machine with the CLI 2.0.4 or later installed. For options on running bash CLI scripts on Windows client, see [Running the Azure CLI in Windows](#). You can also run the script using 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. Click the **Try it** button in the script that follows, which invokes a Cloud Shell that logs you can log in to your Azure account with. To execute the script, click the **Copy** button and paste the contents into your Cloud Shell, then press **Enter**.

```
#!/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
```

6. 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 `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
```

7. 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 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](#).

8. **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.

9. **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 [AzureRm](#) modules. If you're new to Azure PowerShell, see [Azure PowerShell overview](#).
2. Start a PowerShell session.
3. In PowerShell, log in to Azure 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="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. Log in to Azure to create the virtual network (Resource Manager) by entering the `Connect-AzureRmAccount` 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](#).
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-AzureRmResourceGroup -Name myResourceGroup -Location eastus

# Create the virtual network (Resource Manager).
$vnet1 = New-AzureRmVirtualNetwork ` 
-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-AzureRmSubscription` 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-AzureRmVirtualNetworkPeering `  
-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-AzureRmVirtualNetworkPeering `  
-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 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](#).

9. **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.
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 2.0 to delete the virtual network (Resource Manager) with the following command:

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

2. Use the Azure CLI 1.0 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-AzureRmResourceGroup -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

4/18/2018 • 14 min 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](#).

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**.
 - In the **Add permissions** 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//resourceGroups/myResoureGroupB/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.
 - In the **myVnetA - Peerings** blade that appeared, click **+ Add**
 - 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](#).
 - 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.

1. [Install](#) the Azure CLI 1.0 to create the virtual network (classic).
2. Open a CLI session and log in to Azure as UserB using the `azure 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](#).
3. Run the CLI in Service Management mode by entering the `azure config mode asm` command.
4. Enter the following command to create the virtual network (classic):

```
azure network vnet create --vnet myVnetB --address-space 10.1.0.0 --cidr 16 --location "East US"
```

5. The remaining steps must be completed using a bash shell with the Azure CLI 2.0.4 or later [installed](#), or by using 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. Click the **Try it** button in the scripts that follow, which opens a Cloud Shell that logs you in to your Azure account. For options on running bash CLI scripts on a Windows client, see [Running the Azure CLI in Windows](#).
6. 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 2.0 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.

7. Log UserB out of Azure and log in as UserA in the CLI 2.0.
8. 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

```

9. 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

```

10. 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](#).

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.

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 [AzureRm](#) 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-AzureRmAccount` 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-AzureRmSubscription` 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-AzureRmRoleAssignment ` 
-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-AzureRmAccount` 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'
$location='eastus'

# Create a resource group.
New-AzureRmResourceGroup ` 
-Name $rgName ` 
-Location $location

# Create virtual network A.
$vnetA = New-AzureRmVirtualNetwork ` 
-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-AzureRmSubscription` 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-AzureRmRoleAssignment ` 
-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-AzureRmVirtualNetworkPeering ` 
-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-AzureRmVirtualNetworkPeering ` 
-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 2.0 to delete the virtual network (Resource Manager) with the following command:

```
az group delete --name myResourceGroupA --yes
```

2. Log in to Azure using the Azure CLI 1.0 to delete the virtual network (classic) with the following commands:

```
azure config mode asm

azure network vnet delete --vnet myVnetB --quiet
```

PowerShell

1. At the PowerShell command prompt, enter the following command to delete the virtual network (Resource Manager):

```
Remove-AzureRmResourceGroup -Name myResourceGroupA -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="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 a VM with a static public IP address using the Azure portal

4/16/2018 • 2 min to read • [Edit Online](#)

You can create virtual machines (VMs) in Azure and expose them to the public Internet by using a public IP address. By default, Public IPs are dynamic and the address associated to them may change when the VM is deleted or stopped/deallocated. To guarantee that the VM always uses the same public IP address, you need to create a static Public IP.

Before you can implement static Public IPs in VMs, it is necessary to understand when you can use static Public IPs, and how they are used. Read the [IP addressing overview](#) to learn more about IP addressing in Azure.

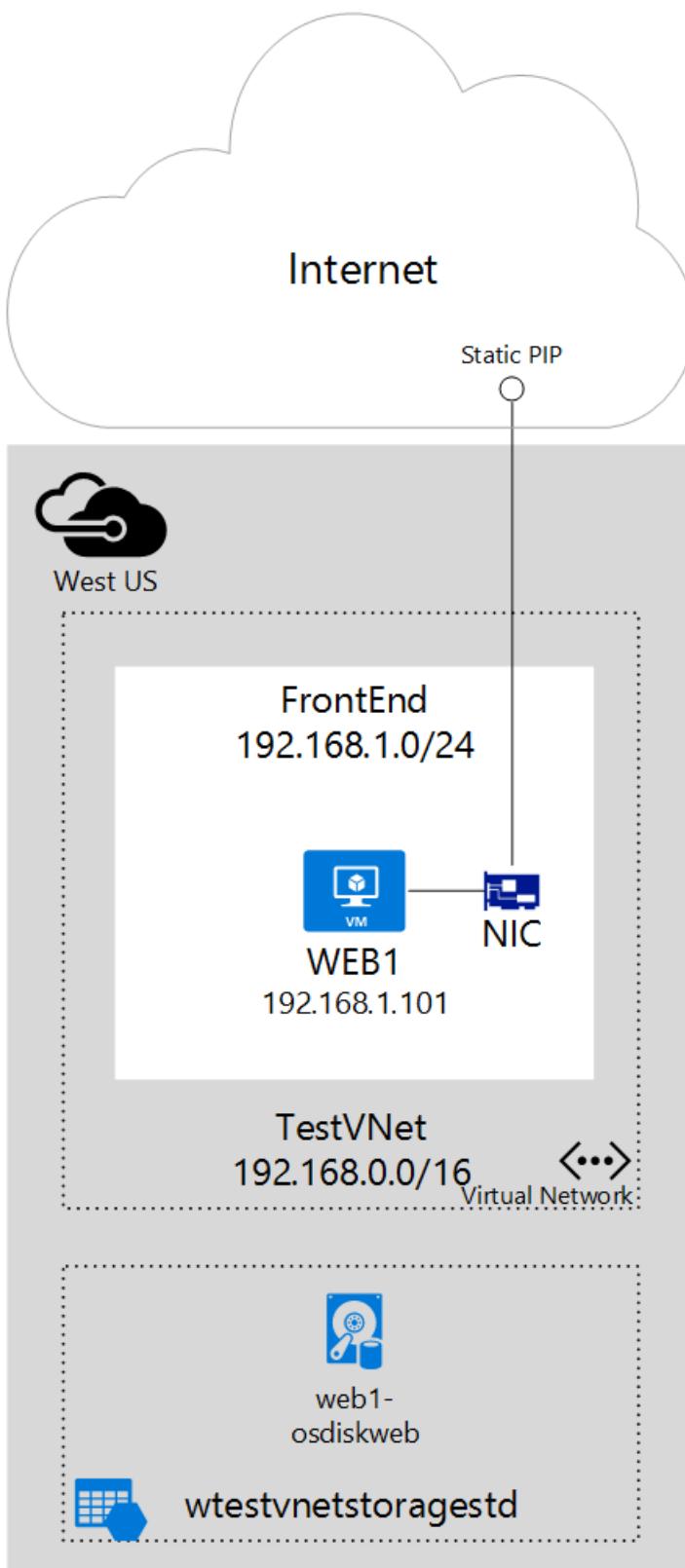
NOTE

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.

Scenario

This document will walk through a deployment that uses a static public IP address allocated to a virtual machine (VM). In this scenario, you have a single VM with its own static public IP address. The VM is part of a subnet named **FrontEnd** and also has a static private IP address (**192.168.1.101**) in that subnet.

You may need a static IP address for web servers that require SSL connections in which the SSL certificate is linked to an IP address.



You can follow the steps below to deploy the environment shown in the figure above.

Create a VM with a static public IP

To create a VM with a static public IP address in the Azure portal, complete the following steps:

1. From a browser, navigate to the [Azure portal](#) and, if necessary, sign in with your Azure account.
2. On the top left-hand corner of the portal, click **Create a resource** > **Compute** > **Windows Server 2012 R2 Datacenter**.
3. In the **Select a deployment model** list, select **Resource Manager** and click **Create**.
4. In the **Basics** pane, enter the VM information as follows, and then click **OK**.

* Name
WEB1 ✓

* User name
adminUser ✓

* Password
***** ✓

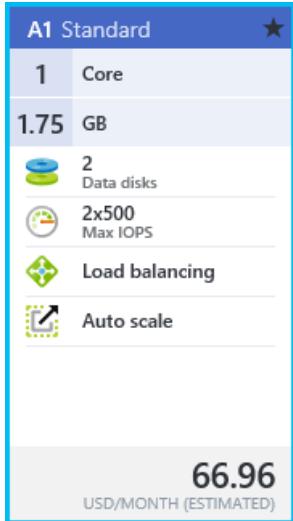
* Subscription
Microsoft Azure Internal Consumption

* Resource group
PublicIPTest ✗ ✓

Select existing

* Location
East US 2

- In the **Choose a size** pane, click **A1 Standard** as follows, and then click **Select**.



- In the **Settings** pane, click **Public IP address**, then in the **Create public IP address** pane, under **Assignment**, click **Static** as follows. And, then click **OK**.

* Name
WEB1 ✓

Assignment
 Dynamic Static

- In the **Settings** pane, click **OK**.
- Review the **Summary** pane, as follows, and then click **OK**.

Basics	
Subscription	Microsoft Azure Internal Consumption
Resource group	(new) PublicIPTest
Location	East US 2
Settings	
Computer name	WEB1
User name	adminUser
Size	Standard A1
Disk type	Standard
Storage account	(new) publiciptest3109
Virtual network	(new) PublicIPTest
Subnet	(new) default (10.6.0.0/24)
Public IP address	(new) WEB1
Network security group	(new) WEB1
Availability set	None
Diagnostics	Enabled
Diagnostics storage account	(new) publiciptest3109

9. Notice the new tile in your dashboard.



10. Once the VM is created, the **Settings** pane displays as follows:

The screenshot shows the Azure portal interface for a virtual machine named 'WEB1'. The left pane displays the 'Essentials' section with basic information like Resource group (PublicIPTest), Status (Running), Location (East US 2), Subscription name (Microsoft Azure Internal Consumption), and Subscription ID (628dad04-b5d1-4f10-b3a4-dc61d88cf97c). It also shows the computer name (WEB1), size (Standard A1), operating system (Windows), and public IP address (104.208.232.131). The right pane is titled 'Settings' and contains sections for INVESTIGATE (Audit logs, Boot diagnostics, Reset password, Troubleshoot, New support request), MANAGE (Properties, Disks, Network interfaces, Availability set, Extensions, Size), MONITOR (Alert rules, Diagnostics), and RESOURCE MANAGEMENT (Users, Tags). Below the settings, there's a 'Monitoring' section showing a chart of CPU utilization over time, with a value of 0.08% displayed prominently.

Set IP addresses within the operating system

You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's 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.

Next steps

Any network traffic can flow to and from the VM created in this article. You can define inbound and outbound security rules within a network security group that limit the traffic that can flow to and from the network interface, the subnet, or both. To learn more about network security groups, see [Network security group overview](#).

Create a VM with a static public IP address using PowerShell

4/16/2018 • 4 min to read • [Edit Online](#)

You can create virtual machines (VMs) in Azure and expose them to the public Internet by using a public IP address. By default, Public IPs are dynamic and the address associated to them may change when the VM is deleted or stopped/deallocated. To guarantee that the VM always uses the same public IP address, you need to create a static Public IP.

Before you can implement static Public IPs in VMs, it is necessary to understand when you can use static Public IPs, and how they are used. Read the [IP addressing overview](#) to learn more about IP addressing in Azure.

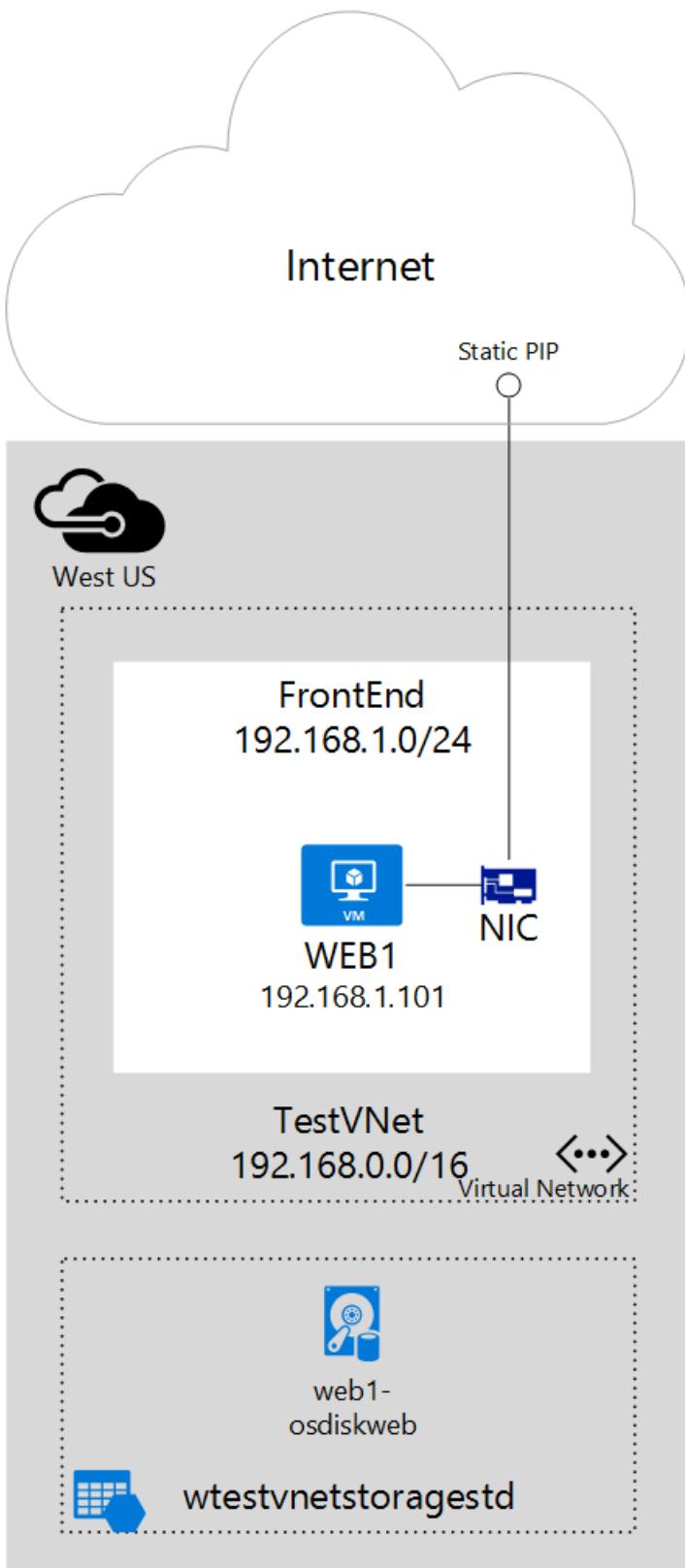
NOTE

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.

Scenario

This document will walk through a deployment that uses a static public IP address allocated to a virtual machine (VM). In this scenario, you have a single VM with its own static public IP address. The VM is part of a subnet named **FrontEnd** and also has a static private IP address (**192.168.1.101**) in that subnet.

You may need a static IP address for web servers that require SSL connections in which the SSL certificate is linked to an IP address.



You can follow the steps below to deploy the environment shown in the figure above.

Prerequisite: Install the Azure PowerShell module

To perform the steps in this article, you need to [install and configure the Azure PowerShell module](#). Be sure to complete all of the instructions. After the installation is finished, sign in to Azure and select your subscription.

NOTE

You need an Azure account to complete these steps. If you don't have an Azure account, you can sign up for a [free trial](#).

Start your script

You can download the full PowerShell script used [here](#). Follow the steps below to change the script to work in your environment.

Change the values of the variables below based on the values you want to use for your deployment. The following values map to the scenario used in this article:

```
# Set variables resource group
$rgName          = "IaaSStory"
.setLocation      = "West US"

# Set variables for VNet
$vnetName        = "WTestVNet"
$vnetPrefix      = "192.168.0.0/16"
$subnetName      = "FrontEnd"
$subnetPrefix    = "192.168.1.0/24"

# Set variables for storage
$stdStorageAccountName = "iaasstorystorage"

# Set variables for VM
$vmSize          = "Standard_A1"
$diskSize         = 127
$publisher        = "MicrosoftWindowsServer"
$offer            = "WindowsServer"
$sku              = "2012-R2-Datacenter"
$version          = "latest"
$vmName           = "WEB1"
$osDiskName       = "osdisk"
$nicName          = "NICWEB1"
$privateIPAddress = "192.168.1.101"
$pipName          = "PIPWEB1"
$dnsName          = "iaasstoryws1"
```

Create the necessary resources for your VM

Before creating a VM, you need a resource group, VNet, public IP, and NIC to be used by the VM.

1. Create a new resource group.

```
New-AzureRmResourceGroup -Name $rgName -Location $location
```

2. Create the VNet and subnet.

```
$vnet = New-AzureRmVirtualNetwork -ResourceGroupName $rgName -Name $vnetName ` 
    -AddressPrefix $vnetPrefix -Location $location

Add-AzureRmVirtualNetworkSubnetConfig -Name $subnetName ` 
    -VirtualNetwork $vnet -AddressPrefix $subnetPrefix

Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

3. Create the public IP resource.

```
$pip = New-AzureRmPublicIpAddress -Name $pipName -ResourceGroupName $rgName ` 
    -AllocationMethod Static -DomainNameLabel $dnsName -Location $location
```

4. Create the network interface (NIC) for the VM in the subnet created above, with the public IP. Notice the

first cmdlet retrieving the VNet from Azure, this is necessary since a `Set-AzureRmVirtualNetwork` was executed to change the existing VNet.

```
$vnet = Get-AzureRmVirtualNetwork -Name $vnetName -ResourceGroupName $rgName  
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet -Name $subnetName  
$nic = New-AzureRmNetworkInterface -Name $nicName -ResourceGroupName $rgName  
    -Subnet $subnet -Location $location -PrivateIpAddress $privateIPAddress  
    -PublicIpAddress $pip
```

5. Create a storage account to host the VM OS drive.

```
$stdStorageAccount = New-AzureRmStorageAccount -Name $stdStorageAccountName  
    -ResourceGroupName $rgName -Type Standard_LRS -Location $location
```

Create the VM

Now that all necessary resources are in place, you can create a new VM.

1. Create the configuration object for the VM.

```
$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmSize
```

2. Get credentials for the VM local administrator account.

```
$cred = Get-Credential -Message "Type the name and password for the local administrator account."
```

3. Create a VM configuration object.

```
$vmConfig = Set-AzureRmVMOperatingSystem -VM $vmConfig -Windows -ComputerName $vmName  
    -Credential $cred -ProvisionVMAgent -EnableAutoUpdate
```

4. Set the operating system image for the VM.

```
$vmConfig = Set-AzureRmVMSourceImage -VM $vmConfig -PublisherName $publisher  
    -Offer $offer -Skus $sku -Version $version
```

5. Configure the OS disk.

```
$osVhdUri = $stdStorageAccount.PrimaryEndpoints.Blob.ToString() + "vhds/" + $osDiskName + ".vhf"  
$vmConfig = Set-AzureRmVMOSDisk -VM $vmConfig -Name $osDiskName -VhdUri $osVhdUri -CreateOption  
fromImage
```

6. Add the NIC to the VM.

```
$vmConfig = Add-AzureRmVMNetworkInterface -VM $vmConfig -Id $nic.Id -Primary
```

7. Create the VM.

```
New-AzureRmVM -VM $vmConfig -ResourceGroupName $rgName -Location $location
```

8. Save the script file.

Run the script

After making any necessary changes, run the previous script. The virtual machine is created after a few minutes.

Set IP addresses within the operating system

You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's 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.

Next steps

Any network traffic can flow to and from the VM created in this article. You can define inbound and outbound security rules within a network security group that limit the traffic that can flow to and from the network interface, the subnet, or both. To learn more about network security groups, see [Network security group overview](#).

Create a VM with a static public IP address using the Azure CLI

4/16/2018 • 5 min to read • [Edit Online](#)

You can create virtual machines (VMs) in Azure and expose them to the public Internet by using a public IP address. By default, Public IPs are dynamic and the address associated to them may change when the VM is deleted or stopped/deallocated. To guarantee that the VM always uses the same public IP address, you need to create a static Public IP.

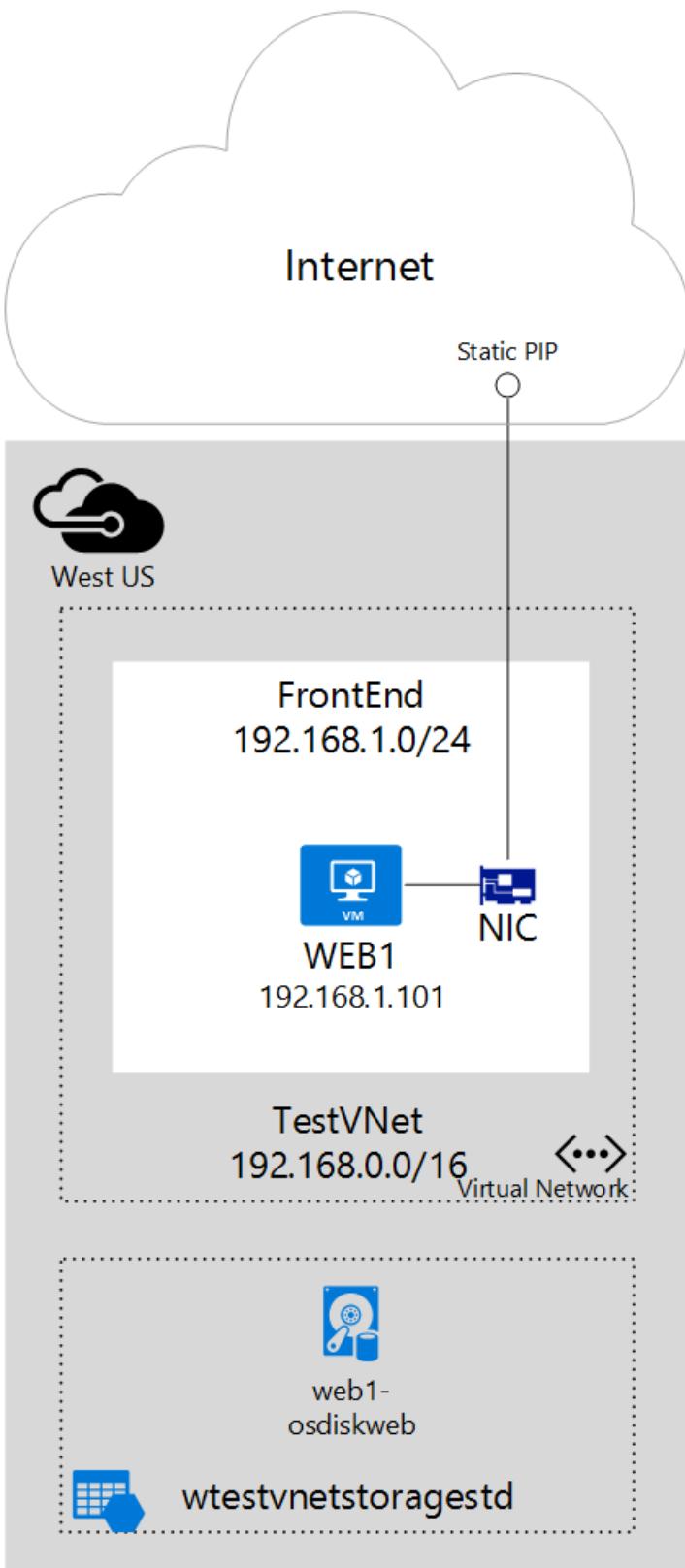
Before you can implement static Public IPs in VMs, it is necessary to understand when you can use static Public IPs, and how they are used. Read the [IP addressing overview](#) to learn more about IP addressing in Azure.

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.

Scenario

This document will walk through a deployment that uses a static public IP address allocated to a virtual machine (VM). In this scenario, you have a single VM with its own static public IP address. The VM is part of a subnet named **FrontEnd** and also has a static private IP address (**192.168.1.101**) in that subnet.

You may need a static IP address for web servers that require SSL connections in which the SSL certificate is linked to an IP address.



You can follow the steps below to deploy the environment shown in the figure above.

Create the VM

The values in "" for the variables in the steps that follow create resources with settings from the scenario. Change the values, as appropriate, for your environment.

1. Install the [Azure CLI 2.0](#) 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`.

4. Create the VM by executing the script that follows on a Linux or Mac computer. The Azure public IP address, virtual network, network interface, and VM resources must all exist in the same location. Though the resources don't all have to exist in the same resource group, in the following script they do.

```
RgName="IaaSStory"
Location="westus"

# Create a resource group.

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. The DnsName must be unique within the
# Azure location it's created in. 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="PIPWEB1"
DnsName="iaasstoryws1"
az network public-ip create \
--name $PipName \
--resource-group $RgName \
--location $Location \
--allocation-method Static \
--dns-name $DnsName

# Create a virtual network with one subnet

VnetName="TestVNet"
VnetPrefix="192.168.0.0/16"
SubnetName="FrontEnd"
SubnetPrefix="192.168.1.0/24"
az network vnet create \
--name $VnetName \
--resource-group $RgName \
--location $Location \
--address-prefix $VnetPrefix \
--subnet-name $SubnetName \
--subnet-prefix $SubnetPrefix

# Create a network interface connected to the VNet with a static private IP address and associate the public
# IP address
# resource to the NIC.

NicName="NICWEB1"
PrivateIpAddress="192.168.1.101"
az network nic create \
--name $NicName \
--resource-group $RgName \
--location $Location \
--subnet $SubnetName \
--vnet-name $VnetName \
--private-ip-address $PrivateIpAddress \
--public-ip-address $PipName

# Create a new VM with the NIC

VmName="WEB1"

# Replace the value for the VmSize variable with a value from the
# https://docs.microsoft.com/azure/virtual-machines/virtual-machines-linux-sizes article.
VmSize="Standard_DS1"

# Replace the value for the OsImage variable 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.
$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
# If creating a Windows VM, remove the previous line and you'll be prompted for the password you want to
configure for the VM.

```

In addition to creating a VM, 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 2.0](#) article for details.
- Virtual network, subnet, NIC, and public IP address resources. 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`.

Validate VM creation and public IP address

1. Enter the command `az resource list --resource-group IaaSStory --output table` to see a list of the resources created by the script. There should be five resources in the returned output: network interface, disk, public IP address, virtual network, and a virtual machine.
2. Enter the command `az network public-ip show --name PIPWEB1 --resource-group IaaSStory --output table`. In the returned output, note the value of **IpAddress** and that the value of **PublicIpAllocationMethod** is *Static*.
3. Before executing the following command, remove the <>, replace *Username* with the name you used for the **Username** variable in the script, and replace *ipAddress* with the **IpAddress** from the previous step. Run the following command to connect to the VM: `ssh -i ~/.ssh/azure_id_rsa <Username>@<ipAddress>`.

Remove the VM and associated resources

It's recommended that you delete the resources created in this exercise if you won't use them in production. VM, public IP address, and disk resources incur charges, as long as they're provisioned. To remove the resources created during this exercise, complete the following steps:

1. To view the resources in the resource group, run the `az resource list --resource-group IaaSStory` command.
2. Confirm there are no resources in the resource group, other than the resources created by the script in this article.
3. To delete all resources created in this exercise, run the `az group delete -n IaaSStory` command. The command deletes the resource group and all the resources it contains.

Set IP addresses within the operating system

You should never manually assign the public IP address assigned to an Azure virtual machine within the virtual machine's 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.

Next steps

Any network traffic can flow to and from the VM created in this article. You can define inbound and outbound security rules within a network security group that limit the traffic that can flow to and from the network interface, the subnet, or both. To learn more about network security groups, see [Network security group overview](#).

Configure private IP addresses for a virtual machine using the Azure portal

4/11/2018 • 5 min to read • [Edit Online](#)

Your IaaS virtual machines (VMs) and PaaS role instances in a virtual network automatically receive a private IP address from a range that you specify, based on the subnet they are connected to. That address is retained by the VMs and role instances, until they are decommissioned. You decommission a VM or role instance by stopping it from PowerShell, the Azure CLI, or the Azure portal. In those cases, once the VM or role instance starts again, it will receive an available IP address from the Azure infrastructure, which might not be the same it previously had. If you shut down the VM or role instance from the guest operating system, it retains the IP address it had.

In certain cases, you want a VM or role instance to have a static IP address, for example, if your VM is going to run DNS or will be a domain controller. You can do so by setting a static private IP address.

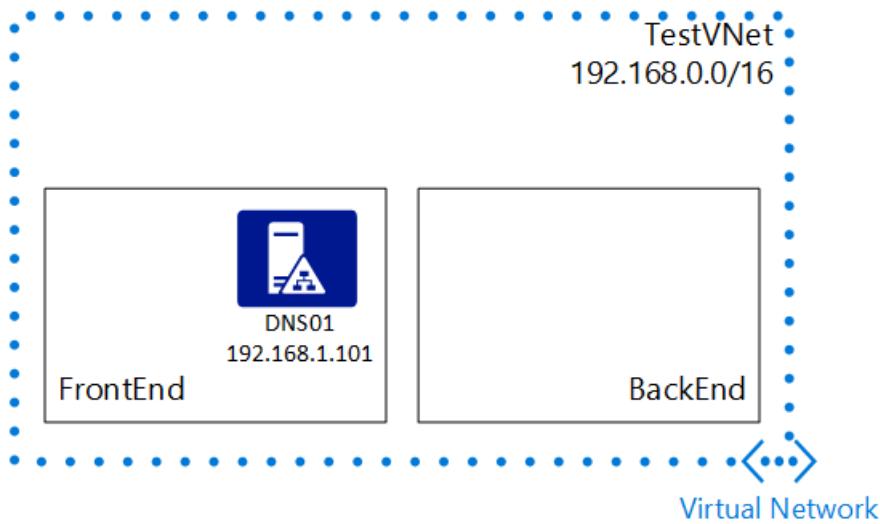
IMPORTANT

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**.

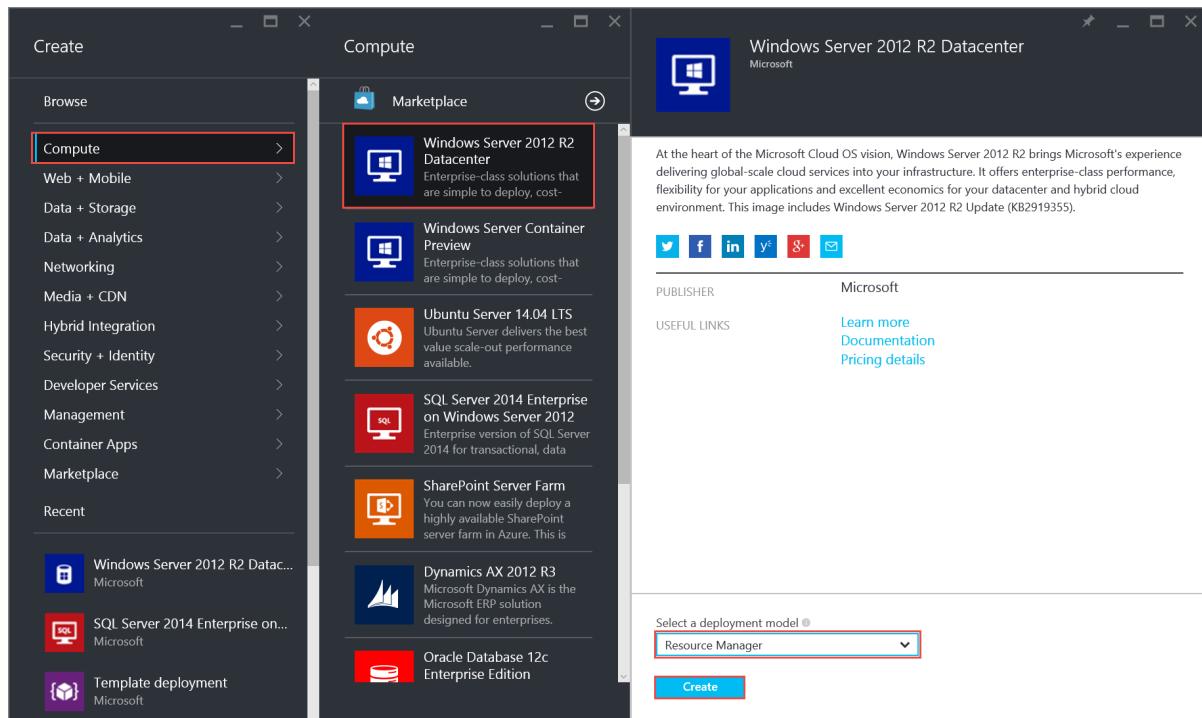
The following sample steps expect a simple environment already created. If you want to run the steps as they are displayed in this document, first build the test environment described in [Create a virtual network](#).

How to create a VM for testing static private IP addresses

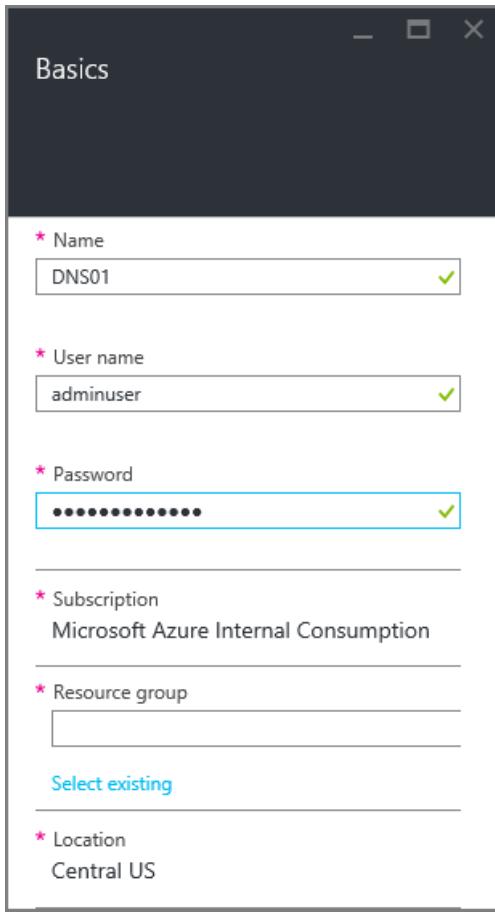
You cannot set a static private IP address during the creation of a VM in the Resource Manager deployment mode by using the Azure portal. You must create the VM first, then set its private IP to be static.

To create a VM named *DNS01* in the *FrontEnd* subnet of a VNet named *TestVNet*, follow these steps:

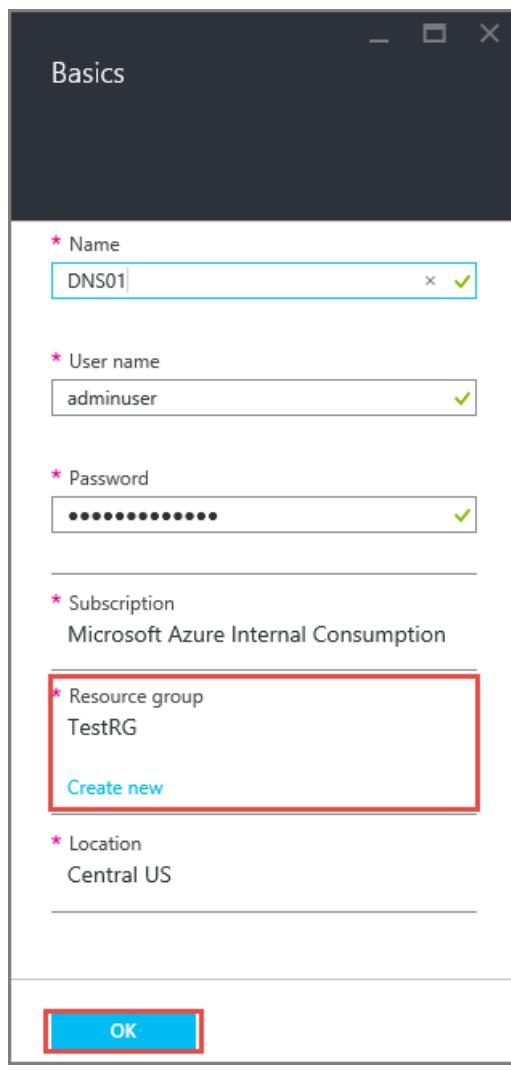
1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Click **Create a resource** > **Compute** > **Windows Server 2012 R2 Datacenter**, notice that the **Select a deployment model** list already shows **Resource Manager**, and then click **Create**, as seen in the following figure.



3. 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.



4. 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**.



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

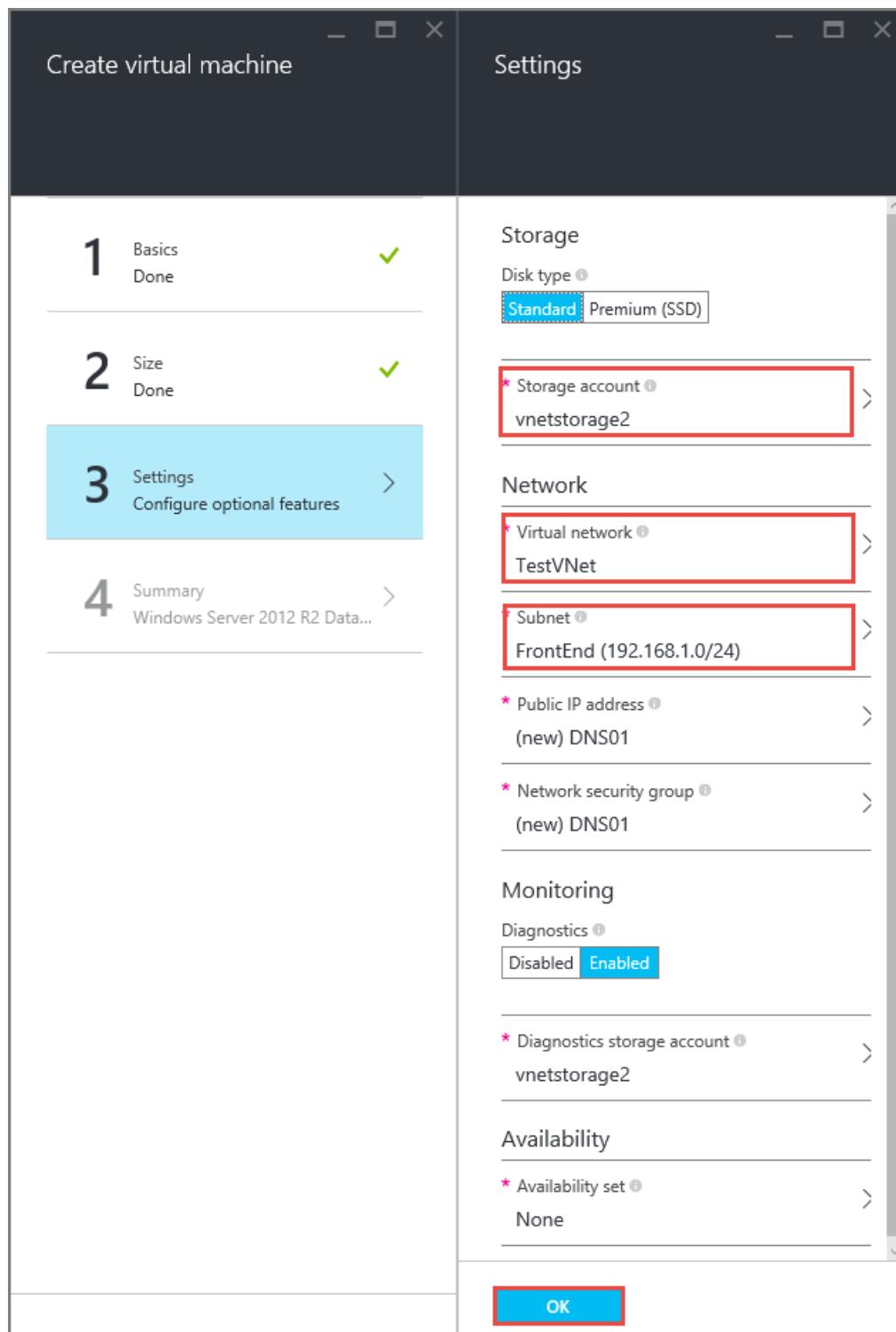
D1 Standard	A1 Standard
1 Core	1 Core
3.5 GB	1.75 GB
2 Data disks	2 Data disks
2x500 Max IOPS	2x500 Max IOPS
50 GB Local SSD	Load balancing
Load balancing	Auto scale
Auto scale	
118.30 USD/MONTH (ESTIMATED)	66.96 USD/MONTH (ESTIMATED)

Select

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.

NAME	PUBLIC IP ADDRESS	PRIVATE IP ADDRESS	SECURITY GROUP
dns01995	40.122.203.66	192.168.1.6	DNS01

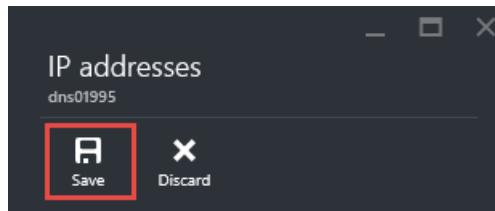
2. In the **Network interface** pane, click **All settings > IP addresses** and notice the **Assignment** and **IP address** values.

The screenshot shows the Azure portal interface for managing a network interface named "dns01995". The main pane displays the "Essentials" and "Operations" sections. The "Operations" section shows 4 events for the VM. The "IP addresses" pane is open, showing the "IP addresses" settings for the "dns01995" interface. It details the Public IP address settings (Enabled) and the Private IP address settings (Virtual network: TestVNet, Subnet: FrontEnd (192.168.1.0/24)). The "Assignment" dropdown is set to "Static", and the IP address is listed as "192.168.1.6". A red box highlights the "Assignment" dropdown and the IP address input field.

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](#).

Configure private IP addresses for a virtual machine using PowerShell

4/11/2018 • 6 min to read • [Edit Online](#)

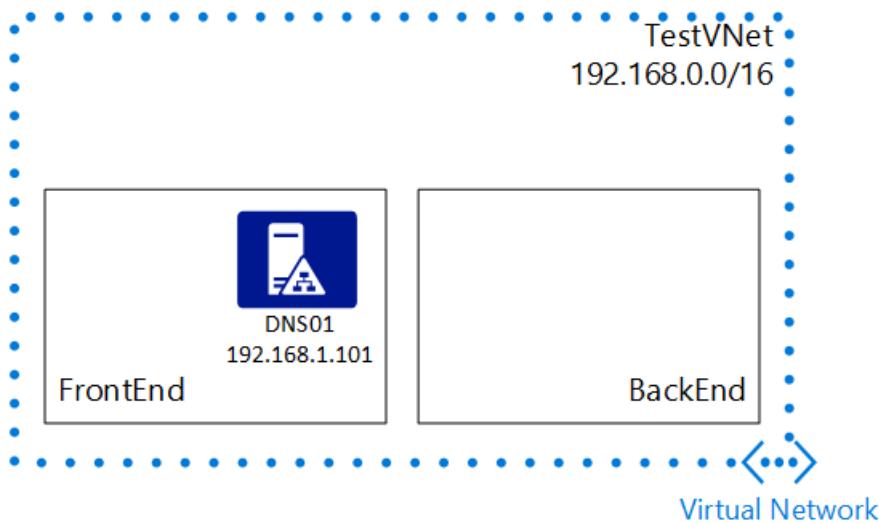
Your IaaS virtual machines (VMs) and PaaS role instances in a virtual network automatically receive a private IP address from a range that you specify, based on the subnet they are connected to. That address is retained by the VMs and role instances, until they are decommissioned. You decommission a VM or role instance by stopping it from PowerShell, the Azure CLI, or the Azure portal. In those cases, once the VM or role instance starts again, it will receive an available IP address from the Azure infrastructure, which might not be the same it previously had. If you shut down the VM or role instance from the guest operating system, it retains the IP address it had.

In certain cases, you want a VM or role instance to have a static IP address, for example, if your VM is going to run DNS or will be a domain controller. You can do so by setting a static private IP address.

Azure has two deployment models: Azure Resource Manager and classic. Microsoft recommends creating resources through the Resource Manager deployment model. To learn more about the differences between the two models, read the [Understand Azure deployment models](#) 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 sample PowerShell commands below expect a simple environment already created based on the scenario above. If you want to run the commands as they are displayed in this document, first build the test environment described in [Create a virtual network](#).

Create a VM with a static private IP address

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*, follow the steps below:

1. Set variables for the storage account, location, resource group, and credentials to be used. You will need to

enter a user name and password for the VM. The storage account and resource group must already exist.

```
$stName = "vnetstorage"
$locName = "Central US"
$rgName = "TestRG"
$cred = Get-Credential -Message "Type the name and password of the local administrator account."
```

2. Retrieve the virtual network and subnet you want to create the VM in.

```
$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName TestRG -Name TestVNet
$subnet = $vnet.Subnets[0].Id
```

3. If necessary, create a public IP address to access the VM from the Internet.

```
$pip = New-AzureRmPublicIpAddress -Name TestPIP -ResourceGroupName $rgName ` 
-Location $locName -AllocationMethod Dynamic
```

4. Create a NIC using the static private IP address you want to assign to the VM. Make sure the IP is from the subnet range you are adding the VM to. This is the main step for this article, where you set the private IP to be static.

```
$nic = New-AzureRmNetworkInterface -Name TestNIC -ResourceGroupName $rgName ` 
-Location $locName -SubnetId $vnet.Subnets[0].Id -PublicIpAddressId $pip.Id ` 
-PrivateIpAddress 192.168.1.101
```

5. Create the VM with the NIC:

```
$vm = New-AzureRmVMConfig -VMName DNS01 -VMSize "Standard_A1"
$vm = Set-AzureRmVMOperatingSystem -VM $vm -Windows -ComputerName DNS01 ` 
-Credential $cred -ProvisionVMAgent -EnableAutoUpdate
$vm = Set-AzureRmVMSourceImage -VM $vm -PublisherName MicrosoftWindowsServer ` 
-Offer WindowsServer -Skus 2012-R2-Datacenter -Version "latest"
$vm = Add-AzureRmVMNetworkInterface -VM $vm -Id $nic.Id
$osDiskUri = $storageAcc.PrimaryEndpoints.Blob.ToString() + "vhds/WindowsVMosDisk.vhd"
$vm = Set-AzureRmVMOSDisk -VM $vm -Name "windowsvmosdisk" -VhdUri $osDiskUri ` 
-CreateOption fromImage
New-AzureRmVM -ResourceGroupName $rgName -Location $locName -VM $vm
```

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.

Retrieve static private IP address information for a network interface

To view the static private IP address information for the VM created with the script above, run the following PowerShell command and observe the values for *PrivateIpAddress* and *PrivateIpAllocationMethod*:

```
Get-AzureRmNetworkInterface -Name TestNIC -ResourceGroupName TestRG
```

Expected output:

```

Name : TestNIC
ResourceGroupName : TestRG
Location : centralus
Id :
/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC
Etag : W/"[Id]"
ProvisioningState : Succeeded
Tags :
VirtualMachine : {
    "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/DNS01"
}
IpConfigurations : [
    {
        "Name": "ipconfig1",
        "Etag": "W/"[Id]\\"",
        "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC/ipConfigurations/ipconfig1",
        "PrivateIpAddress": "192.168.1.101",
        "PrivateIpAllocationMethod": "Static",
        "Subnet": {
            "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd"
        },
        "PublicIpAddress": {
            "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/TestPIP"
        },
        "LoadBalancerBackendAddressPools": [],
        "LoadBalancerInboundNatRules": [],
        "ProvisioningState": "Succeeded"
    }
]
DnsSettings : {
    "DnsServers": [],
    "AppliedDnsServers": [],
    "InternalDnsNameLabel": null,
    "InternalFqdn": null
}
EnableIPForwarding : False
NetworkSecurityGroup : null
Primary : True

```

Remove a static private IP address from a network interface

To remove the static private IP address added to the VM in the script above, run the following PowerShell commands:

```

$nic=Get-AzureRmNetworkInterface -Name TestNIC -ResourceGroupName TestRG
$nic.IpConfigurations[0].PrivateIpAllocationMethod = "Dynamic"
Set-AzureRmNetworkInterface -NetworkInterface $nic

```

Expected output:

```

Name : TestNIC
ResourceGroupName : TestRG
Location : centralus
Id :
/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC
Etag : W/"[Id]"
ProvisioningState : Succeeded
Tags :
VirtualMachine : {
    "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/WindowsVM"
}
IpConfigurations : [
    {
        "Name": "ipconfig1",
        "Etag": "W/"[Id]\\"",
        "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC/ipConfigurations/ipconfig1",
        "PrivateIpAddress": "192.168.1.101",
        "PrivateIpAllocationMethod": "Dynamic",
        "Subnet": {
            "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd"
        },
        "PublicIpAddress": {
            "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/TestPIP"
        },
        "LoadBalancerBackendAddressPools": [],
        "LoadBalancerInboundNatRules": [],
        "ProvisioningState": "Succeeded"
    }
]
DnsSettings : {
    "DnsServers": [],
    "AppliedDnsServers": [],
    "InternalDnsNameLabel": null,
    "InternalFqdn": null
}
EnableIPForwarding : False
NetworkSecurityGroup : null
Primary : True

```

Add a static private IP address to a network interface

To add a static private IP address to the VM created using the script above, run the following commands:

```

$nic=Get-AzureRmNetworkInterface -Name TestNIC -ResourceGroupName TestRG
$nic.IpConfigurations[0].PrivateIpAllocationMethod = "Static"
$nic.IpConfigurations[0].PrivateIpAddress = "192.168.1.101"
Set-AzureRmNetworkInterface -NetworkInterface $nic

```

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.

Change the allocation method for a private IP address assigned to a network interface

A private IP address is assigned to a NIC with the static or dynamic allocation method. Dynamic IP addresses can change after starting a VM that was previously in the stopped (deallocated) state. This can potentially cause issues if the VM is hosting a service that requires the same IP address, even after restarts from a stopped (deallocated) state. Static IP addresses are retained until the VM is deleted. To change the allocation method of an IP address, run the following script, which changes the allocation method from dynamic to static. If the allocation method for the current private IP address is static, change *Static* to *Dynamic* before executing the script.

```
$RG = "TestRG"
$NIC_name = "testnic1"

$nic = Get-AzureRmNetworkInterface -ResourceGroupName $RG -Name $NIC_name
$nic.IpConfigurations[0].PrivateIpAllocationMethod = 'Static'
Set-AzureRmNetworkInterface -NetworkInterface $nic
$IP = $nic.IpConfigurations[0].PrivateIpAddress

Write-Host "The allocation method is now set to"$nic.IpConfigurations[0].PrivateIpAllocationMethod"for the IP
address" $IP"." -NoNewline
```

If you don't know the name of the NIC, you can view a list of NICs within a resource group by entering the following command:

```
Get-AzureRmNetworkInterface -ResourceGroupName $RG | Where-Object {$_ .ProvisioningState -eq 'Succeeded'}
```

Next steps

Learn about managing [IP address settings](#).

Configure private IP addresses for a virtual machine using the Azure CLI

4/11/2018 • 5 min to read • [Edit Online](#)

Your IaaS virtual machines (VMs) and PaaS role instances in a virtual network automatically receive a private IP address from a range that you specify, based on the subnet they are connected to. That address is retained by the VMs and role instances, until they are decommissioned. You decommission a VM or role instance by stopping it from PowerShell, the Azure CLI, or the Azure portal. In those cases, once the VM or role instance starts again, it will receive an available IP address from the Azure infrastructure, which might not be the same it previously had. If you shut down the VM or role instance from the guest operating system, it retains the IP address it had.

In certain cases, you want a VM or role instance to have a static IP address, for example, if your VM is going to run DNS or will be a domain controller. You can do so by setting a static private IP address.

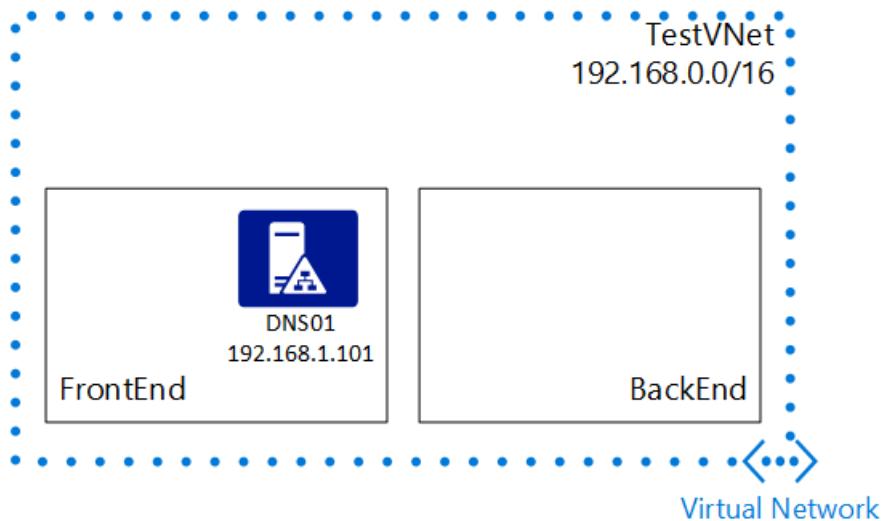
IMPORTANT

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 2.0](#) 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 `az 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:

- 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"
  }
}
```

- Run the **azure vm set** command to change the NIC used by the VM.

```
azure vm set -g TestRG -n DNS01 -N 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](#).

Create and manage a Windows virtual machine that has multiple NICs

4/16/2018 • 8 min 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, or a network dedicated to a monitoring or backup solution. 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

Make sure that you have the [latest Azure PowerShell version installed and configured](#).

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-AzureRmResourceGroup -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-AzureRmVirtualNetworkSubnetConfig](#). The following example defines the subnets for *mySubnetFrontEnd* and *mySubnetBackEnd*:

```
$mySubnetFrontEnd = New-AzureRmVirtualNetworkSubnetConfig -Name "mySubnetFrontEnd" `  
    -AddressPrefix "192.168.1.0/24"  
$mySubnetBackEnd = New-AzureRmVirtualNetworkSubnetConfig -Name "mySubnetBackEnd" `  
    -AddressPrefix "192.168.2.0/24"
```

2. Create your virtual network and subnets with [New-AzureRmVirtualNetwork](#). The following example creates a virtual network named *myVnet*:

```
$myVnet = New-AzureRmVirtualNetwork -ResourceGroupName "myResourceGroup" `  
    -Location "EastUs" `  
    -Name "myVnet" `  
    -AddressPrefix "192.168.0.0/16" `  
    -Subnet $mySubnetFrontEnd,$mySubnetBackEnd
```

Create multiple NICs

Create two NICs with [New-AzureRmNetworkInterface](#). 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-AzureRmNetworkInterface -ResourceGroupName "myResourceGroup" ` 
    -Name "myNic1" ` 
    -Location "EastUs" ` 
    -SubnetId $frontEnd.Id

$backEnd = $myVnet.Subnets | ?{$_ .Name -eq 'mySubnetBackEnd'}
$myNic2 = New-AzureRmNetworkInterface -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-AzureRmVMConfig](#). The following example defines a VM named *myVM* and uses a VM size that supports more than two NICs (*Standard_DS3_v2*):

```
$vmConfig = New-AzureRmVMConfig -VMName "myVM" -VMSize "Standard_DS3_v2"
```

3. Create the rest of your VM configuration with [Set-AzureRmVMOperatingSystem](#) and [Set-AzureRmVMSourceImage](#). The following example creates a Windows Server 2016 VM:

```

$vmConfig = Set-AzureRmVMOperatingSystem -VM $vmConfig ` 
    -Windows ` 
    -ComputerName "myVM" ` 
    -Credential $cred ` 
    -ProvisionVMAgent ` 
    -EnableAutoUpdate
$vmConfig = Set-AzureRmVMSourceImage -VM $vmConfig ` 
    -PublisherName "MicrosoftWindowsServer" ` 
    -Offer "WindowsServer" ` 
    -Skus "2016-Datacenter" ` 
    -Version "latest"

```

4. Attach the two NICs that you previously created with [Add-AzureRmVMNetworkInterface](#):

```

$vmConfig = Add-AzureRmVMNetworkInterface -VM $vmConfig -Id $myNic1.Id -Primary
$vmConfig = Add-AzureRmVMNetworkInterface -VM $vmConfig -Id $myNic2.Id

```

5. Create your VM with [New-AzureRmVM](#):

```
New-AzureRmVM -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-AzureRmVM](#). The following example deallocates the VM named *myVM* in *myResourceGroup*:

```
Stop-AzureRmVM -Name "myVM" -ResourceGroupName "myResourceGroup"
```

2. Get the existing configuration of the VM with [Get-AzureRmVm](#). The following example gets information for the VM named *myVM* in *myResourceGroup*:

```
$vm = Get-AzureRmVm -Name "myVM" -ResourceGroupName "myResourceGroup"
```

3. The following example creates a virtual NIC with [New-AzureRmNetworkInterface](#) named *myNic3* that is attached to *mySubnetBackEnd*. The virtual NIC is then attached to the VM named *myVM* in *myResourceGroup* with [Add-AzureRmVMNetworkInterface](#):

```
# Get info for the back end subnet
$myVnet = Get-AzureRmVirtualNetwork -Name "myVnet" -ResourceGroupName "myResourceGroup"
$backEnd = $myVnet.Subnets | ?{$_ . Name -eq 'mySubnetBackEnd'}`

# Create a virtual NIC
$myNic3 = New-AzureRmNetworkInterface -ResourceGroupName "myResourceGroup" ` 
    -Name "myNic3" ` 
    -Location "EastUs" ` 
    -SubnetId $backEnd.Id

# Get the ID of the new virtual NIC and add to VM
$nicId = (Get-AzureRmNetworkInterface -ResourceGroupName "myResourceGroup" -Name "MyNic3").Id
Add-AzureRmVMNetworkInterface -VM $vm -Id $nicId | Update-AzureRmVm -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-AzureRmVM -VM $vm -ResourceGroupName "myResourceGroup"
```

4. Start the VM with [Start-AzureRmVm](#):

```
Start-AzureRmVM -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-AzureRmVM](#). The following example deallocates the VM named *myVM* in *myResourceGroup*:

```
Stop-AzureRmVM -Name "myVM" -ResourceGroupName "myResourceGroup"
```

2. Get the existing configuration of the VM with [Get-AzureRmVm](#). The following example gets information for the VM named *myVM* in *myResourceGroup*:

```
$vm = Get-AzureRmVm -Name "myVM" -ResourceGroupName "myResourceGroup"
```

3. Get information about the NIC remove with [Get-AzureRmNetworkInterface](#). 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-AzureRmNetworkInterface -ResourceGroupName "myResourceGroup" -Name "myNic3").Id
```

4. Remove the NIC with [Remove-AzureRmVMNetworkInterface](#) and then update the VM with [Update-AzureRmVm](#). The following example removes *myNic3* as obtained by `$nicId` in the preceding step:

```
Remove-AzureRmVMNetworkInterface -VM $vm -NetworkInterfaceIDs $nicId | `  
Update-AzureRmVm -ResourceGroupName "myResourceGroup"
```

5. Start the VM with [Start-AzureRmVm](#):

```
Start-AzureRmVM -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

4/16/2018 • 5 min to read • [Edit Online](#)

You can create a virtual machine (VM) in Azure that has multiple virtual network interfaces (NICs) attached to it. A common scenario is to have different subnets for front-end and back-end connectivity, or a network dedicated to a monitoring or backup solution. This article details how to create a VM with multiple NICs attached to it and how to add or remove NICs from an existing VM. Different [VM sizes](#) support a varying number of NICs, so size your VM accordingly.

This article details how to create a VM with multiple NICs with the Azure CLI 2.0. You can also perform these steps with the [Azure CLI 1.0](#).

Create supporting resources

Install the latest [Azure CLI 2.0](#) 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 192.168.0.0/16 \
--subnet-name mySubnetFrontEnd \
--subnet-prefix 192.168.1.0/24
```

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 192.168.2.0/24
```

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 2.0. 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

When you add multiple NICs to a Linux VM, you need to create routing rules. These rules allow the VM to send and receive traffic that belongs to a specific NIC. Otherwise, traffic that belongs to *eth1*, for example, cannot be processed correctly by the defined default route.

To correct this routing issue, first add two routing tables to */etc/iproute2/rt_tables* as follows:

```
echo "200 eth0-rt" >> /etc/iproute2/rt_tables
echo "201 eth1-rt" >> /etc/iproute2/rt_tables
```

To make the change persistent and applied during network stack activation, edit */etc/sysconfig/network-scripts/ifcfg-eth0* and */etc/sysconfig/network-scripts/ifcfg-eth1*. Alter the line "*NM_CONTROLLED=yes*" to "*NM_CONTROLLED=no*". Without this step, the additional rules/routing are not automatically applied.

Next, extend the routing tables. Let's assume we have the following setup in place:

Routing

```
default via 10.0.1.1 dev eth0 proto static metric 100
10.0.1.0/24 dev eth0 proto kernel scope link src 10.0.1.4 metric 100
10.0.1.0/24 dev eth1 proto kernel scope link src 10.0.1.5 metric 101
168.63.129.16 via 10.0.1.1 dev eth0 proto dhcp metric 100
169.254.169.254 via 10.0.1.1 dev eth0 proto dhcp metric 100
```

Interfaces

```
lo: inet 127.0.0.1/8 scope host lo
eth0: inet 10.0.1.4/24 brd 10.0.1.255 scope global eth0
eth1: inet 10.0.1.5/24 brd 10.0.1.255 scope global eth1
```

You would then create the following files and add the appropriate rules and routes to each:

- */etc/sysconfig/network-scripts/rule-eth0*

```
from 10.0.1.4/32 table eth0-rt
to 10.0.1.4/32 table eth0-rt
```

- */etc/sysconfig/network-scripts/route-eth0*

```
10.0.1.0/24 dev eth0 table eth0-rt
default via 10.0.1.1 dev eth0 table eth0-rt
```

- */etc/sysconfig/network-scripts/rule-eth1*

```
from 10.0.1.5/32 table eth1-rt
to 10.0.1.5/32 table eth1-rt
```

- `/etc/sysconfig/network-scripts/route-eth1`

```
10.0.1.0/24 dev eth1 table eth1-rt
default via 10.0.1.1 dev eth1 table eth1-rt
```

To apply the changes, restart the *network* service as follows:

```
systemctl restart network
```

The routing rules are now correctly in place and you can connect with either interface as needed.

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.

Assign multiple IP addresses to virtual machines using the Azure portal

4/16/2018 • 12 min 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.

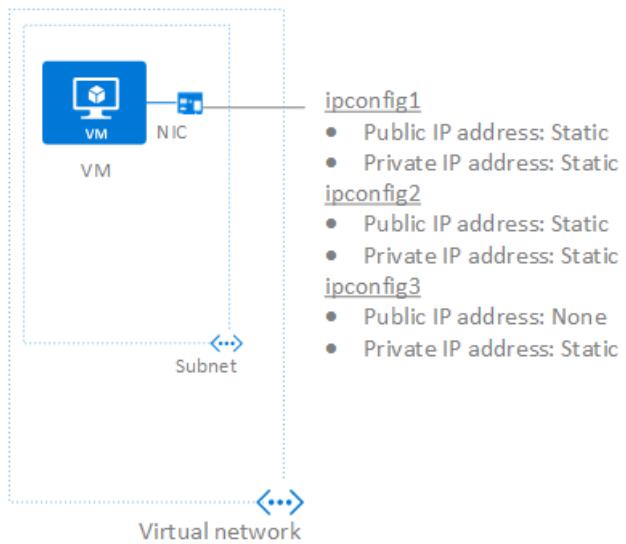
There is a limit to how many private IP addresses can be assigned to a NIC. There is also a limit to how many public IP addresses that can be used in an Azure subscription. See the [Azure limits](#) article for details.

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

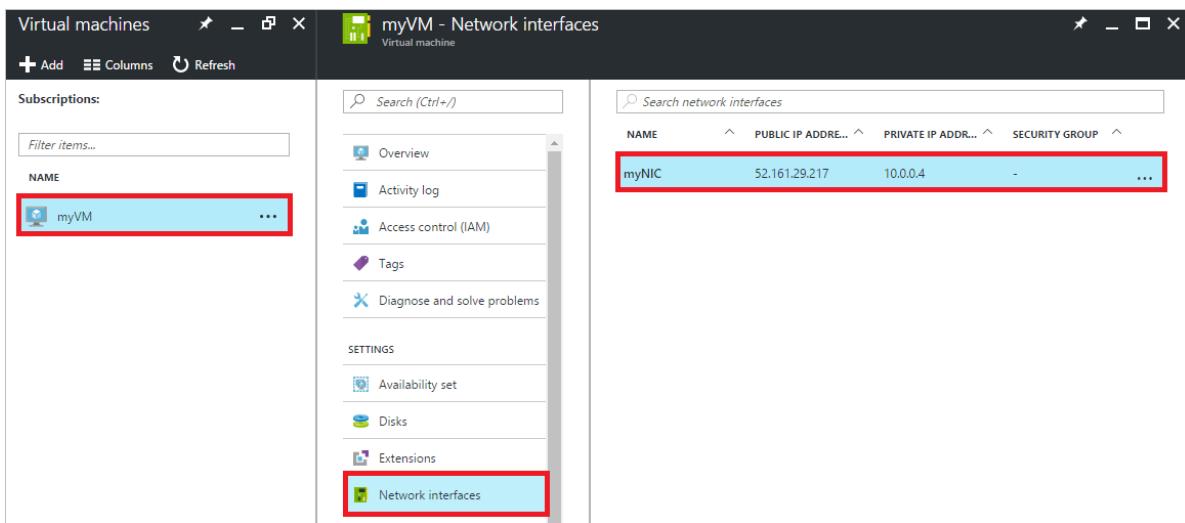
If you want to create a VM with multiple IP addresses, or a static private IP address, you must create it using PowerShell or the Azure CLI. To learn how, click the PowerShell or CLI options at the top of this article. You can create a VM with a single dynamic private IP address and (optionally) a single 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.

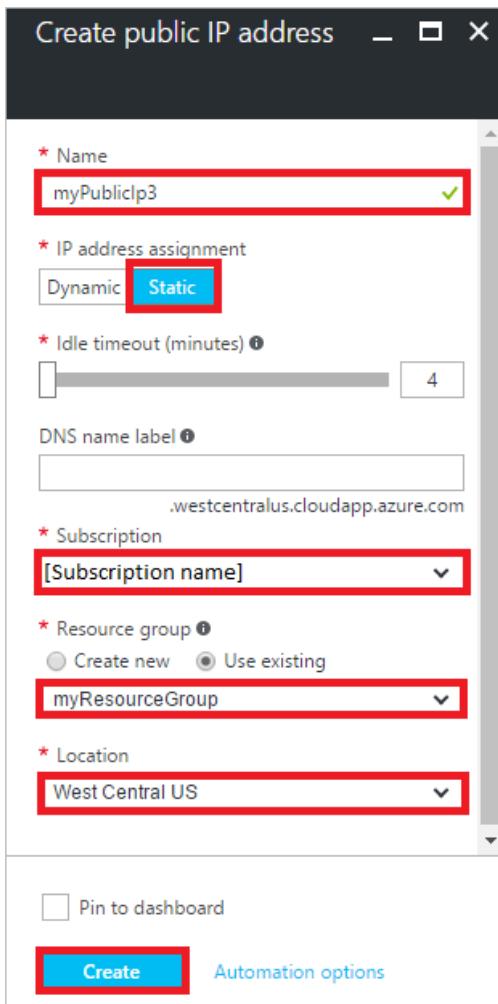
NOTE

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 login 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 following 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 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)

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 (Redhat, 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

4/18/2018 • 16 min 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.

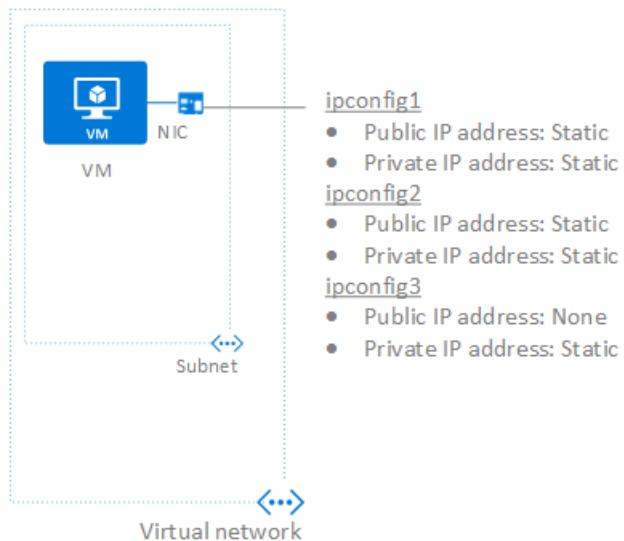
There is a limit to how many private IP addresses can be assigned to a NIC. There is also a limit to how many public IP addresses that can be used in an Azure subscription. See the [Azure limits](#) article for details.

This article explains how to create a virtual machine (VM) through the Azure Resource Manager deployment model using 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.



- ipconfig1
 - Public IP address: Static
 - Private IP address: Static
- ipconfig2
 - Public IP address: Static
 - Private IP address: Static
- ipconfig3
 - Public IP address: None
 - Private IP address: Static

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-AzureRmAccount` 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-AzureRmResourceGroup ` 
-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-AzureRmVirtualNetworkSubnetConfig ` 
-Name MySubnet ` 
-AddressPrefix 10.0.0.0/24

# Create a virtual network
$VNet = New-AzureRmVirtualNetwork ` 
-ResourceGroupName $RgName ` 
-Location $Location ` 
-Name MyVNet ` 
-AddressPrefix 10.0.0.0/16 ` 
-Subnet $subnetConfig

# Get the subnet object
$Subnet = Get-AzureRmVirtualNetworkSubnetConfig -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-AzureRmNetworkSecurityRuleConfig ` 
-Name MyNsgRuleRDP ` 
-Protocol Tcp ` 
-Direction Inbound ` 
-Priority 1000 ` 
-SourceAddressPrefix * ` 
-SourcePortRange * ` 
-DestinationAddressPrefix * ` 
-DestinationPortRange 3389 -Access Allow

# Create a network security group
$NSG = New-AzureRmNetworkSecurityGroup ` 
-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-AzureRmPrivateIPAddressAvailability -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 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-AzureRmPublicIpAddress ` 
-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-AzureRmNetworkInterfaceIpConfig ` 
-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-AzureRmPublicIpAddress ` 
-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-AzureRmNetworkInterfaceIpConfig ` 
-Name $IpConfigName2 ` 
-Subnet $Subnet ` 
-PrivateIpAddress 10.0.0.5 ` 
-PublicIpAddress $PublicIP2

$IpConfigName3 = "IpConfig-3"
$IpConfig3      = New-AzureRmNetworkInterfaceIpConfig ` 
-Name $IPConfigName3 ` 
-Subnet $Subnet ` 
-PrivateIpAddress 10.0.0.6

```

8. Create the NIC and associate the three IP configurations to it:

```
$NIC = New-AzureRmNetworkInterface `  
-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 susername and  
password for the VM you're reating.  
$cred = Get-Credential  
  
# Create a virtual machine configuration  
$VmConfig = New-AzureRmVMConfig `  
-VMName MyVM `  
-VMSize Standard_DS1_v2 | `  
Set-AzureRmVMOperatingSystem -Windows `  
-ComputerName MyVM `  
-Credential $cred | `  
Set-AzureRmVMSourceImage `  
-PublisherName MicrosoftWindowsServer `  
-Offer WindowsServer `  
-Skus 2016-Datacenter `  
-Version latest | `  
Add-AzureRmVMNetworkInterface `  
-Id $NIC.Id  
  
# Create the VM  
New-AzureRmVM `  
-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-AzureRmNetworkInterface | Format-Table Name, ResourceGroupName, Location
```

3. Create a variable and set it to the existing NIC by typing the following command:

```
$MyNIC = Get-AzureRmNetworkInterface -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-AzureRMVirtualNetwork -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-AzureRmPrivateIPAddressAvailability -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-AzureRmNetworkInterfaceIpConfig -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-AzureRmPublicIpAddress `  
-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-AzureRmNetworkInterfaceIpConfig `  
-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-AzureRmPublicIpAddress  
-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-AzureRmNetworkInterfaceIpConfig  
-Name IpConfig-3  
-NetworkInterface $mynic  
-Subnet $Subnet  
-PublicIpAddress $myPublicIp3
```

6. Set the NIC with the new IP configuration by entering the following command:

```
Set-AzureRmNetworkInterface -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 login 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)

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 (Redhat, 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 the Azure CLI

4/16/2018 • 15 min 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.

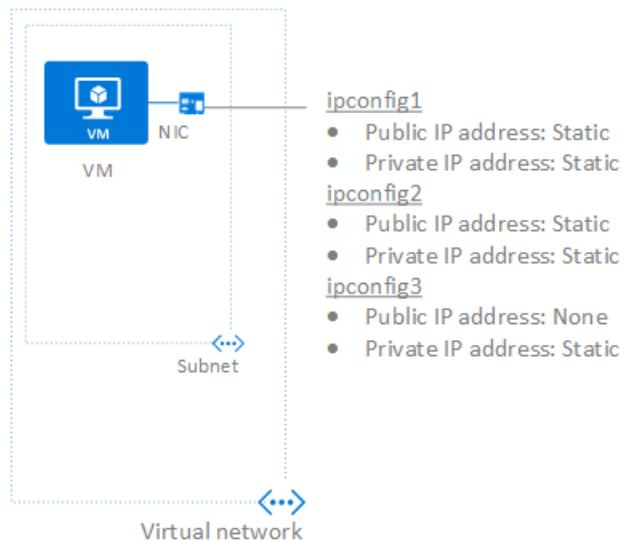
There is a limit to how many private IP addresses can be assigned to a NIC. There is also a limit to how many public IP addresses that can be used in an Azure subscription. See the [Azure limits](#) article for details.

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 2.0](#) 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.

azure 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 estcentralus` 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 2.0](#) 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 2.0 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 login 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 following 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.
 - 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)

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 (Redhat, 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:

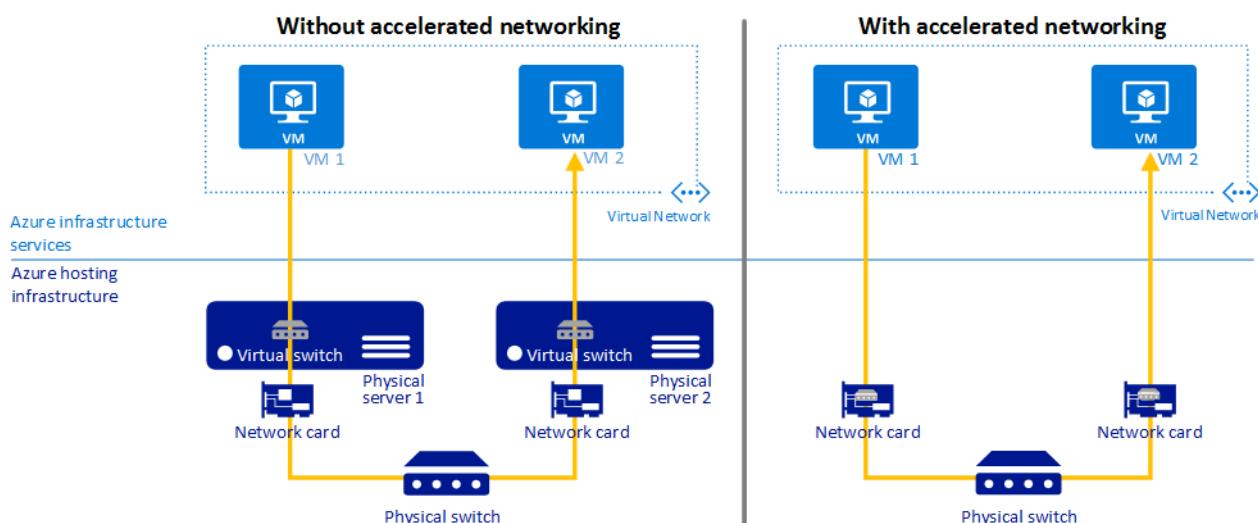
```
echo 150 custom >> /etc/iproute2/rt_tables  
  
ip rule add from 10.0.0.5 lookup custom  
ip route add default via 10.0.0.1 dev eth2 table custom
```

- 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

Create a Windows virtual machine with Accelerated Networking

5/7/2018 • 9 min 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, read the [Hyper-V network virtualization and virtual switch](#) article.

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**

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, and Ms/Mms

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

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, under **Settings**, select **Enabled**, under **Accelerated networking**. The option to enable accelerated networking doesn't appear in the portal unless you've selected a [supported operating system](#) and [VM size](#). After the virtual machine is created, you need to complete the instructions in [Confirm the driver is installed in the operating system](#).

Create a virtual network

Install [Azure PowerShell](#) version 5.1.1 or later. To find your currently installed version, run

```
Get-Module -ListAvailable AzureRM
```

If you need to install or upgrade, install the latest version of the AzureRM module from the [PowerShell Gallery](#). In a PowerShell session, log in to an Azure account using [Connect-AzureRmAccount](#).

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-AzureRmResourceGroup](#). The following example creates a resource group named *myResourceGroup* in the *centralus* location:

```
New-AzureRmResourceGroup -Name "myResourceGroup" -Location "centralus"
```

First, create a subnet configuration with [New-AzureRmVirtualNetworkSubnetConfig](#). The following example creates a subnet named *mySubnet*:

```
$subnet = New-AzureRmVirtualNetworkSubnetConfig `  
    -Name "mySubnet" `  
    -AddressPrefix "192.168.1.0/24"
```

Create a virtual network with [New-AzureRmVirtualNetwork](#), with the *mySubnet* subnet.

```
$vnet = New-AzureRmVirtualNetwork -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-AzureRmNetworkSecurityRuleConfig](#).

```
$rdp = New-AzureRmNetworkSecurityRuleConfig `  
    -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-AzureRmNetworkSecurityGroup](#) 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-AzureRmNetworkSecurityGroup `  
    -ResourceGroupName myResourceGroup `  
    -Location centralus `  
    -Name "myNsg" `  
    -SecurityRules $rdp
```

Associate the network security group to the *mySubnet* subnet with [Set-AzureRmVirtualNetworkSubnetConfig](#). The rule in the network security group is effective for all resources deployed in the subnet.

```
Set-AzureRmVirtualNetworkSubnetConfig `  
    -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-AzureRmPublicIpAddress](#). 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-AzureRmPublicIpAddress `  
    -ResourceGroupName myResourceGroup `  
    -Name 'myPublicIp' `  
    -location centralus `  
    -AllocationMethod Dynamic
```

Create a network interface with [New-AzureRmNetworkInterface](#) 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-AzureRmNetworkInterface `  
    -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-AzureRmVMConfig](#). The following example defines a VM named *myVM* with a VM size that supports Accelerated Networking (*Standard_DS4_v2*):

```
$vmConfig = New-AzureRmVMConfig -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-AzureRmVMOperatingSystem](#) and [Set-AzureRmVMSourceImage](#). The following example creates a Windows Server 2016 VM:

```
$vmConfig = Set-AzureRmVMOperatingSystem -VM $vmConfig `  
    -Windows `  
    -ComputerName "myVM" `  
    -Credential $cred `  
    -ProvisionVMAgent `  
    -EnableAutoUpdate  
$vmConfig = Set-AzureRmVMSourceImage -VM $vmConfig `  
    -PublisherName "MicrosoftWindowsServer" `  
    -Offer "WindowsServer" `  
    -Skus "2016-Datacenter" `  
    -Version "latest"
```

Attach the network interface that you previously created with [Add-AzureRmVMNetworkInterface](#):

```
$vmConfig = Add-AzureRmVMNetworkInterface -VM $vmConfig -Id $nic.Id
```

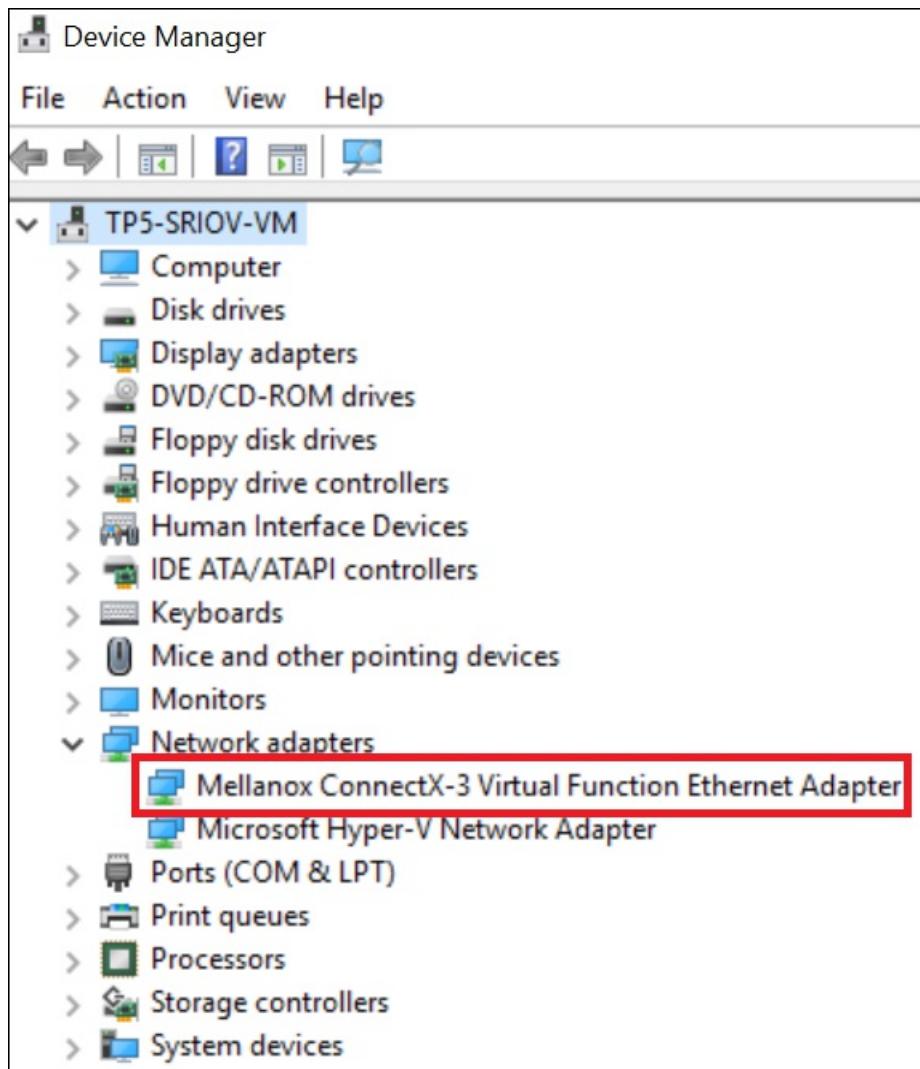
Finally, create your VM with [New-AzureRmVM](#):

```
New-AzureRmVM -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-AzureRmVM -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-AzureRmNetworkInterface -ResourceGroupName "myResourceGroup" `  
-Name "myNic"  
  
$nic.EnableAcceleratedNetworking = $true  
  
$nic | Set-AzureRmNetworkInterface
```

Restart your VM or, if in an Availability Set, all the VMs in the Set and confirm that Accelerated Networking is enabled:

```
Start-AzureRmVM -ResourceGroup "myResourceGroup" `  
-Name "myVM"
```

VMSS

VMSS is slightly different but follows the same workflow. First, stop the VMs:

```
Stop-AzureRmVmss -ResourceGroupName "myResourceGroup" `  
-VMScaleSetName "myScaleSet"
```

Once the VMs are stopped, update the Accelerated Networking property under the network interface:

```
$vmss = Get-AzureRmVmss -ResourceGroupName "myResourceGroup" `  
-VMScaleSetName "myScaleSet"  
  
$vmss.VirtualMachineProfile.NetworkProfile.NetworkInterfaceConfigurations[0].EnableAcceleratedNetworking =  
$true  
  
Update-AzureRmVmss -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-AzureRmVmss -ResourceGroupName "myResourceGroup" `  
-VMSScaleSetName "myScaleSet" `  
-VirtualMachineScaleSet $vmss
```

Finally, restart the VMSS:

```
Start-AzureRmVmss -ResourceGroupName "myResourceGroup" `  
-VMSScaleSetName "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.

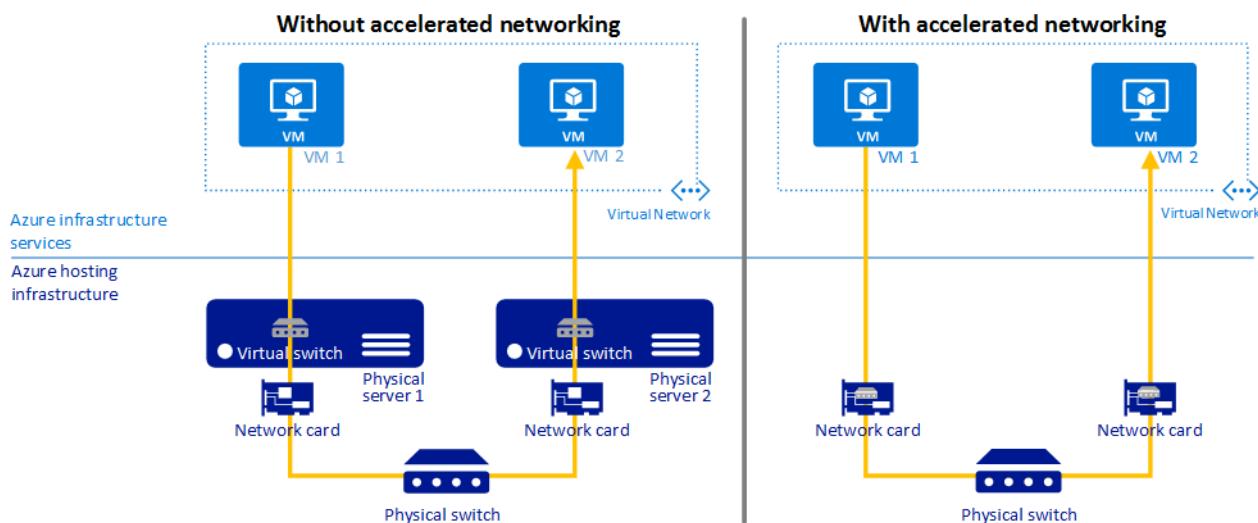
A VM with Accelerated Networking enabled cannot be resized to a VM instance that does not support Accelerated Networking using the resize operation. Instead, to resize one of these VMs:

- 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

5/2/2018 • 9 min 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 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.

Supported operating systems

The following distributions are supported out of the box from the Azure Gallery:

- **Ubuntu 16.04**
- **SLES 12 SP3**
- **RHEL 7.4**
- **CentOS 7.4**
- **CoreOS Linux**
- **Debian "Stretch" with backports kernel**
- **Oracle Linux 7.4**

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, and Ms/Mms.

For more information on VM instances, see [Linux VM sizes](#).

Regions

Available in all public Azure regions as well as Azure Government Clouds.

Network interface creation

Accelerated networking can only be enabled for a new NIC. It cannot be enabled for an existing NIC.

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

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, under **Settings**, select **Enabled**, under **Accelerated networking**. The option to enable accelerated networking doesn't appear in the portal unless you've selected a [supported operating system](#) and [VM size](#). After the virtual machine is created, you need to complete the instructions in [Confirm that accelerated networking is enabled](#).

Create a virtual network

Install the latest [Azure CLI 2.0](#) 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.

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 myVM -n 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.

Virtual machine network bandwidth

1/11/2018 • 2 min 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.

Next steps

- [Optimize network throughput for a virtual machine operating system](#)
- [Test network throughput for a virtual machine](#).

Configure a VNet-to-VNet VPN gateway connection using PowerShell

4/18/2018 • 17 min 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.

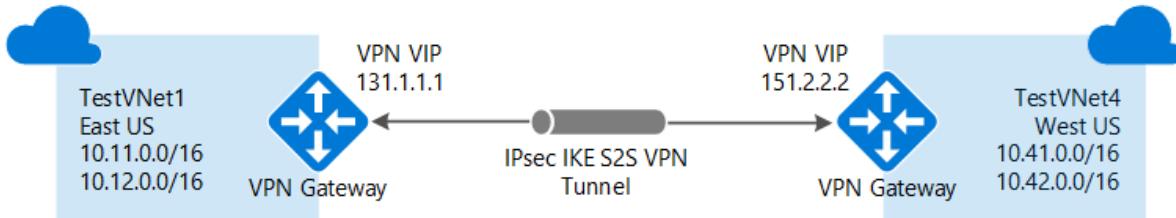
VNet-to-VNet communication can be combined with multi-site configurations. This lets you establish network topologies that combine cross-premises connectivity with inter-virtual network connectivity.

Which VNet-to-VNet steps should I use?

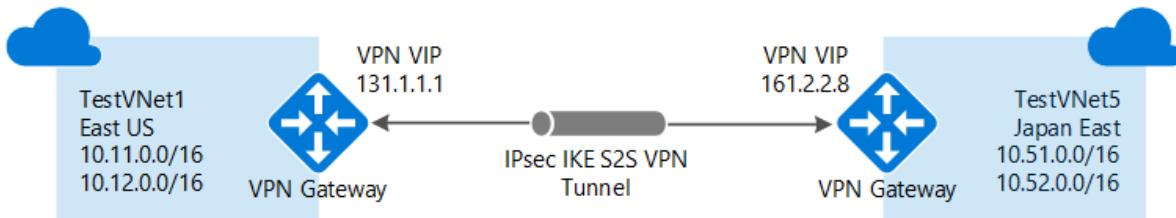
In this article, you see two different sets of steps. One set of steps for [VNets that reside in the same subscription](#) and one for [VNets that reside in different subscriptions](#). The key difference between the sets is that you must use separate PowerShell sessions when configuring the connections 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

Before beginning, you need to install the latest version of the Azure Resource Manager PowerShell cmdlets, at least 4.0 or later. For more information about installing the PowerShell cmdlets, 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. 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.

```
$Sub1 = "Replace_With_Your_Subscription_Name"
$RG1 = "TestRG1"
$Location1 = "East US"
$VNetName1 = "TestVNet1"
$FESubName1 = "FrontEnd"
$BESubName1 = "Backend"
$GWSubName1 = "GatewaySubnet"
$VNetPrefix11 = "10.11.0.0/16"
$VNetPrefix12 = "10.12.0.0/16"
$FESubPrefix1 = "10.11.0.0/24"
$BESubPrefix1 = "10.12.0.0/24"
$GWSubPrefix1 = "10.12.255.0/27"
$GWName1 = "VNet1GW"
$GWIPName1 = "VNet1GWIP"
$GWIPconfName1 = "gwipconf1"
$Connection14 = "VNet1toVNet4"
$Connection15 = "VNet1toVNet5"
```

2. Connect to your account. Use the following example to help you connect:

```
Connect-AzureRmAccount
```

Check the subscriptions for the account.

```
Get-AzureRmSubscription
```

Specify the subscription that you want to use.

```
Select-AzureRmSubscription -SubscriptionName $Sub1
```

3. Create a new resource group.

```
New-AzureRmResourceGroup -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.

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-AzureRmVirtualNetworkSubnetConfig -Name $FESubName1 -AddressPrefix $FESubPrefix1  
$besub1 = New-AzureRmVirtualNetworkSubnetConfig -Name $BESubName1 -AddressPrefix $BESubPrefix1  
$gwsu1 = New-AzureRmVirtualNetworkSubnetConfig -Name $GWSuName1 -AddressPrefix $GWSuPrefix1
```

5. Create TestVNet1.

```
New-AzureRmVirtualNetwork -Name $VNetName1 -ResourceGroupName $RG1  
-Location $Location1 -AddressPrefix $VNetPrefix11,$VNetPrefix12 -Subnet $fesub1,$besub1,$gwsu1
```

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-AzureRmPublicIpAddress -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-AzureRmVirtualNetwork -Name $VNetName1 -ResourceGroupName $RG1  
$subnet1 = Get-AzureRmVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet1  
$gwipconf1 = New-AzureRmVirtualNetworkGatewayIpConfig -Name $GWIPconfName1  
-Subnet $subnet1 -PublicIpAddress $gwpip1
```

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-AzureRmVirtualNetworkGateway -Name $GWName1 -ResourceGroupName $RG1 `  
-Location $Location1 -IpConfigurations $gwipconf1 -GatewayType Vpn `  
-VpnType RouteBased -GatewaySku VpnGw1
```

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. This step can be done within the same PowerShell session because it is in the same subscription.

1. 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"  
$GWSubName4 = "GatewaySubnet"  
$VnetPrefix41 = "10.41.0.0/16"  
$VnetPrefix42 = "10.42.0.0/16"  
$FESubPrefix4 = "10.41.0.0/24"  
$BESubPrefix4 = "10.42.0.0/24"  
$GWSubPrefix4 = "10.42.255.0/27"  
$GWName4 = "VNet4GW"  
$GWIPName4 = "VNet4GWIP"  
$GWIPconfName4 = "gwipconf4"  
$Connection41 = "VNet4toVNet1"
```

2. Create a new resource group.

```
New-AzureRmResourceGroup -Name $RG4 -Location $Location4
```

3. Create the subnet configurations for TestVNet4.

```
$fesub4 = New-AzureRmVirtualNetworkSubnetConfig -Name $FESubName4 -AddressPrefix $FESubPrefix4  
$besub4 = New-AzureRmVirtualNetworkSubnetConfig -Name $BESubName4 -AddressPrefix $BESubPrefix4  
$gwsub4 = New-AzureRmVirtualNetworkSubnetConfig -Name $GWSubName4 -AddressPrefix $GWSubPrefix4
```

4. Create TestVNet4.

```
New-AzureRmVirtualNetwork -Name $VnetName4 -ResourceGroupName $RG4 `  
-Location $Location4 -AddressPrefix $VnetPrefix41,$VnetPrefix42 -Subnet $fesub4,$besub4,$gwsub4
```

5. Request a public IP address.

```
$gwpip4 = New-AzureRmPublicIpAddress -Name $GWIPName4 -ResourceGroupName $RG4 `  
-Location $Location4 -AllocationMethod Dynamic
```

6. Create the gateway configuration.

```
$vnet4 = Get-AzureRmVirtualNetwork -Name $vnetName4 -ResourceGroupName $RG4  
$subnet4 = Get-AzureRmVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet4  
$gwipconf4 = New-AzureRmVirtualNetworkGatewayIpConfig -Name $GWIPconfName4 -Subnet $subnet4 -  
PublicIpAddress $gwip4
```

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

```
New-AzureRmVirtualNetworkGateway -Name $GWName4 -ResourceGroupName $RG4 `  
-Location $Location4 -IpConfigurations $gwipconf4 -GatewayType Vpn `  
-VpnType RouteBased -GatewaySku VpnGw1
```

Step 4 - Create the connections

1. Get both virtual network gateways. If both of the gateways are in the same subscription, as they are in the example, you can complete this step in the same PowerShell session.

```
$vnet1gw = Get-AzureRmVirtualNetworkGateway -Name $GWName1 -ResourceGroupName $RG1  
$vnet4gw = Get-AzureRmVirtualNetworkGateway -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-AzureRmVirtualNetworkGatewayConnection -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-AzureRmVirtualNetworkGatewayConnection -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 a different subscription. 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.

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-AzureRmAccount
```

Check the subscriptions for the account.

```
Get-AzureRmSubscription
```

Specify the subscription that you want to use.

```
Select-AzureRmSubscription -SubscriptionName $Sub5
```

3. Create a new resource group.

```
New-AzureRmResourceGroup -Name $RG5 -Location $Location5
```

4. Create the subnet configurations for TestVNet5.

```
$fesub5 = New-AzureRmVirtualNetworkSubnetConfig -Name $FESubName5 -AddressPrefix $FESubPrefix5  
$besub5 = New-AzureRmVirtualNetworkSubnetConfig -Name $BESubName5 -AddressPrefix $BESubPrefix5  
$gwsb5 = New-AzureRmVirtualNetworkSubnetConfig -Name $GWSubName5 -AddressPrefix $GWSubPrefix5
```

5. Create TestVNet5.

```
New-AzureRmVirtualNetwork -Name $VnetName5 -ResourceGroupName $RG5 -Location $Location5 `  
-AddressPrefix $VnetPrefix51,$VnetPrefix52 -Subnet $fesub5,$besub5,$gwsb5
```

6. Request a public IP address.

```
$gwpip5 = New-AzureRmPublicIpAddress -Name $GWIPName5 -ResourceGroupName $RG5 `  
-Location $Location5 -AllocationMethod Dynamic
```

7. Create the gateway configuration.

```
$vnet5 = Get-AzureRmVirtualNetwork -Name $VnetName5 -ResourceGroupName $RG5  
$subnet5 = Get-AzureRmVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet5  
$gwipconf5 = New-AzureRmVirtualNetworkGatewayIpConfig -Name $GWIPconfName5 -Subnet $subnet5 -  
PublicIpAddress $gwpip5
```

8. Create the TestVNet5 gateway.

```
New-AzureRmVirtualNetworkGateway -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. Log in and connect to Subscription 1 before running the following example:

```
$vnet1gw = Get-AzureRmVirtualNetworkGateway -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/resourceGroups/TestRG1/providers/Microsoft.Network/virtualNetworkGateways/VNet1GW
```

2. [Subscription 5] Get the virtual network gateway for Subscription 5. Log in and connect to Subscription 5 before running the following example:

```
$vnet5gw = Get-AzureRmVirtualNetworkGateway -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 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-AzureRmVirtualNetworkGatewayConnection -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 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-AzureRmVirtualNetworkGatewayConnection -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 VPN 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-AzureRmVirtualNetworkGatewayConnection' 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-AzureRmVirtualNetworkGatewayConnection -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. If you are looking for 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 using 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. Refer to the [VPN Gateway pricing page](#) for details. If you are connecting your VNets using VNet Peering, rather than VPN Gateway, see the [Virtual Network pricing page](#).

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 AAD Tenants?

Yes, VNet-to-VNet connections using Azure VPN gateways work across AAD Tenants.

Is VNet-to-VNet traffic secure?

Yes, it is 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 are not in the same subscription, do the subscriptions need to be associated with the same AD 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 public Azure and the Chinese / German / US Gov Azure instances. For these scenarios, consider using a Site-to-Site VPN connection.

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 [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 does not support connecting virtual machines or cloud services that are not 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 are connected together.

Can I used 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 be using 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

4/18/2018 • 21 min 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

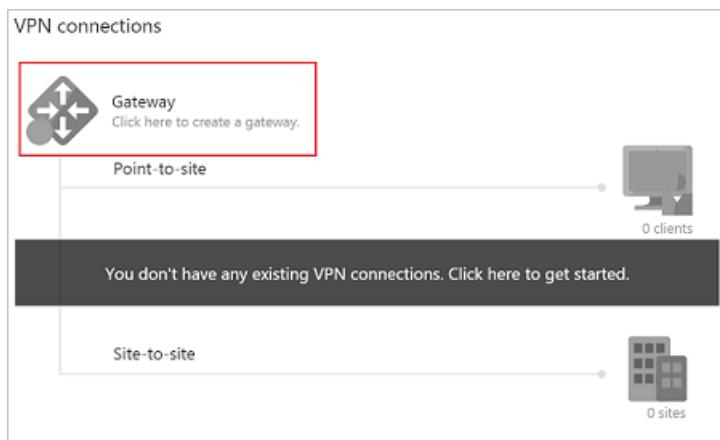
If you don't have a classic VNet and are running these steps as an exercise, you can create a VNet by using [this article](#) and the [Example](#) settings values from above.

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

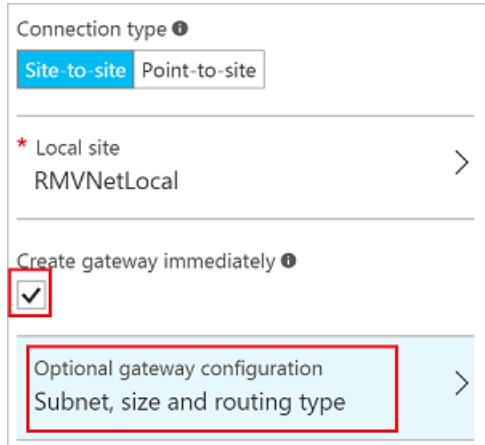
1. Navigate to **All resources** and locate the **ClassicVNet** in the list.
2. On the **Overview** page, in the **VPN connections** section, click **Gateway** to create a gateway.



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

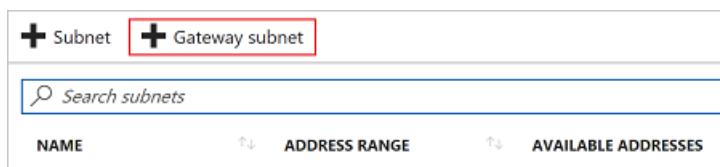
If you don't have a Resource Manager VNet and are running these steps as an exercise, create a virtual network with the steps in [Create a virtual network](#), using the example values.

2. Create a gateway subnet

Example value: GatewaySubnet = 192.168.0.0/26

Before creating a virtual network gateway, you first need to create the gateway subnet. Create a gateway subnet with CIDR count of /28 or larger (/27, /26, etc.). If you are creating this as part of an exercise, you can use the Example values.

1. In the [portal](#), navigate to the Resource Manager virtual network for which you want to create a virtual network gateway.
2. In the **Settings** section of your VNet page, click **Subnets** to expand the Subnets page.
3. On the **Subnets** page, click **+Gateway subnet** to open the **Add subnet** page.



4. The **Name** for your subnet is automatically filled in with the value 'GatewaySubnet'. This value is required in order for Azure to recognize the subnet as the gateway subnet. Adjust the auto-filled **Address range** values to match your configuration requirements, then click **OK** at the bottom of the page to create the subnet.



IMPORTANT

When working with gateway subnets, avoid associating a network security group (NSG) to the gateway subnet. Associating a network security group to this subnet may cause your VPN gateway to stop functioning as expected. For more information about network security groups, see [What is a network security group?](#)

3. Create a virtual network gateway

Example values:

- Virtual network gateway name = RMGateway
- Gateway type = VPN
- VPN type = Route-based
- SKU = VpnGw1
- Location = East US
- Virtual network = RMVNet
- First IP configuration = rmgwpip

1. In the portal, on the left side, click + and type 'virtual network gateway' in search. Locate **Virtual network gateway** in the search return and click the entry. On the **Virtual network gateway** page, click **Create** at the bottom of the page to open the **Create virtual network gateway** page.
2. On the **Create virtual network gateway** page, fill in the values for your virtual network gateway.

Create virtual network gateway

* Name

Gateway type [?](#)

VPN ExpressRoute

VPN type [?](#)

Route-based Policy-based

* SKU [?](#)

VpnGw1

Enable active-active mode [?](#)

* Virtual network [?](#) >

Choose a virtual network

* First IP configuration [?](#) >

Create gateway IP configuration

Configure BGP ASN

* Subscription

Windows Azure Internal Consumption

Resource group [?](#)

-

* Location [?](#)

West US

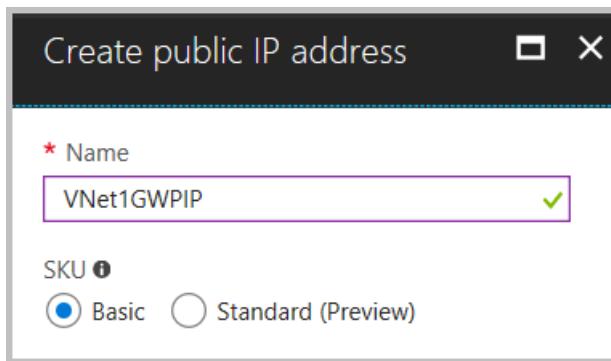
Pin to dashboard

Create [Automation options](#)

Provisioning a virtual network gateway may take up to 45 minutes.

3. On the **Create virtual network gateway** page, specify the values for your virtual network gateway.
 - **Name:** Name your gateway. This is not the same as naming a gateway subnet. It's the name of the gateway object you are creating.
 - **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](#).
 - **Location:** You may need to scroll to see Location. Adjust the **Location** field to point to the location where your virtual network is located. If the location is not pointing to the region where your virtual network resides, when you select a virtual network in the next step, it will not appear in the drop-down list.

- **Virtual network:** Choose the virtual network to which you want to add this gateway. Click **Virtual network** to open the 'Choose a virtual network' page. Select the VNet. If you don't see your VNet, make sure the Location field is pointing to the region in which your virtual network is located.
- **Gateway subnet address range:** You will only see this setting if you did not previously create a gateway subnet for your virtual network. If you previously created a valid gateway subnet, this setting will not appear.
- **First IP configuration:** The 'Choose public IP address' page creates a 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. VPN Gateway currently only supports *Dynamic* Public IP address allocation. However, this does not mean that the IP address changes after it has been assigned to your VPN gateway. 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.
 - First, click **Create gateway IP configuration** to open the 'Choose public IP address' page, then click **+Create new** to open the 'Create public IP address' page.
 - Next, input a **Name** for your public IP address. Leave the SKU as **Basic** unless there is a specific reason to change it to something else, then click **OK** at the bottom of this page to save your changes.



4. Verify the settings. You can select **Pin to dashboard** at the bottom of the page if you want your gateway to appear on the dashboard.
5. Click **Create** to begin creating the VPN gateway. The settings are validated and you'll see the "Deploying Virtual network gateway" tile on the dashboard. Creating a gateway can take up to 45 minutes. You may need to refresh your portal page to see the completed status.

After the gateway is created, 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. You can click the connected device (your virtual network gateway) to view more information.

4. 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, from **All resources**, click **+Add**.
2. In the **Everything** page search box, type **Local network gateway**, then click to return a list of resources. Click **Local network gateway** to open the page, then click **Create** to open the **Create local network gateway** page.

Create local network gateway X

* Name

* IP address ⓘ

Address space ⓘ
 Add additional address range ...

Configure BGP settings

* Subscription
 Windows Azure Internal Consumption ▾

* Resource group ⓘ
 Create new Use existing

* Location
 East US ▾

Pin to dashboard

Create [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. If possible, use something intuitive, such as **ClassicVNetLocal** or **TestVNet1Local**. This makes it easier for you to identify the local network gateway in the portal.
- **IP address:** Specify a valid Public **IP address** for the VPN device or virtual network gateway to which you want to connect.
 - **If this local network represents an on-premises location:** Specify the Public IP address of the VPN device that you want to connect to. It cannot be behind NAT and has to be reachable by

Azure.

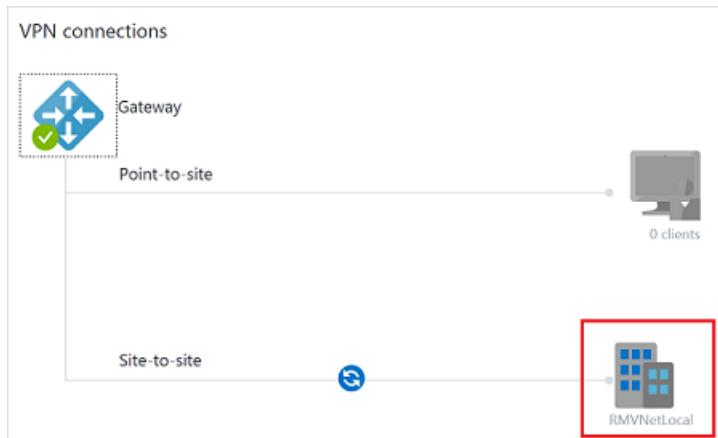
- **If this local network represents another VNet:** Specify the Public IP address that was assigned to the virtual network gateway for that VNet.
- **If you don't yet have the IP address:** You can make up a valid placeholder IP address, and then come back and modify this setting before connecting.
- **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 to which you connect.
- **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. Click **Create** to create the local network gateway.

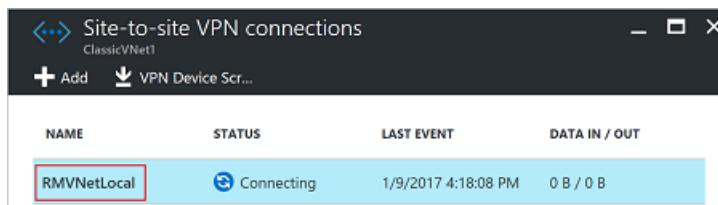
Section 3 – Modify the classic VNet local site settings

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.

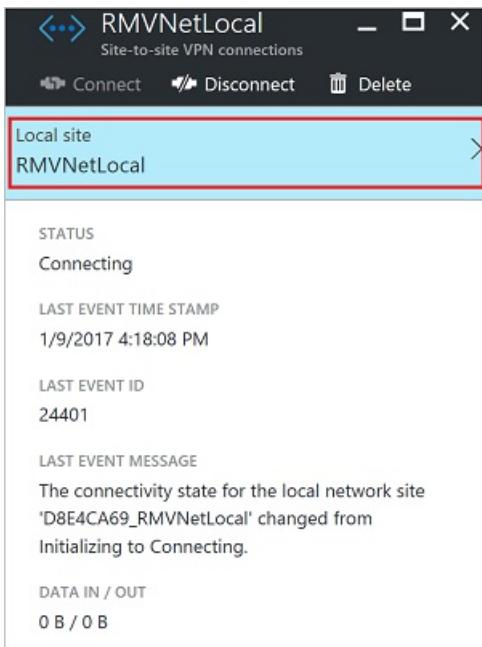
1. In the Azure portal, navigate to the classic virtual network.
2. On the page for your virtual network, click **Overview**.
3. In the **VPN connections** section, click the name of your local site in the graphic.



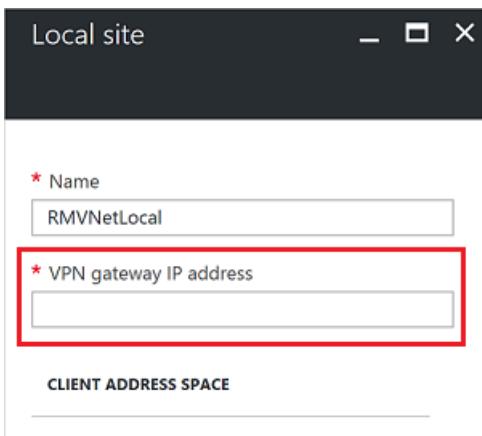
4. On the **Site-to-site VPN connections** page, click the name of the site.



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-AzureRmAccount
```

Get a list of your Azure subscriptions.

```
Get-AzureRmSubscription
```

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

```
Select-AzureRmSubscription -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
```

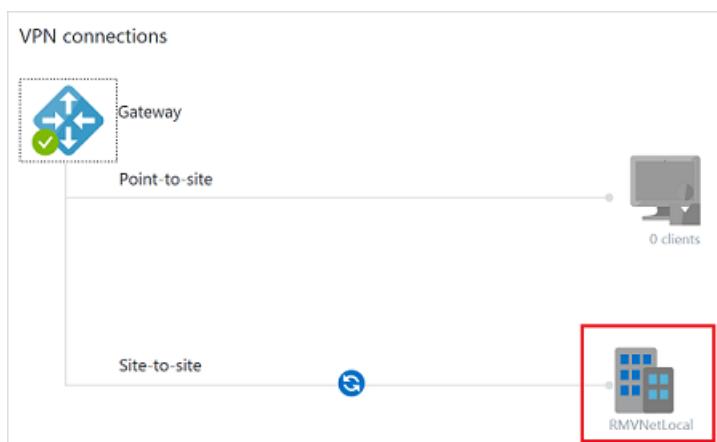
Section 6 – Verify your connections

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.

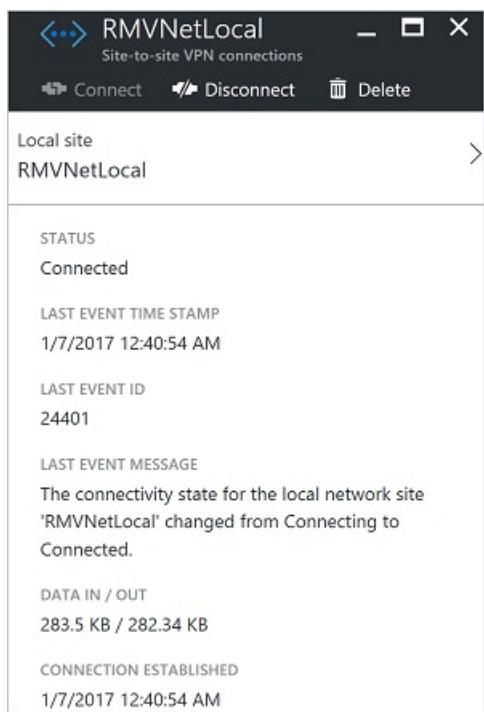
1. In the [Azure portal](#), click **All resources** and navigate to your classic virtual network.
2. On the virtual network blade, click **Overview** to access the **VPN connections** section of the blade.
3. On the VPN connections graphic, click the site.



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
Status Connected	Data out 3.14 KB
Location East US	Virtual network
Subscription name	Virtual network gateway
Subscription ID	Local network gateway

VNet-to-VNet FAQ

The VNet-to-VNet FAQ applies to VPN Gateway connections. If you are looking for 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 using 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. Refer to the [VPN Gateway pricing page](#) for details. If you are connecting your VNets using VNet Peering, rather than VPN Gateway, see the [Virtual Network pricing page](#).

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 AAD Tenants?

Yes, VNet-to-VNet connections using Azure VPN gateways work across AAD Tenants.

Is VNet-to-VNet traffic secure?

Yes, it is 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 are not in the same subscription, do the subscriptions need to be associated with the same AD 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 public Azure and the Chinese / German / US Gov Azure instances. For these scenarios, consider using a Site-to-Site VPN connection.

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 [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 does not support connecting virtual machines or cloud services that are not 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 are connected together.

Can I used 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 be using 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?

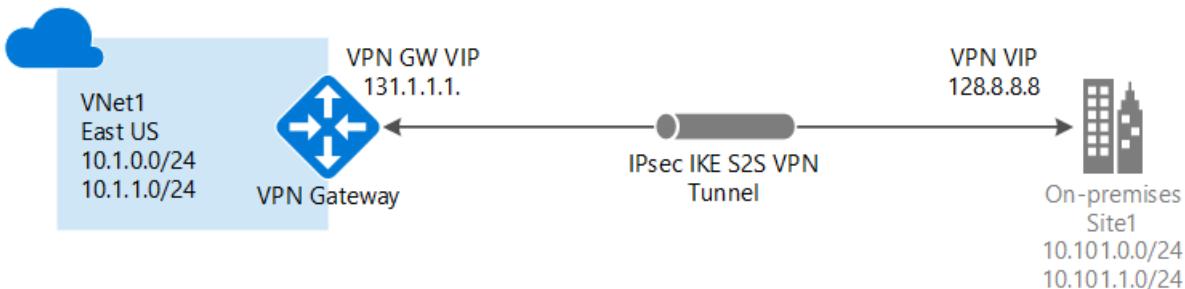
No. You can't have overlapping IP address ranges.

Create a Site-to-Site connection in the Azure portal

4/9/2018 • 19 min 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. This IP address cannot be located behind a NAT.
- 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](#).

- **VNet Name:** TestVNet1
- **Address Space:** 10.1.0.0/16
- **Subscription:** The subscription you want to use
- **Resource Group:** TestRG1
- **Location:** East US
- **Subnet:** FrontEnd: 10.1.0.0/24, BackEnd: 10.1.1.0/24 (optional for this exercise)
- **Gateway Subnet name:** GatewaySubnet (this will auto-fill in the portal)
- **Gateway Subnet address range:** 10.1.255.0/27
- **DNS Server:** 8.8.8.8 - Optional. The IP address of your DNS server.
- **Virtual Network Gateway Name:** VNet1GW

- **Public IP:** 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 a browser, navigate to the [Azure portal](#) and sign in with your Azure account.
2. Click **Create a resource**. 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. Near the bottom of the Virtual Network page, from the **Select a deployment model** list, select **Resource Manager**, and then click **Create**. This opens the 'Create virtual network' page.

Create virtual network

* Name
VNet1

* Address space ⓘ
10.1.0.0/16
10.1.0.0 - 10.1.255.255 (65536 addresses)

* Subscription
Windows Azure Internal Consumption

* Resource group
 Create new Use existing
TestRG1

* Location
East US

Subnet

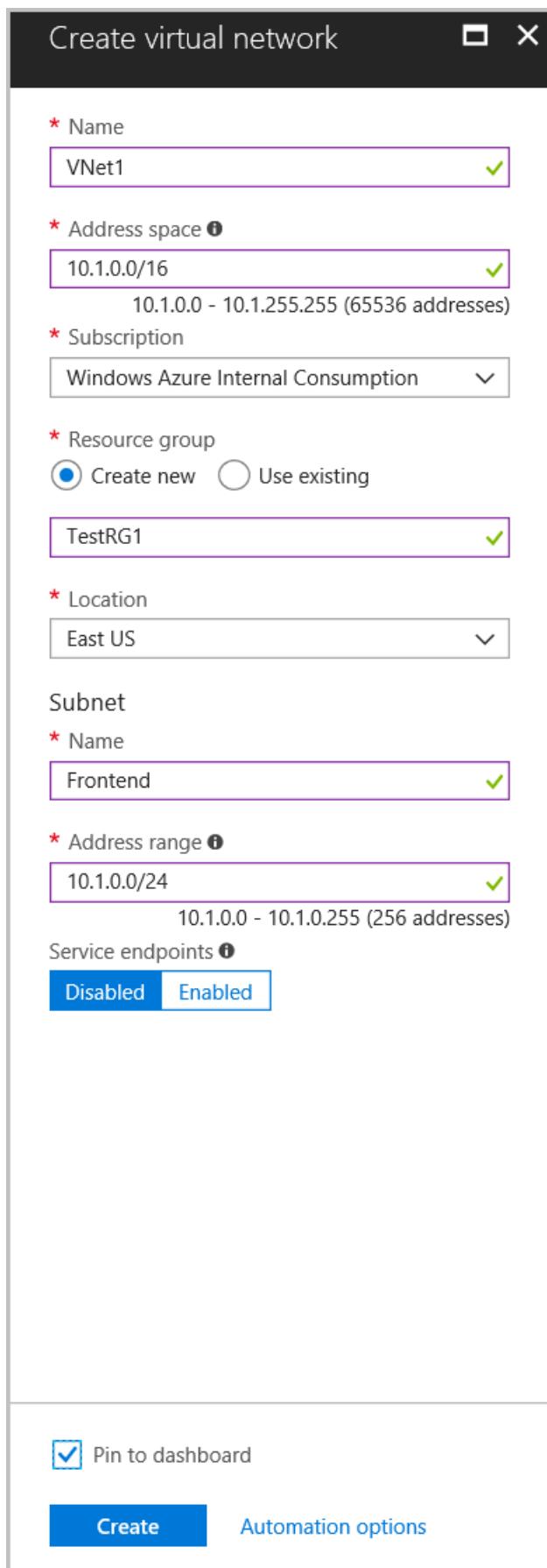
* Name
Frontend

* Address range ⓘ
10.1.0.0/24
10.1.0.0 - 10.1.0.255 (256 addresses)

Service endpoints ⓘ
 Disabled Enabled

Pin to dashboard

Create [Automation options](#)



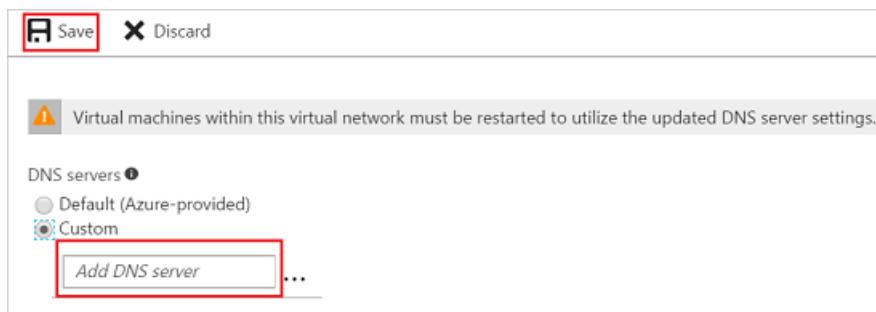
- On the **Create virtual network** page, configure the VNet settings. When you fill in the fields, the red exclamation mark becomes a green check mark when the characters entered in the field are valid.
 - Name:** Enter the name for your virtual network. In this example, we use VNet1.
 - Address space:** Enter the address space. If you have multiple address spaces to add, add your first address space. You can add additional address spaces later, after creating the VNet. Make sure that the address space that you specify does not overlap with the address space for your on-premises location.

- **Subscription:** Verify that the subscription listed is the correct one. You can change subscriptions by using the drop-down.
 - **Resource group:** Select an existing resource group, or create a new one by typing a name for your new resource group. If you are creating a new group, name the resource group according to your planned configuration values. For more information about resource groups, visit [Azure Resource Manager Overview](#).
 - **Location:** Select the location for your VNet. The location determines where the resources that you deploy to this VNet will reside.
 - **Subnet:** Add the first subnet name and subnet address range. You can add additional subnets and the gateway subnet later, after creating this VNet.
5. Select **Pin to dashboard** if you want to be able to find your VNet easily on the dashboard, and then click **Create**. After clicking **Create**, you will see a tile on your dashboard that will reflect the progress of your VNet. The tile changes as the VNet is being created.

2. Specify a DNS server

DNS is not required to create a Site-to-Site connection. However, if you want to have name resolution for resources that are deployed to your virtual network, you should specify a DNS server. This setting lets you specify the DNS server that you want to use for name resolution for this virtual network. It does not create a DNS server. For more information about name resolution, see [Name Resolution for VMs and role instances](#).

1. On the **Settings** page for your virtual network, navigate to **DNS Servers** and click to open the **DNS servers** page.



- **DNS Servers:** Select **Custom**.
 - **Add DNS server:** Enter the IP address of the DNS server that you want to use for name resolution.
2. When you are done adding DNS servers, click **Save** at the top of the page.

3. Create the gateway subnet

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. The subnet must be named 'GatewaySubnet' in order for Azure to deploy the gateway resources. You can't specify a different subnet to deploy the gateway resources to. If you don't have a subnet named 'GatewaySubnet', when you create your VPN gateway, it will fail.

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.

1. In the portal, navigate to the virtual network for which you want to create a virtual network gateway.
2. In the **Settings** section of your VNet page, click **Subnets** to expand the Subnets page.
3. On the **Subnets** page, click **+Gateway subnet** at the top to open the **Add subnet** page.

The screenshot shows the 'Add subnet' dialog box. At the top, it says 'Add subnet' and 'VNet1'. There are two required fields: 'Name' (GatewaySubnet) and 'Address range (CIDR block)' (10.1.255.0/27). Below these are optional sections: 'Route table' (None) and 'Service endpoints' (Services: 0 selected).

4. The **Name** for your subnet is automatically filled in with the value 'GatewaySubnet'. The GatewaySubnet value is required in order for Azure to recognize the subnet as the gateway subnet. Adjust the auto-filled **Address range** values to match your configuration requirements.

NAME	ADDRESS RANGE	AVAILABLE ADDRESSES
GatewaySubnet	10.1.255.0/27	251 + 5 Azure reserved addresses

5. To create the subnet, click **OK** at the bottom of the page.

IMPORTANT

When working with gateway subnets, avoid associating a network security group (NSG) to the gateway subnet. Associating a network security group to this subnet may cause your VPN gateway to stop functioning as expected. For more information about network security groups, see [What is a network security group?](#)

4. Create the VPN gateway

1. On the left side of the portal page, click **+** and type 'Virtual Network Gateway' in search. In **Results**, locate and click **Virtual network gateway**.
2. At the bottom of the 'Virtual network gateway' page, click **Create**. This opens the **Create virtual network gateway** page.
3. On the **Create virtual network gateway** page, specify the values for your virtual network gateway.

Create virtual network gateway

* Name
VNet1GW ✓

Gateway type i
VPN ExpressRoute

VPN type i
Route-based Policy-based

* SKU i
VpnGw1

Enable active-active mode i

* Virtual network i >
VNet1

* First IP configuration >
VNet1GWIP

Configure BGP ASN

* Subscription
Windows Azure Internal Consumption

Resource group i
TestRG1

Pin to dashboard

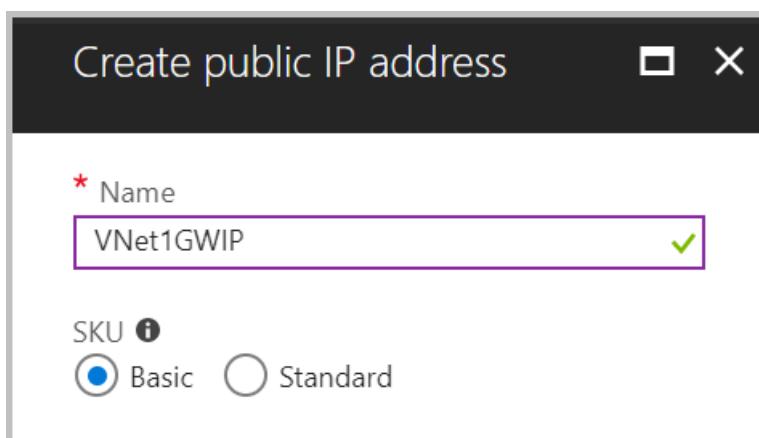
Create Automation options

Provisioning a virtual network gateway may take up to 45 minutes.

- **Name:** Name your gateway. This is not the same as naming a gateway subnet. It's the name of the gateway object you are creating.
- **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](#).

- **Location:** You may need to scroll to see Location. Adjust the **Location** field to point to the location where your virtual network is located. If the location is not pointing to the region where your virtual network resides, when you select a virtual network in the next step, it will not appear in the drop-down list.
- **Virtual network:** Choose the virtual network to which you want to add this gateway. Click **Virtual network** to open the 'Choose a virtual network' page. Select the VNet. If you don't see your VNet, make sure the Location field is pointing to the region in which your virtual network is located.
- **Gateway subnet address range:** You will only see this setting if you did not previously create a gateway subnet for your virtual network. If you previously created a valid gateway subnet, this setting will not appear.
- **First IP configuration:** The 'Choose public IP address' page creates a 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. VPN Gateway currently only supports *Dynamic* Public IP address allocation. However, this does not mean that the IP address changes after it has been assigned to your VPN gateway. 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.
 - First, click **Create gateway IP configuration** to open the 'Choose public IP address' page, then click **+Create new** to open the 'Create public IP address' page.
 - Next, input a **Name** for your public IP address. Leave the SKU as **Basic** unless there is a specific reason to change it to something else, then click **OK** at the bottom of this page to save your changes.



4. Verify the settings. You can select **Pin to dashboard** at the bottom of the page if you want your gateway to appear on the dashboard.
5. Click **Create** to begin creating the VPN gateway. The settings are validated and you'll see the "Deploying Virtual network gateway" tile on the dashboard. Creating a gateway can take up to 45 minutes. You may need to refresh your portal page to see the completed status.

After the gateway is created, 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. You can click the connected device (your virtual network gateway) to view more information.

5. 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.

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.

Create local network gateway X

* Name
 ✓

* IP address i
 ✓

Address space i

10.101.1.0/24 ...

10.101.0.0/24 ...

...

Configure BGP settings

* Subscription
 ▼

* Resource group i
 Create new Use existing

▼

* Location
 ▼

Pin to dashboard

Create [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. The IP address cannot be behind NAT and has to be reachable by Azure. 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 at the bottom of the page to create the local network gateway.

6. 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,

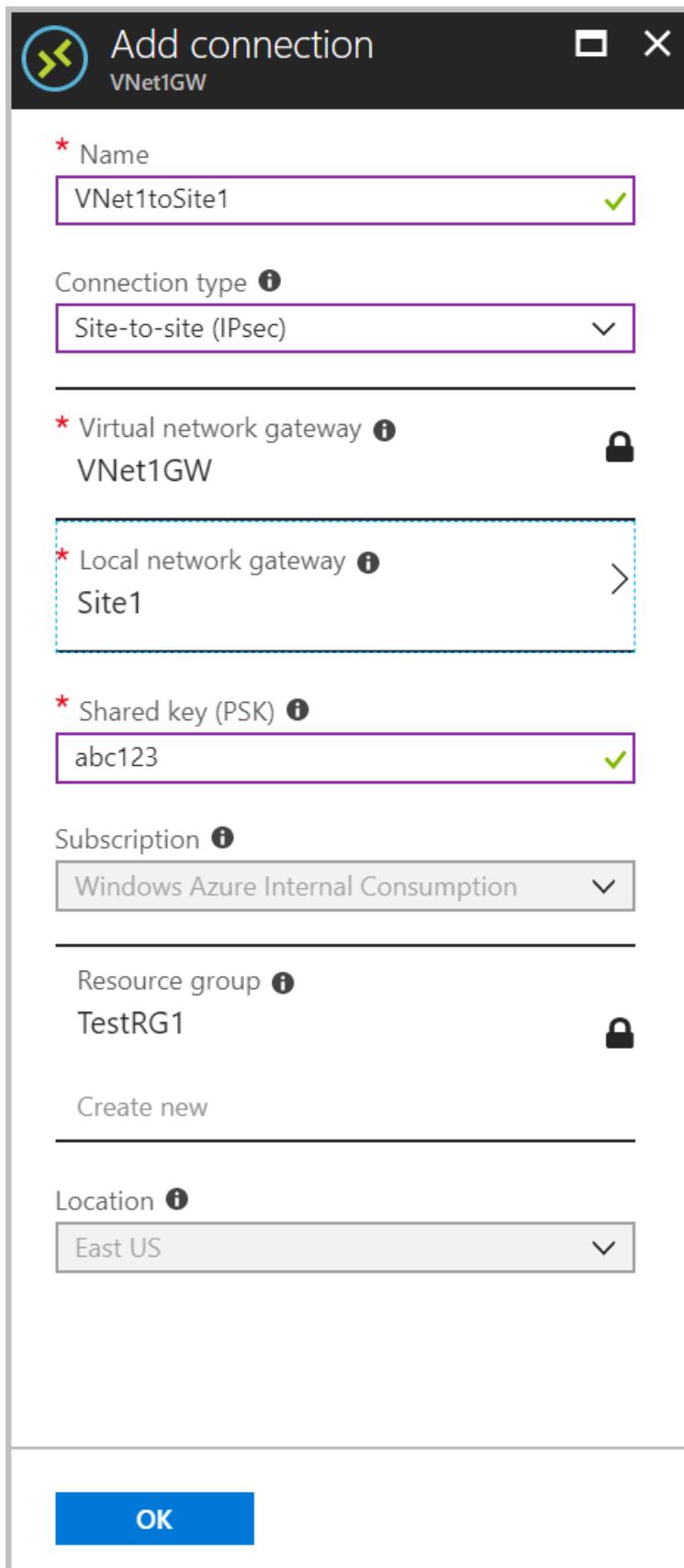
Authentication method, encryption and hashing algorithms, SA lifetime, PFS, and DPD, in addition to other parameter information that you need to complete your configuration.

- 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](#).

7. Create the VPN connection

Create the Site-to-Site VPN connection between your virtual network gateway and your on-premises VPN device.

1. Navigate to and open the page for your virtual network gateway. There are multiple ways to navigate. You can navigate to the gateway 'VNet1GW' by going to **TestVNet1 -> Overview -> Connected devices -> VNet1GW**.
2. On the page for VNet1GW, 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*.

8. 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](#), 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
Status Connected	Data out 3.14 KB
Location East US	Virtual network
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-AzureRmVM
$Nics = Get-AzureRmNetworkInterface | 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

3/9/2018 • 4 min 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 [Configure routing](#) article for 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 [Configure routing](#). 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 Peering location (Silicon Valley). It also lists Peering entries: Azure private (Enabled, 172.16.0.0/30, 172.16.0.4/30), Azure public (Disabled), and Microsoft (Disabled).
- Middle Window (Peerings):** Shows the 'Peerings' section with a table of peering configurations. One entry is highlighted with a red box: 'Azure private' (Type), 'Enabled' (Status), '172.16.0.0/30' (Primary subnet), and '172.16.0.4/30' (Secondary subnet).
- Right Window (Peering Details):** A detailed view of the selected peering entry. It shows 'TYPE: Azure private', 'STATUS: Enabled', '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 a red box around the 'Add' button.
- Middle Window (Connections):** Shows the 'Connections' table with a red box around the 'Connections' item in the navigation menu.
- Right Window (Add connection):** The 'Add connection' dialog box. It includes fields for Name (ER-VNet-Connection), Connection type (ExpressRoute), Virtual network gateway (Demo-VNet-GW), ExpressRoute circuit (ER-Demo-Ckt-SV), Subscription (ExpressRoute-Demo), Resource group (USWest-ER-Demo-RG), Location (West US), and a 'New' checkbox.

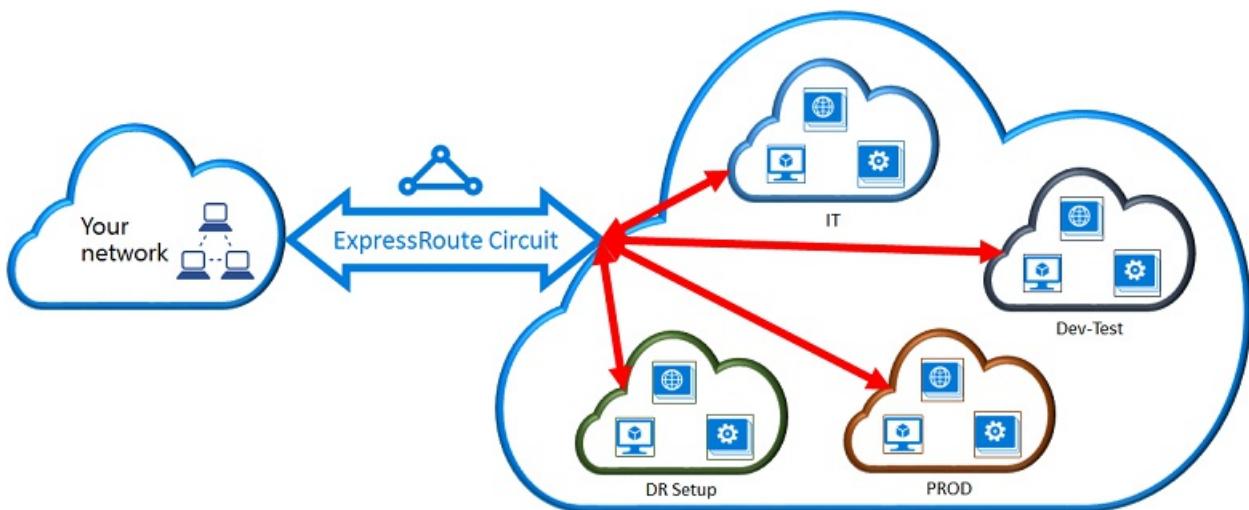
3. After your connection has been successfully configured, your connection object will show the information for the connection.

The figure shows two windows side-by-side:

- Left Window (Connections):** Shows the 'Connections' table with one entry: 'ER-VNet-Connection' (Status: Succeeded, Connection Type: ExpressRoute, Peer: Demo-VNet-GW).
- Right Window (ER-VNet-Connection):** A detailed view of the 'ER-VNet-Connection' object. It shows the 'Essentials' tab with information: Resource group (USWest-ER-Demo-RG), Status (Succeeded), Location (West US), Subscription name (ExpressRoute-Demo), and Subscription ID. A red box highlights the 'Virtual network' field, which lists 'Demo-VNet' and 'Virtual network gateway' (Demo-VNet-GW (13.88.21.182)).

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.

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. The left sidebar lists various management options like Overview, Activity log, Access control (IAM), Tags, and more. The 'Authorizations' section is selected and highlighted with a red box. A new authorization entry is being added, with the 'NAME' field containing 'IDemo' also highlighted with a red box. The 'PROVISIONING STATE' is listed as 'Pending' and 'USE STATUS' as 'Available'.

- Once the configuration is saved, copy the **Resource ID** and the **Authorization Key**.

This screenshot shows the same 'Authorizations' blade after the configuration has been saved. The 'IDemo' authorization now has a 'PROVISIONING STATE' of 'Success' and an 'AUTHORIZATION KEY' displayed in the 'KEY STATUS' column. The 'NAME' field still contains 'IDemo'.

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's main dashboard. On the left, there's a navigation bar with 'Preview' and 'Microsoft Azure' tabs. Below them is a 'New' button with a green plus sign, which is highlighted with a red box. To the right, there's a list of service categories: All resources, Resource groups, App Services, SQL databases, SQL data warehouses, NoSQL (DocumentDB), Virtual machines, and Load balancers.

- 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 list shows various connection-related items. The first item in the list is 'Connection' by Microsoft, which is highlighted with a red box. To the right of the results, there is a detailed description of what a VPN Connection is, mentioning it connects two Azure virtual networks or a virtual network to your local network using Internet Protocol security (IPsec). It also describes ExpressRoute and VNet-to-VNet connections. At the bottom right of the results area, there is a large red box around the '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 current step is 'Basics'. On the left, a navigation pane shows icons for Compute, Networking, Storage, and more. The main area has three steps: 1. Basics (Configure basic settings), 2. Settings (Configure connection settings), and 3. Summary (Review and create). Step 1 is active and highlighted in blue. On the right, the 'Basics' configuration panel shows the following fields: 'Connection type' dropdown set to 'ExpressRoute' (highlighted with a red box), 'Subscription' dropdown, 'Resource group' dropdown set to 'Default-Networking' (with 'Use existing' radio button selected), 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**.

A screenshot of the 'Create connection' wizard in the Microsoft Azure portal, showing the 'Settings' step. The left navigation pane is visible. The main area shows the 'Settings' configuration panel with the following fields: 'Virtual network gateway' dropdown, 'Redeem authorization' checkbox (highlighted with a red box and checked), 'Authorization key' input field, 'Peer circuit URI' input field (with a green checkmark icon), and 'Connection name' input field.

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](#).

4 min to read •

Virtual appliance scenario

10/30/2017 • 8 min 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 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 ARM 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 DMZ/protected network scenario.

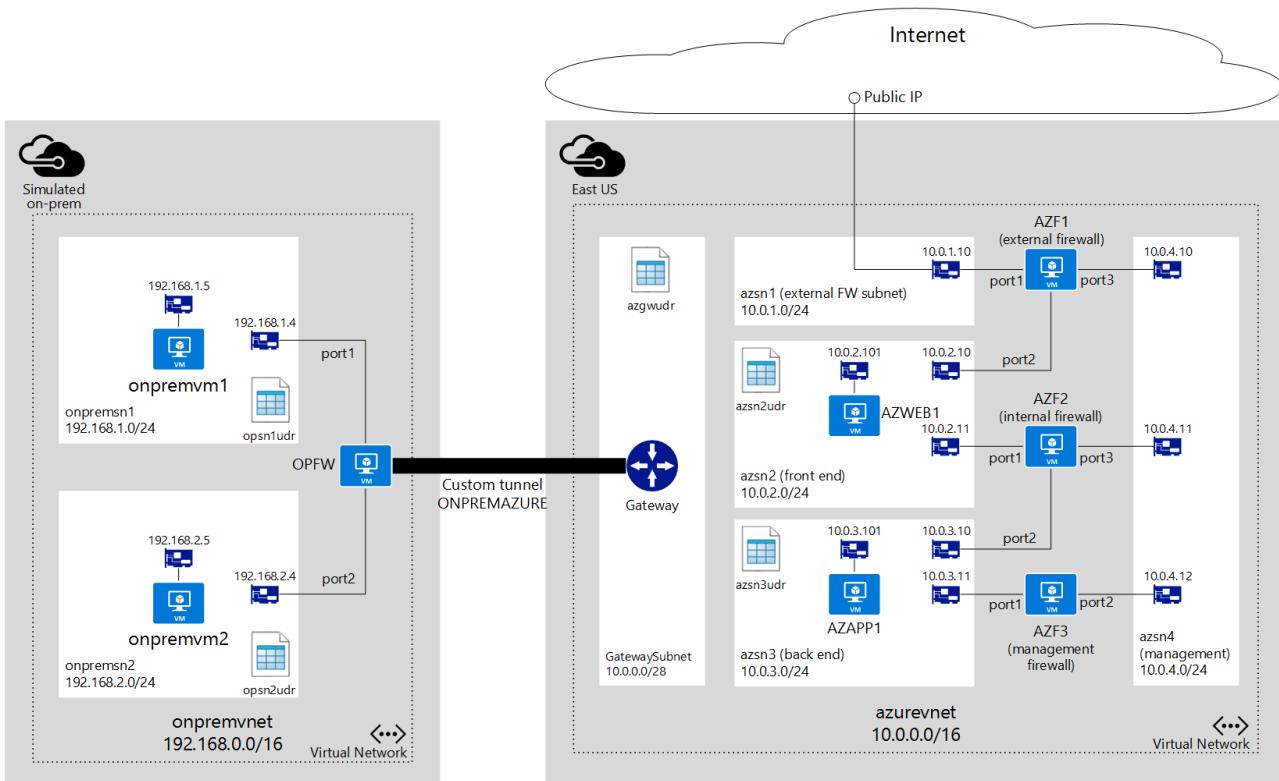
Considerations

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 AZF3

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 AZF2
0.0.0.0/0	10.0.2.10	Allows all other traffic to be routed through AZF1

azsn3udr

DESTINATION	NEXT HOP	EXPLANATION
10.0.2.0/24	10.0.3.10	Allows traffic to azsn2 to flow from app server to the webserver through AZF2

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 onpremsn2 through OPFW

onpremsn2udr

DESTINATION	NEXT HOP	EXPLANATION
10.0.3.0/24	192.168.2.4	Allows traffic to the backed subnet in Azure through OPFW

DESTINATION	NEXT HOP	EXPLANATION
192.168.1.0/24	192.168.2.4	Allows traffic to onpremsn1 through OPFW

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, log on to **onpremvn2** and ping 10.0.3.101 to test connectivity between **onpremsn2** and **azsn3**.

4 min to read •

Example 1 – Build a simple DMZ using NSGs with an Azure Resource Manager template

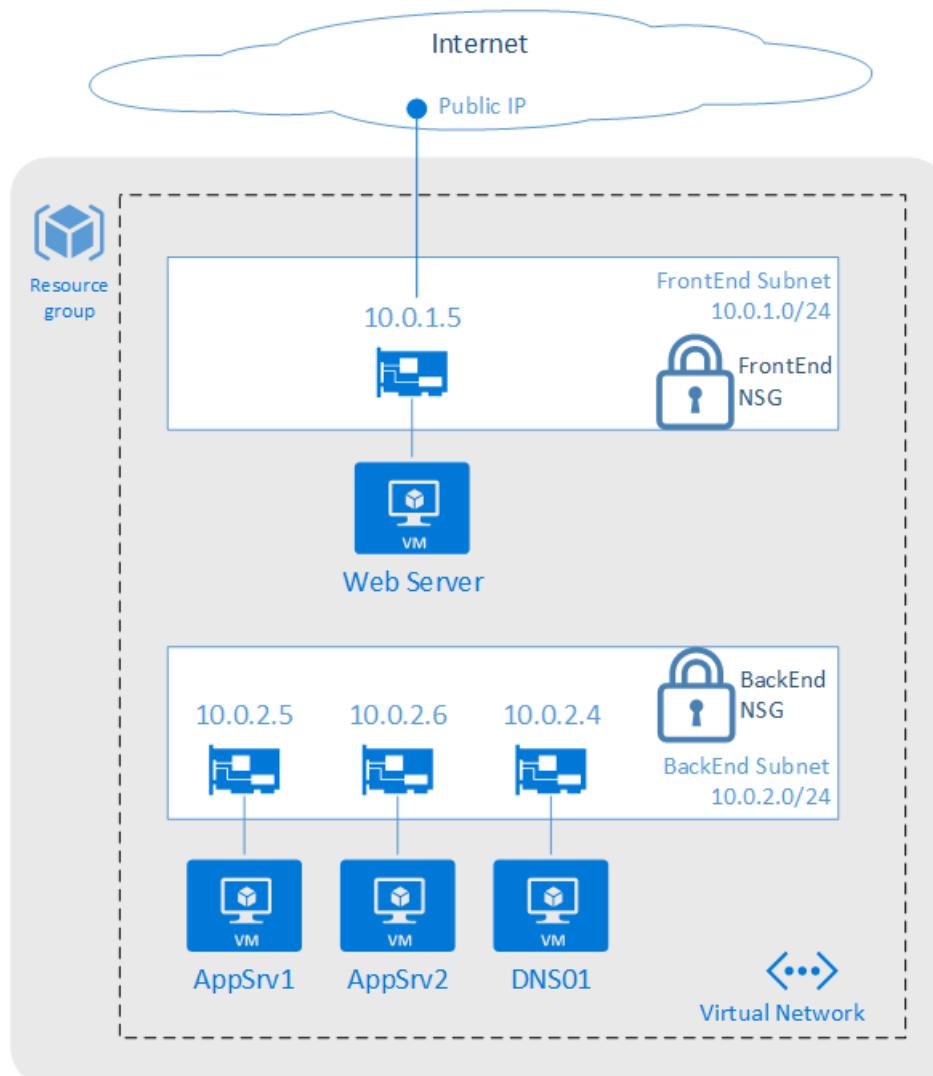
6/27/2017 • 14 min to read • [Edit Online](#)

[Return to the Security Boundary Best Practices Page](#)

This example creates a primitive DMZ with four Windows servers and Network Security Groups. This example describes each of the relevant template sections to provide a deeper understanding of each step. There is also a Traffic Scenario section to provide an in-depth step-by-step look at how traffic proceeds through the layers of defense in the DMZ. Finally, in the references section is the complete template code and instructions to build this environment to test and experiment with various scenarios.

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.



[Environment description](#)

In this example a subscription contains the following resources:

- A single resource group
- A Virtual Network with two subnets; "FrontEnd" and "BackEnd"
- A Network Security Group that is applied to both subnets
- A Windows Server that represents an application web server ("IIS01")
- Two windows servers that represent application back-end servers ("AppVM01", "AppVM02")
- A Windows server that represents a DNS server ("DNS01")
- A public IP address associated with the application web server

In the references section, there is a link to an Azure Resource Manager template that builds the environment described in this example. Building the VMs and Virtual Networks, although done by the example template, are not described in detail in this document.

To build this environment (detailed instructions are in the references section of this document);

1. Deploy the Azure Resource Manager Template at: [Azure Quickstart Templates](#)
2. Install the sample application at: [Sample Application Script](#)

NOTE

To RDP to any back-end servers in this instance, the IIS server is used as a "jump box." First RDP to the IIS server and then from the IIS Server RDP to the back-end server. Alternately a Public IP can be associated with each server NIC for easier RDP.

The following sections provide a detailed description of the Network Security Group and how it functions for this example by walking through key lines of the Azure Resource Manager Template.

Network Security Groups (NSG)

For this example, an NSG group is built and then loaded with six rules.

TIP

Generally speaking, you should create your specific "Allow" rules first and then the more generic "Deny" rules last. The assigned priority dictates which rules are evaluated first. Once traffic is found to apply to a specific rule, no further rules are evaluated. NSG rules can apply in either in the inbound or outbound direction (from the perspective of the subnet).

Declaratively, the following rules are being built for inbound traffic:

1. Internal DNS traffic (port 53) is allowed
2. RDP traffic (port 3389) from the Internet to any VM is allowed
3. HTTP traffic (port 80) from the Internet to web server (IIS01) is allowed
4. Any traffic (all ports) from IIS01 to AppVM1 is allowed
5. Any traffic (all ports) from the Internet to the entire VNet (both subnets) is Denied
6. Any traffic (all ports) from the Frontend subnet to the Backend subnet is Denied

With these rules bound to each subnet, if an HTTP request was inbound from the Internet to the web server, both rules 3 (allow) and 5 (deny) would apply, but since rule 3 has a higher priority only it would apply and rule 5 would not come into play. Thus the HTTP request would be allowed to the web server. If that same traffic was trying to reach the DNS01 server, rule 5 (Deny) would be the first to apply and the traffic would not be allowed to pass to the server. Rule 6 (Deny) blocks the Frontend subnet from talking to the Backend subnet (except for allowed traffic in rules 1 and 4), this rule-set protects the Backend network in case an attacker compromises the web application on the Frontend, the attacker would have limited access to the Backend "protected" network (only to resources

exposed on the AppVM01 server).

There is a default outbound rule that allows traffic out to the internet. For this example, we're allowing outbound traffic and not modifying any outbound rules. To apply security policy to traffic in both directions, User Defined Routing is required and is explored in "Example 3" on the [Security Boundary Best Practices Page](#).

Each rule is discussed in more detail as follows:

1. A Network Security Group resource must be instantiated to hold the rules:

```
"resources": [  
    {  
        "apiVersion": "2015-05-01-preview",  
        "type": "Microsoft.Network/networkSecurityGroups",  
        "name": "[variables('NSGName')]",  
        "location": "[resourceGroup().location]",  
        "properties": {}  
    }  
]
```

2. The first rule in this example allows DNS traffic between all internal networks to the DNS server on the backend subnet. The rule has some important parameters:

- "destinationAddressPrefix" - Rules can use a special type of address prefix called a "Default Tag", these tags are system-provided identifiers that allow an easy way to address a larger category of address prefixes. This rule uses the Default Tag "Internet" to signify any address outside of the VNet. Other prefix labels are VirtualNetwork and AzureLoadBalancer.
- "Direction" signifies in which direction of traffic flow this rule takes effect. The direction is from the perspective of the subnet or Virtual Machine (depending on where this NSG is bound). Thus if Direction is "Inbound" and traffic is entering the subnet, the rule would apply and traffic leaving the subnet would not be affected by this rule.
- "Priority" sets the order in which a traffic flow is evaluated. The lower the number the higher the priority. When a rule applies to a specific traffic flow, no further rules are processed. Thus if a rule with priority 1 allows traffic, and a rule with priority 2 denies traffic, and both rules apply to traffic then the traffic would be allowed to flow (since rule 1 had a higher priority it took effect and no further rules were applied).
- "Access" signifies if traffic affected by this rule is blocked ("Deny") or allowed ("Allow").

```
"properties": {},  
"securityRules": [  
    {  
        "name": "enable_dns_rule",  
        "properties": {  
            "description": "Enable Internal DNS",  
            "protocol": "*",  
            "sourcePortRange": "*",  
            "destinationPortRange": "53",  
            "sourceAddressPrefix": "VirtualNetwork",  
            "destinationAddressPrefix": "10.0.2.4",  
            "access": "Allow",  
            "priority": 100,  
            "direction": "Inbound"  
        }  
    },  
],
```

3. This rule allows RDP traffic to flow from the internet to the RDP port on any server on the bound subnet.

```
{
  "name": "enable_rdp_rule",
  "properties": {
    "description": "Allow RDP",
    "protocol": "Tcp",
    "sourcePortRange": "*",
    "destinationPortRange": "3389",
    "sourceAddressPrefix": "*",
    "destinationAddressPrefix": "*",
    "access": "Allow",
    "priority": 110,
    "direction": "Inbound"
  }
},
}
```

4. This rule allows inbound internet traffic to hit the web server. This rule does not change the routing behavior. The rule only allows traffic destined for IIS01 to pass. Thus if traffic from the Internet had the web server as its destination this rule would allow it and stop processing further rules. (In the rule at priority 140 all other inbound internet traffic is blocked). If you're only processing HTTP traffic, this rule could be further restricted to only allow Destination Port 80.

```
{
  "name": "enable_web_rule",
  "properties": {
    "description": "Enable Internet to [variables('VM01Name')]",
    "protocol": "Tcp",
    "sourcePortRange": "*",
    "destinationPortRange": "80",
    "sourceAddressPrefix": "Internet",
    "destinationAddressPrefix": "10.0.1.5",
    "access": "Allow",
    "priority": 120,
    "direction": "Inbound"
  }
},
}
```

5. This rule allows traffic to pass from the IIS01 server to the AppVM01 server, a later rule blocks all other Frontend to Backend traffic. To improve this rule, if the port is known that should be added. For example, if the IIS server is hitting only SQL Server on AppVM01, the Destination Port Range should be changed from "*" (Any) to 1433 (the SQL port) thus allowing a smaller inbound attack surface on AppVM01 should the web application ever be compromised.

```
{
  "name": "enable_app_rule",
  "properties": {
    "description": "Enable [variables('VM01Name')] to [variables('VM02Name')]",
    "protocol": "*",
    "sourcePortRange": "*",
    "destinationPortRange": "*",
    "sourceAddressPrefix": "10.0.1.5",
    "destinationAddressPrefix": "10.0.2.5",
    "access": "Allow",
    "priority": 130,
    "direction": "Inbound"
  }
},
}
```

6. This rule denies traffic from the internet to any servers on the network. With the rules at priority 110 and 120, the effect is to allow only inbound internet traffic to the firewall and RDP ports on servers and blocks everything else. This rule is a "fail-safe" rule to block all unexpected flows.

```
{
  "name": "deny_internet_rule",
  "properties": {
    "description": "Isolate the [variables('VNetName')] VNet from the Internet",
    "protocol": "*",
    "sourcePortRange": "*",
    "destinationPortRange": "*",
    "sourceAddressPrefix": "Internet",
    "destinationAddressPrefix": "VirtualNetwork",
    "access": "Deny",
    "priority": 140,
    "direction": "Inbound"
  }
},
}
```

7. The final rule denies traffic from the Frontend subnet to the Backend subnet. Since this rule is an Inbound only rule, reverse traffic is allowed (from the Backend to the Frontend).

```
{
  "name": "deny_frontend_rule",
  "properties": {
    "description": "Isolate the [variables('Subnet1Name')] subnet from the [variables('Subnet2Name')] subnet",
    "protocol": "*",
    "sourcePortRange": "*",
    "destinationPortRange": "*",
    "sourceAddressPrefix": "[variables('Subnet1Prefix')]",
    "destinationAddressPrefix": "[variables('Subnet2Prefix')]",
    "access": "Deny",
    "priority": 150,
    "direction": "Inbound"
  }
}
```

Traffic scenarios

(Allowed) Internet to web server

1. An internet user requests an HTTP page from the public IP address of the NIC associated with the IIS01 NIC
2. The Public IP address passes traffic to the VNet towards IIS01 (the web server)
3. Frontend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to IIS01) does apply, traffic is allowed, stop rule processing
4. Traffic hits internal IP address of the web server IIS01 (10.0.1.5)
5. IIS01 is listening for web traffic, receives this request and starts processing the request
6. IIS01 asks the SQL Server on AppVM01 for information
7. No outbound rules on Frontend subnet, traffic is allowed
8. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to Firewall) doesn't apply, move to next rule
 - d. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
9. AppVM01 receives the SQL Query and responds
10. Since there are no outbound rules on the Backend subnet, the response is allowed
11. Frontend subnet begins inbound rule processing:

- a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
12. The IIS server receives the SQL response and completes the HTTP response and sends to the requester
 13. Since there are no outbound rules on the Frontend subnet, the response is allowed and the Internet User receives the web page requested.

(Allowed) RDP to IIS server

1. A Server Admin on internet requests an RDP session to IIS01 on the public IP address of the NIC associated with the IIS01 NIC (this public IP address can be found via the Portal or PowerShell)
2. The Public IP address passes traffic to the VNet towards IIS01 (the web server)
3. Frontend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) does apply, traffic is allowed, stop rule processing
4. With no outbound rules, default rules apply and return traffic is allowed
5. RDP session is enabled
6. IIS01 prompts for the user name and password

NOTE

To RDP to any back-end servers in this instance, the IIS server is used as a "jump box." First RDP to the IIS server and then from the IIS Server RDP to the back-end server.

(Allowed) Web server DNS look-up on DNS server

1. Web Server, IIS01, needs a data feed at www.data.gov, but needs to resolve the address.
2. The network configuration for the VNet lists DNS01 (10.0.2.4 on the Backend subnet) as the primary DNS server, IIS01 sends the DNS request to DNS01
3. No outbound rules on Frontend subnet, traffic is allowed
4. Backend subnet begins inbound rule processing:
 - NSG Rule 1 (DNS) does apply, traffic is allowed, stop rule processing
5. DNS server receives the request
6. DNS server doesn't have the address cached and asks a root DNS server on the internet
7. No outbound rules on Backend subnet, traffic is allowed
8. Internet DNS server responds, since this session was initiated internally, the response is allowed
9. DNS server caches the response, and responds to the initial request back to IIS01
10. No outbound rules on Backend subnet, traffic is allowed
11. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed
12. IIS01 receives the response from DNS01

(Allowed) Web server access file on AppVM01

1. IIS01 asks for a file on AppVM01
2. No outbound rules on Frontend subnet, traffic is allowed
3. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to IIS01) doesn't apply, move to next rule

- d. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
- 4. AppVM01 receives the request and responds with file (assuming access is authorized)
- 5. Since there are no outbound rules on the Backend subnet, the response is allowed
- 6. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
- 7. The IIS server receives the file

(Denied) RDP to backend

1. An internet user tries to RDP to server AppVM01
2. Since there are no Public IP addresses associated with this servers NIC, this traffic would never enter the VNet and wouldn't reach the server
3. However if a Public IP address was enabled for some reason, NSG rule 2 (RDP) would allow this traffic

NOTE

To RDP to any back-end servers in this instance, the IIS server is used as a "jump box." First RDP to the IIS server and then from the IIS Server RDP to the back-end server.

(Denied) Web to backend server

1. An internet user tries to access a file on AppVM01
2. Since there are no Public IP addresses associated with this servers NIC, this traffic would never enter the VNet and wouldn't reach the server
3. If a Public IP address was enabled for some reason, NSG rule 5 (Internet to VNet) would block this traffic

(Denied) Web DNS look-up on DNS server

1. An internet user tries to look up an internal DNS record on DNS01
2. Since there are no Public IP addresses associated with this servers NIC, this traffic would never enter the VNet and wouldn't reach the server
3. If a Public IP address was enabled for some reason, NSG rule 5 (Internet to VNet) would block this traffic (Note: that Rule 1 (DNS) would not apply because the requests source address is the internet and Rule 1 only applies to the local VNet as the source)

(Denied) SQL access on the web server

1. An internet user requests SQL data from IIS01
2. Since there are no Public IP addresses associated with this servers NIC, this traffic would never enter the VNet and wouldn't reach the server
3. If a Public IP address was enabled for some reason, the Frontend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to IIS01) does apply, traffic is allowed, stop rule processing
4. Traffic hits internal IP address of the IIS01 (10.0.1.5)
5. IIS01 isn't listening on port 1433, so no response to the request

Conclusion

This example is a relatively simple and straight forward way of isolating the back-end subnet from inbound traffic.

More examples and an overview of network security boundaries can be found [here](#).

References

Azure Resource Manager template

This example uses a predefined Azure Resource Manager template in a GitHub repository maintained by Microsoft and open to the community. This template can be deployed straight out of GitHub, or downloaded and modified to fit your needs.

The main template is in the file named "azuredeploy.json." This template can be submitted via PowerShell or CLI (with the associated "azuredeploy.parameters.json" file) to deploy this template. I find the easiest way is to use the "Deploy to Azure" button on the README.md page at GitHub.

To deploy the template that builds this example from GitHub and the Azure portal, follow these steps:

1. From a browser, navigate to the [Template](#)
2. Click the "Deploy to Azure" button (or the "Visualize" button to see a graphical representation of this template)
3. Enter the Storage Account, User Name, and Password in the Parameters blade, then click **OK**
4. Create a Resource Group for this deployment (You can use an existing one, but I recommend a new one for best results)
5. If necessary, change the Subscription and Location settings for your VNet.
6. Click **Review legal terms**, read the terms, and click **Purchase** to agree.
7. Click **Create** to begin the deployment of this template.
8. Once the deployment finishes successfully, navigate to the Resource Group created for this deployment to see the resources configured inside.

NOTE

This template enables RDP to the IIS01 server only (find the Public IP for IIS01 on the Portal). To RDP to any back-end servers in this instance, the IIS server is used as a "jump box." First RDP to the IIS server and then from the IIS Server RDP to the back-end server.

To remove this deployment, delete the Resource Group and all child resources will also be deleted.

Sample application scripts

Once the template runs successfully, you can set up the web server and app server with a simple web application to allow testing with this DMZ configuration. To install a sample application for this, and other DMZ Examples, one has been provided at the following link: [Sample Application Script](#)

Next steps

- Deploy this example
- Build the sample application
- Test different traffic flows through this DMZ

Add network interfaces to or remove network interfaces from virtual machines

4/18/2018 • 8 min 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

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 5.2.0 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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 2.0](#). 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-AzureRmVM</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.

TOOL	COMMAND
CLI	az vm nic add (reference) or detailed steps
PowerShell	Add-AzureRmVMNetworkInterface (reference) or detailed steps

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	az vm show
PowerShell	Get-AzureRmVM

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	az vm nic remove (reference) or detailed steps
PowerShell	Remove-AzureRMVMNetworkInterface (reference) or detailed steps

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.
- If a network interface has a private IPv6 address assigned to it, you must add (attach) it to a VM when you create the VM. You cannot add a network interface with an assigned IPv6 address to a VM after you create the VM. If you add a network interface with an assigned private IPv6 address when you create a virtual machine, you can only add that network interface to the virtual machine, regardless of how many network interfaces the VM size supports. See [Manage network interface IP addresses](#) to learn more about how to assign IP addresses to network interfaces.
- 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, read the following articles:

Commands

TASK	TOOL
Create a VM with multiple NICs	CLI , PowerShell
Create a single NIC VM with multiple IPv4 addresses	CLI , PowerShell
Create a single NIC VM with a private IPv6 address (behind an Azure Load Balancer)	CLI , PowerShell , Azure Resource Manager template

Name resolution for resources in Azure virtual networks

4/18/2018 • 12 min 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 may want to use the Azure DNS Private Zones feature, which is currently in Public Preview. 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.	Azure DNS Private Zones or Azure-provided name resolution	Hostname or FQDN
Name resolution between VMs in different virtual networks or role instances in different cloud services.	Azure DNS Private Zones or, Customer-managed DNS servers forwarding queries between virtual networks for resolution by Azure (DNS proxy). See Name resolution using your own DNS server .	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 Name resolution using your own DNS server .	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 Name resolution using your own DNS server .	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 Name resolution using your own DNS server .	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 Name resolution using your own DNS server .	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 Name resolution using your own DNS server .	FQDN only
Reverse DNS for internal IPs.	Name resolution using your own DNS server .	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.

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.
- **OpenLogic (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/head**.
 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.
- **OpenLogic** (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.

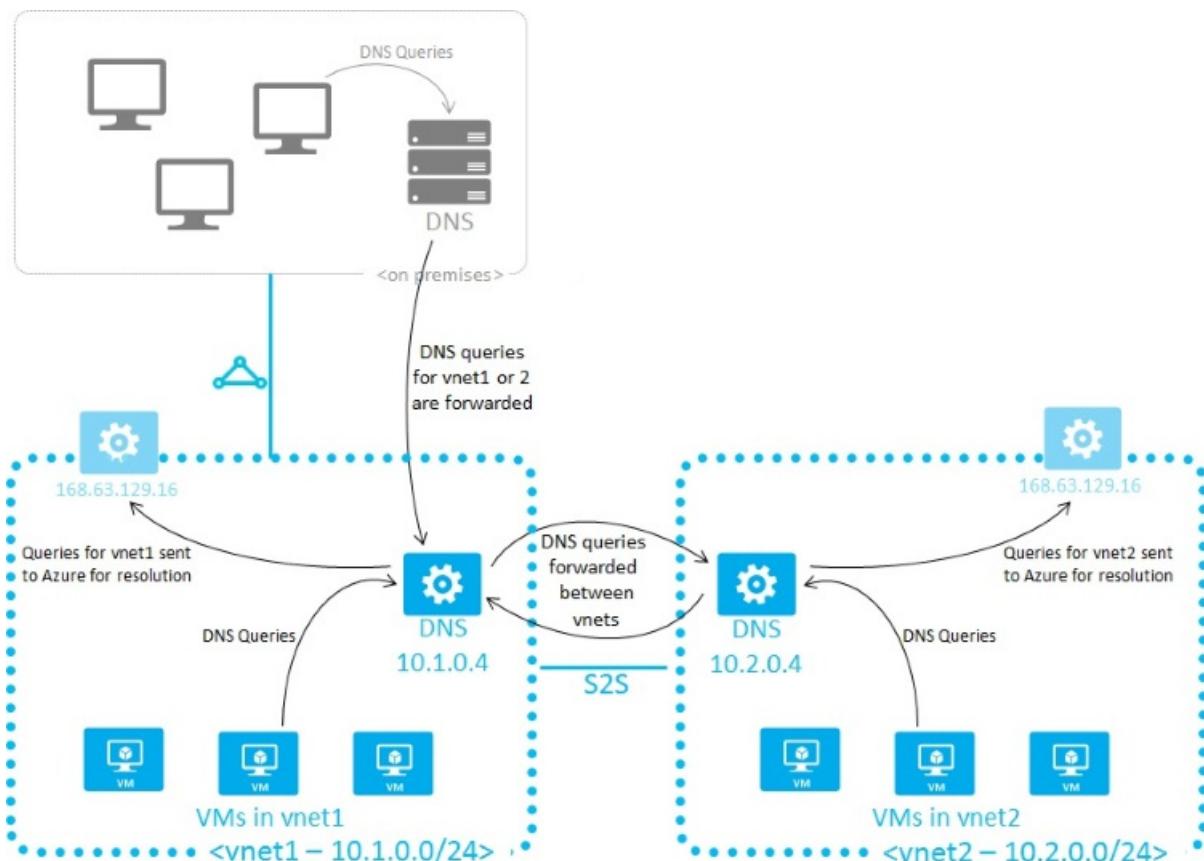
DNS forwarding also enables DNS resolution between virtual networks, and allows your on-premises machines to resolve Azure-provided host names. In order to resolve a VM's host name, the DNS server VM must reside in the same virtual network, and be configured to forward host name queries to Azure. Because 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 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-AzureRmNetworkInterface](#) 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.

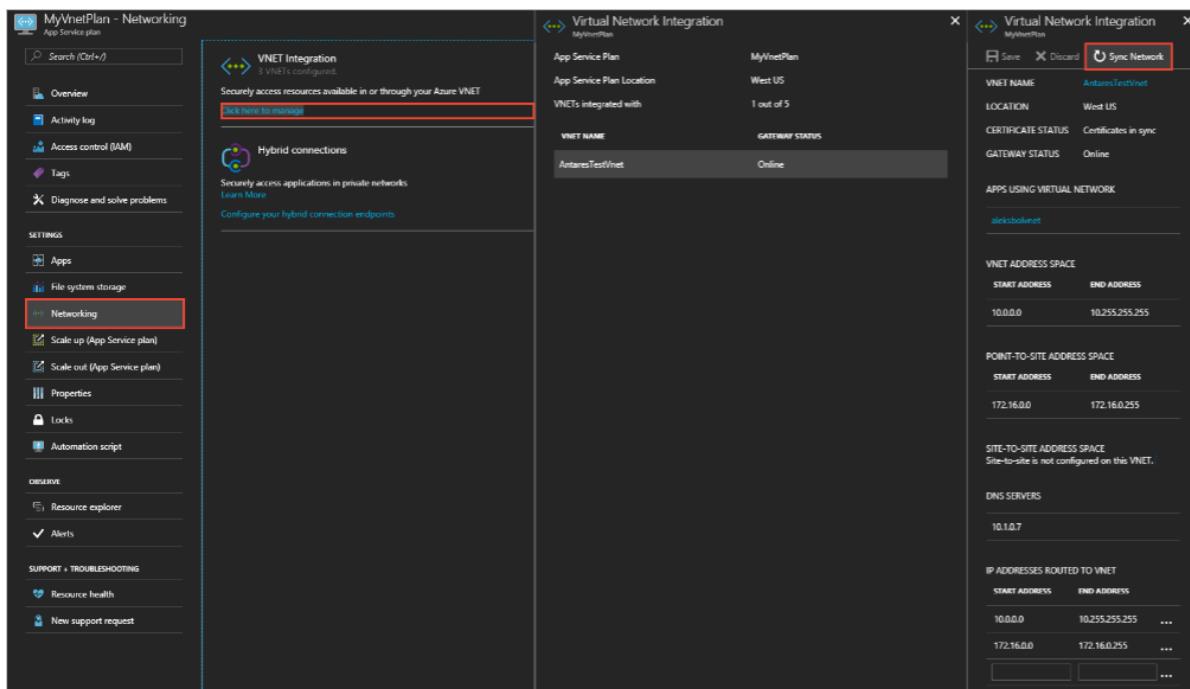
NOTE

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. For additional performance analysis and optimizations when you are using Windows Server as your DNS server, see [Name resolution performance of a recursive Windows DNS Server 2012 R2](#).

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 Windows VMs. This is because they might get erased during service heal when the virtual network adaptor gets replaced.

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](#).

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, you need to restart each affected VM for the changes to take effect.

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

4/18/2018 • 3 min 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

2/8/2018 • 3 min 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:

```
mkdir lis4.2.3-5  
cd lis4.2.3-5  
wget https://download.microsoft.com/download/6/8/F/68FE11B8-FAA4-4F8D-8C7D-74DA7F2CFC8C/lis-rpms-4.2.3-  
5.tar.gz  
tar xvzf lis-rpms-4.2.3-5.tar.gz  
cd LISISO  
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

4/19/2018 • 2 min 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.

Azure Portal

You can use the [Azure portal](#) to view the host names for virtual machines on the overview blade for a virtual machine. Keep in mind that the blade shows a value for **Name** and **Host Name**. Although they are initially the same, changing the host name will not change the name of the virtual machine or role instance.

Role instances can also be viewed in the Azure portal, but when you list the instances in a cloud service, the host name is not displayed. You will see a name for each instance, but that name does not represent the host name.

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](#)

Create, change, or delete a virtual network

4/25/2018 • 13 min to read • [Edit Online](#)

Learn how to create and delete a virtual network and change settings, like DNS servers and IP address spaces, for an existing virtual network.

A virtual network is a representation of your own network in the cloud. A virtual network is a logical isolation of the Azure cloud that is dedicated to your Azure subscription. For each virtual network that you create, you can:

- Choose an address space to assign. An address space consists of one or more address ranges that are defined by using Classless Inter-Domain Routing (CIDR) notation, like 10.0.0.0/16.
- Choose to use the Azure-provided DNS server, or use your own DNS server. All resources that are connected to the virtual network are assigned this DNS server to resolve names within the virtual network.
- Segment the virtual network into subnets, each with its own address range, within the address space of the virtual network. To learn how to create, change, and delete subnets, see [Add, change, or delete 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 5.2.0 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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 2.0](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

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)

Although you can define only one address range when you create the virtual network, 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 Azures 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.

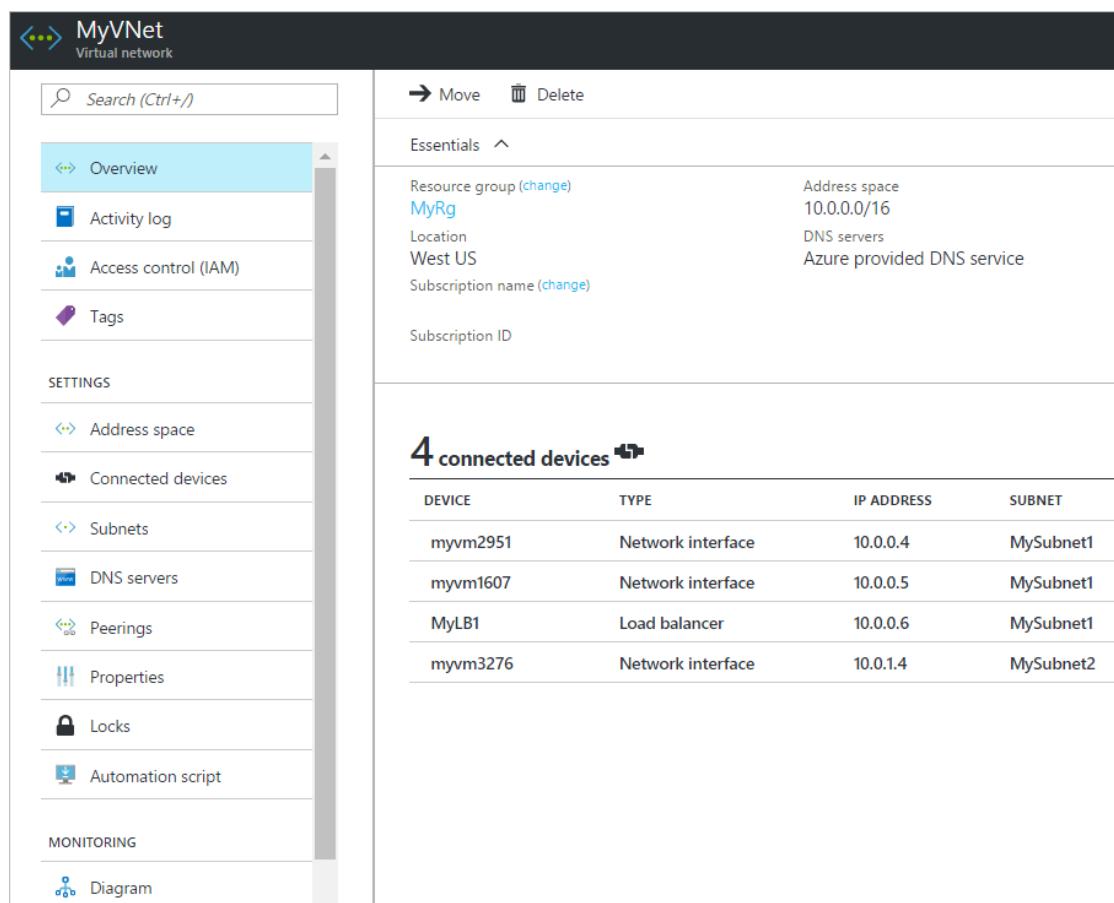
Commands

- Azure CLI: [az network vnet create](#)
- PowerShell: [New-AzureRmVirtualNetwork](#)

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** appears 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
myvm1607	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 appears 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-AzureRmVirtualNetwork`

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 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)

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** appears 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-AzureRmVirtualNetwork](#)

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** appears 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-AzureRmVirtualNetwork](#)

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** appears 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: [azure network vnet delete](#)
- PowerShell: [Remove-AzureRmVirtualNetwork](#)

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 permissions listed in the following table:

OPERATION	OPERATION NAME
Microsoft.Network/virtualNetworks/read	Get Virtual Network
Microsoft.Network/virtualNetworks/write	Create or Update Virtual Network
Microsoft.Network/virtualNetworks/delete	Delete Virtual Network

Next steps

- To create a VM and then connect it to a virtual network, see [Create a virtual network and connect VMs](#).
- To filter network traffic between subnets within a virtual network, see [Create network security groups](#).
- To peer a virtual network to another virtual network, see [Create a virtual network peering](#).
- To learn about options for connecting a virtual network to an on-premises network, see [About VPN Gateway](#).

Add, change, or delete a virtual network subnet

4/18/2018 • 6 min to read • [Edit Online](#)

Learn how to add, change, or delete a virtual network subnet. If you're not familiar with virtual networks, before you add, change, or delete a subnet, we recommend that you read [Azure Virtual Network overview](#) and [Create, change, or delete a virtual network](#). All Azure resources deployed into a virtual network are deployed into a subnet within a virtual network.

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 5.2.0 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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 2.0](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

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.
 - **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. To remove a service endpoint, unselect the service you want to remove the service endpoint for. To learn more about service endpoints, 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.

6. To add the subnet to the virtual network that you selected, select **OK**.

Commands

- Azure CLI: [az network vnet subnet create](#)
- PowerShell: [Add-AzureRmVirtualNetworkSubnetConfig](#)

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](#).
 - For information about changing **Network security group**, **Route table**, **Users**, and **Service endpoints**, see step 5 in [Add a subnet](#).
5. Select **Save**.

Commands

- Azure CLI: [az network vnet subnet update](#)
- PowerShell: [Set-AzureRmVirtualNetworkSubnetConfig](#)

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 delete](#)
- PowerShell: [Remove-AzureRmVirtualNetworkSubnetConfig](#)

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 permissions listed in the following table:

OPERATION	OPERATION NAME
Microsoft.Network/virtualNetworks/subnets/read	Get Virtual Network Subnet
Microsoft.Network/virtualNetworks/subnets/write	Create or Update Virtual Network Subnet
Microsoft.Network/virtualNetworks/subnets/delete	Delete Virtual Network Subnet
Microsoft.Network/virtualNetworks/subnets/join/action	Join Virtual Network
Microsoft.Network/virtualNetworks/subnets/joinViaServiceEndpoint/action	Join Service to a Subnet
Microsoft.Network/virtualNetworks/subnets/virtualMachines/read	Get Virtual Network Subnet Virtual Machines

Create, change, or delete a virtual network peering

4/30/2018 • 15 min to read • [Edit Online](#)

Learn how to create, change, or delete a virtual network peering. Virtual network peering enables you to connect virtual networks through the Azure backbone network. Once peered, the virtual networks are still managed as separate resources. If you're not familiar with virtual network peering, we recommend reading the [Virtual network peering overview](#) and completing the [Create a virtual network peering tutorial](#), before completing the tasks in this article.

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 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 5.5.0 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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 2.0.29 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` with an account that has the [necessary permissions](#) to work with peering, to create a connection with Azure.

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** appears 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. From the list of virtual networks, select the virtual network you want to create a peering for.
4. Under **SETTINGS**, select **Peerings**.
5. Select **+ Add**.
6. 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//resourceGroups//providers/Microsoft.Network/virtualNetworks/`. 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](#).

- **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** default tag for network security groups encompasses the virtual network and peered virtual network. To learn more about network security group default tags, read the [Network security groups overview](#) article. 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 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). You cannot enable this option if you're peering virtual networks in different regions.

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. This 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. You cannot enable this option if you're peering virtual networks in different regions.

- **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. You cannot enable this option if you're peering virtual networks in different regions. To learn more about using a gateway for transit, see [Configure a VPN gateway for transit in a virtual network peering](#)

7. 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** appears 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. From the list of virtual networks, select the virtual network you want to change peering settings for.
4. Under **SETTINGS**, select **Peerings**.
5. Select the peering you want to view or change settings for.
6. Change the appropriate setting. Read about the options for each setting in [step 6](#) of Create a peering.
7. 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-AzureRmVirtualNetworkPeering` to retrieve view peering settings and `Set-AzureRmVirtualNetworkPeering` 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** appears 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. From the list of virtual networks, select the virtual network you want to delete a peering for.
4. Under **SETTINGS**, select **Peerings**.
5. 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.
6. 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-AzureRmVirtualNetworkPeering`

Requirements and constraints

- 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.
- The subscriptions that both virtual networks that you want to peer are in, must be associated to the same Azure Active Directory tenant. If you don't already have an AD tenant, you can quickly [create one](#). You can use a [VPN Gateway](#) to connect two virtual networks that exist in different subscriptions associated to different Active Directory tenants.
- 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.
- There is a nominal charge for ingress and egress traffic that utilizes a virtual network peering. For more information, see the [pricing page](#).
- You can peer virtual networks in the same region, or different regions. The following constraints do not apply when both virtual networks are in the *same* region, but do apply when the virtual networks are globally peered:
 - The virtual networks can exist in any Azure public cloud region, but not in Azure national clouds.
 - Resources in one virtual network cannot communicate with the IP address of an Azure internal load balancer in the peered virtual network. The load balancer and the resources that communicate with it must be in the same virtual network.
 - You cannot use remote gateways or allow gateway transit. To use remote gateways or allow gateway transit, both virtual networks in the peering must exist in the same region.

Permissions

The accounts you use to create a virtual network peering must have the necessary role or permissions. For example, if you are peering two virtual networks named *myVnetA* and *myVnetB*, your account must be assigned the following minimum role or permissions for each virtual network:

VIRTUAL NETWORK	DEPLOYMENT MODEL	ROLE	PERMISSIONS
myVnetA	Resource Manager	Network Contributor	Microsoft.Network/virtualNetworks/virtualNetworkPeerings/write
	Classic	Classic Network Contributor	N/A
myVnetB	Resource Manager	Network Contributor	Microsoft.Network/virtualNetworks/peer
	Classic	Classic Network Contributor	Microsoft.ClassicNetwork/virtualNetworks/peer

Learn more about [built-in roles](#) and assigning specific permissions to [custom roles](#) (Resource Manager only).

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](#).

Manage Azure DDoS Protection Standard using the Azure portal

4/17/2018 • 8 min 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>. 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 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 Create new 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 Use existing , and then select myResourceGroup
Location	East US
DDoS protection	Select Standard and then under DDoS protection , select myDdosProtectionPlan . 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.

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**.

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 Virtual network for Azure services , except for Azure App Service Environments and Azure VPN Gateway. To continue with this tutorial, you can quickly create a Windows or Linux virtual machine.
Metric	Under DDoS attack or not
Threshold	1 - 1 means you are under attack. 0 means you are not under attack.
Period	Select whatever value you choose.
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:

Alerts fired: 1

Activity log errors: 0

Service Health:

- Service Issues: 0
- Planned Maintenance: 0
- Health Advisories: 0

Alert sources (1) Application Insights (2)

Filter alerts...

NAME	STATUS	CONDITION	SOURCE
DDoS Attack -App Gateway...	⚠ Warning	Failed locations >= 3	Metrics

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.

Configure logging for DDoS protection metrics

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 following data** 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 OMS Log Analytics service. To learn more about this option, see [Collect logs for use in Log Analytics](#).

To simulate a DDoS attack to validate logging, see [Validate DDoS detection](#).

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.

The metric names present different packet types, and bytes vs. packets, with a basic construct of tag names on each

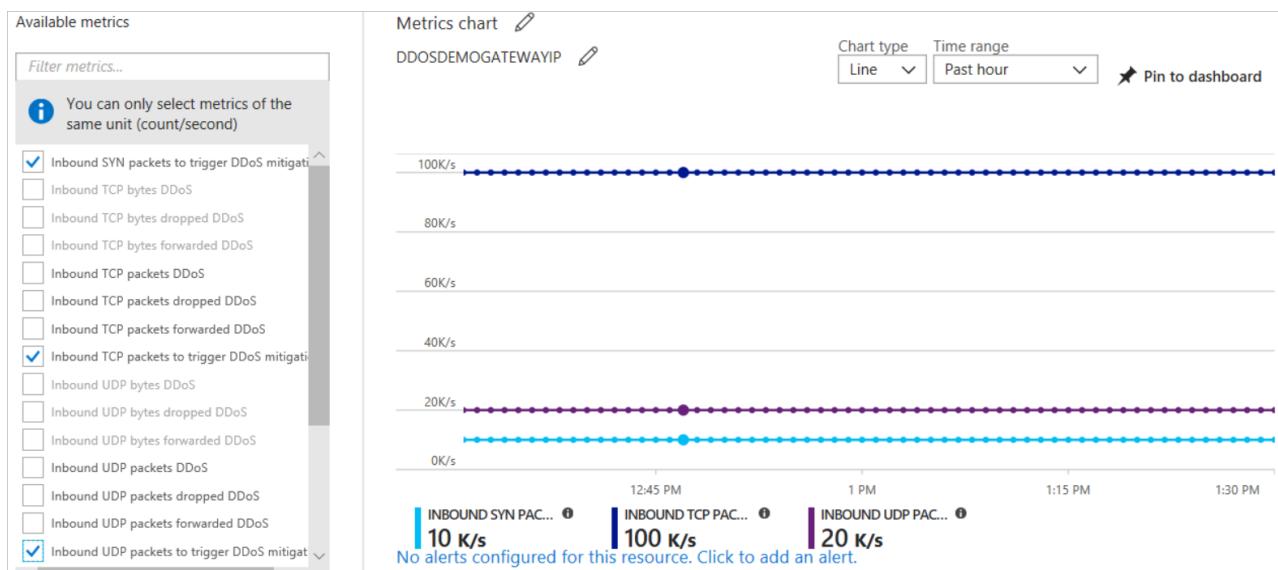
metric as follows:

- **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, 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.

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

Create, change, or delete a network security group

5/3/2018 • 10 min 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

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 5.4.1 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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 2.0](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

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. In the top-left corner of the portal, 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-AzureRmNetworkSecurityGroup`

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-AzureRmNetworkSecurityGroup](#)

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-AzureRmNetworkSecurityGroup](#)

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-AzureRmNetworkSecurityGroup](#)

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-AzureRmNetworkSecurityGroup](#)

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
Source	Select Any, IP Addresses , or Service Tag .	If you select IP Addresses , you must then specify Source IP addresses/CIDR ranges . 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 Azure limits for details. If you select Service Tag , you must 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 Service tags
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 Azure limits for details.
Destination	Select Any, IP addresses , or Virtual Network .	If you select IP addresses , you must then specify Destination IP addresses/CIDR ranges . Similar to Source and Source IP addresses/CIDR ranges , you can specify a single, or multiple addresses or ranges, and there are limits to the number you can specify. Selecting Virtual network , which is a service tag, means that traffic is allowed to all IP addresses within the address space of the virtual network.

SETTING	VALUE	DETAILS
Destination port ranges	Specify a single value, or comma-separated list of values.	Similar to Source port ranges , you can specify a single, or multiple ports and ranges, and there are limits to the number you can specify.
Protocol	Select Any , TCP , or UDP .	
Action	Select Allow or Deny .	
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.
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.	

You cannot specify an [application security group](#) for the **Source** or **Destination** settings using the portal.

You can however, using the Azure CLI or PowerShell. The settings for **Outbound security rules** are similar, so they are not covered separately.

Commands

- Azure CLI: [az network nsg rule create](#)
- PowerShell: [New-AzureRmNetworkSecurityRuleConfig](#)

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-AzureRmNetworkSecurityRuleConfig](#)

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-AzureRmNetworkSecurityRuleConfig`

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-AzureRmSecurityRuleConfig`

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-AzureRmSecurityRuleConfig`

Work with application security groups

An application security group contains zero or more network interfaces. To learn more, see [application security groups](#). You cannot work with application security groups in the portal, but you can use PowerShell or the Azure CLI. All network interfaces in an application security group must exist in the same virtual network. The first network interface added to an application security group determines which virtual network all subsequent network interfaces must be in. 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

- Azure CLI: `az network asg create`
- PowerShell: `New-AzureRmApplicationSecurityGroup`

View all application security groups

- Azure CLI: `az network asg list`
- PowerShell: `Get-AzureRmApplicationSecurityGroup`

View details of a specific application security group

- Azure CLI: `az network asg show`
- PowerShell: `Get-AzureRmApplicationSecurityGroup`

Change an application security group

While you can change some settings such as tags and permissions for an existing application security group, you can't change its name or location.

- 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. You must 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](#).

Commands

- Azure CLI: `az network asg delete`
- PowerShell: `Remove-AzureRmApplicationSecurityGroup`

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 table:

OPERATION	OPERATION NAME
Microsoft.Network/ruleTables/read	Get network security group
Microsoft.Network/ruleTables/write	Create or update network security group
Microsoft.Network/ruleTables/delete	Delete network security group
Microsoft.Network/ruleTables/join/action	Join network security group
Microsoft.Network/ruleTables/rules/read	Get rule
Microsoft.Network/ruleTables/rules/write	Create or update rule
Microsoft.Network/ruleTables/rules/delete	Delete rule
Microsoft.Network/networkInterfaces/effectiveruleTable/action	Get Network Interface Effective network security group
Microsoft.Network/networkWatchers/nextHop/action	Gets the next hop from a VM

The *Join network security group* operation is required to associate a network security group to a subnet.

Log analytics for network security groups (NSGs)

2/16/2018 • 3 min to read • [Edit Online](#)

You can enable the following diagnostic log categories for NSGs:

- **Event:** Contains entries for which NSG rules are applied to VMs and instance roles based on MAC address. The status for these rules is collected every 60 seconds.
- **Rule counter:** Contains entries for how many times each NSG rule is applied to deny or allow traffic.

NOTE

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, reference the [Understanding Azure deployment models](#) article.

Activity logging (previously known as audit or operational logs) 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

Read the [Overview of the Azure Activity Log](#) article to learn more about activity logs.

Enable diagnostic logging

Diagnostic logging must be enabled for *each* NSG you want to collect data for. The [Overview of Azure Diagnostic Logs](#) article explains where diagnostic logs can be sent. If you don't have an existing NSG, complete the steps in the [Create a network security group](#) article to create one. You can enable NSG diagnostic logging using any of the following methods:

Azure portal

To use the portal to enable logging, login to the [portal](#). Click **All services**, then type *network security groups*. Select the NSG you want to enable logging for. Follow the instructions for non-compute resources in the [Enable diagnostic logs in the portal](#) article. Select **NetworkSecurityGroupEvent**, **NetworkSecurityGroupRuleCounter**, or both categories of logs.

PowerShell

To use PowerShell to enable logging, follow the instructions in the [Enable diagnostic logs via PowerShell](#) article. Evaluate the following information before entering a command from the article:

- You can determine the value to use for the `-ResourceId` parameter by replacing the following [text], as appropriate, then entering the command
`Get-AzureRmNetworkSecurityGroup -Name [nsg-name] -ResourceGroupName [resource-group-name]`. The ID output from the command looks similar to `/subscriptions/[Subscription Id]/resourceGroups/[resource-group]/providers/Microsoft.Network/networkSecurityGroups/[NSG name]`.
- If you only want to collect data from log category add `-Categories [category]` to the end of the command in the article, where category is either *NetworkSecurityGroupEvent* or *NetworkSecurityGroupRuleCounter*. If you

don't use the `-Categories` parameter, data collection is enabled for both log categories.

Azure command-line interface (CLI)

To use the CLI to enable logging, follow the instructions in the [Enable diagnostic logs via CLI](#) article. Evaluate the following information before entering a command from the article:

- You can determine the value to use for the `-ResourceId` parameter by replacing the following [text], as appropriate, then entering the command `azure network nsg show [resource-group-name] [nsg-name]`. The ID output from the command looks similar to `/subscriptions/[Subscription Id]/resourceGroups/[resource-group]/providers/Microsoft.Network/networkSecurityGroups/[NSG name]`.
- If you only want to collect data from log category add `-Categories [category]` to the end of the command in the article, where category is either `NetworkSecurityGroupEvent` or `NetworkSecurityGroupRuleCounter`. If you don't use the `-Categories` parameter, data collection is enabled for both log categories.

Logged data

JSON-formatted data is written for both logs. The specific data written for each log type is listed in the following sections:

Event log

This log contains information about which NSG rules are applied to VMs and cloud service role instances, based on MAC address. The following example data is logged for each event:

```
{  
    "time": "[DATE-TIME]",  
    "systemId": "007d0441-5d6b-41f6-8bfd-930db640ec03",  
    "category": "NetworkSecurityGroupEvent",  
    "resourceId": "/SUBSCRIPTIONS/[SUBSCRIPTION-ID]/RESOURCEGROUPS/[RESOURCE-GROUP-  
NAME]/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/[NSG-NAME]",  
    "operationName": "NetworkSecurityGroupEvents",  
    "properties": {  
        "vnetResourceGuid": "{5E8AEC16-C728-441F-B0CA-B791E1DBC2F4}",  
        "subnetPrefix": "192.168.1.0/24",  
        "macAddress": "00-0D-3A-92-6A-7C",  
        "primaryIPv4Address": "192.168.1.4",  
        "ruleName": "UserRule_default-allow-rdp",  
        "direction": "In",  
        "priority": 1000,  
        "type": "allow",  
        "conditions": {  
            "protocols": "6",  
            "destinationPortRange": "3389-3389",  
            "sourcePortRange": "0-65535",  
            "sourceIP": "0.0.0.0/0",  
            "destinationIP": "0.0.0.0/0"  
        }  
    }  
}
```

Rule counter log

This log contains information about each rule applied to resources. The following example data is logged each time a rule is applied:

```
{  
    "time": "[DATE-TIME]",  
    "systemId": "007d0441-5d6b-41f6-8bfd-930db640ec03",  
    "category": "NetworkSecurityGroupRuleCounter",  
    "resourceId": "/SUBSCRIPTIONS/[SUBSCRIPTION ID]/RESOURCEGROUPS/[RESOURCE-GROUP-  
NAME]TESTRG/PROVIDERS/MICROSOFT.NETWORK/NETWORKSECURITYGROUPS/[NSG-NAME]",  
    "operationName": "NetworkSecurityGroupCounters",  
    "properties": {  
        "vnetResourceGuid": "{5E8AEC16-C728-441F-B0CA-791E1DBC2F4}",  
        "subnetPrefix": "192.168.1.0/24",  
        "macAddress": "00-0D-3A-92-6A-7C",  
        "primaryIPv4Address": "192.168.1.4",  
        "ruleName": "UserRule_default-allow-rdp",  
        "direction": "In",  
        "type": "allow",  
        "matchedConnections": 125  
    }  
}
```

View and analyze logs

To learn how to view activity log data, read the [Overview of the Azure Activity Log](#) article. To learn how to view diagnostic log data, read the [Overview of Azure Diagnostic Logs](#) article. If you send diagnostics data to Log Analytics, you can use the [Azure Network Security Group analytics](#) (preview) management solution for enhanced insights.

Create, change, or delete a route table

4/18/2018 • 10 min 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 not familiar with Azure routing, we recommend reading the [Routing overview](#) and completing the [Route network traffic with a route table](#) tutorial, before completing tasks in this article.

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 5.2.0 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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 2.0](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

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**. The **Disable BGP route propagation** option prevents on-premises routes from being propagated via BGP to the network interfaces in any subnet that the route table is associated to. If your virtual network is not connected to an Azure network gateway (VPN or ExpressRoute), leave the option *Disabled*.

Commands

- Azure CLI: `az network route-table create`
- PowerShell: `New-AzureRmRouteTable`

View route tables

In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appears in the search results, select it. The route tables that exist in your subscription are listed.

Commands

- Azure CLI: [az network route-table list](#)
- PowerShell: [Get-AzureRmRouteTable](#)

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** appears 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](#)

Commands

- Azure CLI: [az network route-table show](#)
- PowerShell: [Get-AzureRmRouteTable](#)

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** appears 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.

Commands

- Azure CLI: [az network route-table update](#)
- PowerShell: [Set-AzureRmRouteTable](#)

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, if using BGP with a VPN gateway). 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** appears 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**.

Commands

- Azure CLI: [az network vnet subnet update](#)
- PowerShell: [Set-AzureRmVirtualNetworkSubnetConfig](#)

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** appears 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**.

Commands

- Azure CLI: [az network vnet subnet update](#)
- PowerShell: [Set-AzureRmVirtualNetworkSubnetConfig](#)

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** appears 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**.

Commands

- Azure CLI: [az network route-table delete](#)
- PowerShell: [Delete-AzureRmRouteTable](#)

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** appears 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**.

Commands

- Azure CLI: [az network route-table route create](#)
- PowerShell: [New-AzureRmRouteConfig](#)

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** appears 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**.

Commands

- Azure CLI: [az network route-table route list](#)
- PowerShell: [Get-AzureRmRouteConfig](#)

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** appears 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.

Commands

- Azure CLI: [az network route-table route show](#)
- PowerShell: [Get-AzureRmRouteConfig](#)

Change a route

1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appears 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**.

Commands

- Azure CLI: [az network route-table route update](#)
- PowerShell: [Set-AzureRmRouteConfig](#)

Delete a route

1. In the search box at the top of the portal, enter *route tables* in the search box. When **Route tables** appears 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**.

Commands

- Azure CLI: [az network route-table route delete](#)
- PowerShell: [Remove-AzureRmRouteConfig](#)

View effective routes

The effective routes for each network interface attached to a virtual machine are a combination route tables 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** appears 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](#).

Commands

- Azure CLI: [az network nic show-effective-route-table](#)
- PowerShell: [Get-AzureRmEffectiveRouteTable](#)

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](#).

Commands

- Azure CLI: [az network watcher show-next-hop](#)
- PowerShell: [Get-AzureRmNetworkWatcherNextHop](#)

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 permissions listed in the following table:

OPERATION	OPERATION NAME
Microsoft.Network/routeTables/read	Get route table
Microsoft.Network/routeTables/write	Create or update route table
Microsoft.Network/routeTables/delete	Delete route table
Microsoft.Network/routeTables/join/action	Join route table
Microsoft.Network/routeTables/routes/read	Get route
Microsoft.Network/routeTables/routes/write	Create or update route
Microsoft.Network/routeTables/routes/delete	Delete route
Microsoft.Network/networkInterfaces/effectiveRouteTable/acti on	Get Network Interface Effective Route Table
Microsoft.Network/networkWatchers/nextHop/action	Gets the next hop from a VM

The *Join route table* operation is required to associate a route table to a subnet.

Create, change, or delete a network interface

4/24/2018 • 19 min 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

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 5.4.1 or later. Run `Get-Module -ListAvailable AzureRM` 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-AzureRmAccount` 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 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 Azure with must be assigned, at a minimum, permissions for the Network Contributor role for your subscription. To learn more about assigning roles and permissions to accounts, see [Built-in roles for Azure role-based access control](#).

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 Naming conventions . 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.
Private IP address assignment	Yes	In this setting, you're choosing the assignment method for the IPv4 address. Choose from the following assignment methods: Dynamic : When selecting this option, Azure automatically assigns the next available address from the address space of the subnet you selected. Static : 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.

Setting	Required?	Details
Network security group	No	Leave set to None , select an existing network security group , or create a network security group . 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 Troubleshoot network security groups .
Subscription	Yes	Select one of your Azure subscriptions . The virtual machine you attach a network interface to and the virtual network you connect it to must exist in the same subscription.
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 Private IP address (IPv6) checkbox is checked)	Yes, if the Private IP address (IPv6) checkbox is checked.	This name is assigned to a secondary IP configuration for the network interface. To learn more about IP configurations, see View network interface settings .
Resource group	Yes	Select an existing resource group 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 location , 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, but the Azure CLI and PowerShell do. To learn more about application security groups, see [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-AzureRmNetworkInterface</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**:

The screenshot shows the Azure portal interface for a network interface named "mywebserver256". The main content area displays the following details:

Setting	Value
Private IP address	10.0.0.4
Virtual network/subnet	MyVNet/Front-end
Public IP address	13.64.66.250 (MyWebServer-ip)
Network security group	MyWebServer-nsg
Attached to	MyWebServer

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-AzureRmNetworkInterface</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.
5. Select **Save**.

Commands

TOOL	COMMAND
CLI	az network nic update
PowerShell	Set-AzureRmNetworkInterface

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	az network nic update
PowerShell	Set-AzureRmNetworkInterface

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	az network nic ip-config update
PowerShell	Set-AzureRmNetworkInterfaceIpConfig

Add to or remove from application security groups

The portal doesn't provide the option to assign a network interface to, or remove a network interface from application security groups, but the Azure CLI and PowerShell do. To learn more about application security groups, see [Application security groups](#) and [Create an application security group](#).

Commands

TOOL	COMMAND
CLI	az network nic update
PowerShell	Set-AzureRmNetworkInterface

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-AzureRmNetworkInterface](#)

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 ... on the right side of the network interface you want to delete from the list of network interfaces.
3. 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	az network nic delete
PowerShell	Remove-AzureRmNetworkInterface

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-AzureRmEffectiveNetworkSecurityGroup](#)

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-AzureRmEffectiveRouteTable](#)

Next steps

To create a virtual machine with multiple network interfaces or IP addresses, see the following articles:

TASK	TOOL
Create a VM with multiple NICs	CLI , PowerShell
Create a single NIC VM with multiple IPv4 addresses	CLI , PowerShell
Create a single NIC VM with a private IPv6 address (behind an Azure Load Balancer)	CLI , PowerShell , Azure Resource Manager template

TASK

TOOL

Add, change, or remove IP addresses for an Azure network interface

4/18/2018 • 16 min 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

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 5.2.0 or later. Run `Get-Module -ListAvailable Azurerm` 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-AzurermAccount` 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 2.0](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

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 cannot use the portal to add an IPv6 address to an existing network interface (though you can use the portal to add a private IPv6 address to a network interface when you create the network interface). You can use PowerShell or the CLI to 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 that is not attached to a virtual machine. You cannot use any tool to add a public IPv6 address to a network interface. See [IPv6](#) for details about using IPv6 addresses.

1. Log in to the [Azure portal](#) with an account that is assigned (at a minimum) permissions for the Network Contributor role for your subscription. Read the [Built-in roles for Azure role-based access control](#) article to learn more about assigning roles and permissions to accounts.
2. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*.

When **network interfaces** appears in the search results, click it.

3. In the **Network interfaces** blade that appears, click the network interface you want to add an IPv4 address for.
4. Click **IP configurations** in the **SETTINGS** section of the blade for the network interface you selected.
5. Click **+ Add** in the blade that opens for IP configurations.
6. Specify the following, then click **OK** to close the **Add IP configuration** blade:

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 primary IP configuration, your only option is Secondary .
Private IP address assignment method	Yes	Dynamic: Azure assigns the next available address for the subnet address range the network interface is deployed in. Static: You assign an unused address for the subnet address range the network interface is deployed in.
Public IP address	No	Disabled: No public IP address resource is currently associated to the IP configuration. Enabled: Select an existing IPv4 Public IP address, or create a new one. To learn how to create a public IP address, read the Public IP addresses article.

7. 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	az network nic ip-config create
PowerShell	Add-AzureRmNetworkInterfaceIpConfig

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. Log in to the [Azure portal](#) with an account that is assigned (at a minimum) permissions for the Network Contributor role for your subscription. Read the [Built-in roles for Azure role-based access control](#) article to learn more about assigning roles and permissions to accounts.
2. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appears in the search results, click it.
3. In the **Network interfaces** blade that appears, click the network interface you want to view or change IP address settings for.
4. Click **IP configurations** in the **SETTINGS** section of the blade for the network interface you selected.
5. Click the IP configuration you want to modify from the list in the blade that opens for IP configurations.
6. Change the settings, as desired, using the information about the settings in step 6 of the [Add an IP configuration](#) section of this article. Click **Save** to close the blade for the IP configuration you changed.

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, read the [Assign multiple IP addresses to virtual machines](#) 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	az network nic ip-config update
PowerShell	Set-AzureRMNetworkInterfaceIpConfig

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. Log in to the [Azure portal](#) with an account that is assigned (at a minimum) permissions for the Network Contributor role for your subscription. Read the [Built-in roles for Azure role-based access control](#) article to learn more about assigning roles and permissions to accounts.
2. In the box that contains the text *Search resources* at the top of the Azure portal, type *network interfaces*. When **network interfaces** appears in the search results, click it.
3. In the **Network interfaces** blade that appears, click the network interface you want to remove IP addresses from.
4. Click **IP configurations** in the **SETTINGS** section of the blade for the network interface you selected.
5. Right-click a [secondary](#) IP configuration (you cannot delete the [primary](#) configuration) you want to delete, click **Delete**, then click **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.
6. Close the **IP configurations** blade.

Commands

TOOL	COMMAND
CLI	az network nic ip-config delete
PowerShell	Remove-AzureRmNetworkInterfaceIpConfig

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 or secondary IP configuration. You can however, assign a public IPv6 address to an Azure load balancer, which can load balance traffic to a virtual machine's private IPv6 address. For more information, see [details and limitations for IPv6](#).

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 address assigned to it, if the private IP address is IPv4. If the private IP address is IPv6, you cannot assign a public IPv4 or IPv6 address to the IP configuration. Assigning multiple IP 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](#) addresses enable a virtual machine to communicate with other resources in a virtual network or other connected networks. A virtual machine cannot be communicated inbound to, nor can the virtual machine communicate outbound with a private [IPv6](#) address, with one exception. A virtual machine can communicate with the Azure load balancer using an IPv6 address. For more information, see [details and limitations for IPv6](#).

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. You can only assign a private IPv6 address using the dynamic assignment method.

Static

You can (optionally) assign a public or private static IPv4 address to an IP configuration. You cannot assign a static public or private IPv6 address to an IP configuration. To learn more about how Azure assigns static public IPv4 addresses, see the [Public IP address](#) article.

- **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 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 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.

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. You cannot add an IP configuration with an IPv6 address using the portal. Use PowerShell or the CLI to add an IP configuration with a private IPv6 address to an existing network interface. The network interface cannot be attached to an existing VM.

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 2.0 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 details 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 NICs	CLI , PowerShell
Create a single NIC VM with multiple IPv4 addresses	CLI , PowerShell
Create a single NIC VM with a private IPv6 address (behind an Azure Load Balancer)	CLI , PowerShell , Azure Resource Manager template

Create, change, or delete a public IP address

4/18/2018 • 10 min 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 other Azure resources enables:

- Inbound Internet connectivity to resources such as Azure Virtual Machines, Azure Virtual Machine Scale Sets, Azure VPN Gateway, Application gateways, and Internet-facing Azure Load Balancers. Azure resources cannot receive inbound communication from the Internet without an assigned public IP address. While some Azure resources are inherently accessible through public IP addresses, other resources must have public IP addresses assigned to them to be accessible from the Internet.
- 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. 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, read the [Understand outbound connections](#) article.

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 5.2.0 or later. Run `Get-Module -ListAvailable Azurerm` 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-AzureRmAccount` 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 2.0](#). If you are running the Azure CLI locally, you also need to run `az login` to create a connection with Azure.

Public IP addresses have a nominal charge. To view the pricing, read the [IP address pricing](#) page.

Create a public IP address

1. In the box that contains the text *Search resources* at the top of the Azure portal, type *public ip address*. When **Public IP addresses** appears in the search results, click it.
2. Click **+ Add** in the **Public IP address** blade that appears.
3. Enter or select values for the following settings in the **Create public IP address** blade that appears, then click **Create**:

SETTING	REQUIRED?	DETAILS
SKU	Yes	<p>All public IP addresses created before the introduction of SKUs are Basic 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 is not allowed.</p> <p>Basic SKU: If you are creating a public IP address in a region that supports availability zones, the Availability zone setting is set to <i>None</i> by default. You can choose to select an availability zone to guarantee a specific zone for your public IP address.</p> <p>Standard 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 Availability zone setting is set to <i>Zone-redundant</i> by default. For more information about availability zones, see the Availability zone setting. The standard SKU is required if you associate the address to a Standard load balancer. To learn more about standard load balancers, see Azure load balancer standard SKU. 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.</p>
Name	Yes	The name must be unique within the resource group you select.

Setting	Required?	Details
IP Version	Yes	Select IPv4 or IPv6. While public IPv4 addresses can be assigned to several Azure resources, an IPv6 public IP address can only be assigned to an Internet-facing load balancer. The load balancer can load balance IPv6 traffic to Azure virtual machines. Learn more about load balancing IPv6 traffic to virtual machines . If you selected the Standard SKU , you do not have the option to select <i>IPv6</i> . You can only create an IPv4 address when using the Standard SKU .
IP address assignment	Yes	Dynamic: Dynamic addresses are assigned only after the public IP address is associated to a network interface attached to a virtual machine and the virtual machine is started for the first time. Dynamic addresses can change if the virtual machine the network interface is attached to is stopped (deallocated). The address remains the same if the virtual machine is rebooted or stopped (but not deallocated). Static: Static addresses are assigned when the public IP address is created. Static addresses do not change even if the virtual machine is put in the stopped (deallocated) state. The address is only released when the network interface is deleted. You can change the assignment method after the network interface is created. If you select <i>IPv6</i> for the IP version , the assignment method is <i>Dynamic</i> . If you select <i>Standard</i> for SKU , the assignment method is <i>Static</i> .
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 IPv6 for IP Version , 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 Use Azure DNS with an Azure public IP address .
Create an IPv6 (or IPv4) address	No	Whether IPv6 or IPv4 is displayed is dependent on what you select for IP Version . For example, if you select IPv4 for IP Version , IPv6 is displayed here. If you select Standard for SKU , you don't have the option to create an IPv6 address.
Name (Only visible if you checked the Create an IPv6 (or IPv4) address checkbox)	Yes, if you select the Create an IPv6 (or IPv4) checkbox.	The name must be different than the name you enter for the first Name 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 checked the Create an IPv6 (or IPv4) address checkbox)	Yes, if you select the Create an IPv6 (or IPv4) checkbox.	If the checkbox says Create an IPv4 address , you can select an assignment method. If the checkbox says Create an IPv6 address , you cannot select an assignment method, as it must be Dynamic .
Subscription	Yes	Must exist in the same subscription as the resource you want to associate the public IP address to.

Setting	Required?	Details
Resource group	Yes	Can exist in the same, or different, resource group as the resource you want to associate the public IP address to.
Location	Yes	Must exist in the same location , also referred to as region, as the resource you want to associate the public IP address to.
Availability zone	No	This setting only appears if you select a supported location. For a list of supported locations, see Availability zones overview . If you selected the Basic SKU, None 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 Standard 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 versions for the public IP address resources.

Tool	Command
CLI	az network public-ip create
PowerShell	New-AzureRmPublicIpAddress

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** appears in the search results, click it.
- In the **Public IP addresses** blade that appears, click the name of the public IP address you want to view, change settings for, or delete.
- In the blade that appears for the public IP address, complete one of the following options depending on whether you want to view, delete, or change the public IP address.
 - View:** The **Overview** section of the blade 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, click **Delete** in the **Overview** section of the blade. If the

address is currently associated to an IP configuration, it cannot be deleted. If the address is currently associated with a configuration, click **Dissociate** to dissociate the address from the IP configuration.

- **Change:** Click **Configuration**. Change settings using the information in step 4 of the [Create a public IP address](#) section of this article. 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 click **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, click **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	az network public-ip-list to list public IP addresses, az network public-ip-show to show settings; az network public-ip update to update; az network public-ip delete to delete
PowerShell	Get-AzureRmPublicIpAddress to retrieve a public IP address object and view its settings, Set-AzureRmPublicIpAddress to update settings; Remove-AzureRmPublicIpAddress to delete

Next steps

Assign public IP addresses when creating the following Azure resources:

- [Windows](#) or [Linux](#) virtual machines
- [Internet-facing Azure Load Balancer](#)
- [Azure Application Gateway](#)
- [Site-to-site connection using an Azure VPN Gateway](#)
- [Azure Virtual Machine Scale Set](#)

Move a VM (Classic) or Cloud Services role instance to a different subnet using PowerShell

4/19/2018 • 1 min to read • [Edit Online](#)

You can use PowerShell to move your VMs (Classic) from one subnet to another in the same virtual network (VNet). Role instances can be moved by editing the CSCFG file, rather than using PowerShell.

NOTE

This article explains how to move VMs deployed through the classic deployment model only.

Why move VMs to another subnet? Subnet migration is useful when the older subnet is too small and cannot be expanded due to existing running VMs in that subnet. In that case, you can create a new, larger subnet and migrate the VMs to the new subnet, then after migration is complete, you can delete the old empty subnet.

How to move a VM to another subnet

To move a VM, run the Set-AzureSubnet PowerShell cmdlet, using the example below as a template. In the example below, we are moving TestVM from its present subnet, to Subnet-2. Be sure to edit the example to reflect your environment. Note that whenever you run the Update-AzureVM cmdlet as part of a procedure, it will restart your VM as part of the update process.

```
Get-AzureVM -ServiceName TestVMCloud -Name TestVM `| Set-AzureSubnet -SubnetNames Subnet-2 `| Update-AzureVM
```

If you specified a static internal private IP for your VM, you'll have to clear that setting before you can move the VM to a new subnet. In that case, use the following:

```
Get-AzureVM -ServiceName TestVMCloud -Name TestVM `| Remove-AzureStaticVNetIP `| Update-AzureVM  
Get-AzureVM -ServiceName TestVMCloud -Name TestVM `| Set-AzureSubnet -SubnetNames Subnet-2 `| Update-AzureVM
```

To move a role instance to another subnet

To move a role instance, edit the CSCFG file. In the example below, we are moving "Role0" in virtual network *VNETName* from its present subnet to *Subnet-2*. Because the role instance was already deployed, you'll just change the Subnet name = Subnet-2. Be sure to edit the example to reflect your environment.

```
<NetworkConfiguration>
  <VirtualNetworkSite name="VNETName" />
  <AddressAssignments>
    <InstanceAddress roleName="Role0">
      <Subnets><Subnet name="Subnet-2" /></Subnets>
    </InstanceAddress>
  </AddressAssignments>
</NetworkConfiguration>
```

Troubleshoot Network Security Groups using the Azure Portal

4/11/2018 • 8 min to read • [Edit Online](#)

If you configured Network Security Groups (NSGs) on your virtual machine (VM) and are experiencing VM connectivity issues, this article provides an overview of diagnostics capabilities for NSGs to help troubleshoot further.

NSGs enable you to control the types of traffic that flow in and out of your virtual machines (VMs). NSGs can be applied to subnets in an Azure Virtual Network (VNet), network interfaces (NIC), or both. The effective rules applied to a NIC are an aggregation of the rules that exist in the NSGs applied to a NIC and the subnet it is connected to. Rules across these NSGs can sometimes conflict with each other and impact a VM's network connectivity.

You can view all the effective security rules from your NSGs, as applied on your VM's NICs. This article shows how to troubleshoot VM connectivity issues using these rules in the Azure Resource Manager deployment model. If you're not familiar with VNet and NSG concepts, read the [Virtual network](#) and [Network security groups](#) overview articles.

Using Effective Security Rules to troubleshoot VM traffic flow

The scenario that follows is an example of a common connection problem:

A VM named *VM1* is part of a subnet named *Subnet1* within a VNet named *WestUS-VNet1*. An attempt to connect to the VM using RDP over TCP port 3389 fails. NSGs are applied at both the NIC *VM1-NIC1* and the subnet *Subnet1*. Traffic to TCP port 3389 is allowed in the NSG associated with the network interface *VM1-NIC1*, however TCP ping to VM1's port 3389 fails.

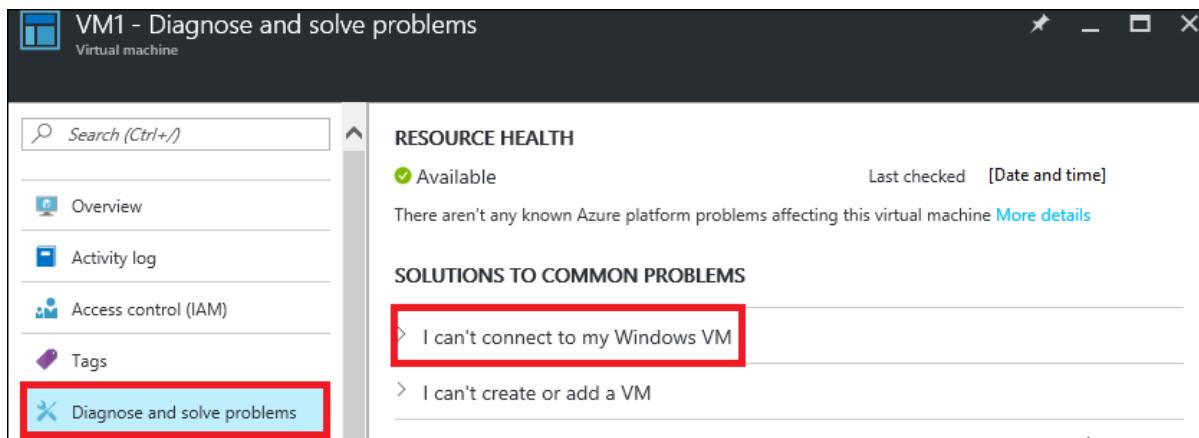
While this example uses TCP port 3389, the following steps can be used to determine inbound and outbound connection failures over any port.

View effective security rules for a virtual machine

Complete the following steps to troubleshoot NSGs for a VM:

You can view full list of the effective security rules on a NIC, from the VM itself. You can also add, modify, and delete both NIC and subnet NSG rules from the effective rules blade, if you have permissions to perform these operations.

1. Login to the Azure portal at <https://portal.azure.com> with an Azure account. Your account must be assigned the `Microsoft.Network/networkInterfaces/effectiveNetworkSecurityGroups/action` operation for the network interface. To learn how to assign operations to accounts, see [Create custom roles for Azure Role-Based Access Control](#).
2. Click **All services**, then click **Virtual machines** in the list that appears.
3. Select a VM to troubleshoot from the list that appears and a VM blade with options appears.
4. Click **Diagnose & solve problems** and then select a common problem. For this example, **I can't connect to my Windows VM** is selected.



5. Steps appear under the problem, as shown in the following picture:

This screenshot shows the detailed view for the 'I can't connect to my Windows VM' problem. It includes the 'RESOURCE HEALTH' section at the top, followed by the 'SOLUTIONS TO COMMON PROBLEMS' section. Under the 'I can't connect to my Windows VM' item, there's a heading 'Recommended steps' with the following list:

1. Review your VM's [console screenshot](#) to correct boot problems
2. Reset Remote Access to address remote server issues
[Reset remote access using PowerShell or CLI](#)
3. Restart the Virtual Machine to address startup issues by clicking 'Restart' at the top of the VM resource blade
4. Address Azure host issues by [redeploying](#), which will migrate the VM to a new Azure host
5. To connect to your VM via RDP, please review [effective security group rules](#) to ensure inbound "Allow" NSG rule exists for RDP port(3389)
6. RDP to your VM from Internet will not work with force tunneling enabled. Review [effective routes](#)
With force tunneling, all outbound traffic destined to Internet will be redirected to on-premises

Click *effective security group rules* in the list of recommended steps.

6. The **Get effective security rules** blade appears, as shown in the following picture:

Notice the following sections of the picture:

- **Scope:** Set to *VM1*, the VM selected in step 3.
- **Network interface:** *VM1-NIC1* is selected. A VM can have multiple network interfaces (NIC). Each NIC can have unique effective security rules. When troubleshooting, you may need to view the effective security rules for each NIC.
- **Associated NSGs:** NSGs can be applied to both the NIC and the subnet the NIC is connected to. In the picture, an NSG has been applied to both the NIC and the subnet it's connected to. You can click on the NSG names to directly modify rules in the NSGs.
- **VM1-nsg tab:** The list of rules displayed in the picture is for the NSG applied to the NIC. Several default rules are created by Azure whenever an NSG is created. You can't remove the default rules, but you can override them with rules of higher priority. To learn more about default rules, read the [NSG overview](#) article.
- **DESTINATION column:** Some of the rules have text in the column, while others have address prefixes. The text is the name of default tags applied to the security rule when it was created. The tags are system-provided identifiers that represent multiple prefixes. Selecting a rule with a tag, such as *AllowInternetOutBound*, lists the prefixes in the **Address prefixes** blade.
- **Download:** The list of rules can be long. You can download a .csv file of the rules for offline analysis by clicking **Download** and saving the file.
- **AllowRDP Inbound rule:** This rule allows RDP connections to the VM.

7. Click the **Subnet1-NSG** tab to view the effective rules from the NSG applied to the subnet, as shown in the following picture:

Notice the **denyRDP Inbound** rule. Inbound rules applied at the subnet are evaluated before rules applied at the network interface. Since the deny rule is applied at the subnet, the request to connect to TCP 3389 fails, because the allow rule at the NIC is never evaluated.

The *denyRDP* rule is the reason why the RDP connection is failing. Removing it should resolve the problem.

NOTE

If the VM associated with the NIC is not in a running state, or NSGs haven't been applied to the NIC or subnet, no rules are shown.

8. To edit NSG rules, click **Subnet1-NSG** in the **Associated NSGs** section. This opens the **Subnet1-NSG** blade. You can directly edit the rules by clicking on **Inbound security rules**.

The screenshot shows the 'Subnet1-NSG' blade in the Azure portal. The left pane displays the 'Effective security rules' grid, which is currently empty. The right pane contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and SETTINGS. Under SETTINGS, 'Inbound security rules' is highlighted with a red box. The main content area shows the 'Inbound rules' table with one row:

NAME	PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
denyRDP	1000	Internet (76 prefixes)	0-65535	0.0.0.0/0	3389-3389	TCP	Deny

9. After removing the *denyRDP* inbound rule in the **Subnet1-NSG** and adding an *allowRDP* rule, the effective rules list looks like the following picture:

The screenshot shows the 'Subnet1-NSG' blade after changes. The left pane displays the 'Effective security rules' grid, which is currently empty. The right pane contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, and SETTINGS. Under SETTINGS, 'Inbound security rules' is highlighted with a red box. The main content area shows the 'Inbound rules' table with four rows:

NAME	PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS
allowRDP	100	Internet (76 prefixes)	0-65535	0.0.0.0/0	3389-3389	TCP	Allow
AllowVnetInBound	65000	VirtualNetwork (4 prefixes)	0-65535	VirtualNetwork (4 prefixes)	0-65535	All	Allow
AllowAzureLoadBalancer...	65001	AzureLoadBalancer (1 prefixes)	0-65535	0.0.0.0/0	0-65535	All	Allow
DenyAllInBound	65500	0.0.0.0/0	0-65535	0.0.0.0/0	0-65535	All	Deny

Confirm that TCP port 3389 is open by opening an RDP connection to the VM or using the PsPing tool. You can learn more about PsPing by reading the [PsPing download page](#).

View effective security rules for a network interface

If your VM traffic flow is impacted for a specific NIC, you can view a full list of the effective rules for the NIC from the network interfaces context by completing the following steps:

1. Login to the Azure portal at <https://portal.azure.com>.

2. Click **All services**, then click **Network interfaces** in the list that appears.
3. Select a network interface. In the following picture, a NIC named VM1-NIC1 is selected.

The screenshot shows the Azure portal interface for managing network security rules. The title bar says "VM1-NIC1 - Effective security rules". The left sidebar has a tree view with "Effective security rules" selected. The main area displays two tables of security rules:

Inbound rules								
NAME	PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS	
allowRDP	1000	Internet (76 prefixes)	0-65535	0.0.0.0/0	3389-3389	TCP	Allow	
AllowVnetInBound	65000	VirtualNetwork (4 prefixes)	0-65535	VirtualNetwork (4 prefixes)	0-65535	All	Allow	
AllowAzureLoadBalancer...	65001	AzureLoadBalancer (1 prefixes)	0-65535	0.0.0.0/0	0-65535	All	Allow	
DenyAllInBound	65500	0.0.0.0/0	0-65535	0.0.0.0/0	0-65535	All	Deny	

Outbound rules								
NAME	PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL	ACCESS	
AllowVnetOutBound	65000	VirtualNetwork (4 prefixes)	0-65535	VirtualNetwork (4 prefixes)	0-65535	All	Allow	
AllowInternetOutBound	65001	0.0.0.0/0	0-65535	Internet (76 prefixes)	0-65535	All	Allow	
DefaultOutboundDenyAll	65500	*	0-65535	*	0-65535	All	Deny	

Notice that the **Scope** is set to the network interface selected. To learn more about the additional information shown, read step 6 of the **Troubleshoot NSGs for a VM** section of this article.

NOTE

If an NSG is removed from a network interface, the subnet NSG is still effective on the given NIC. In this case, the output would only show rules from the subnet NSG. Rules only appear if the NIC is attached to a VM.

4. You can directly edit rules for NSGs associated with a NIC and a subnet. To learn how, read step 8 of the **View effective security rules for a virtual machine** section of this article.

View effective security rules for a network security group (NSG)

When modifying NSG rules, you may want to review the impact of the rules being added on a particular VM. You can view a full list of the effective security rules for all the NICs that a given NSG is applied to, without having to switch context from the given NSG blade. To troubleshoot effective rules within an NSG, complete the following steps:

1. Login to the Azure portal at <https://portal.azure.com>.
2. Click **All services**, then click **Network security groups** in the list that appears.
3. Select an NSG. In the following picture, an NSG named VM1-nsg was selected.

The screenshot shows the 'Effective security rules' page for the network security group 'VM1-nsg'. The 'Scope' is set to 'Network security group (VM1-nsg)'. Under 'Virtual machine', 'VM1' is selected. Under 'Associated NSGs', both 'VM1-nsg (Network interface)' and 'Subnet1-NSG (Subnet)' are listed. The 'Inbound rules' and 'Outbound rules' tables show standard Azure default rules. The 'Effective security rules' link in the left sidebar is highlighted.

Notice the following sections of the previous picture:

- **Scope:** Set to the NSG selected.
- **Virtual machine:** When an NSG is applied to a subnet, it's applied to all network interfaces attached to all VMs connected to the subnet. This list shows all VMs this NSG is applied to. You can select any VM from the list.

NOTE

If an NSG is applied to only an empty subnet, VMs will not be listed. If an NSG is applied to a NIC which is not associated with a VM, those NICs will also not be listed.

- **Network Interface:** A VM can have multiple network interfaces. You can select a network interface attached to the selected VM.
 - **AssociatedNSGs:** At any time, a NIC can have up to two effective NSGs, one applied to the NIC and the other to the subnet. Although the scope is selected as VM1-nsg, if the NIC has an effective subnet NSG, the output will show both NSGs.
4. You can directly edit rules for NSGs associated with a NIC or subnet. To learn how, read step 8 of the [View effective security rules for a virtual machine](#) section of this article.

To learn more about the additional information shown, read step 6 of the [View effective security rules for a virtual machine](#) section of this article.

NOTE

Though a subnet and NIC can each have only one NSG applied to them, an NSG can be associated to multiple NICs and multiple subnets.

Considerations

Consider the following points when troubleshooting connectivity problems:

- Default NSG rules will block inbound access from the internet and only permit VNet inbound traffic. Rules

should be explicitly added to allow inbound access from Internet, as required.

- If there are no NSG security rules causing a VM's network connectivity to fail, the problem may be due to:
 - Firewall software running within the VM's operating system
 - Routes configured for virtual appliances or on-premises traffic. Internet traffic can be redirected to on-premises via forced-tunneling. An RDP/SSH connection from the Internet to your VM may not work with this setting, depending on how the on-premises network hardware handles this traffic. Read the [Troubleshooting Routes](#) article to learn how to diagnose route problems that may be impeding the flow of traffic in and out of the VM.
- If you have peered VNets, by default, the VIRTUAL_NETWORK tag will automatically expand to include prefixes for peered VNets. You can view these prefixes in the **ExpandedAddressPrefix** list, to troubleshoot any issues related to VNet peering connectivity.
- Effective security rules are only shown if there is an NSG associated with the VM's NIC and or subnet.
- If there are no NSGs associated with the NIC or subnet and you have a public IP address assigned to your VM, all ports will be open for inbound and outbound access. If the VM has a public IP address, applying NSGs to the NIC or subnet is strongly recommended.

Troubleshoot Network Security Groups using Azure PowerShell

4/11/2018 • 6 min to read • [Edit Online](#)

If you configured Network Security Groups (NSGs) on your virtual machine (VM) and are experiencing VM connectivity issues, this article provides an overview of diagnostics capabilities for NSGs to help troubleshoot further.

NSGs enable you to control the types of traffic that flow in and out of your virtual machines (VMs). NSGs can be applied to subnets in an Azure Virtual Network (VNet), network interfaces (NIC), or both. The effective rules applied to a NIC are an aggregation of the rules that exist in the NSGs applied to a NIC and the subnet it is connected to. Rules across these NSGs can sometimes conflict with each other and impact a VM's network connectivity.

You can view all the effective security rules from your NSGs, as applied on your VM's NICs. This article shows how to troubleshoot VM connectivity issues using these rules in the Azure Resource Manager deployment model. If you're not familiar with VNet and NSG concepts, read the [Virtual network](#) and [Network security groups](#) overview articles.

Using Effective Security Rules to troubleshoot VM traffic flow

The scenario that follows is an example of a common connection problem:

A VM named *VM1* is part of a subnet named *Subnet1* within a VNet named *WestUS-VNet1*. An attempt to connect to the VM using RDP over TCP port 3389 fails. NSGs are applied at both the NIC *VM1-NIC1* and the subnet *Subnet1*. Traffic to TCP port 3389 is allowed in the NSG associated with the network interface *VM1-NIC1*, however TCP ping to *VM1*'s port 3389 fails.

While this example uses TCP port 3389, the following steps can be used to determine inbound and outbound connection failures over any port.

Detailed Troubleshooting Steps

Complete the following steps to troubleshoot NSGs for a VM:

1. Start an Azure PowerShell session and login to Azure. If you're not familiar with using Azure PowerShell, read the [How to install and configure Azure PowerShell](#) article. Your account must be assigned the `Microsoft.Network/networkInterfaces/effectiveNetworkSecurityGroups/action` operation for the network interface. To learn how to assign operations to accounts, see [Create custom roles for Azure Role-Based Access Control](#).
2. Enter the following command to return all NSG rules applied to a NIC named *VM1-NIC1* in the resource group *RG1*:

```
Get-AzureRmEffectiveNetworkSecurityGroup -NetworkInterfaceName VM1-NIC1 -ResourceGroupName RG1
```

TIP

If you don't know the name of a NIC, enter the following command to retrieve the names of all NICs in a resource group:

```
Get-AzureRmNetworkInterface -ResourceGroupName RG1 | Format-Table Name
```

The following text is a sample of the effective rules output returned for the VM1-NIC1 NIC:

```
NetworkSecurityGroup : {
    "Id": "/subscriptions/[Subscription
ID]/resourceGroups/RG1/providers/Microsoft.Network/networkSecurityGroups/VM1-NIC1-NSG"
}
Association : {
    "NetworkInterface": {
        "Id": "/subscriptions/[Subscription
ID]/resourceGroups/RG1/providers/Microsoft.Network/networkInterfaces/VM1-NIC1"
    }
}
EffectiveSecurityRules : [
{
{
    "Name": "securityRules/allowRDP",
    "Protocol": "Tcp",
    "SourcePortRange": "0-65535",
    "DestinationPortRange": "3389-3389",
    "SourceAddressPrefix": "Internet",
    "DestinationAddressPrefix": "0.0.0.0/0",
    "ExpandedSourceAddressPrefix": [... ],
    "ExpandedDestinationAddressPrefix": [],
    "Access": "Allow",
    "Priority": 1000,
    "Direction": "Inbound"
},
{
    "Name": "defaultSecurityRules/AllowVnetInBound",
    "Protocol": "All",
    "SourcePortRange": "0-65535",
    "DestinationPortRange": "0-65535",
    "SourceAddressPrefix": "VirtualNetwork",
    "DestinationAddressPrefix": "VirtualNetwork",
    "ExpandedSourceAddressPrefix": [
        "10.9.0.0/16",
        "168.63.129.16/32",
        "10.0.0.0/16",
        "10.1.0.0/16"
    ],
    "ExpandedDestinationAddressPrefix": [
        "10.9.0.0/16",
        "168.63.129.16/32",
        "10.0.0.0/16",
        "10.1.0.0/16"
    ],
    "Access": "Allow",
    "Priority": 65000,
    "Direction": "Inbound"
},...
]
NetworkSecurityGroup : {
    "Id":
        "/subscriptions/[Subscription
ID]/resourceGroups/RG1/providers/Microsoft.Network/networkSecurityGroups/Subnet1-NSG"
}
Association : {
    "Subnet": {
        "Id": "
```

```

        ...
        "/subscriptions/[Subscription
ID]/resourceGroups/RG1/providers/Microsoft.Network/virtualNetworks/WestUS-VNet1/subnets/Subnet1"
    }
}
EffectiveSecurityRules : [
{
    "Name": "securityRules/denyRDP",
    "Protocol": "Tcp",
    "SourcePortRange": "0-65535",
    "DestinationPortRange": "3389-3389",
    "SourceAddressPrefix": "Internet",
    "DestinationAddressPrefix": "0.0.0.0/0",
    "ExpandedSourceAddressPrefix": [
        ...
    ],
    "ExpandedDestinationAddressPrefix": [],
    "Access": "Deny",
    "Priority": 1000,
    "Direction": "Inbound"
},
{
    "Name": "defaultSecurityRules/AllowVnetInBound",
    "Protocol": "All",
    "SourcePortRange": "0-65535",
    "DestinationPortRange": "0-65535",
    "SourceAddressPrefix": "VirtualNetwork",
    "DestinationAddressPrefix": "VirtualNetwork",
    "ExpandedSourceAddressPrefix": [
        "10.9.0.0/16",
        "168.63.129.16/32",
        "10.0.0.0/16",
        "10.1.0.0/16"
    ],
    "ExpandedDestinationAddressPrefix": [
        "10.9.0.0/16",
        "168.63.129.16/32",
        "10.0.0.0/16",
        "10.1.0.0/16"
    ],
    "Access": "Allow",
    "Priority": 65000,
    "Direction": "Inbound"
},...
]

```

Note the following information in the output:

- There are two **NetworkSecurityGroup** sections: One is associated with a subnet (*Subnet1*) and one is associated with a NIC (*VM1-NIC1*). In this example, an NSG has been applied to each.
- **Association** shows the resource (subnet or NIC) a given NSG is associated with. If the NSG resource is moved/disassociated immediately before running this command, you may need to wait a few seconds for the change to reflect in the command output.
- The rule names that are prefaced with *defaultSecurityRules*: When an NSG is created, several default security rules are created within it. Default rules can't be removed, but they can be overridden with higher priority rules. Read the [NSG overview](#) article to learn more about NSG default security rules.
- **ExpandedAddressPrefix** expands the address prefixes for NSG default tags. Tags represent multiple address prefixes. Expansion of the tags can be useful when troubleshooting VM connectivity to/from specific address prefixes. For example, with VNET peering, VIRTUAL_NETWORK tag expands to show peered VNet prefixes in the previous output.

NOTE

The command only shows effective rules if an NSG is associated with either a subnet, a NIC, or both. A VM may have multiple NICs with different NSGs applied. When troubleshooting, run the command for each NIC.

3. To ease filtering over larger number of NSG rules, enter the following commands to troubleshoot further:

```
$NSGs = Get-AzureRmEffectiveNetworkSecurityGroup -NetworkInterfaceName VM1-NIC1 -ResourceGroupName RG1  
$NSGs.EffectiveSecurityRules | Sort-Object Direction, Access, Priority | Out-GridView
```

A filter for RDP traffic (TCP port 3389), is applied to the grid view, as shown in the following picture:

Name	Protocol	SourcePortRange	DestinationPortRange	SourceAddressPrefix	DestinationAddressPrefix	ExpandedSourceAddressPrefix	ExpandedDestinationAddressPrefix	Access	Priority	Direction
securityRules/Allow-All-RDP	Tcp	0-65535	3389-3389	0.0.0.0/0	0.0.0.0/0			Allow	100	Inbound
securityRules/Deny-All-RDP	Tcp	0-65535	3389-3389	0.0.0.0/0	0.0.0.0/0			Deny	100	Inbound

4. As you can see in the grid view, there are both allow and deny rules for RDP. The output from step 2 shows that the *DenyRDP* rule is in the NSG applied to the subnet. For inbound rules, NSGs applied to the subnet are processed first. If a match is found, the NSG applied to the network interface is not processed. In this case, the *DenyRDP* rule from the subnet blocks RDP to the VM (**VM1**).

NOTE

A VM may have multiple NICs attached to it. Each may be connected to a different subnet. Since the commands in the previous steps are run against a NIC, it's important to ensure that you specify the NIC you're having the connectivity failure to. If you're not sure, you can always run the commands against each NIC attached to the VM.

5. To RDP into VM1, change the *Deny RDP* (3389) rule to *Allow RDP(3389)* in the **Subnet1-NSG** NSG. Confirm that TCP port 3389 is open by opening an RDP connection to the VM or using the PsPing tool. You can learn more about PsPing by reading the [PsPing download page](#)

You can or remove rules from an NSG by using the information in the output from the following command:

```
Get-Help *-AzureRmNetworkSecurityRuleConfig
```

Considerations

Consider the following points when troubleshooting connectivity problems:

- Default NSG rules will block inbound access from the internet and only permit VNet inbound traffic. Rules should be explicitly added to allow inbound access from Internet, as required.
- If there are no NSG security rules causing a VM's network connectivity to fail, the problem may be due to:
 - Firewall software running within the VM's operating system
 - Routes configured for virtual appliances or on-premises traffic. Internet traffic can be redirected to on-premises via forced-tunneling. An RDP/SSH connection from the Internet to your VM may not work with this setting, depending on how the on-premises network hardware handles this traffic. Read the [Troubleshooting Routes](#) article to learn how to diagnose route problems that may be impeding the flow of traffic in and out of the VM.
- If you have peered VNets, by default, the VIRTUAL_NETWORK tag will automatically expand to include prefixes for peered VNets. You can view these prefixes in the **ExpandedAddressPrefix** list, to troubleshoot any issues related to VNet peering connectivity.

- Effective security rules are only shown if there is an NSG associated with the VM's NIC and or subnet.
- If there are no NSGs associated with the NIC or subnet and you have a public IP address assigned to your VM, all ports will be open for inbound and outbound access. If the VM has a public IP address, applying NSGs to the NIC or subnet is strongly recommended.

Troubleshoot routes using the Azure Portal

4/11/2018 • 7 min to read • [Edit Online](#)

If you are experiencing network connectivity issues to or from your Azure Virtual Machine (VM), routes may be impacting your VM traffic flows. This article provides an overview of diagnostics capabilities for routes to help troubleshoot further.

Route tables are associated with subnets and are effective on all network interfaces (NIC) in that subnet. The following types of routes can be applied to each network interface:

- **System routes:** By default, every subnet created in an Azure Virtual Network (VNet) has system route tables that allow local VNet traffic, on-premises traffic via VPN gateways, and Internet traffic. System routes also exist for peered VNets.
- **BGP routes:** Propagated to network interfaces through ExpressRoute or site-to-site VPN connections. Learn more about BGP routing by reading the [BGP with VPN gateways](#) and [ExpressRoute overview](#) articles.
- **User-defined routes (UDR):** If you are using network virtual appliances or are forced-tunneling traffic to an on-premises network via a site-to-site VPN, you may have user-defined routes (UDRs) associated with your subnet route table. If you're not familiar with UDRs, read the [user-defined routes](#) article.

With the various routes that can be applied to a network interface, it can be difficult to determine which aggregate routes are effective. To help troubleshoot VM network connectivity, you can view all the effective routes for a network interface in the Azure Resource Manager deployment model.

Using Effective Routes to troubleshoot VM traffic flow

This article uses the following scenario as an example to illustrate how to troubleshoot the effective routes for a network interface:

A VM (VM1) connected to the VNet (VNet1, prefix: 10.9.0.0/16) fails to connect to a VM(VM3) in a newly peered VNet (VNet3, prefix 10.10.0.0/16). There are no UDRs or BGP routes applied to VM1-NIC1 network interface connected to the VM, only system routes are applied.

This article explains how to determine the cause of the connection failure, using effective routes capability in Azure Resource Management deployment model. While the example uses only system routes, the same steps can be used to determine inbound and outbound connection failures over any route type.

NOTE

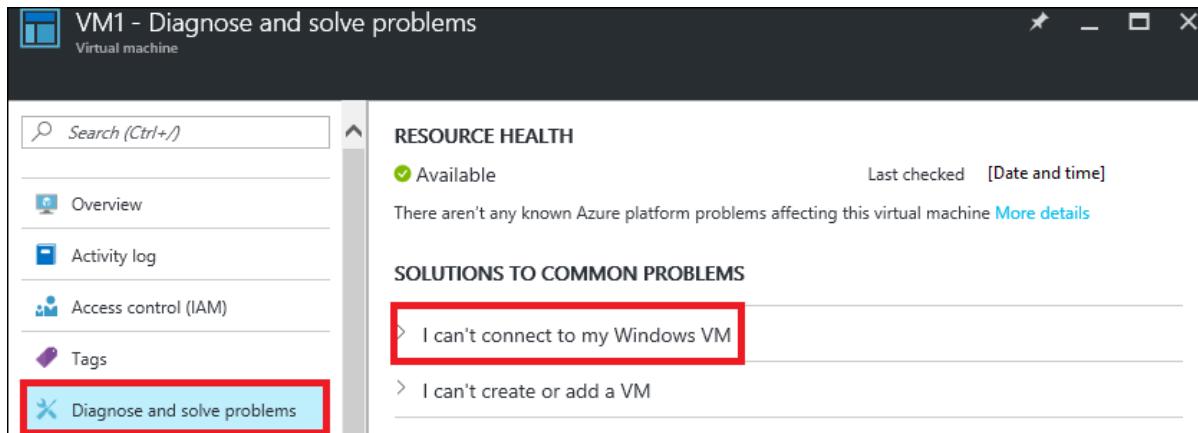
If your VM has more than one NIC attached, check effective routes for each of the NICs to diagnose network connectivity issues to and from a VM.

View effective routes for a virtual machine

To see the aggregate routes that are applied to a VM, complete the following steps:

1. Login to the Azure portal at <https://portal.azure.com>. Your account must be assigned the `Microsoft.Network/networkInterfaces/effectiveRouteTable/action` operation for the network interface. To learn how to assign operations to accounts, see [Create custom roles for Azure Role-Based Access Control](#).
2. Click **All services**, then click **Virtual machines** in the list that appears.
3. Select a VM to troubleshoot from the list that appears and a VM blade with options appears.
4. Click **Diagnose & solve problems** and then select a common problem. For this example, **I can't connect**

to my Windows VM is selected.



5. Steps appear under the problem, as shown in the following picture:

RESOURCES

vm1

RESOURCE HEALTH

Available Last checked 9/22/2016 12:01:00 PM

There aren't any known Azure platform problems affecting this virtual machine [More details](#)

SOLUTIONS TO COMMON PROBLEMS

› I can't connect to my Windows VM

› I can't create or add a VM

Recommended steps

To resolve common issues, try one or more of the following steps.

1. Review your VM's [console screenshot](#) to correct boot problems
2. Reset Remote Access to address remote server issues
[Reset remote access using PowerShell or CLI](#)
3. Restart the Virtual Machine to address startup issues by clicking 'Restart' at the top of the VM resource blade
4. Address Azure host issues by [redeploying](#), which will migrate the VM to a new Azure host
5. To connect to your VM via RDP, please review [effective security group rules](#) to ensure inbound "Allow" NSG rule exists for RDP port(3389)
6. RDP to your VM from Internet will not work with force tunneling enabled. Review [effective routes](#)
With force tunneling, all outbound traffic destined to Internet will be redirected to on-premises
7. If you're getting an RDP license error, use 'mstsc/admin' as a work around. If needed, uninstall or buy an RDS license.
[Address Remote Desktop License Server error](#)

Click **effective routes** in the list of recommended steps.

6. The **Effective routes** blade appears, as shown in the following picture:

Scope Virtual machine (VM1)

Network interface VM1-NIC1

Effective routes

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE
Default	Active	10.9.0.0/16	VnetLocal
Default	Active	0.0.0.0/0	Internet
Default	Active	10.0.0.0/8	None

If your VM has only one NIC, it is selected by default. If you have more than one NIC, select the NIC for which you want to view the effective routes.

NOTE

If the VM associated with the NIC is not in a running state, effective routes will not be shown. Only the first 200 effective routes are shown in the portal. For the full list, click **Download**. You can further filter on the results from the downloaded .csv file.

Notice the following in the output:

- **Source:** Indicates the type of route. System routes are shown as *Default*, UDRs are shown as *User* and gateway routes (static or BGP) are shown as *VPNGateway*.
 - **State:** Indicates state of the effective route. Possible values are *Active* or *Invalid*.
 - **AddressPrefixes:** Specifies the address prefix of the effective route in CIDR notation.
 - **nextHopType:** Indicates the next hop for the given route. Possible values are *VirtualAppliance*, *Internet*, *VNetLocal*, *VNetPeering*, or *Null*. A value of *Null* for **nextHopType** in a UDR may indicate an invalid route. For example, if **nextHopType** is *VirtualAppliance* and the network virtual appliance VM is not in a provisioned/running state. If **nextHopType** is *VPNGateway* and there is no gateway provisioned/running in the given VNet, the route may become invalid.
7. There is no route listed to the *WestUS-VNET3* VNet (Prefix 10.10.0.0/16) from the *WestUS-VNet1* (Prefix 10.9.0.0/16) in the picture in the previous step. In the following picture, the peering link is in the *Disconnected* state:

NAME	PEERING STATUS	PEER	GATEWAY TRANSIT
PeerLink1	Disconnected	WestUS-VNET3	Disabled

The bi-directional link for the peering is broken, which explains why VM1 could not connect to VM3 in the *WestUS-VNet3* VNet.

8. The following picture shows the routes after establishing the bi-directional peering link:

NAME	PEERING STATUS	PEER	GATEWAY TRANSIT
PeerLink1	Connected	WestUS-VNET3	Disabled

For more troubleshooting scenarios for forced-tunneling and route evaluation, read the [Considerations](#) section of this article.

View effective routes for a network interface

If network traffic flow is impacted for a particular network interface (NIC), you can view a full list of effective routes on a NIC directly. To see the aggregate routes that are applied to a NIC, complete the following steps:

1. Login to the Azure portal at <https://portal.azure.com>.
2. Click **All services**, then click **Network interfaces**
3. Search the list for the name of a NIC, or select it from the list that appears. In this example, **VM1-NIC1** is selected.
4. Select **Effective routes** in the **Network interface** blade, as shown in the following picture:



The **Scope** defaults to the network interface selected.

A screenshot of the Azure portal showing the 'Effective routes' table for a specific network interface. The table has columns for SOURCE, STATE, ADDRESS PREFIXES, and NEXT HOP TYPE. There are three rows of data. The first row has a SOURCE of 'Default', a STATE of 'Active', an ADDRESS PREFIX of '10.9.0.0/16', and a NEXT HOP TYPE of 'VnetLocal'. The second row has a SOURCE of 'Default', a STATE of 'Active', an ADDRESS PREFIX of '10.10.0.0/16', and a NEXT HOP TYPE of 'VNetPeering'. The third row has a SOURCE of 'Default', a STATE of 'Active', an ADDRESS PREFIX of '0.0.0.0/0', and a NEXT HOP TYPE of 'Internet'. A red box highlights the 'Scope' header and the value 'Network interface (VM1-NIC1)' in the first column of the table.

Scope	Network interface (VM1-NIC1)
Default	Active
Default	Active
Default	Active

View effective routes for a route table

When modifying user-defined routes (UDRs) in a route table, you may want to review the impact of the routes being added on a particular VM. A route table can be associated with any number of subnets. You can now view all the effective routes for all the NICs that a given route table is applied to, without having to switch context from the given route table blade.

For this example, a UDR (*UDRoute*) is specified in a route table (*UDRouteTable*). This route sends all Internet traffic from *Subnet1* in the *WestUS-VNet1* VNet, through a network virtual appliance (NVA), in *Subnet2* of the same VNet. The route is shown in the following picture:

The screenshot shows the Azure portal interface. On the left, there's a sidebar with 'Network interfaces' and a search bar. Below it, 'Subscriptions: All 3 selected' is shown with 'VM1-NIC1' selected. A dropdown menu shows 'All subscriptions'. Under 'NAME', 'VM1-NIC1' is listed with a red box around it. On the right, the main area has a title 'VM1-NIC1 - Effective routes' and a 'Network interface' subtitle. A navigation menu on the right includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'IP configurations', 'DNS servers', 'Network security group', 'Effective security rules', and 'Effective routes'. The 'Effective routes' item is highlighted with a red box.

To see the aggregate routes for a route table, complete the following steps:

1. Login to the Azure portal at <https://portal.azure.com>.
2. Click **All services**, then click **Route tables**
3. Search the list for the route table you want to see aggregate routes for and select it. In this example, **UDRouteTable** is selected. A blade for the selected route table appears, as shown in the following picture:

The screenshot shows the Azure portal interface. On the left, there's a sidebar with 'Route tables' and a search bar. Below it, 'Subscriptions: All 3 selected' is shown with 'UDRouteTable' selected. A dropdown menu shows 'All subscriptions'. Under 'NAME', 'UDRouteTable' is listed with a red box around it. On the right, the main area has a title 'UDRouteTable - Effective routes' and a 'Route table' subtitle. A navigation menu on the right includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Routes', 'Subnets', and 'Effective routes'. The 'Effective routes' item is highlighted with a red box.

4. Select **Effective Routes** in the **Route table** blade. The **Scope** is set to the route table you selected.
5. A route table can be applied to multiple subnets. Select the **Subnet** you want to review from the list. In this example, **Subnet1** is selected.
6. Select a **Network Interface**. All NICs connected to the selected subnet are listed. In this example, **VM1-NIC1** is selected.

The screenshot shows the 'UDRouteTable - Effective routes' blade in the Azure portal. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Routes, Subnets, and Effective routes (which is selected). The main area shows a table of effective routes. At the top, there are filters for Scope (Route table (UDRouteTable)), Subnet (Subnet1), and Network interface (VM1-NIC1). The table has columns: SOURCE, STATE, ADDRESS PREFIXES, NEXT HOP TYPE, NEXT HOP TYPE IP ADDRESS, and USER DEFINED ROUTE. The rows are: Default (Active, 10.9.0.0/16, VnetLocal, -, -), Default (Active, 10.10.0.0/16, VNetPeering, -, -), and User (Active, 0.0.0.0/0, VirtualAppliance, 10.9.1.4, UDRoute). A red box highlights the filter section and the User-defined route row.

NOTE

If the NIC is not associated with a running VM, no effective routes are shown.

Considerations

A few things to keep in mind when reviewing the list of routes returned:

- Routing is based on Longest Prefix Match (LPM) among UDRs, BGP and system routes. If there is more than one route with the same LPM match, then a route is selected based on its origin in the following order:
 - User-defined route
 - BGP route
 - System (Default) route

With effective routes, you can only see effective routes that are LPM match based on all the available routes. By showing how the routes are actually evaluated for a given NIC, this makes it a lot easier to troubleshoot specific routes that may be impacting connectivity to/from your VM.

- If you have UDRs and are sending traffic to a network virtual appliance (NVA), with *VirtualAppliance* as **nextHopType**, ensure that IP forwarding is enabled on the NVA receiving the traffic or packets are dropped.
- If Forced tunneling is enabled, all outbound Internet traffic will be routed to on-premises. RDP/SSH from Internet to your VM may not work with this setting, depending on how the on-premises handles this traffic. Forced-tunneling can be enabled:
 - If using site-to-site VPN, by setting a user-defined route (UDR) with nextHopType as VPN Gateway
 - If a default route is advertised over BGP
- For VNet peering traffic to work correctly, a system route with **nextHopType** *VNetPeering* must exist for the peered VNet's prefix range. If such a route doesn't exist and the VNet peering link looks OK:
 - Wait a few seconds and retry if it's a newly established peering link. It occasionally takes longer to propagate routes to all the network interfaces in a subnet.
 - Network Security Group (NSG) rules may be impacting the traffic flows. For more information, see the [Troubleshoot Network Security Groups](#) article.

Troubleshoot routes using Azure PowerShell

4/11/2018 • 5 min to read • [Edit Online](#)

If you are experiencing network connectivity issues to or from your Azure Virtual Machine (VM), routes may be impacting your VM traffic flows. This article provides an overview of diagnostics capabilities for routes to help troubleshoot further.

Route tables are associated with subnets and are effective on all network interfaces (NIC) in that subnet. The following types of routes can be applied to each network interface:

- **System routes:** By default, every subnet created in an Azure Virtual Network (VNet) has system route tables that allow local VNet traffic, on-premises traffic via VPN gateways, and Internet traffic. System routes also exist for peered VNets.
- **BGP routes:** Propagated to network interfaces through ExpressRoute or site-to-site VPN connections. Learn more about BGP routing by reading the [BGP with VPN gateways](#) and [ExpressRoute overview](#) articles.
- **User-defined routes (UDR):** If you are using network virtual appliances or are forced-tunneling traffic to an on-premises network via a site-to-site VPN, you may have user-defined routes (UDRs) associated with your subnet route table. If you're not familiar with UDRs, read the [user-defined routes](#) article.

With the various routes that can be applied to a network interface, it can be difficult to determine which aggregate routes are effective. To help troubleshoot VM network connectivity, you can view all the effective routes for a network interface in the Azure Resource Manager deployment model.

Using Effective Routes to troubleshoot VM traffic flow

This article uses the following scenario as an example to illustrate how to troubleshoot the effective routes for a network interface:

A VM (*VM1*) connected to the VNet (*VNet1*, prefix: 10.9.0.0/16) fails to connect to a VM(*VM3*) in a newly peered VNet (*VNet3*, prefix 10.10.0.0/16). There are no UDRs or BGP routes applied to VM1-NIC1 network interface connected to the VM, only system routes are applied.

This article explains how to determine the cause of the connection failure, using effective routes capability in Azure Resource Management deployment model. While the example uses only system routes, the same steps can be used to determine inbound and outbound connection failures over any route type.

NOTE

If your VM has more than one NIC attached, check effective routes for each of the NICs to diagnose network connectivity issues to and from a VM.

View effective routes for a virtual machine

To see the aggregate routes that are applied to a VM, complete the following steps:

View effective routes for a network interface

To see the aggregate routes that are applied to a network interface, complete the following steps:

1. Start an Azure PowerShell session and login to Azure. If you're not familiar with Azure PowerShell, read the [How to install and configure Azure PowerShell](#) article. Your account must be assigned the `Microsoft.Network/networkInterfaces/effectiveRouteTable/action` operation for the network interface. To learn how to assign operations to accounts, see [Create custom roles for Azure Role-Based Access Control](#).

2. The following command returns all routes applied to a network interface named *VM1-NIC1* in the resource group *RG1*.

```
Get-AzureRmEffectiveRouteTable -NetworkInterfaceName VM1-NIC1 -ResourceGroupName RG1
```

TIP

If you don't know the name of a network interface, type the following command to retrieve the names of all network interfaces in a resource group.*

```
Get-AzureRmNetworkInterface -ResourceGroupName RG1 | Format-Table Name
```

The following output looks similar to the output for each route applied to the subnet the NIC is connected to:

```
Name :  
State : Active  
AddressPrefix : {10.9.0.0/16}  
NextHopType : VNetLocal  
NextHopIpAddress : {}  
  
Name :  
State : Active  
AddressPrefix : {0.0.0.0/16}  
NextHopType : Internet  
NextHopIpAddress : {}
```

Notice the following in the output:

- **Name:** Name of the effective route may be empty, unless explicitly specified, for user-defined routes.
- **State:** Indicates state of the effective route. Possible values are "Active" or "Invalid"
- **AddressPrefixes:** Specifies the address prefix of the effective route in CIDR notation.
- **nextHopType:** Indicates the next hop for the given route. Possible values are *VirtualAppliance*, *Internet*, *VNetLocal*, *VNetPeering*, or *Null*. A value of *Null* for **nextHopType** in a UDR may indicate an invalid route. For example, if **nextHopType** is *VirtualAppliance* and the network virtual appliance VM is not in a provisioned/running state. If **nextHopType** is *VPNGateway* and there is no gateway provisioned/running in the given VNet, the route may become invalid.
- **NextHopIpAddress:** Specifies the IP address of the next hop of the effective route.

The following command returns the routes in an easier to view table:

```
Get-AzureRmEffectiveRouteTable -NetworkInterfaceName VM1-NIC1 -ResourceGroupName RG1 | Format-Table
```

The following output is some of the output received for the scenario described previously:

Name	State	AddressPrefix	NextHopType	NextHopIpAddress
Active	{10.9.0.0/16}	VnetLocal	{}	
Active	{0.0.0.0/0}	Internet	{}	

3. There is no route listed to the *WestUS-VNet3* VNet (Prefix 10.10.0.0/16)** from *WestUS-VNet1* (Prefix 10.9.0.0/16) in the output from the previous step. As shown in the following picture, the VNet peering link with the *WestUS-VNet3* VNet is in the *Disconnected* state.

Peering Status				
Name	Peering Status	Peer	Gateway Transit	...
PeerLink1	Disconnected	WestUS-VNET3	Disabled	...

The bi-directional link for the peering is broken, which explains why VM1 could not connect to VM3 in the *WestUS-VNet3* VNet. Setup a bi-directional VNet peering link again for *WestUS-VNet1* and *WestUS-VNet3* VNets. The output returned after the VNet peering link is correctly established follows:

Name	State	AddressPrefix	NextHopType	NextHopIpAddress
Active	{10.9.0.0/16}	VnetLocal	{}	
Active	{10.10.0.0/16}	VNetPeering	{}	
Active	{0.0.0.0/0}	Internet	{}	

Once you determine the issue, you can add, remove, or change routes and route tables. Type the following command to see a list of the commands used to do so:

```
Get-Help *-AzureRmRouteConfig
```

Considerations

A few things to keep in mind when reviewing the list of routes returned:

- Routing is based on Longest Prefix Match (LPM) among UDRs, BGP and system routes. If there is more than one route with the same LPM match, then a route is selected based on its origin in the following order:
 - User-defined route
 - BGP route
 - System (Default) route
 With effective routes, you can only see effective routes that are LPM match based on all the available routes. By showing how the routes are actually evaluated for a given NIC, this makes it a lot easier to troubleshoot specific routes that may be impacting connectivity to/from your VM.
- If you have UDRs and are sending traffic to a network virtual appliance (NVA), with *VirtualAppliance* as **nextHopType**, ensure that IP forwarding is enabled on the NVA receiving the traffic or packets are dropped.
- If Forced tunneling is enabled, all outbound Internet traffic will be routed to on-premises. RDP/SSH from Internet to your VM may not work with this setting, depending on how the on-premises handles this traffic. Forced-tunneling can be enabled:
 - If using site-to-site VPN, by setting a user-defined route (UDR) with nextHopType as VPN Gateway
 - If a default route is advertised over BGP
- For VNet peering traffic to work correctly, a system route with **nextHopType VNetPeering** must exist for the peered VNet's prefix range. If such a route doesn't exist and the VNet peering link looks OK:
 - Wait a few seconds and retry if it's a newly established peering link. It occasionally takes longer to propagate routes to all the network interfaces in a subnet.
 - Network Security Group (NSG) rules may be impacting the traffic flows. For more information, see the [Troubleshoot Network Security Groups](#) article.

Bandwidth/Throughput testing (NTTTCP)

1/26/2018 • 3 min to read • [Edit Online](#)

When testing network throughput performance in Azure, it's best to use a tool that targets the network for testing and minimizes the use of other resources that could impact performance. NTTTCP is recommended.

Copy the tool to two Azure VMs of the same size. One VM functions as SENDER and the other as RECEIVER.

Deploying VMs for testing

For the purposes of this test, the two VMs should be in either the same Cloud Service or the same Availability Set so that we can use their internal IPs and exclude the Load Balancers from the test. It is possible to test with the VIP but this kind of testing is outside the scope of this document.

Make a note of the RECEIVER's IP address. Let's call that IP "a.b.c.r"

Make a note of the number of cores on the VM. Let's call this "#num_cores"

Run the NTTTCP test for 300 seconds (or 5 minutes) on the sender VM and receiver VM.

Tip: When setting up this test for the first time, you might try a shorter test period to get feedback sooner. Once the tool is working as expected, extend the test period to 300 seconds for the most accurate results.

NOTE

The sender **and** receiver must specify **the same** test duration parameter (-t).

To test a single TCP stream for 10 seconds:

Receiver parameters: nttcp -r -t 10 -P 1

Sender parameters: nttcp -s10.27.33.7 -t 10 -n 1 -P 1

NOTE

The preceding sample should only be used to confirm your configuration. Valid examples of testing are covered later in this document.

Testing VMs running WINDOWS:

Get NTTTCP onto the VMs.

Download the latest version: <https://gallery.technet.microsoft.com/NTtcp-Version-528-Now-f8b12769>

Or search for it if moved: <https://www.bing.com/search?q=nttcp+download> < -- should be first hit

Consider putting NTTTCP in separate folder, like c:\tools

Allow NTTTCP through the Windows firewall

On the RECEIVER, create an Allow rule on the Windows Firewall to allow the NTTTCP traffic to arrive. It's easiest to allow the entire NTTTCP program by name rather than to allow specific TCP ports inbound.

Allow nttcp through the Windows Firewall like this:

```
netsh advfirewall firewall add rule program=<PATH>\nttcp.exe name="nttcp" protocol=any dir=in action=allow enable=yes profile=ANY
```

For example, if you copied ntttcp.exe to the "c:\tools" folder, this would be the command:

```
netsh advfirewall firewall add rule program=c:\tools\ntttcp.exe name="ntttcp" protocol=any dir=in action=allow enable=yes profile=ANY
```

Running NTTTCP tests

Start NTTTCP on the RECEIVER (**run from CMD**, not from PowerShell):

```
ntttcp -r -m [2*#num_cores],*,a.b.c.r -t 300
```

If the VM has four cores and an IP address of 10.0.0.4, it would look like this:

```
ntttcp -r -m 8,*,10.0.0.4 -t 300
```

Start NTTTCP on the SENDER (**run from CMD**, not from PowerShell):

```
ntttcp -s -m 8,*,10.0.0.4 -t 300
```

Wait for the results.

Testing VMs running LINUX:

Use nttcp-for-linux. It is available from <https://github.com/Microsoft/ntttcp-for-linux>

On the Linux VMs (both SENDER and RECEIVER), run these commands to prepare nttcp-for-linux on your VMs:

CentOS - Install Git:

```
yum install gcc -y  
yum install git -y
```

Ubuntu - Install Git:

```
apt-get -y install build-essential  
apt-get -y install git
```

Make and Install on both:

```
git clone https://github.com/Microsoft/ntttcp-for-linux  
cd nttcp-for-linux/src  
make && make install
```

As in the Windows example, we assume the Linux RECEIVER's IP is 10.0.0.4

Start NTTTCP-for-Linux on the RECEIVER:

```
ntttcp -r -t 300
```

And on the SENDER, run:

```
ntttcp -s10.0.0.4 -t 300
```

Test length defaults to 60 seconds if no time parameter is given

Testing between VMs running Windows and LINUX:

On this scenarios we should enable the no-sync mode so the test can run. This is done by using the **-N flag** for Linux, and **-ns flag** for Windows.

From Linux to Windows:

Receiver :

```
nttcp -r -m <2 x nr cores>,*,<Windows server IP>
```

Sender :

```
nttcp -s -m <2 x nr cores>,*,<Windows server IP> -N -t 300
```

From Windows to Linux:

Receiver :

```
nttcp -r -m <2 x nr cores>,*,<Linux server IP>
```

Sender :

```
nttcp -s -m <2 x nr cores>,*,<Linux server IP> -ns -t 300
```

Testing Cloud Service Instances:

You need to add following section into your ServiceDefinition.csdef

```
<Endpoints>
  <InternalEndpoint name="Endpoint3" protocol="any" />
</Endpoints>
```

Next steps

- Depending on results, there may be room to [Optimize network throughput machines](#) for your scenario.
- Read about how [bandwidth is allocated to virtual machines](#)
- Learn more with [Azure Virtual Network frequently asked questions \(FAQ\)](#)

Troubleshooting: Failed to delete a virtual network in Azure

12/12/2017 • 2 min to read • [Edit Online](#)

You might receive errors when you try to delete a virtual network in Microsoft Azure. This article provides troubleshooting steps to help you resolve this problem.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Troubleshooting guidance

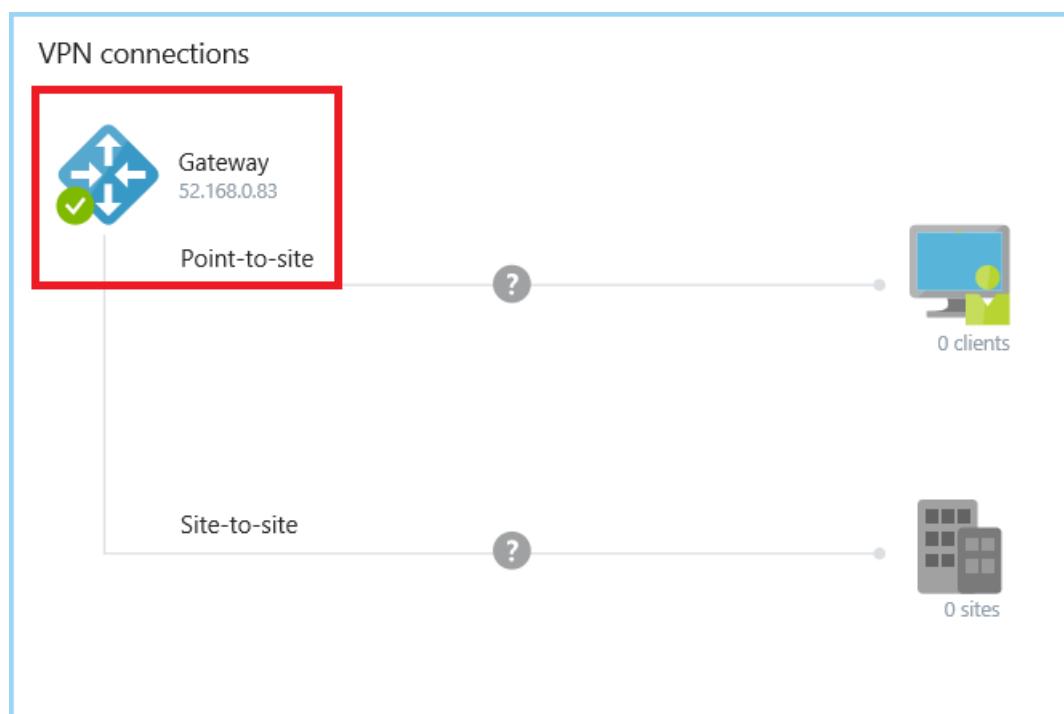
1. [Check whether a virtual network gateway is running in the virtual network.](#)
2. [Check whether an application gateway is running in the virtual network.](#)
3. [Check whether Azure Active Directory Domain Service is enabled in the virtual network.](#)
4. [Check whether the virtual network is connected to other resource.](#)
5. [Check whether a virtual machine is still running in the virtual network.](#)
6. [Check whether the virtual network is stuck in migration.](#)

Troubleshooting steps

Check whether a virtual network gateway is running in the virtual network

To remove the virtual network, you must first remove the virtual network gateway.

For classic virtual networks, go to the **Overview** page of the classic virtual network in the Azure portal. In the **VPN connections** section, if the gateway is running in the virtual network, you will see the IP address of the gateway.



For virtual networks, go to the **Overview** page of the virtual network. Check **Connected devices** for the virtual

network gateway.

Connected devices				
DEVICE	TYPE	IP ADDRESS	SUBNET	
testrbac914	Network interface	10.0.0.4	default	
thomasvm195	Network interface	10.0.0.5	default	
Gateway01	Virtual network gateway	-	GatewaySubnet	

Before you can remove the gateway, first remove any **Connection** objects in the gateway.

Check whether an application gateway is running in the virtual network

Go to the **Overview** page of the virtual network. Check the **Connected devices** for the application gateway.

Connected devices				
DEVICE	TYPE	IP ADDRESS	SUBNET	
testrbac914	Network interface	10.0.0.4	default	
thomasvm195	Network interface	10.0.0.5	default	
AppGateway-Test	Application gateway		gateway	

If there is an application gateway, you must remove it before you can delete the virtual network.

Check whether Azure Active Directory Domain Service is enabled in the virtual network

If the Active Directory Domain Service is enabled and connected to the virtual network, you cannot delete this virtual network.

The screenshot shows the Azure AD Domain Services blade. On the left, there's a navigation menu with 'Overview', 'Activity log', and 'Access control (IAM)'. Below that is a 'MANAGE' section with a 'Properties' button. On the right, it displays 'DNS DOMAIN NAME' (with placeholder '<Domain name>'), 'LOCATION' (set to 'East Asia'), and a large 'AVAILABLE IN VIRTUAL NETWORK/SUBNET' section containing '<VNet name/subnet>'.

To disable the service, see [Disable Azure Active Directory Domain Services using the Azure portal](#).

Check whether the virtual network is connected to other resource

Check for Circuit Links, connections, and virtual network peerings. Any of these can cause a virtual network deletion to fail.

The recommended deletion order is as follows:

1. Gateway connections
2. Gateways
3. IPs
4. Virtual network peerings
5. App Service Environment (ASE)

Check whether a virtual machine is still running in the virtual network

Make sure that no virtual machine is in the virtual network.

Check whether the virtual network is stuck in migration

If the virtual network is stuck in a migration state, it cannot be deleted. Run the following command to abort the migration, and then delete the virtual network.

```
Move-AzureVirtualNetwork -VirtualNetworkName "Name" -Abort
```

Next steps

- [Azure Virtual Network](#)
- [Azure Virtual Network frequently asked questions \(FAQ\)](#)

Troubleshooting connectivity problems between Azure VMs

9/27/2017 • 3 min to read • [Edit Online](#)

You might experience connectivity problems between Azure virtual machines (VMs). This article provides troubleshooting steps to help you resolve this problem.

If your Azure issue is not addressed in this article, visit the Azure forums on [MSDN and Stack Overflow](#). You can post your issue in these forums, or post to [@AzureSupport on Twitter](#). You also can submit an Azure support request. To submit a support request, on the [Azure support](#) page, select **Get support**.

Symptom

One Azure VM cannot connect to another Azure VM.

Troubleshooting guidance

1. [Check whether NIC is misconfigured](#)
2. [Check whether network traffic is blocked by NSG or UDR](#)
3. [Check whether network traffic is blocked by VM firewall](#)
4. [Check whether VM app or service is listening on the port](#)
5. [Check whether the problem is caused by SNAT](#)
6. [Check whether traffic is blocked by ACLs for the classic VM](#)
7. [Check whether the endpoint is created for the classic VM](#)
8. [Try to connect to a VM network share](#)
9. [Check Inter-Vnet connectivity](#)

Troubleshooting steps

Follow these steps to troubleshoot the problem. After you complete each step, check whether the problem is resolved.

Step 1: Check whether NIC is misconfigured

Follow the steps in [How to reset network interface for Azure Windows VM](#).

If the problem occurs after you modify the network interface (NIC), follow these steps:

Multi-NIC VMs

1. Add a NIC.
2. Fix the problems in the bad NIC or remove the bad NIC. Then add the NIC again.

For more information, see [Add network interfaces to or remove from virtual machines](#).

Single-NIC VM

- [Redeploy Windows VM](#)
- [Redeploy Linux VM](#)

Step 2: Check whether network traffic is blocked by NSG or UDR

Use [Network Watcher IP Flow Verify](#) and [NSG Flow Logging](#) to determine whether there is a Network Security Group (NSG) or User-Defined Route (UDR) that is interfering with traffic flow.

Step 3: Check whether network traffic is blocked by VM firewall

Disable the firewall, and then test the result. If the problem is resolved, verify the firewall settings, and then re-enable the firewall.

Step 4: Check whether VM app or service is listening on the port

You can use one of the following methods to check whether the VM app or service is listening on the port.

- Run the following commands to check whether the server is listening on that port.

Windows VM

```
netstat -ano
```

Linux VM

```
netstat -l
```

- Run the **telnet** command on the virtual machine itself to test the port. If the test fails, the application or service is not configured to listen on that port.

Step 5: Check whether the problem is caused by SNAT

In some scenarios, the VM is placed behind a load balance solution that has a dependency on resources outside of Azure. In these scenarios, if you experience intermittent connection problems, the problem may be caused by [SNAT port exhaustion](#). To resolve the issue, create a VIP (or ILPIP for classic) for each VM that is behind the load balancer and secure with NSG or ACL.

Step 6: Check whether traffic is blocked by ACLs for the classic VM

An access control list (ACL) provides the ability to selectively permit or deny traffic for a virtual machine endpoint. For more information, see [Manage the ACL on an endpoint](#).

Step 7: Check whether the endpoint is created for the classic VM

All VMs that you create in Azure by using the classic deployment model can automatically communicate over a private network channel with other virtual machines in the same cloud service or virtual network. However, computers on other virtual networks require endpoints to direct the inbound network traffic to a virtual machine. For more information, see [How to set up endpoints](#).

Step 8: Try to connect to a VM network share

If you cannot connect to a VM network share, the problem may be caused by unavailable NICs in the VM. To delete the unavailable NICs, see [How to delete the unavailable NICs](#)

Step 9: Check Inter-Vnet connectivity

Use [Network Watcher IP Flow Verify](#) and [NSG Flow Logging](#) to determine whether there is a NSG or UDR that is interfering with traffic flow. You can also verify your Inter-Vnet configuration [here](#).

Need help? Contact support.

If you still need help, [contact support](#) to get your issue resolved quickly.

Configure reverse lookup zones for an SMTP banner check

2/9/2018 • 1 min to read • [Edit Online](#)

This article describes how to use a reverse zone in Azure DNS and create a Reverse DNS (PTR) record for SMTP Banner Check.

Symptom

If you host an SMTP server in Microsoft Azure, you may receive the following error message when send or receive a message from remote mail servers:

554: No PTR Record

Solution

For a virtual IP address in Azure, the reverse records are created in Microsoft owned domain zones, not custom domain zones.

To configure PTR records in Microsoft owned zones, use the -ReverseFqdn property on the PublicIpAddress resource. For more information, see [Configure reverse DNS for services hosted in Azure](#).

When you configure the PTR records, make sure that the IP address and the reverse FQDN are owned by the subscription. If you try to set a reverse FQDN that does not belong to the subscription, you receive the following error message:

```
Set-AzureRmPublicIpAddress : ReverseFqdn mail.contoso.com that PublicIPAddress ip01 is trying to use does not belong to subscription <Subscription ID>. One of the following conditions need to be met to establish ownership:
```

- 1) ReverseFqdn matches fqdn of any public ip resource under the subscription;
- 2) ReverseFqdn resolves to the fqdn (through CName records chain) of any public ip resource under the subscription;
- 3) It resolves to the ip address (through CName and A records chain) of a static public ip resource under the subscription.

If you manually change your SMTP banner to match our default reverse FQDN, the remote mail server can still fail because it may expect the SMTP banner host to match the MX record for the domain.

Create a virtual network (classic) with multiple subnets

4/19/2018 • 5 min to read • [Edit Online](#)

IMPORTANT

Azure has two [different deployment models](#) for creating and working with resources: Resource Manager and classic. This article covers using the classic deployment model. Microsoft recommends creating most new virtual networks through the [Resource Manager](#) deployment model.

In this tutorial, learn how to create a basic Azure virtual network (classic) that has separate public and private subnets. You can create Azure resources, like Virtual machines and Cloud services in a subnet. Resources created in virtual networks (classic) can communicate with each other, and with resources in other networks connected to a virtual network.

Learn more about all [virtual network](#) and [subnet](#) settings.

WARNING

Virtual networks (classic) are immediately deleted by Azure when a [subscription is disabled](#). Virtual networks (classic) are deleted regardless of whether resources exist in the virtual network. If you later re-enable the subscription, resources that existed in the virtual network must be recreated.

You can create a virtual network (classic) by using the [Azure portal](#), the [Azure command-line interface \(CLI\) 1.0](#), or [PowerShell](#).

Portal

1. In an Internet browser, go to the [Azure portal](#). Log in using your [Azure account](#). If you don't have an Azure account, you can sign up for a [free trial](#).
2. Click **Create a resource** in the portal.
3. Enter *Virtual network* in the **Search the Marketplace** box at the top of the **New** pane that appears. Click **Virtual network** when it appears in the search results.
4. Select **Classic** in the **Select a deployment model** box in the **Virtual Network** pane that appears, then click **Create**.
5. Enter the following values on the **Create virtual network (classic)** pane and then click **Create**:

SETTING	VALUE
Name	myVnet
Address space	10.0.0.0/16
Subnet name	Public
Subnet address range	10.0.0.0/24
Resource group	Leave Create new selected, and then enter myResourceGroup .

SETTING	VALUE
Subscription and location	Select your subscription and location.

If you're new to Azure, learn more about [resource groups](#), [subscriptions](#), and [locations](#) (also referred to as *regions*).

6. In the portal, you can create only one subnet when you create a virtual network. In this tutorial, you create a second subnet after you create the virtual network. You might later create Internet-accessible resources in the **Public** subnet. You also might create resources that aren't accessible from the Internet in the **Private** subnet. To create the second subnet, enter **myVnet** in the **Search resources** box at the top of the page. Click **myVnet** when it appears in the search results.
7. Click **Subnets** (in the **SETTINGS** section) on the **Create virtual network (classic)** pane that appears.
8. Click **+Add** on the **myVnet - Subnets** pane that appears.
9. Enter **Private** for **Name** on the **Add subnet** pane. Enter **10.0.1.0/24** for **Address range**. Click **OK**.
10. On the **myVnet - Subnets** pane, you can see the **Public** and **Private** subnets that you created.
11. **Optional:** When you finish this tutorial, you might want to delete the resources that you created, so that you don't incur usage charges:
 - Click **Overview** on the **myVnet** pane.
 - Click the **Delete** icon on the **myVnet** pane.
 - To confirm the deletion, click **Yes** in the **Delete virtual network** box.

Azure CLI

1. You can either [install and configure the Azure CLI](#), or use the CLI within 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. To get help for CLI commands, type `azure <command> --help`.
2. In a CLI session, log in to Azure with the command that follows. If you click **Try it** in the box below, a Cloud Shell opens. You can log in to your Azure subscription, without entering the following command:

```
azure login
```

3. To ensure the CLI is in Service Management mode, enter the following command:

```
azure config mode asm
```

4. Create a virtual network with a private subnet:

```
azure network vnet create --vnet myVnet --address-space 10.0.0.0 --cidr 16 --subnet-name Private --subnet-start-ip 10.0.0.0 --subnet-cidr 24 --location "East US"
```

5. Create a public subnet within the virtual network:

```
azure network vnet subnet create --name Public --vnet-name myVnet --address-prefix 10.0.1.0/24
```

6. Review the virtual network and subnets:

```
azure network vnet show --vnet myVnet
```

7. **Optional:** You might want to delete the resources that you created when you finish this tutorial, so that you don't incur usage charges:

```
azure network vnet delete --vnet myVnet --quiet
```

NOTE

Though you can't specify a resource group to create a virtual network (classic) in using the CLI, Azure creates the virtual network in a resource group named *Default-Networking*.

PowerShell

1. Install the latest version of the PowerShell [Azure](#) module. If you're new to Azure PowerShell, see [Azure PowerShell overview](#).
2. Start a PowerShell session.
3. In PowerShell, log in to Azure by entering the `Add-AzureAccount` command.
4. Change the following path and filename, as appropriate, then export your existing network configuration file:

```
Get-AzureVNetConfig -ExportToFile c:\azure\NetworkConfig.xml
```

5. To create a virtual network with public and private subnets, use any text editor to add the **VirtualNetworkSite** element that follows to the network configuration file.

```
<VirtualNetworkSite name="myVnet" Location="East US">
  <AddressSpace>
    <AddressPrefix>10.0.0.0/16</AddressPrefix>
  </AddressSpace>
  <Subnets>
    <Subnet name="Private">
      <AddressPrefix>10.0.0.0/24</AddressPrefix>
    </Subnet>
    <Subnet name="Public">
      <AddressPrefix>10.0.1.0/24</AddressPrefix>
    </Subnet>
  </Subnets>
</VirtualNetworkSite>
```

Review the full [network configuration file schema](#).

6. Import the network configuration file:

```
Set-AzureVNetConfig -ConfigurationPath c:\azure\NetworkConfig.xml
```

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.

7. Review the virtual network and subnets:

```
Get-AzureVNetSite -VNetName "myVnet"
```

8. **Optional:** You might want to delete the resources that you created when you finish this tutorial, so that you don't incur usage charges. To delete the virtual network, complete steps 4-6 again, this time removing the **VirtualNetworkSite** element you added in step 5.

NOTE

Though you can't specify a resource group to create a virtual network (classic) in using PowerShell, Azure creates the virtual network in a resource group named *Default-Networking*.

Next steps

- To learn about all virtual network and subnet settings, see [Manage virtual networks](#) and [Manage virtual network subnets](#). You have various options for using virtual networks and subnets in a production environment to meet different requirements.
- To filter inbound and outbound subnet traffic, create and apply [network security groups](#) to subnets.
- Create a [Windows](#) or a [Linux](#) virtual machine, and then connect it to an existing virtual network.
- To connect two virtual networks in the same Azure location, create a [virtual network peering](#) between the virtual networks. You can peer a virtual network (Resource Manager) to a virtual network (classic), but you cannot create a peering between two virtual networks (classic).
- Connect the virtual network to an on-premises network by using a [VPN Gateway](#) or [Azure ExpressRoute](#) circuit.

Create a virtual network (classic) by using the Azure portal

4/19/2018 • 2 min to read • [Edit Online](#)

An Azure virtual network (VNet) is a representation of your own network in the cloud. You can control your Azure network settings and define DHCP address blocks, DNS settings, security policies, and routing. You can also further segment your VNet into subnets and deploy Azure IaaS virtual machines (VMs) and PaaS role instances, in the same way you can deploy physical and virtual machines to your on-premises datacenter. In essence, you can expand your network to Azure, bringing your own IP address blocks. Read the [virtual network overview](#) if you are not familiar with VNets.

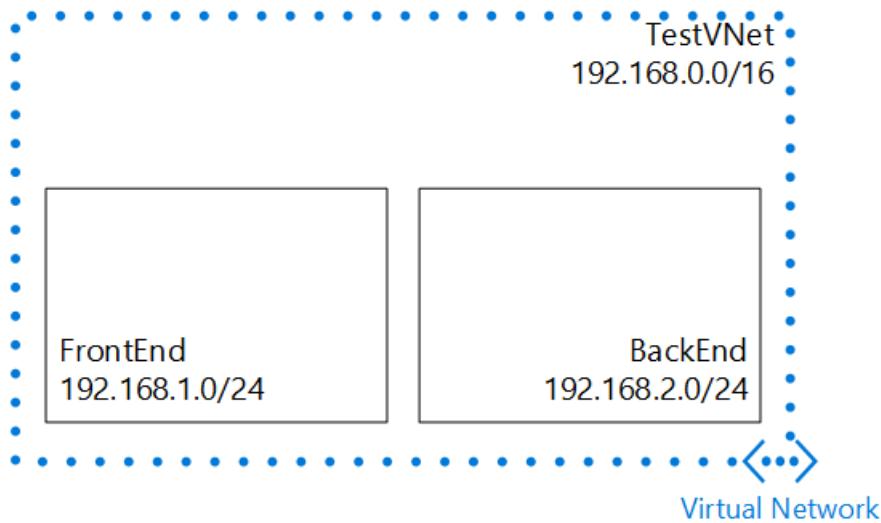
IMPORTANT

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 document covers creating a VNet by using the classic deployment model. You can also [create a virtual network in the Resource Manager deployment model by using the Azure portal](#).

Scenario

To illustrate how to create a VNet and subnets, this document uses the following scenario:



In this scenario you create a VNet named **TestVNet**, with a reserved CIDR block of **192.168.0.0/16**. The VNet contains the following subnets:

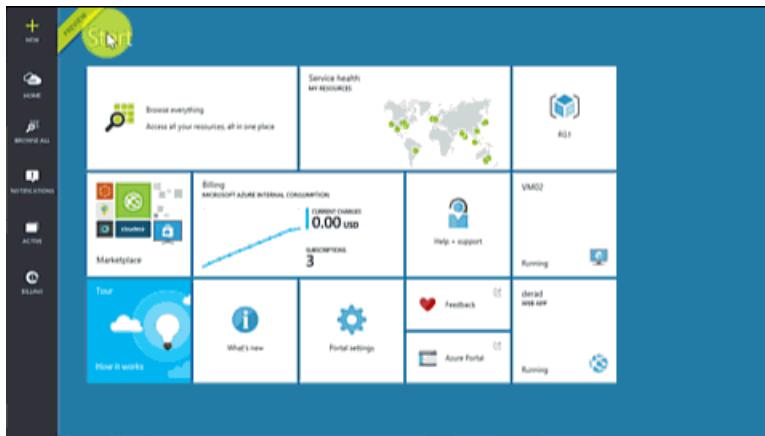
- **FrontEnd**, using **192.168.1.0/24** as its CIDR block.
- **BackEnd**, using **192.168.2.0/24** as its CIDR block.

How to create a classic VNet in the Azure portal

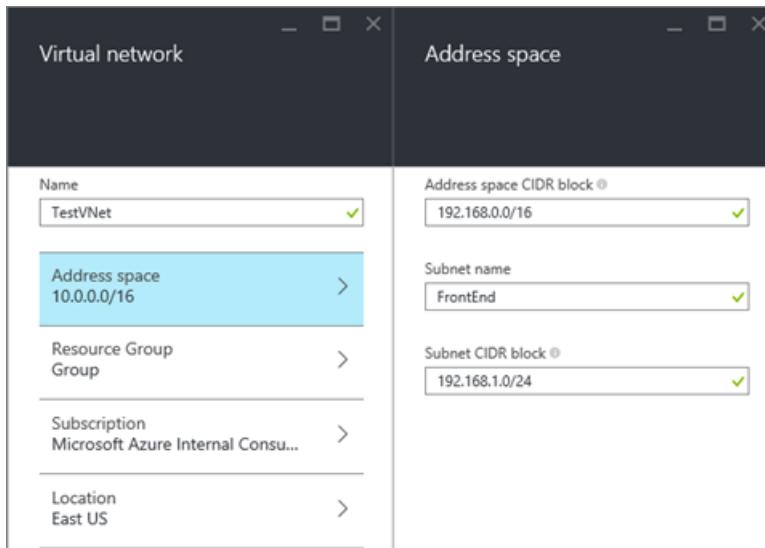
To create a classic VNet based on the preceding scenario, follow these steps.

1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.

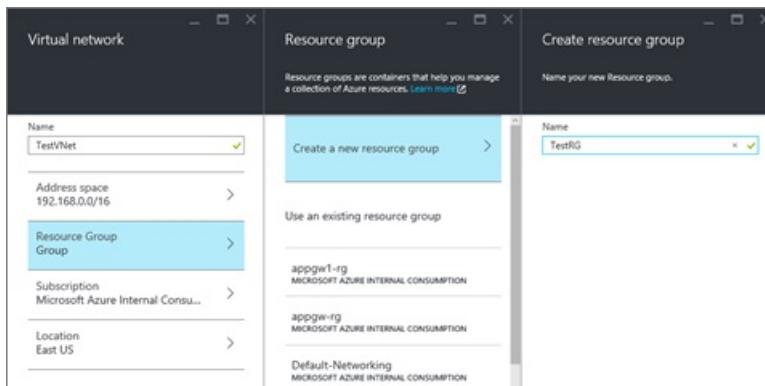
2. Click **Create a resource** > **Networking** > **Virtual network**. Notice that the **Select a deployment model** list already shows **Classic**. 3. Click **Create** as shown in the following figure.



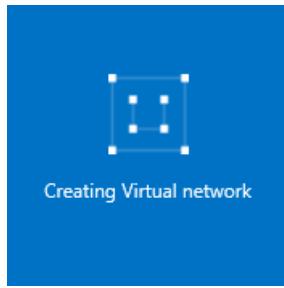
3. On the **Virtual network** pane, type the **Name** of the VNet, and then click **Address space**. Configure your address space settings for the VNet and its first subnet, then click **OK**. The following figure shows the CIDR block settings for our scenario.



4. Click **Resource Group** and select a resource group to add the VNet to, or click **Create new resource group** to add the VNet to a new resource group. The following figure shows the resource group settings for a new resource group called **TestRG**. For more information about resource groups, visit [Azure Resource Manager Overview](#).



5. If necessary, change the **Subscription** and **Location** settings for your VNet.
 6. If you do not want to see the VNet as a tile in the **Startboard**, disable **Pin to Startboard**.
 7. Click **Create** and notice the tile named **Creating Virtual network** as shown in the following figure.



8. Wait for the VNet to be created, and when you see the tile, click it to add more subnets.



9. You should see the **Configuration** for your VNet as shown.

A screenshot of the Azure portal showing the configuration of a virtual network named "TestVNet".

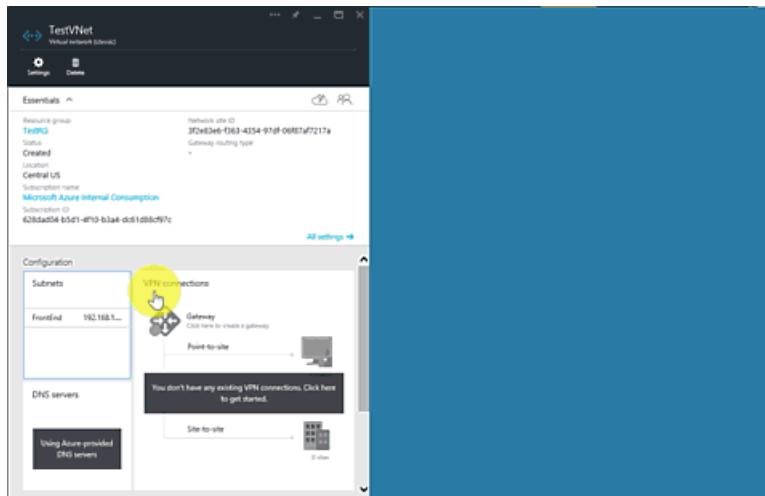
The left sidebar shows the "Configuration" section with the following details:

- Subnets:** A table with one entry: "FrontEnd" with IP range "192.168.1....".
- DNS servers:** A button labeled "Using Azure-provided DNS servers".

The main area is titled "VPN connections" and contains the following sections:

- Gateway:** A "Point-to-site" connection is listed, represented by a computer monitor icon.
- A message box states: "You don't have any existing VPN connections. Click here to get started."
- Site-to-site:** A "Site-to-site" connection is listed, represented by a server rack icon, with "0 sites" indicated.

10. Click **Subnets > Add**, then type a **Name** and specify an **Address range (CIDR block)** for your subnet, and then click **OK**. The following figure shows the settings for our current scenario.



Create a virtual network (classic) using a network configuration file with PowerShell

4/19/2018 • 2 min to read • [Edit Online](#)

An Azure virtual network (VNet) is a representation of your own network in the cloud. You can control your Azure network settings and define DHCP address blocks, DNS settings, security policies, and routing. You can also further segment your VNet into subnets and deploy Azure IaaS virtual machines (VMs) and PaaS role instances, in the same way you can deploy physical and virtual machines to your on-premises datacenter. In essence, you can expand your network to Azure, bringing your own IP address blocks. Read the [virtual network overview](#) if you are not familiar with VNets.

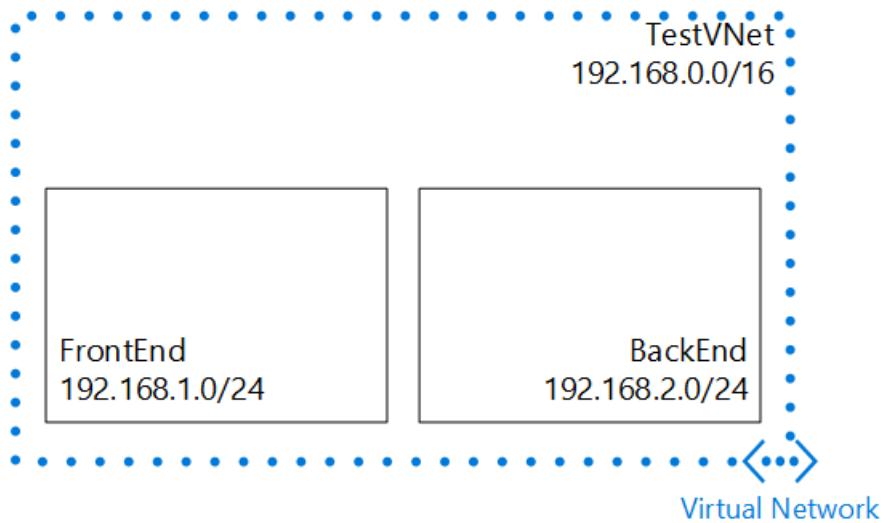
IMPORTANT

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 document covers creating a VNet by using the classic deployment model. You can also [create a virtual network in the Resource Manager deployment model](#).

Scenario

To illustrate how to create a VNet and subnets, this document uses the following scenario:



In this scenario you create a VNet named **TestVNet**, with a reserved CIDR block of **192.168.0.0/16**. The VNet contains the following subnets:

- **FrontEnd**, using **192.168.1.0/24** as its CIDR block.
- **BackEnd**, using **192.168.2.0/24** as its CIDR block.

How to create a virtual network using a network config file from PowerShell

Azure uses an xml file to define all virtual networks available to a subscription. You can download this file, edit it to

modify or delete existing virtual networks, and create new virtual networks. In this tutorial, you learn how to download this file, referred to as network configuration (or netcfg) file, and edit it to create a new virtual network. To learn more about the network configuration file, see the [Azure virtual network configuration schema](#).

To create a virtual network with a netcfg file using PowerShell, complete the following steps:

1. If you have never used Azure PowerShell, complete the steps in the [How to Install and Configure Azure PowerShell](#) article, then sign in to Azure and select your subscription.
2. From the Azure PowerShell console, use the **Get-AzureVnetConfig** cmdlet to download the network configuration file to a directory on your computer by running the following command:

```
Get-AzureVNetConfig -ExportToFile c:\azure\NetworkConfig.xml
```

Expected output:

```
XMLConfiguration
-----
<?xml version="1.0" encoding="utf-8"?>...
```

3. Open the file you saved in step 2 using any XML or text editor application, and look for the element. If you have any networks already created, each network is displayed as its own element.
4. To create the virtual network described in this scenario, add the following XML just under the element:

```
<?xml version="1.0" encoding="utf-8"?>
<NetworkConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.microsoft.com/ServiceHosting/2011/07/NetworkConfiguration">
    <VirtualNetworkConfiguration>
        <VirtualNetworkSites>
            <VirtualNetworkSite name="TestVNet" Location="East US">
                <AddressSpace>
                    <AddressPrefix>192.168.0.0/16</AddressPrefix>
                </AddressSpace>
                <Subnets>
                    <Subnet name="FrontEnd">
                        <AddressPrefix>192.168.1.0/24</AddressPrefix>
                    </Subnet>
                    <Subnet name="BackEnd">
                        <AddressPrefix>192.168.2.0/24</AddressPrefix>
                    </Subnet>
                </Subnets>
            </VirtualNetworkSite>
        </VirtualNetworkSites>
    </VirtualNetworkConfiguration>
</NetworkConfiguration>
```

5. Save the network configuration file.
6. From the Azure PowerShell console, use the **Set-AzureVnetConfig** cmdlet to upload the network configuration file by running the following command:

```
Set-AzureVNetConfig -ConfigurationPath c:\azure\NetworkConfig.xml
```

Returned output:

OperationDescription	OperationId	OperationStatus
Set-AzureVNetConfig <Id>		Succeeded

If **OperationStatus** is not *Succeeded* in the returned output, check the xml file for errors and complete step 6 again.

- From the Azure PowerShell console, use the **Get-AzureVnetSite** cmdlet to verify that the new network was added by running the following command:

```
Get-AzureVNetSite -VNetName TestVNet
```

The returned (abbreviated) output includes the following text:

```
AddressSpacePrefixes : {192.168.0.0/16}
Location          : Central US
Name              : TestVNet
State             : Created
Subnets           : {FrontEnd, BackEnd}
OperationDescription : Get-AzureVNetSite
OperationStatus    : Succeeded
```

Create a virtual network (classic) by using the Azure CLI

4/19/2018 • 3 min to read • [Edit Online](#)

An Azure virtual network (VNet) is a representation of your own network in the cloud. You can control your Azure network settings and define DHCP address blocks, DNS settings, security policies, and routing. You can also further segment your VNet into subnets and deploy Azure IaaS virtual machines (VMs) and PaaS role instances, in the same way you can deploy physical and virtual machines to your on-premises datacenter. In essence, you can expand your network to Azure, bringing your own IP address blocks. Read the [virtual network overview](#) if you are not familiar with VNets.

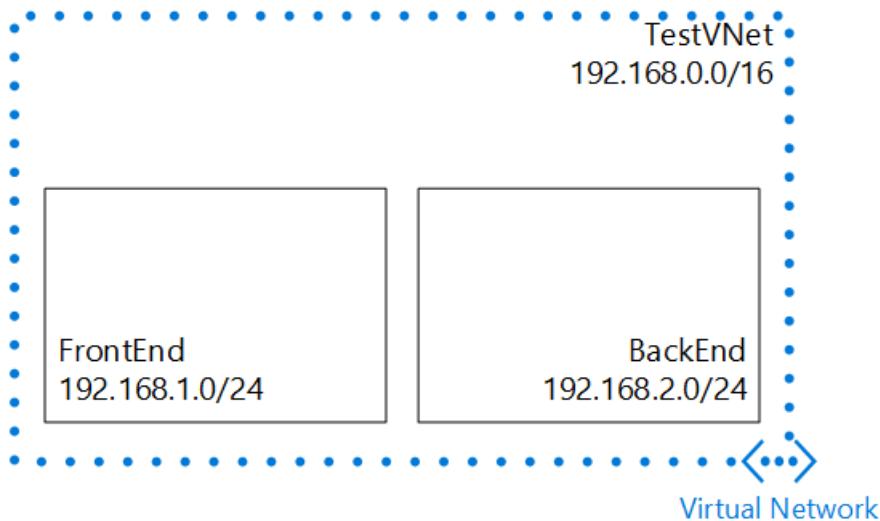
IMPORTANT

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 document covers creating a VNet by using the classic deployment model. You can also [create a virtual network in the Resource Manager deployment model by using the Azure CLI](#).

Scenario

To illustrate how to create a VNet and subnets, this document uses the following scenario:



In this scenario you create a VNet named **TestVNet**, with a reserved CIDR block of **192.168.0.0/16**. The VNet contains the following subnets:

- **FrontEnd**, using **192.168.1.0/24** as its CIDR block.
- **BackEnd**, using **192.168.2.0/24** as its CIDR block.

How to create a classic VNet using Azure CLI

You can use the Azure CLI to manage your Azure resources from the command prompt from any computer running Windows, Linux, or OSX.

1. If you have never used Azure CLI, see [Install and Configure the Azure CLI](#) and follow the instructions up to the point where you select your Azure account and subscription.

2. To create a VNet and a subnet, run the **azure network vnet create** command:

```
azure network vnet create --vnet TestVNet -e 192.168.0.0 -i 16 -n FrontEnd -p 192.168.1.0 -r 24 -l "Central US"
```

Expected output:

```
info: Executing command network vnet create
+ Looking up network configuration
+ Looking up locations
+ Setting network configuration
info: network vnet create command OK
```

- **--vnet**. Name of the VNet to be created. For the scenario, *TestVNet*
- **-e (or --address-space)**. VNet address space. For the scenario, *192.168.0.0*
- **-i (or -cidr)**. Network mask in CIDR format. For the scenario, *16*.
- **-n (or --subnet-name)**. Name of the first subnet. For the scenario, *FrontEnd*.
- **-p (or --subnet-start-ip)**. Starting IP address for subnet, or subnet address space. For the scenario, *192.168.1.0*.
- **-r (or --subnet-cidr)**. Network mask in CIDR format for subnet. For the scenario, *24*.
- **-l (or --location)**. Azure region where the VNet is created. For the scenario, *Central US*.

3. To create a subnet, run the **azure network vnet subnet create** command:

```
azure network vnet subnet create -t TestVNet -n BackEnd -a 192.168.2.0/24
```

The expected output for the previous command:

```
info: Executing command network vnet subnet create
+ Looking up network configuration
+ Creating subnet "BackEnd"
+ Setting network configuration
+ Looking up the subnet "BackEnd"
+ Looking up network configuration
data: Name : BackEnd
data: Address prefix : 192.168.2.0/24
info: network vnet subnet create command OK
```

- **-t (or --vnet-name)**. Name of the VNet where the subnet will be created. For the scenario, *TestVNet*.
- **-n (or --name)**. Name of the new subnet. For the scenario, *BackEnd*.
- **-a (or --address-prefix)**. Subnet CIDR block. For the scenario, *192.168.2.0/24*.

4. To view the properties of the new vnet, run the **azure network vnet show** command:

```
azure network vnet show
```

Expected output for the previous command:

```
info: Executing command network vnet show
Virtual network name: TestVNet
+ Looking up the virtual network sites
data: Name : TestVNet
data: Location : Central US
data: State : Created
data: Address space : 192.168.0.0/16
data: Subnets:
data: Name : FrontEnd
data: Address prefix : 192.168.1.0/24
data:
data: Name : BackEnd
data: Address prefix : 192.168.2.0/24
data:
info: network vnet show command OK
```

Specifying DNS settings in a virtual network configuration file

4/19/2018 • 1 min to read • [Edit Online](#)

A network configuration file has two elements that you can use to specify Domain Name System (DNS) settings: **DnsServers** and **DnsServerRef**. You can add a list of DNS servers by specifying their IP addresses and reference names to the **DnsServers** element. You can then use a **DnsServerRef** element to specify which DNS server entries from the DnsServers element are used for different network sites within your virtual network.

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 classic deployment model.

The network configuration file may contain the following elements. The title of each element is linked to a page that provides additional information about the element value settings.

IMPORTANT

For information about how to configure the network configuration file, see [Configure a Virtual Network Using a Network Configuration File](#). For information about each element contained in the network configuration file, see [Azure Virtual Network Configuration Schema](#).

Dns Element

```
<Dns>
  <DnsServers>
    <DnsServer name="ID1" IPAddress="IPAddress1" />
    <DnsServer name="ID2" IPAddress="IPAddress2" />
    <DnsServer name="ID3" IPAddress="IPAddress3" />
  </DnsServers>
</Dns>
```

WARNING

The **name** attribute in the **DnsServer** element is used only as a reference for the **DnsServerRef** element. It does not represent the host name for the DNS server. Each **DnsServer** attribute value must be unique across the entire Microsoft Azure subscription.

Virtual Network Sites Element

```
<DnsServersRef>
  <DnsServerRef name="ID1" />
  <DnsServerRef name="ID2" />
  <DnsServerRef name="ID3" />
</DnsServersRef>
```

NOTE

In order to specify this setting for the Virtual Network Sites element, it must be previously defined in the DNS element. The `DnsServerRef name` in the Virtual Network Sites element must refer to a name value specified in the DNS element for `DnsServer name`.

Next steps

- Understand the [Azure Virtual Network Configuration Schema](#).
- Understand the [Azure Service Configuration Schema](#).
- [Configure a virtual network using Network configuration files](#).

Specifying DNS Settings in a Service Configuration File

4/19/2018 • 1 min to read • [Edit Online](#)

DNS elements

A service configuration file may contain a `DnsServers` element with a list of IPv4 addresses for the Domain Name System (DNS) servers that the service will use. Settings in the service configuration file take precedence over settings in the network configuration file. For more information, see [Azure Service Configuration Schema \(.cscfg File\)](#).

NetworkConfiguration element

```
<DnsServers>
  <DnsServer name="ID1" IPAddress="IPAddress1" />
  <DnsServer name="ID2" IPAddress="IPAddress2" />
  <DnsServer name="ID3" IPAddress="IPAddress3" />
</DnsServers>
```

WARNING

The `name` attribute in the `DnsServer` element is used only as a reference name. It does not represent the host name for the DNS server. Each `DnsServer` attribute value must be unique across the entire Microsoft Azure subscription.

See Also

[Azure Service Configuration Schema \(.cscfg\)](#)

[Azure Virtual Network Configuration Schema](#)

[Configure a Virtual Network Using Network Configuration Files](#)

[About Virtual Network settings in the Management Portal](#)

Configure a virtual network (classic) using a network configuration file

4/19/2018 • 4 min to read • [Edit Online](#)

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the classic deployment model. Microsoft recommends that most new deployments use the Resource Manager deployment model.

You can create and configure a virtual network (classic) with a network configuration file using the Azure command-line interface (CLI) 1.0 or Azure PowerShell. You cannot create or modify a virtual network through the Azure Resource Manager deployment model using a network configuration file. You cannot use the Azure portal to create or modify a virtual network (classic) using a network configuration file, however you can use the Azure portal to create a virtual network (classic), without using a network configuration file.

Creating and configuring a virtual network (classic) with a network configuration file requires exporting, changing, and importing the file.

Export a network configuration file

You can use PowerShell or the Azure CLI to export a network configuration file. PowerShell exports an XML file, while the Azure CLI exports a json file.

PowerShell

1. [Install Azure PowerShell and sign in to Azure](#).
2. Change the directory (and ensure it exists) and filename in the following command as desired, then run the command to export the network configuration file:

```
Get-AzureVNetConfig -ExportToFile c:\azure\networkconfig.xml
```

Azure CLI 1.0

1. [Install the Azure CLI 1.0](#). Complete the remaining steps from an Azure CLI 1.0 command prompt.
2. Log in to Azure by entering the `azure login` command.
3. Ensure you're in asm mode by entering the `azure config mode asm` command.
4. Change the directory (and ensure it exists) and filename in the following command as desired, then run the command to export the network configuration file:

```
azure network export c:\azure\networkconfig.json
```

Create or modify a network configuration file

A network configuration file is an XML file (when using PowerShell) or a json file (when using the Azure CLI). You can edit the file in any text, or XML/json editor. The [Network configuration file schema settings](#) article includes details for all settings. For additional explanation of the settings, see [View virtual networks and settings](#). The changes you make to the file:

- Must comply with the schema, or importing the network configuration file will fail.
- Overwrite any existing network settings for your subscription, so use extreme caution when making modifications. For example, reference the example network configuration files that follow. Say the original file contained two **VirtualNetworkSite** instances, and you changed it, as shown in the examples. When you import the file, Azure deletes the virtual network for the **VirtualNetworkSite** instance you removed in the file. This simplified scenario assumes no resources were in the virtual network, as if there were, the virtual network could not be deleted, and the import would fail.

IMPORTANT

Azure considers a subnet that has something deployed to it as **in use**. When a subnet is in use, it cannot be modified. Before modifying subnet information in a network configuration file, move anything that you have deployed to the subnet to a different subnet that isn't being modified. See [Move a VM or Role Instance to a Different Subnet](#) for details.

Example XML for use with PowerShell

The following example network configuration file creates a virtual network named *myVirtualNetwork* with an address space of *10.0.0.0/16* in the *East US* Azure region. The virtual network contains one subnet named *mySubnet* with an address prefix of *10.0.0.0/24*.

```
<?xml version="1.0" encoding="utf-8"?>
<NetworkConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://schemas.microsoft.com/ServiceHosting/2011/07/NetworkConfiguration">
    <VirtualNetworkConfiguration>
        <Dns />
        <VirtualNetworkSites>
            <VirtualNetworkSite name="myVirtualNetwork" Location="East US">
                <AddressSpace>
                    <AddressPrefix>10.0.0.0/16</AddressPrefix>
                </AddressSpace>
                <Subnets>
                    <Subnet name="mySubnet">
                        <AddressPrefix>10.0.0.0/24</AddressPrefix>
                    </Subnet>
                </Subnets>
            </VirtualNetworkSite>
        </VirtualNetworkSites>
    </VirtualNetworkConfiguration>
</NetworkConfiguration>
```

If the network configuration file you exported contains no contents, you can copy the XML in the previous example, and paste it into a new file.

Example JSON for use with the Azure CLI 1.0

The following example network configuration file creates a virtual network named *myVirtualNetwork* with an address space of *10.0.0.0/16* in the *East US* Azure region. The virtual network contains one subnet named *mySubnet* with an address prefix of *10.0.0.0/24*.

```
{  
    "VirtualNetworkConfiguration" : {  
        "Dns" : "",  
        "VirtualNetworkSites" : [  
            {  
                "AddressSpace" : [ "10.0.0.0/16" ],  
                "Location" : "East US",  
                "Name" : "myVirtualNetwork",  
                "Subnets" : [  
                    {  
                        "AddressPrefix" : "10.0.0.0/24",  
                        "Name" : "mySubnet"  
                    }  
                ]  
            }  
        ]  
    }  
}
```

If the network configuration file you exported contains no contents, you can copy the json in the previous example, and paste it into a new file.

Import a network configuration file

You can use PowerShell or the Azure CLI to import a network configuration file. PowerShell imports an XML file, while the Azure CLI imports a json file. If the import fails, confirm that the file complies with the [network configuration schema](#).

PowerShell

1. [Install Azure PowerShell and sign in to Azure](#).
2. Change the directory and filename in the following command as necessary, then run the command to import the network configuration file:

```
Set-AzureVNetConfig -ConfigurationPath c:\azure\networkconfig.xml
```

Azure CLI 1.0

1. [Install the Azure CLI 1.0](#). Complete the remaining steps from an Azure CLI 1.0 command prompt.
2. Log in to Azure by entering the `azure login` command.
3. Ensure you're in asm mode by entering the `azure config mode asm` command.
4. Change the directory and filename in the following command as necessary, then run the command to import the network configuration file:

```
azure network import c:\azure\networkconfig.json
```

Migrate a virtual network (classic) from an affinity group to a region

4/19/2018 • 2 min to read • [Edit Online](#)

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the classic deployment model. Microsoft recommends that most new deployments use the Resource Manager deployment model.

Affinity groups ensure that resources created within the same affinity group are physically hosted by servers that are close together, enabling these resources to communicate quicker. In the past, affinity groups were a requirement for creating virtual networks (classic). At that time, the network manager service that managed virtual networks (classic) could only work within a set of physical servers or scale unit. Architectural improvements have increased the scope of network management to a region.

As a result of these architectural improvements, affinity groups are no longer recommended, or required for virtual networks (classic). The use of affinity groups for virtual networks (classic) is replaced by regions. Virtual networks (classic) that are associated with regions are called regional virtual networks.

We recommend that you don't use affinity groups in general. Aside from the virtual network requirement, affinity groups were also important to use to ensure resources, such as compute (classic) and storage (classic), were placed near each other. However, with the current Azure network architecture, these placement requirements are no longer necessary.

IMPORTANT

Although it is still technically possible to create a virtual network that is associated with an affinity group, there is no compelling reason to do so. Many virtual network features, such as network security groups, are only available when using a regional virtual network, and are not available for virtual networks that are associated with affinity groups.

Edit the network configuration file

1. Export the network configuration file. To learn how to export a network configuration file using PowerShell or the Azure command-line interface (CLI) 1.0, see [Configure a virtual network using a network configuration file](#).
2. Edit the network configuration file, replacing **AffinityGroup** with **Location**. You specify an Azure [region](#) for **Location**.

NOTE

The **Location** is the region that you specified for the affinity group that is associated with your virtual network (classic). For example, if your virtual network (classic) is associated with an affinity group that is located in West US, when you migrate, your **Location** must point to West US.

Edit the following lines in your network configuration file, replacing the values with your own:

Old value: <VirtualNetworkSitetename="VNetUSWest" AffinityGroup="VNetDemoAG">

New value: <VirtualNetworkSitetename="VNetUSWest" Location="West US">

3. Save your changes and [import](#) the network configuration to Azure.

NOTE

This migration does NOT cause any downtime to your services.

What to do if you have a VM (classic) in an affinity group

VMs (classic) that are currently in an affinity group do not need to be removed from the affinity group. Once a VM is deployed, it is deployed to a single scale unit. Affinity groups can restrict the set of available VM sizes for a new VM deployment, but any existing VM that is deployed is already restricted to the set of VM sizes available in the scale unit in which the VM is deployed. Because the VM is already deployed to a scale unit, removing a VM from an affinity group has no effect on the VM.

Create a network security group (classic) using PowerShell

4/19/2018 • 2 min to read • [Edit Online](#)

You can use an NSG to control traffic to one or more virtual machines (VMs), role instances, network adapters (NICs), or subnets in your virtual network. An NSG contains access control rules that allow or deny traffic based on traffic direction, protocol, source address and port, and destination address and port. The rules of an NSG can be changed at any time, and changes are applied to all associated instances.

For more information about NSGs, visit [what is an NSG](#).

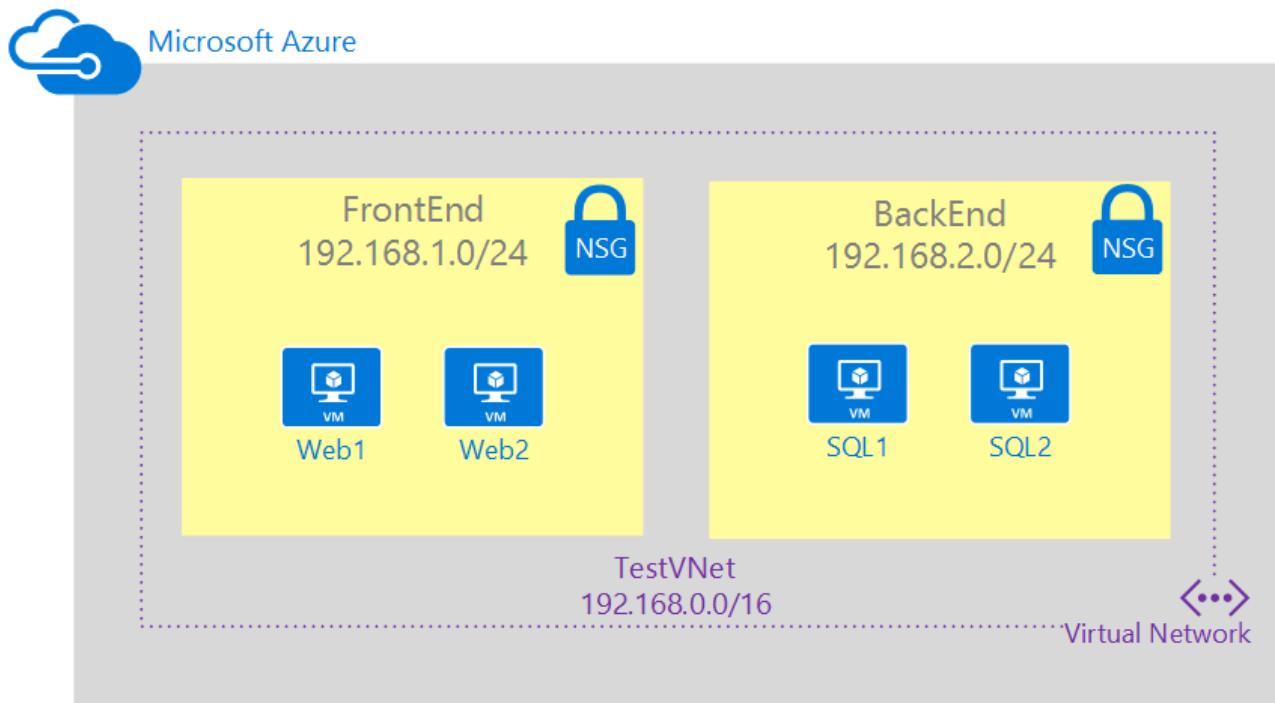
IMPORTANT

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 classic deployment model. You can also [create NSGs in the Resource Manager deployment model](#).

Scenario

To better illustrate how to create NSGs, this document uses the following scenario:



In this scenario, you create an NSG for each subnet in the **TestVNet** virtual network, as follows:

- **NSG-FrontEnd.** The front-end NSG is applied to the *FrontEnd* subnet, and contains two rules:
 - **rdp-rule.** Allows RDP traffic to the *FrontEnd* subnet.
 - **web-rule.** Allows HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd.** The back-end NSG is applied to the *BackEnd* subnet, and contains two rules:

- **sql-rule.** Allows SQL traffic only from the *FrontEnd* subnet.
- **web-rule.** Denies all internet bound traffic from the *BackEnd* subnet.

The combination of these rules create a DMZ-like scenario, where the back-end subnet can only receive incoming traffic for SQL from the front-end subnet, and has no access to the Internet, while the front-end subnet can communicate with the Internet, and receive incoming HTTP requests only.

The sample PowerShell commands below expect a simple environment already created based on the scenario above. If you want to run the commands as they are displayed in this document, first build the test environment by [creating a VNet](#).

Create an NSG for the front-end subnet

1. If you have never used Azure PowerShell, see [How to Install and Configure Azure PowerShell](#).
2. Create a network security group named *NSG-FrontEnd*:

```
New-AzureNetworkSecurityGroup -Name "NSG-FrontEnd" -Location uswest ` 
-Label "Front end subnet NSG"
```

3. Create a security rule allowing access from the internet to port 3389:

```
Get-AzureNetworkSecurityGroup -Name "NSG-FrontEnd" ` 
| Set-AzureNetworkSecurityRule -Name rdp-rule ` 
-Action Allow -Protocol TCP -Type Inbound -Priority 100 ` 
-SourceAddressPrefix Internet -SourcePortRange '*' ` 
-DestinationAddressPrefix '*' -DestinationPortRange '3389'
```

4. Create a security rule allowing access from the internet to port 80:

```
Get-AzureNetworkSecurityGroup -Name "NSG-FrontEnd" ` 
| Set-AzureNetworkSecurityRule -Name web-rule ` 
-Action Allow -Protocol TCP -Type Inbound -Priority 200 ` 
-SourceAddressPrefix Internet -SourcePortRange '*' ` 
-DestinationAddressPrefix '*' -DestinationPortRange '80'
```

Create an NSG for the back-end subnet

1. Create a network security group named *NSG-BackEnd*:

```
New-AzureNetworkSecurityGroup -Name "NSG-BackEnd" -Location uswest ` 
-Label "Back end subnet NSG"
```

2. Create a security rule allowing access from the front-end subnet to port 1433 (default port used by SQL Server):

```
Get-AzureNetworkSecurityGroup -Name "NSG-FrontEnd" ` 
| Set-AzureNetworkSecurityRule -Name rdp-rule ` 
-Action Allow -Protocol TCP -Type Inbound -Priority 100 ` 
-SourceAddressPrefix 192.168.1.0/24 -SourcePortRange '*' ` 
-DestinationAddressPrefix '*' -DestinationPortRange '1433'
```

3. Create a security rule blocking access from the subnet to the internet:

```
Get-AzureNetworkSecurityGroup -Name "NSG-BackEnd" `| Set-AzureNetworkSecurityRule -Name block-internet ` -Action Deny -Protocol '*' -Type Outbound -Priority 200 ` -SourceAddressPrefix '*' -SourcePortRange '*' ` -DestinationAddressPrefix Internet -DestinationPortRange '*'`
```

Create a network security group (classic) using the Azure CLI 1.0

4/19/2018 • 2 min to read • [Edit Online](#)

You can use an NSG to control traffic to one or more virtual machines (VMs), role instances, network adapters (NICs), or subnets in your virtual network. An NSG contains access control rules that allow or deny traffic based on traffic direction, protocol, source address and port, and destination address and port. The rules of an NSG can be changed at any time, and changes are applied to all associated instances.

For more information about NSGs, visit [what is an NSG](#).

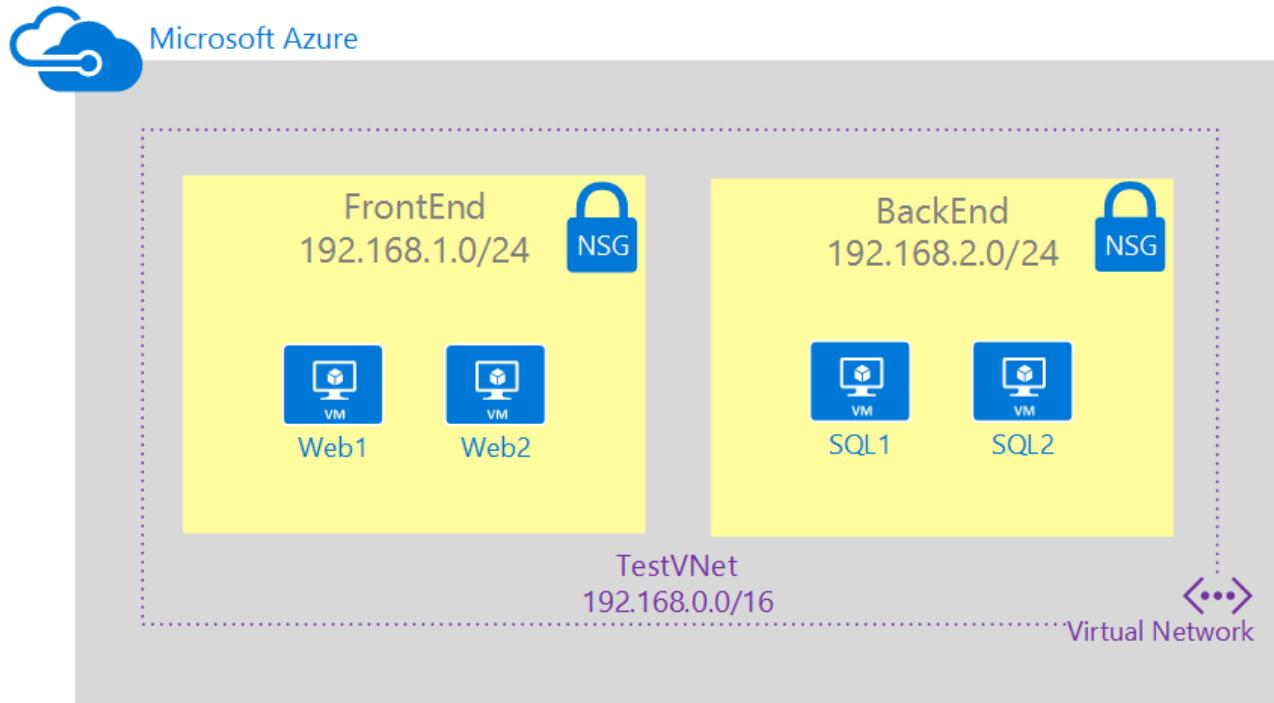
IMPORTANT

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 classic deployment model. You can also [create NSGs in the Resource Manager deployment model](#).

Scenario

To better illustrate how to create NSGs, this document uses the following scenario:



In this scenario, you create an NSG for each subnet in the **TestVNet** virtual network, as follows:

- **NSG-FrontEnd**. The front-end NSG is applied to the *FrontEnd* subnet, and contains two rules:
 - **rdp-rule**. Allows RDP traffic to the *FrontEnd* subnet.
 - **web-rule**. Allows HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd**. The back-end NSG is applied to the *BackEnd* subnet, and contains two rules:

- **sql-rule**. Allows SQL traffic only from the *FrontEnd* subnet.
- **web-rule**. Denies all internet bound traffic from the *BackEnd* subnet.

The combination of these rules create a DMZ-like scenario, where the back-end subnet can only receive incoming traffic for SQL from the front-end subnet, and has no access to the Internet, while the front-end subnet can communicate with the Internet, and receive incoming HTTP requests only.

The following sample Azure CLI commands expect a simple environment already created based on the scenario. If you want to run the commands as they are displayed in this document, first build the test environment by [creating a VNet](#).

Create an NSG for the front-end subnet

1. If you have never used Azure CLI, see [Install and Configure the Azure CLI](#).
2. Switch to classic mode:

```
azure config mode asm
```

3. Create an NSG::

```
azure network nsg create -l uswest -n NSG-FrontEnd
```

4. Create a security rule that allows access to port 3389 (RDP) from the internet:

```
azure network nsg rule create -a NSG-FrontEnd -n rdp-rule -c Allow -p Tcp -r Inbound -y 100 -f Internet
-o * -e * -u 3389
```

5. Create a rule that allows access to port 80 (HTTP) from the internet:

```
azure network nsg rule create -a NSG-FrontEnd -n web-rule -c Allow -p Tcp -r Inbound -y 200 -f Internet
-o * -e * -u 80
```

6. Associate the NSG to the front-end subnet:

```
azure network nsg subnet add -a NSG-FrontEnd --vnet-name TestVNet --subnet-name FrontEnd
```

Create the NSG for the back-end subnet

1. Create the NSG:

```
azure network nsg create -l uswest -n NSG-BackEnd
```

2. Create a rule that allows access to port 1433 (SQL) from the front-end subnet:

```
azure network nsg rule create -a NSG-BackEnd -n sql-rule -c Allow -p Tcp -r Inbound -y 100 -f
192.168.1.0/24 -o * -e * -u 1433
```

3. Create a rule that denies access to the internet:

```
azure network nsg rule create -a NSG-BackEnd -n web-rule -c Deny -p Tcp -r Outbound -y 200 -f * -o * -e Internet -u 80
```

4. Associate the NSG to the back-end subnet:

```
azure network nsg subnet add -a NSG-BackEnd --vnet-name TestVNet --subnet-name BackEnd
```

Control routing and use virtual appliances (classic) using PowerShell

4/19/2018 • 3 min to read • [Edit Online](#)

Although the use of system routes facilitates traffic automatically for your deployment, there are cases in which you want to control the routing of packets through a virtual appliance. You can do so by creating user defined routes that specify the next hop for packets flowing to a specific subnet to go to your virtual appliance instead, and enabling IP forwarding for the VM running as the virtual appliance.

Some of the cases where virtual appliances can be used include:

- Monitoring traffic with an intrusion detection system (IDS)
- Controlling traffic with a firewall

For more information about UDR and IP forwarding, visit [User Defined Routes and IP Forwarding](#).

IMPORTANT

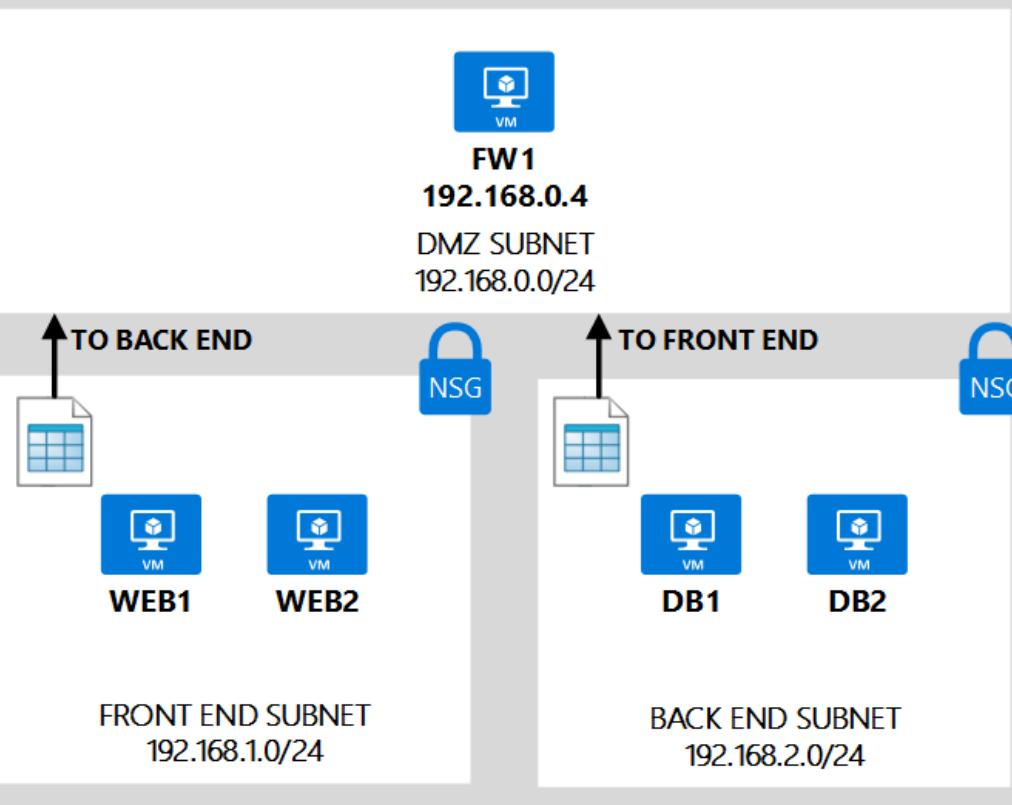
Before you work with Azure resources, it's important to understand that Azure currently has two deployment models: Azure Resource Manager and classic. Make sure you understand [deployment models and tools](#) before you work with any Azure resource. You can view the documentation for different tools by selecting an option at the top of this article. This article covers the classic deployment model.

Scenario

To better illustrate how to create UDRs, this document uses the following scenario:



AZURE REGION



<...>
VIRTUAL NETWORK

In this scenario, you create one UDR for the *Front-end subnet* and another UDR for the *Back-end subnet*, as follows:

- **UDR-FrontEnd.** The front-end UDR is applied to the *FrontEnd* subnet, and contain one route:
 - **RouteToBackend.** This route sends all traffic to the back-end subnet to the **FW1** virtual machine.
- **UDR-BackEnd.** The back-end UDR is applied to the *BackEnd* subnet, and contain one route:
 - **RouteToFrontend.** This route sends all traffic to the front-end subnet to the **FW1** virtual machine.

The combination of these routes ensures that all traffic destined from one subnet to another is routed to the **FW1** virtual machine, which is being used as a virtual appliance. You also need to turn on IP forwarding for the **FW1** VM, to ensure it can receive traffic destined to other VMs.

The sample Azure PowerShell commands below expect a simple environment already created based on the scenario above. If you want to run the commands as they are displayed in this document, create the environment shown in [create a VNet \(classic\) using PowerShell](#).

Prerequisite: Install the Azure PowerShell module

To perform the steps in this article, you need to [install and configure the Azure PowerShell module](#). Be sure to complete all of the instructions. After the installation is finished, sign in to Azure and select your subscription.

NOTE

You need an Azure account to complete these steps. If you don't have an Azure account, you can sign up for a [free trial](#).

Create the UDR for the front end subnet

To create the route table and route needed for the front end subnet based on the scenario above, follow the steps below.

1. Run the following command to create a route table for the front-end subnet:

```
New-AzureRouteTable -Name UDR-FrontEnd -Location uswest  
-Label "Route table for front end subnet"
```

2. Run the following command to create a route in the route table to send all traffic destined to the back-end subnet (192.168.2.0/24) to the **FW1** VM (192.168.0.4):

```
Get-AzureRouteTable UDR-FrontEnd  
| Set-AzureRoute -RouteName RouteToBackEnd -AddressPrefix 192.168.2.0/24  
-NextHopType VirtualAppliance  
-NextHopIpAddress 192.168.0.4
```

3. Run the following command to associate the route table with the **FrontEnd** subnet:

```
Set-AzureSubnetRouteTable -VirtualNetworkName TestVNet  
-SubnetName FrontEnd  
-RouteTableName UDR-FrontEnd
```

Create the UDR for the back-end subnet

To create the route table and route needed for the back end subnet based on the scenario, complete the following steps:

1. Run the following command to create a route table for the back-end subnet:

```
New-AzureRouteTable -Name UDR-BackEnd  
-Location uswest  
-Label "Route table for back end subnet"
```

2. Run the following command to create a route in the route table to send all traffic destined to the front-end subnet (192.168.1.0/24) to the **FW1** VM (192.168.0.4):

```
Get-AzureRouteTable UDR-BackEnd  
| Set-AzureRoute  
-RouteName RouteToFrontEnd  
-AddressPrefix 192.168.1.0/24  
-NextHopType VirtualAppliance  
-NextHopIpAddress 192.168.0.4
```

3. Run the following command to associate the route table with the **BackEnd** subnet:

```
Set-AzureSubnetRouteTable -VirtualNetworkName TestVNet `  
-SubnetName BackEnd `  
-RouteTableName UDR-BackEnd
```

Enable IP forwarding on the FW1 VM

To enable IP forwarding in the FW1 VM, complete the following steps:

1. Run the following command to check the status of IP forwarding:

```
Get-AzureVM -Name FW1 -ServiceName TestRGFW `  
| Get-AzureIPForwarding
```

2. Run the following command to enable IP forwarding for the *FW1* VM:

```
Get-AzureVM -Name FW1 -ServiceName TestRGFW `  
| Set-AzureIPForwarding -Enable
```

Control routing and use virtual appliances (classic) using the Azure CLI

4/19/2018 • 4 min to read • [Edit Online](#)

Although the use of system routes facilitates traffic automatically for your deployment, there are cases in which you want to control the routing of packets through a virtual appliance. You can do so by creating user defined routes that specify the next hop for packets flowing to a specific subnet to go to your virtual appliance instead, and enabling IP forwarding for the VM running as the virtual appliance.

Some of the cases where virtual appliances can be used include:

- Monitoring traffic with an intrusion detection system (IDS)
- Controlling traffic with a firewall

For more information about UDR and IP forwarding, visit [User Defined Routes and IP Forwarding](#).

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.

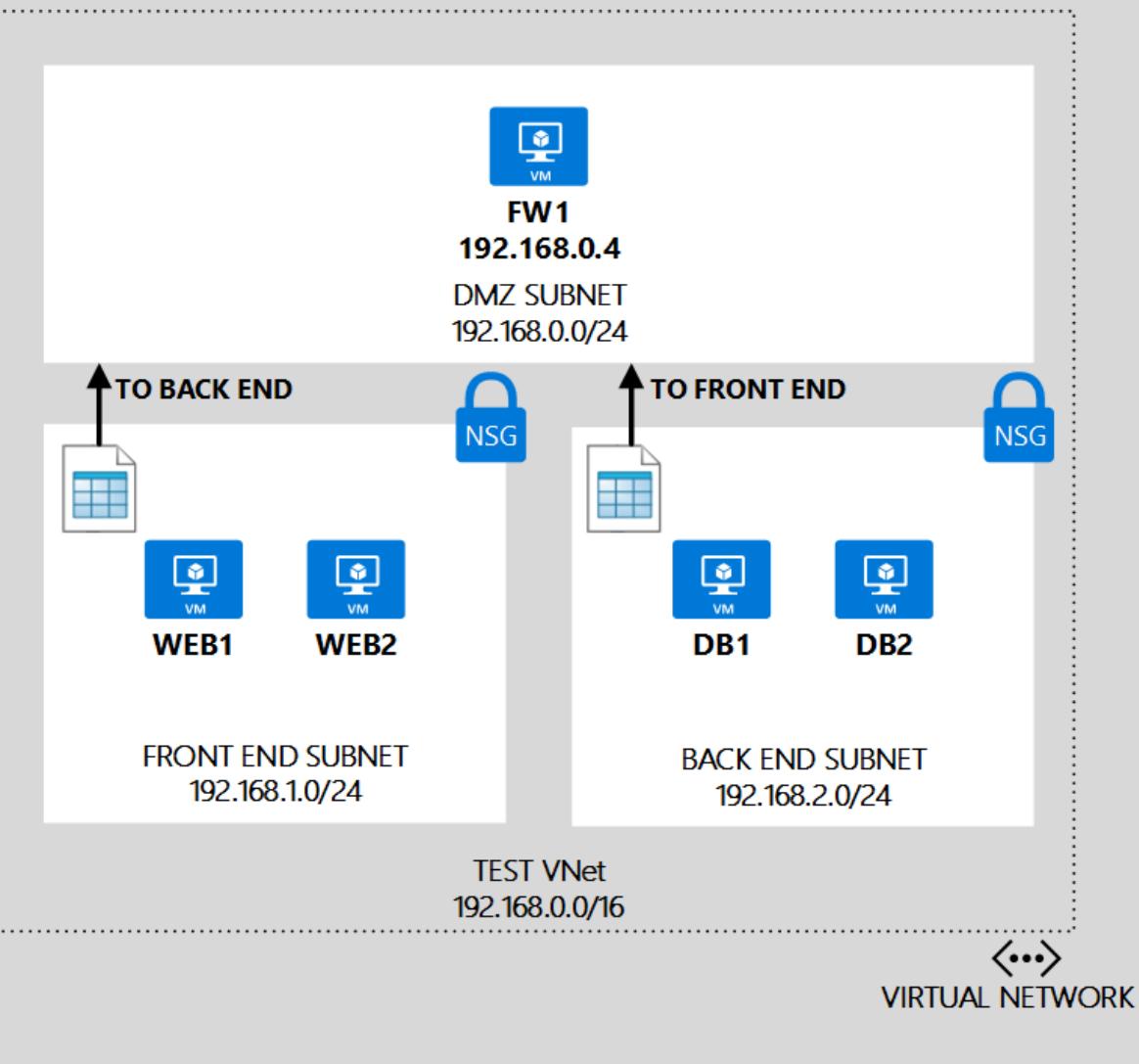
This article covers the classic deployment model. You can also [control routing and use virtual appliances in the Resource Manager deployment model](#).

Scenario

To better illustrate how to create UDRs, this document uses the following scenario:



AZURE REGION



In this scenario, you create one UDR for the *Front-end subnet* and another UDR for the *Back-end subnet*, as follows:

- **UDR-FrontEnd.** The front-end UDR is applied to the *FrontEnd* subnet, and contain one route:
 - **RouteToBackend.** This route sends all traffic to the back-end subnet to the **FW1** virtual machine.
- **UDR-BackEnd.** The back-end UDR is applied to the *BackEnd* subnet, and contain one route:
 - **RouteToFrontend.** This route sends all traffic to the front-end subnet to the **FW1** virtual machine.

The combination of these routes ensures that all traffic destined from one subnet to another is routed to the **FW1** virtual machine, which is being used as a virtual appliance. You also need to turn on IP forwarding for the **FW1** VM, to ensure it can receive traffic destined to other VMs.

The sample Azure CLI commands below expect a simple environment already created based on the scenario above. If you want to run the commands as they are displayed in this document, create the environment shown in [create a VNet \(classic\) using the Azure CLI](#).

Prerequisite: Install the Azure CLI

To perform the steps in this article, you need to [install the Azure Command-Line Interface for Mac, Linux, and Windows \(Azure CLI\)](#) and you need to [log on to Azure](#).

NOTE

If you don't have an Azure account, you need one. Go sign up for a [free trial here](#). In addition, to follow along completely you need to have either [jq](#) or some other JSON parsing tool or library installed.

Create the UDR for the front end subnet

To create the route table and route needed for the front end subnet based on the scenario above, follow the steps below.

1. Run the following command to switch to classic mode:

```
azure config mode asm
```

Output:

```
info: New mode is asm
```

2. Run the following command to create a route table for the front-end subnet:

```
azure network route-table create -n UDR-FrontEnd -l uswest
```

Output:

```
info: Executing command network route-table create
info: Creating route table "UDR-FrontEnd"
info: Getting route table "UDR-FrontEnd"
data: Name : UDR-FrontEnd
data: Location : West US
info: network route-table create command OK
```

Parameters:

- **-l (or --location)**. Azure region where the new NSG will be created. For our scenario, *westus*.
- **-n (or --name)**. Name for the new NSG. For our scenario, *NSG-FrontEnd*.

3. Run the following command to create a route in the route table to send all traffic destined to the back-end subnet (192.168.2.0/24) to the **FW1** VM (192.168.0.4):

```
azure network route-table route set -r UDR-FrontEnd -n RouteToBackEnd -a 192.168.2.0/24 -t
VirtualAppliance -p 192.168.0.4
```

Output:

```
info: Executing command network route-table route set
info: Getting route table "UDR-FrontEnd"
info: Setting route "RouteToBackEnd" in a route table "UDR-FrontEnd"
info: network route-table route set command OK
```

Parameters:

- **-r (or --route-table-name)**. Name of the route table where the route will be added. For our scenario, *UDR-FrontEnd*.

- **-a (or --address-prefix)**. Address prefix for the subnet where packets are destined to. For our scenario, `192.168.2.0/24`.
- **-t (or --next-hop-type)**. Type of object traffic will be sent to. Possible values are *VirtualAppliance*, *VirtualNetworkGateway*, *VNETLocal*, *Internet*, or *None*.
- **-p (or --next-hop-ip-address)**. IP address for next hop. For our scenario, `192.168.0.4`.

4. Run the following command to associate the route table created with the **FrontEnd** subnet:

```
azure network vnet subnet route-table add -t TestVNet -n FrontEnd -r UDR-FrontEnd
```

Output:

```
info: Executing command network vnet subnet route-table add
info: Looking up the subnet "FrontEnd"
info: Looking up network configuration
info: Looking up network gateway route tables in virtual network "TestVNet" subnet "FrontEnd"
info: Associating route table "UDR-FrontEnd" and subnet "FrontEnd"
info: Looking up network gateway route tables in virtual network "TestVNet" subnet "FrontEnd"
data: Route table name : UDR-FrontEnd
data: Location : West US
data: Routes:
info: network vnet subnet route-table add command OK
```

Parameters:

- **-t (or --vnet-name)**. Name of the VNet where the subnet is located. For our scenario, *TestVNet*.
- **-n (or --subnet-name)**. Name of the subnet the route table will be added to. For our scenario, *FrontEnd*.

Create the UDR for the back-end subnet

To create the route table and route needed for the back-end subnet based on the scenario, complete the following steps:

1. Run the following command to create a route table for the back-end subnet:

```
azure network route-table create -n UDR-BackEnd -l uswest
```

2. Run the following command to create a route in the route table to send all traffic destined to the front-end subnet (`192.168.1.0/24`) to the **FW1** VM (`192.168.0.4`):

```
azure network route-table route set -r UDR-BackEnd -n RouteToFrontEnd -a 192.168.1.0/24 -t VirtualAppliance -p 192.168.0.4
```

3. Run the following command to associate the route table with the **BackEnd** subnet:

```
azure network vnet subnet route-table add -t TestVNet -n BackEnd -r UDR-BackEnd
```

Create a VM (Classic) with multiple NICs using PowerShell

4/19/2018 • 5 min to read • [Edit Online](#)

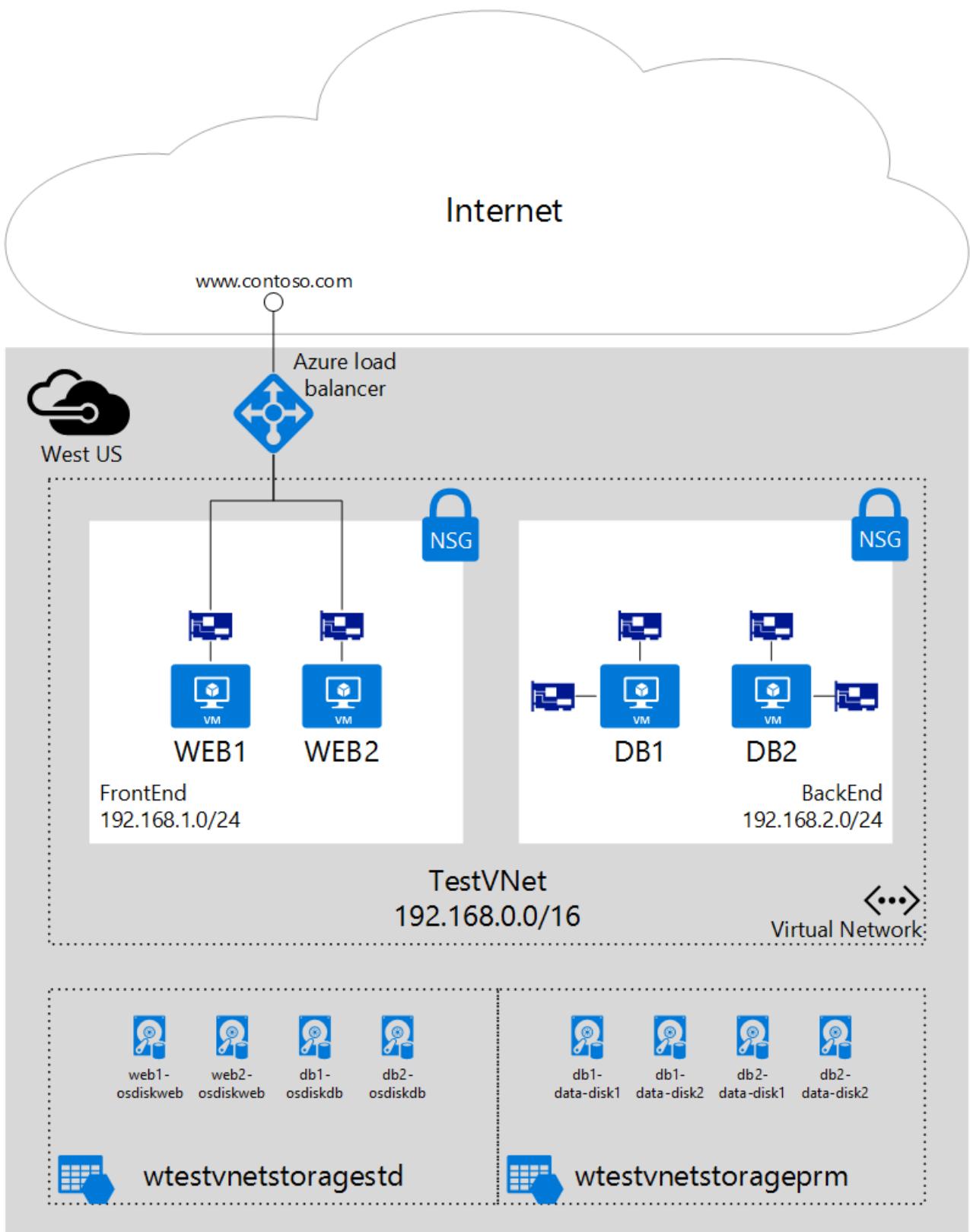
You can create virtual machines (VMs) in Azure and attach multiple network interfaces (NICs) to each of your VMs. Multiple NICs enable separation of traffic types across NICs. For example, one NIC might communicate with the Internet, while another communicates only with internal resources not connected to the Internet. The ability to separate network traffic across multiple NICs is required for many network virtual appliances, such as application delivery and WAN optimization solutions.

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the classic deployment model. Microsoft recommends that most new deployments use the Resource Manager model. Learn how to perform these steps using the [Resource Manager deployment model](#).

Scenario

This document walks through a deployment that uses multiple NICs in VMs in a specific scenario. In this scenario, you have a two-tiered IaaS workload hosted in Azure. Each tier is deployed in its own subnet in a virtual network (VNet). The front-end tier is composed of several web servers, grouped together in a load balancer set for high availability. The back-end tier is composed of several database servers. The database servers are deployed with two NICs each, one for database access, the other for management. The scenario also includes Network Security Groups (NSGs) to control what traffic is allowed to each subnet, and NIC in the deployment. The following picture shows the basic architecture of this scenario:



The following steps use a resource group named *IaaSStory* for the WEB servers and a resource group named *IaaSStory-BackEnd* for the DB servers.

Prerequisites

Before you can create the DB servers, you need to create the *IaaSStory* resource group with all the necessary resources for this scenario. To create these resources, complete the steps that follow. Create a virtual network by following the steps in the [Create a virtual network](#) article.

Prerequisite: Install the Azure PowerShell module

To perform the steps in this article, you need to [install and configure the Azure PowerShell module](#). Be sure to complete all of the instructions. After the installation is finished, sign in to Azure and select your subscription.

NOTE

You need an Azure account to complete these steps. If you don't have an Azure account, you can sign up for a [free trial](#).

Create the back-end VMs

The back-end VMs depend on the creation of the following resources:

- **Backend subnet.** The database servers will be part of a separate subnet, to segregate traffic. The script below expects this subnet to exist in a vnet named *WTestVnet*.
- **Storage account for data disks.** For better performance, the data disks on the database servers will use solid state drive (SSD) technology, which requires a premium storage account. Make sure the Azure location you deploy to support premium storage.
- **Availability set.** All database servers will be added to a single availability set, to ensure at least one of the VMs is up and running during maintenance.

Step 1 - Start your script

You can download the full PowerShell script used [here](#). Follow the steps below to change the script to work in your environment.

1. Change the values of the variables below based on your existing resource group deployed above in [Prerequisites](#).

```
$location          = "West US"  
$vnetName         = "WTestVNet"  
$backendSubnetName = "BackEnd"
```

2. Change the values of the variables below based on the values you want to use for your backend deployment.

```
$backendCSName      = "IaaSStory-Backend"  
$prmStorageAccountName = "iaasstoryprmstorage"  
$avSetName          = "ASDB"  
$vmSize              = "Standard_DS3"  
$diskSize             = 127  
$vmNamePrefix        = "DB"  
$dataDiskSuffix       = "datadisk"  
$ipAddressPrefix     = "192.168.2."  
$numberOfVMs         = 2
```

Step 2 - Create necessary resources for your VMs

You need to create a new cloud service and a storage account for the data disks for all VMs. You also need to specify an image, and a local administrator account for the VMs. To create these resources, complete the following steps:

1. Create a new cloud service.

```
New-AzureService -ServiceName $backendCSName -Location $location
```

2. Create a new premium storage account.

```
New-AzureStorageAccount -StorageAccountName $prmStorageAccountName `  
-Location $location -Type Premium_LRS
```

- Set the storage account created above as the current storage account for your subscription.

```
$subscription = Get-AzureSubscription | where {$_.IsCurrent -eq $true}  
Set-AzureSubscription -SubscriptionName $subscription.SubscriptionName `  
-CurrentStorageAccountName $prmStorageAccountName
```

- Select an image for the VM.

```
$image = Get-AzureVMImage `  
| where{$_._ImageFamily -eq "SQL Server 2014 RTM Web on Windows Server 2012 R2"} `  
| sort PublishedDate -Descending `  
| select -ExpandProperty ImageName -First 1
```

- Set the local administrator account credentials.

```
$cred = Get-Credential -Message "Enter username and password for local admin account"
```

Step 3 - Create VMs

You need to use a loop to create as many VMs as you want, and create the necessary NICs and VMs within the loop. To create the NICs and VMs, execute the following steps.

- Start a `for` loop to repeat the commands to create a VM and two NICs as many times as necessary, based on the value of the `$numberOfVMs` variable.

```
for ($suffixNumber = 1; $suffixNumber -le $numberOfVMs; $suffixNumber++){
```

- Create a `VMConfig` object specifying the image, size, and availability set for the VM.

```
$vmName = $vmNamePrefix + $suffixNumber  
$vmConfig = New-AzureVMConfig -Name $vmName `  
-ImageName $image `  
-InstanceSize $vmSize `  
-AvailabilitySetName $avSetName
```

- Provision the VM as a Windows VM.

```
Add-AzureProvisioningConfig -VM $vmConfig -Windows `  
-AdminUsername $cred.UserName `  
-Password $cred.GetNetworkCredential().Password
```

- Set the default NIC and assign it a static IP address.

```
Set-AzureSubnet      -SubnetNames $backendSubnetName -VM $vmConfig  
Set-AzureStaticVNetIP -IPAddress ($ipAddressPrefix+$suffixNumber+3) -VM $vmConfig
```

- Add a second NIC for each VM.

```
Add-AzureNetworkInterfaceConfig -Name ("RemoteAccessNIC"+$suffixNumber) ` 
-SubnetName $backendSubnetName ` 
-StaticVNetIPAddress ($ipAddressPrefix+(53+$suffixNumber)) ` 
-VM $vmConfig
```

6. Create two data disks for each VM.

```
$dataDisk1Name = $vmName + "-" + $dataDiskSuffix + "-1"
Add-AzureDataDisk -CreateNew -VM $vmConfig ` 
-DiskSizeInGB $diskSize ` 
-DiskLabel $dataDisk1Name ` 
-LUN 0

$dataDisk2Name = $vmName + "-" + $dataDiskSuffix + "-2"
Add-AzureDataDisk -CreateNew -VM $vmConfig ` 
-DiskSizeInGB $diskSize ` 
-DiskLabel $dataDisk2Name ` 
-LUN 1
```

7. Create each VM, and end the loop.

```
New-AzureVM -VM $vmConfig ` 
-ServiceName $backendCSName ` 
-Location $location ` 
-VNetName $vnetName
}
```

Step 4 - Run the script

Now that you downloaded and changed the script based on your needs, run the script to create the back-end database VMs with multiple NICs.

1. Save your script and run it from the **PowerShell** command prompt, or **PowerShell ISE**. You will see the initial output, as shown below.

OperationDescription	OperationId	OperationStatus
New-AzureService	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx	Succeeded
New-AzureStorageAccount	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx	Succeeded
WARNING: No deployment found in service: 'IaaSStory-Backend'.		

2. Fill out the information needed in the credentials prompt and click **OK**. The following output is returned.

New-AzureVM	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx	Succeeded
New-AzureVM	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx	Succeeded

Step 5 - Configure routing within the VM's operating system

Azure DHCP 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. To configure routing for secondary network interfaces, see [Routing within a virtual machine operating system with multiple network interfaces](#).

Create a VM (Classic) with multiple NICs using the Azure CLI 1.0

4/19/2018 • 5 min to read • [Edit Online](#)

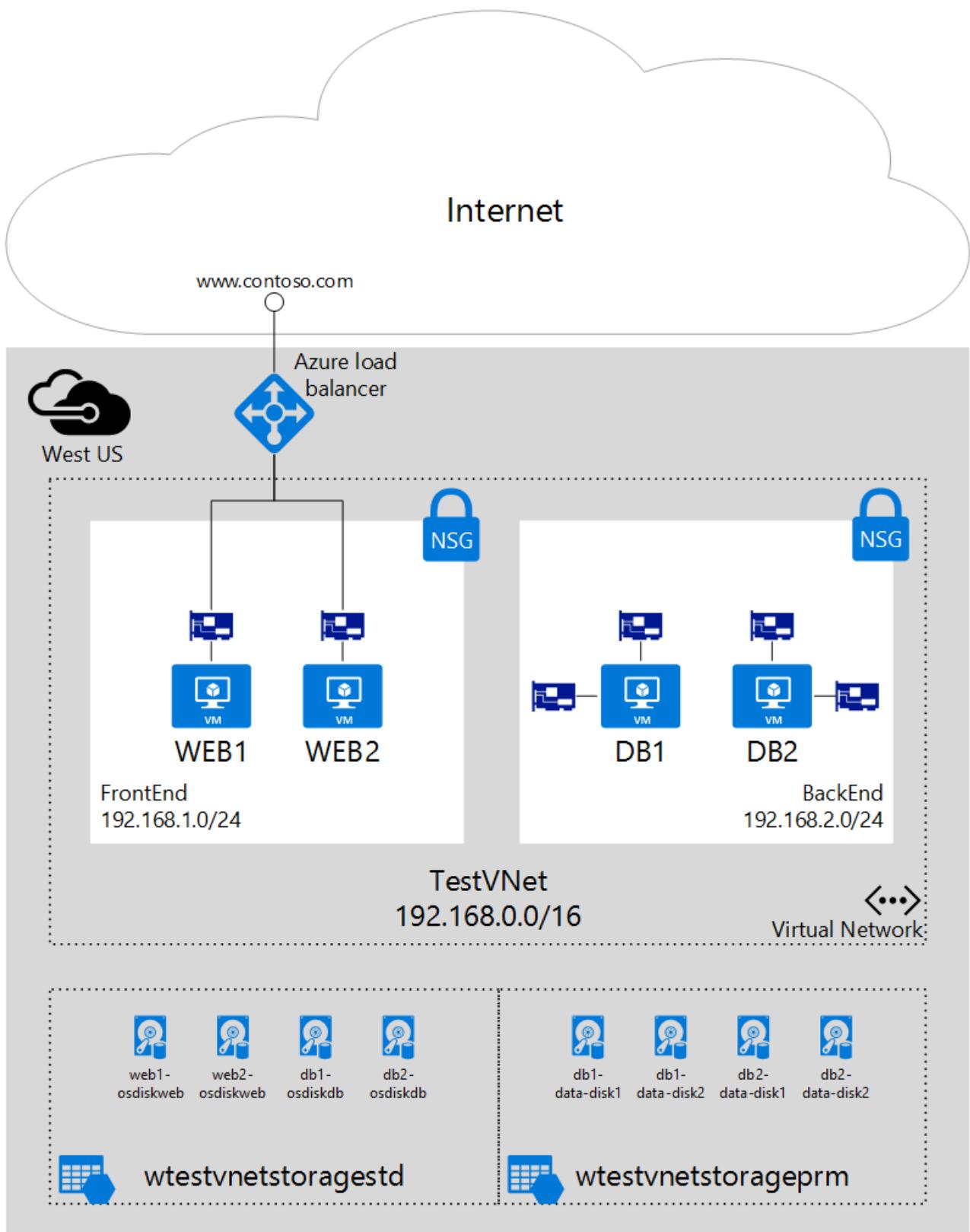
You can create virtual machines (VMs) in Azure and attach multiple network interfaces (NICs) to each of your VMs. Multiple NICs enable separation of traffic types across NICs. For example, one NIC might communicate with the Internet, while another communicates only with internal resources not connected to the Internet. The ability to separate network traffic across multiple NICs is required for many network virtual appliances, such as application delivery and WAN optimization solutions.

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the classic deployment model. Microsoft recommends that most new deployments use the Resource Manager model. Learn how to perform these steps using the [Resource Manager deployment model](#).

Scenario

This document walks through a deployment that uses multiple NICs in VMs in a specific scenario. In this scenario, you have a two-tiered IaaS workload hosted in Azure. Each tier is deployed in its own subnet in a virtual network (VNet). The front-end tier is composed of several web servers, grouped together in a load balancer set for high availability. The back-end tier is composed of several database servers. The database servers are deployed with two NICs each, one for database access, the other for management. The scenario also includes Network Security Groups (NSGs) to control what traffic is allowed to each subnet, and NIC in the deployment. The following picture shows the basic architecture of this scenario:



The following steps use a resource group named *IaaSStory* for the WEB servers and a resource group named *IaaSStory-BackEnd* for the DB servers.

Prerequisites

Before you can create the DB servers, you need to create the *IaaSStory* resource group with all the necessary resources for this scenario. To create these resources, complete the steps that follow. Create a virtual network by following the steps in the [Create a virtual network](#) article.

Prerequisite: Install the Azure CLI

To perform the steps in this article, you need to [install the Azure Command-Line Interface for Mac, Linux, and Windows \(Azure CLI\)](#) and you need to [log on to Azure](#).

NOTE

If you don't have an Azure account, you need one. Go sign up for a [free trial here](#). In addition, to follow along completely you need to have either `jq` or some other JSON parsing tool or library installed.

Deploy the back-end VMs

The back-end VMs depend on the creation of the following resources:

- **Storage account for data disks.** For better performance, the data disks on the database servers will use solid state drive (SSD) technology, which requires a premium storage account. Make sure the Azure location you deploy to support premium storage.
- **NICs.** Each VM will have two NICs, one for database access, and one for management.
- **Availability set.** All database servers will be added to a single availability set, to ensure at least one of the VMs is up and running during maintenance.

Step 1 - Start your script

You can download the full bash script used [here](#). Complete the following steps to change the script to work in your environment:

1. Change the values of the variables below based on your existing resource group deployed above in [Prerequisites](#).

```
location="useast2"
vnetName="WTestVNet"
backendSubnetName="BackEnd"
```

2. Change the values of the variables below based on the values you want to use for your backend deployment.

```
backendCSName="IaaSStory-Backend"
prmStorageAccountName="iaasstoryprmstorage"
image="0b11de9248dd4d87b18621318e037d37__RightImage-Ubuntu-14.04-x64-v14.2.1"
avSetName="ASDB"
vmSize="Standard_DS3"
diskSize=127
vmNamePrefix="DB"
osDiskName="osdiskdb"
dataDiskPrefix="db"
dataDiskName="datadisk"
ipAddressPrefix="192.168.2."
username='adminuser'
password='adminP@ssw0rd'
numberOfVMs=2
```

Step 2 - Create necessary resources for your VMs

1. Create a new cloud service for all backend VMs. Notice the use of the `$backendCSName` variable for the resource group name, and `$location` for the Azure region.

```
azure service create --serviceName $backendCSName \
--location $location
```

2. Create a premium storage account for the OS and data disks to be used by yours VMs.

```
azure storage account create $prmStorageAccountName \
--location $location \
--type PLRS
```

Step 3 - Create VMs with multiple NICs

1. Start a loop to create multiple VMs, based on the `numberOfVMs` variables.

```
for ((suffixNumber=1;suffixNumber<=numberOfVMs;suffixNumber++));  
do
```

2. For each VM, specify the name and IP address of each of the two NICs.

```
nic1Name=$vmNamePrefix$suffixNumber-DA  
x=$((suffixNumber+3))  
ipAddress1=$ipAddressPrefix$x  
  
nic2Name=$vmNamePrefix$suffixNumber-RA  
x=$((suffixNumber+53))  
ipAddress2=$ipAddressPrefix$x
```

3. Create the VM. Notice the usage of the `--nic-config` parameter, containing a list of all NICs with name, subnet, and IP address.

```
azure vm create $backendCSName $image $username $password \  
--connect $backendCSName \  
--vm-name $vmNamePrefix$suffixNumber \  
--vm-size $vmSize \  
--availability-set $avSetName \  
--blob-url $prmStorageAccountName.blob.core.windows.net/vhds/$osDiskName$suffixNumber.vhd \  
--virtual-network-name $vnetName \  
--subnet-names $backendSubnetName \  
--nic-config $nic1Name:$backendSubnetName:$ipAddress1::,$nic2Name:$backendSubnetName:$ipAddress2::
```

4. For each VM, create two data disks.

```
azure vm disk attach-new $vmNamePrefix$suffixNumber \  
$diskSize \  
vhds/$dataDiskPrefix$suffixNumber$dataDiskName-1.vhd  
  
azure vm disk attach-new $vmNamePrefix$suffixNumber \  
$diskSize \  
vhds/$dataDiskPrefix$suffixNumber$dataDiskName-2.vhd  
done
```

Step 4 - Run the script

Now that you downloaded and changed the script based on your needs, run the script to create the back end database VMs with multiple NICs.

1. Save your script and run it from your **Bash** terminal. You will see the initial output, as shown below.

```
info: Executing command service create
info: Creating cloud service
data: Cloud service name IaaSStory-Backend
info: service create command OK
info: Executing command storage account create
info: Creating storage account
info: storage account create command OK
info: Executing command vm create
info: Looking up image 0b11de9248dd4d87b18621318e037d37__RightImage-Ubuntu-14.04-x64-v14.2.1
info: Looking up virtual network
info: Looking up cloud service
info: Getting cloud service properties
info: Looking up deployment
info: Creating VM
```

- After a few minutes, the execution will end and you will see the rest of the output as shown below.

```
info: OK
info: vm create command OK
info: Executing command vm disk attach-new
info: Getting virtual machines
info: Adding Data-Disk
info: vm disk attach-new command OK
info: Executing command vm disk attach-new
info: Getting virtual machines
info: Adding Data-Disk
info: vm disk attach-new command OK
info: Executing command vm create
info: Looking up image 0b11de9248dd4d87b18621318e037d37__RightImage-Ubuntu-14.04-x64-v14.2.1
info: Looking up virtual network
info: Looking up cloud service
info: Getting cloud service properties
info: Looking up deployment
info: Creating VM
info: OK
info: vm create command OK
info: Executing command vm disk attach-new
info: Getting virtual machines
info: Adding Data-Disk
info: vm disk attach-new command OK
info: Executing command vm disk attach-new
info: Getting virtual machines
info: Adding Data-Disk
info: vm disk attach-new command OK
```

Step 5 - Configure routing within the VM's operating system

Azure DHCP 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. To configure routing for secondary network interfaces, see [Routing within a virtual machine operating system with multiple network interfaces](#).

Reserved IP addresses (Classic)

4/19/2018 • 6 min to read • [Edit Online](#)

IP addresses in Azure fall into two categories: dynamic and reserved. Public IP addresses managed by Azure are dynamic by default. That means that the IP address used for a given cloud service (VIP) or to access a VM or role instance directly (ILPIP) can change from time to time, when resources are shut down or stopped (deallocated).

To prevent IP addresses from changing, you can reserve an IP address. Reserved IPs can be used only as a VIP, ensuring that the IP address for the cloud service remains the same, even as resources are shut down or stopped (deallocated). Furthermore, you can convert existing dynamic IPs used as a VIP to a reserved IP address.

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the classic deployment model. Microsoft recommends that most new deployments use the Resource Manager model. Learn how to reserve a static public IP address using the [Resource Manager deployment model](#).

To learn more about IP addresses in Azure, read the [IP addresses](#) article.

When do I need a reserved IP?

- **You want to ensure that the IP is reserved in your subscription.** If you want to reserve an IP address that is not released from your subscription under any circumstance, you should use a reserved public IP.
- **You want your IP to stay with your cloud service even across stopped or deallocated state (VMs).** If you want your service to be accessed by using an IP address that doesn't change, even when VMs in the cloud service are shut down or stop (deallocated).
- **You want to ensure that outbound traffic from Azure uses a predictable IP address.** You may have your on-premises firewall configured to allow only traffic from specific IP addresses. By reserving an IP, you know the source IP address, and don't need to update your firewall rules due to an IP change.

FAQ

1. Can I use a reserved IP for all Azure services?

No. Reserved IPs can only be used for VMs and cloud service instance roles exposed through a VIP.

2. How many reserved IPs can I have?

For details, see the [Azure limits](#) article.

3. Is there a charge for reserved IPs?

Sometimes. For pricing details, see the [Reserved IP Address Pricing Details](#) page.

4. How do I reserve an IP address?

You can use PowerShell, the [Azure Management REST API](#), or the [Azure portal](#) to reserve an IP address in an Azure region. A reserved IP address is associated to your subscription.

5. Can I use a reserved IP with affinity group-based VNets?

No. Reserved IPs are only supported in regional VNets. Reserved IPs are not supported for VNets that are associated with affinity groups. For more information about associating a VNet with a region or affinity group, see the [About Regional VNets and Affinity Groups](#) article.

Manage reserved VIPs

Ensure you have installed and configured PowerShell by completing the steps in the [Install and configure](#)

[PowerShell](#) article.

Before you can use reserved IPs, you must add it to your subscription. To create a reserved IP from the pool of public IP addresses available in the *Central US* location, run the following command:

```
New-AzureReservedIP -ReservedIPName MyReservedIP -Location "Central US"
```

Notice, however, that you cannot specify what IP is being reserved. To view what IP addresses are reserved in your subscription, run the following PowerShell command, and notice the values for *ReservedIPName* and *Address*:

```
Get-AzureReservedIP
```

Expected output:

```
ReservedIPName      : MyReservedIP
Address            : 23.101.114.211
Id                 : d73be9dd-db12-4b5e-98c8-bc62e7c42041
Label              :
Location           : Central US
State              : Created
InUse              : False
ServiceName         :
DeploymentName     :
OperationDescription: Get-AzureReservedIP
OperationId        : 55e4f245-82e4-9c66-9bd8-273e815ce30a
OperationStatus    : Succeeded
```

NOTE

When you create a reserved IP address with PowerShell, you cannot specify a resource group to create the reserved IP in. Azure places it into a resource group named *Default-Networking* automatically. If you create the reserved IP using the [Azure portal](#), you can specify any resource group you choose. If you create the reserved IP in a resource group other than *Default-Networking* however, whenever you reference the reserved IP with commands such as `Get-AzureReservedIP` and `Remove-AzureReservedIP`, you must reference the name *Group resource-group-name reserved-ip-name*. For example, if you create a reserved IP named *myReservedIP* in a resource group named *myResourceGroup*, you must reference the name of the reserved IP as *Group myResourceGroup myReservedIP*.

Once an IP is reserved, it remains associated to your subscription until you delete it. To delete a reserved IP, run the following PowerShell command:

```
Remove-AzureReservedIP -ReservedIPName "MyReservedIP"
```

Reserve the IP address of an existing cloud service

You can reserve the IP address of an existing cloud service by adding the `-ServiceName` parameter. To reserve the IP address of a cloud service *TestService* in the *Central US* location, run the following PowerShell command:

```
New-AzureReservedIP -ReservedIPName MyReservedIP -Location "Central US" -ServiceName TestService
```

Associate a reserved IP to a new cloud service

The following script creates a new reserved IP, then associates it to a new cloud service named *TestService*.

```
New-AzureReservedIP -ReservedIPName MyReservedIP -Location "Central US"

$image = Get-AzureVMImage | ?{$_.ImageName -like "*RightImage-Windows-2012R2-x64*"}

New-AzureVMConfig -Name TestVM -InstanceSize Small -ImageName $image.ImageName ` 
| Add-AzureProvisioningConfig -Windows -AdminUsername adminuser -Password MyP@ssw0rd!! ` 
| New-AzureVM -ServiceName TestService -ReservedIPName MyReservedIP -Location "Central US"
```

NOTE

When you create a reserved IP to use with a cloud service, you still refer to the VM by using *VIP:<port number>* for inbound communication. Reserving an IP does not mean you can connect to the VM directly. The reserved IP is assigned to the cloud service that the VM has been deployed to. If you want to connect to a VM by IP directly, you have to configure an instance-level public IP. An instance-level public IP is a type of public IP (called an ILPIP) that is assigned directly to your VM. It cannot be reserved. For more information, read the [Instance-level Public IP \(ILPIP\)](#) article.

Remove a reserved IP from a running deployment

To remove a reserved IP added to a new cloud service, run the following PowerShell command:

```
Remove-AzureReservedIPAssociation -ReservedIPName MyReservedIP -ServiceName TestService
```

NOTE

Removing a reserved IP from a running deployment does not remove the reservation from your subscription. It simply frees the IP to be used by another resource in your subscription.

Associate a reserved IP to a running deployment

The following commands create a cloud service named *TestService2* with a new VM named *TestVM2*. The existing reserved IP named *MyReservedIP* is then associated to the cloud service.

```
$image = Get-AzureVMImage | ?{$_.ImageName -like "*RightImage-Windows-2012R2-x64*"}

New-AzureVMConfig -Name TestVM2 -InstanceSize Small -ImageName $image.ImageName ` 
| Add-AzureProvisioningConfig -Windows -AdminUsername adminuser -Password MyP@ssw0rd!! ` 
| New-AzureVM -ServiceName TestService2 -Location "Central US"

Set-AzureReservedIPAssociation -ReservedIPName MyReservedIP -ServiceName TestService2
```

Associate a reserved IP to a cloud service by using a service configuration file

You can also associate a reserved IP to a cloud service by using a service configuration (CSCFG) file. The following sample XML shows how to configure a cloud service to use a reserved VIP named *MyReservedIP*:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceConfiguration serviceName="ReservedIPSample"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration" osFamily="4" osVersion="*"
schemaVersion="2014-01.2.3">
  <Role name="WebRole1">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
  <NetworkConfiguration>
    <AddressAssignments>
      <ReservedIPs>
        <ReservedIP name="MyReservedIP"/>
      </ReservedIPs>
    </AddressAssignments>
  </NetworkConfiguration>
</ServiceConfiguration>
```

Next steps

- Understand how [IP addressing](#) works in the classic deployment model.
- Learn about [reserved private IP addresses](#).
- Learn about [Instance Level Public IP \(ILPIP\) addresses](#).

How to set a static internal private IP address using PowerShell (Classic)

4/19/2018 • 2 min to read • [Edit Online](#)

In most cases, you won't need to specify a static internal IP address for your virtual machine. VMs in a virtual network will automatically receive an internal IP address from a range that you specify. But in certain cases, specifying a static IP address for a particular VM makes sense. For example, if your VM is going to run DNS or will be a domain controller. A static internal IP address stays with the VM even through a stop/deprovision state.

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the classic deployment model. Microsoft recommends that most new deployments use the [Resource Manager deployment model](#).

How to verify if a specific IP address is available

To verify if the IP address `10.0.0.7` is available in a vnet named `TestVnet`, run the following PowerShell command and verify the value for `IsAvailable`:

```
Test-AzureStaticVNetIP -VNetName TestVNet -IPAddress 10.0.0.7

IsAvailable      : True
AvailableAddresses : {}
OperationDescription : Test-AzureStaticVNetIP
OperationId       : fd3097e1-5f4b-9cac-8afa-bba1e3492609
OperationStatus    : Succeeded
```

NOTE

If you want to test the command above in a safe environment follow the guidelines in [Create a virtual network \(classic\)](#) to create a vnet named `TestVnet` and ensure it uses the `10.0.0.0/8` address space.

How to specify a static internal IP when creating a VM

The PowerShell script below creates a new cloud service named `TestService`, then retrieves an image from Azure, then creates a VM named `TestVM` in the new cloud service using the retrieved image, sets the VM to be in a subnet named `Subnet-1`, and sets `10.0.0.7` as a static internal IP for the VM:

```
New-AzureService -ServiceName TestService -Location "Central US"
$image = Get-AzureVMImage | ?{$_._.ImageName -like "*RightImage-Windows-2012R2-x64*"}
New-AzureVMConfig -Name TestVM -InstanceSize Small -ImageName $image.ImageName `| Add-AzureProvisioningConfig -Windows -AdminUsername adminuser -Password MyP@ssw0rd!! `| Set-AzureSubnet -SubnetNames Subnet-1 `| Set-AzureStaticVNetIP -IPAddress 10.0.0.7 `| New-AzureVM -ServiceName "TestService" -VNetName TestVnet
```

How to retrieve static internal IP information for a VM

To view the static internal IP information for the VM created with the script above, run the following PowerShell command and observe the values for *IpAddress*:

```
Get-AzureVM -Name TestVM -ServiceName TestService

DeploymentName      : TestService
Name               : TestVM
Label              :
VM                : Microsoft.WindowsAzure.Commands.ServiceManagement.Model.PersistentVM
InstanceStatus     : Provisioning
IpAddress          : 10.0.0.7
InstanceStateDetails : Windows is preparing your computer for first use...
PowerState         : Started
InstanceErrorCode   :
InstanceFaultDomain :
InstanceName        : TestVM
InstanceUpgradeDomain :
InstanceSize        : Small
HostName           : rsR2-797
AvailabilitySetName :
DNSName            : http://testservice000.cloudapp.net/
Status              : Provisioning
GuestAgentStatus    : Microsoft.WindowsAzure.Commands.ServiceManagement.Model.GuestAgentStatus
ResourceExtensionStatusList : {Microsoft.Compute.BGInfo}
PublicIPAddress    :
PublicIPName       :
NetworkInterfaces  : {}
ServiceName         : TestService
OperationDescription : Get-AzureVM
OperationId         : 34c1560a62f0901ab75cde4fed8e8bd1
OperationStatus     : OK
```

How to remove a static internal IP from a VM

To remove the static internal IP added to the VM in the script above, run the following PowerShell command:

```
Get-AzureVM -ServiceName TestService -Name TestVM `| Remove-AzureStaticVNetIP `| Update-AzureVM
```

How to add a static internal IP to an existing VM

To add a static internal IP to the VM created using the script above, run the following command:

```
Get-AzureVM -ServiceName TestService000 -Name TestVM `| Set-AzureStaticVNetIP -IPAddress 10.10.0.7 `| Update-AzureVM
```

Next steps

[Reserved IP](#)

[Instance-Level Public IP \(ILPIP\)](#)

[Reserved IP REST APIs](#)

Configure private IP addresses for a virtual machine (Classic) using the Azure CLI 1.0

4/19/2018 • 4 min to read • [Edit Online](#)

Your IaaS virtual machines (VMs) and PaaS role instances in a virtual network automatically receive a private IP address from a range that you specify, based on the subnet they are connected to. That address is retained by the VMs and role instances, until they are decommissioned. You decommission a VM or role instance by stopping it from PowerShell, the Azure CLI, or the Azure portal. In those cases, once the VM or role instance starts again, it will receive an available IP address from the Azure infrastructure, which might not be the same it previously had. If you shut down the VM or role instance from the guest operating system, it retains the IP address it had.

In certain cases, you want a VM or role instance to have a static IP address, for example, if your VM is going to run DNS or will be a domain controller. You can do so by setting a static private IP address.

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 classic deployment model. You can also [manage a static private IP address in the Resource Manager deployment model](#).

The sample Azure CLI commands below expect a simple environment already created. If you want to run the commands as they are displayed in this document, first build the test environment described in [create a vnet](#).

How to specify a static private IP address when creating a VM

To create a new VM named *DNS01* in a new cloud service named *TestService* based on the scenario above, follow these steps:

1. If you have never used Azure CLI, see [Install and Configure the Azure CLI](#) and follow the instructions up to the point where you select your Azure account and subscription.
2. Run the **azure service create** command to create the cloud service.

```
azure service create TestService --location uscentral
```

Expected output:

```
info:  Executing command service create
info:  Creating cloud service
data:  Cloud service name TestService
info:  service create command OK
```

3. Run the **azure create vm** command to create the VM. Notice the value for a static private IP address. The list shown after the output explains the parameters used.

```
azure vm create -l centralus -n DNS01 -w TestVNet -S "192.168.1.101" TestService  
bd507d3a70934695bc2128e3e5a255ba__RightImage-Windows-2012R2-x64-v14.2 adminuser AdminP@ssw0rd
```

Expected output:

```
info: Executing command vm create  
warn: --vm-size has not been specified. Defaulting to "Small".  
info: Looking up image bd507d3a70934695bc2128e3e5a255ba__RightImage-Windows-2012R2-x64-v14.2  
info: Looking up virtual network  
info: Looking up cloud service  
warn: --location option will be ignored  
info: Getting cloud service properties  
info: Looking up deployment  
info: Retrieving storage accounts  
info: Creating VM  
info: OK  
info: vm create command OK
```

- **-l (or --location)**. Azure region where the VM will be created. For our scenario, *centralus*.
- **-n (or --vm-name)**. Name of the VM to be created.
- **-w (or --virtual-network-name)**. Name of the VNet where the VM will be created.
- **-S (or --static-ip)**. Static private IP address for the VM.
- **TestService**. Name of the cloud service where the VM will be created.
- **bd507d3a70934695bc2128e3e5a255ba__RightImage-Windows-2012R2-x64-v14.2**. Image used to create the VM.
- **adminuser**. Local administrator for the Windows VM.
- **AdminP@ssw0rd**. Local administrator password for the Windows VM.

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 script above, run the following Azure CLI command and observe the value for *Network StaticIP*:

```
azure vm static-ip show DNS01
```

Expected output:

```
info: Executing command vm static-ip show  
info: Getting virtual machines  
data: Network StaticIP "192.168.1.101"  
info: vm static-ip show command OK
```

How to remove a static private IP address from a VM

To remove the static private IP address added to the VM in the script above, run the following Azure CLI command:

```
azure vm static-ip remove DNS01
```

Expected output:

```
info: Executing command vm static-ip remove
info: Getting virtual machines
info: Reading network configuration
info: Updating network configuration
info: vm static-ip remove command OK
```

How to add a static private IP to an existing VM

To add a static private IP address to the VM created using the script above, run the following command:

```
azure vm static-ip set DNS01 192.168.1.101
```

Expected output:

```
info: Executing command vm static-ip set
info: Getting virtual machines
info: Looking up virtual network
info: Reading network configuration
info: Updating network configuration
info: vm static-ip set command OK
```

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. 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 VM, or you can lose connectivity to the virtual machine. 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 [reserved public IP](#) addresses.
- Learn about [instance-level public IP \(ILPIP\)](#) addresses.
- Consult the [Reserved IP REST APIs](#).

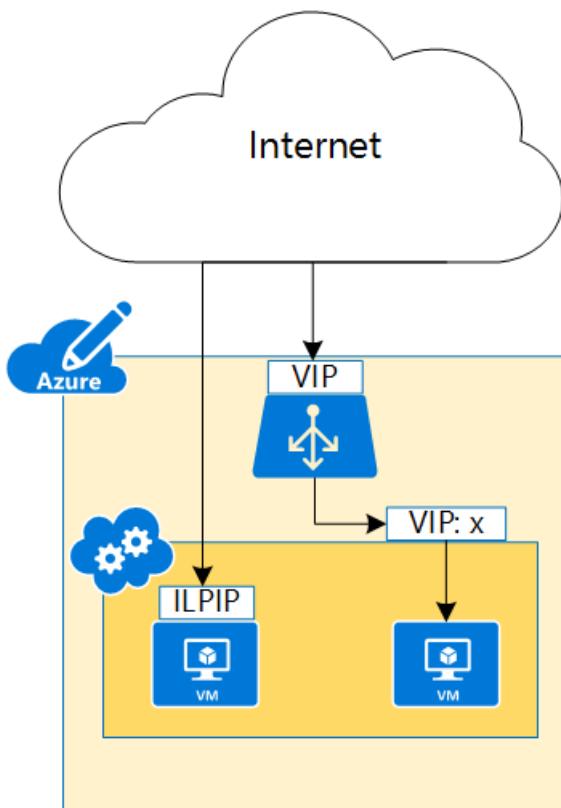
Instance level public IP (Classic) overview

4/19/2018 • 4 min to read • [Edit Online](#)

An instance level public IP (ILPIP) is a public IP address that you can assign directly to a VM or Cloud Services role instance, rather than to the cloud service that your VM or role instance reside in. An ILPIP doesn't take the place of the virtual IP (VIP) that is assigned to your cloud service. Rather, it's an additional IP address that you can use to connect directly to your VM or role instance.

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and classic](#). This article covers using the classic deployment model. Microsoft recommends creating VMs through Resource Manager. Make sure you understand how [IP addresses work in Azure](#).



As shown in Figure 1, the cloud service is accessed using a VIP, while the individual VMs are normally accessed using VIP:<port number>. By assigning an ILPIP to a specific VM, that VM can be accessed directly using that IP address.

When you create a cloud service in Azure, corresponding DNS A records are created automatically to allow access to the service through a fully qualified domain name (FQDN), instead of using the actual VIP. The same process happens for an ILPIP, allowing access to the VM or role instance by FQDN instead of the ILPIP. For instance, if you create a cloud service named *contosoadservice*, and you configure a web role named *contosoweb* with two instances, Azure registers the following A records for the instances:

- contosoweb_IN_0.*contosoadservice.cloudapp.net*
- contosoweb_IN_1.*contosoadservice.cloudapp.net*

NOTE

You can assign only one ILPIP for each VM or role instance. You can use up to 5 ILPIPs per subscription. ILPIPs are not supported for multi-NIC VMs.

Why would I request an ILPIP?

If you want to be able to connect to your VM or role instance by an IP address assigned directly to it, rather than using the cloud service VIP:<port number>, request an ILPIP for your VM or your role instance.

- **Active FTP** - By assigning an ILPIP to a VM, it can receive traffic on any port. Endpoints are not required for the VM to receive traffic. See (https://en.wikipedia.org/wiki/File_Transfer_Protocol#Protocol_overview) [FTP Protocol Overview] for details on the FTP protocol.
- **Outbound IP** - Outbound traffic originating from the VM is mapped to the ILPIP as the source and the ILPIP uniquely identifies the VM to external entities.

NOTE

In the past, an ILPIP address was referred to as a public IP (PIP) address.

Manage an ILPIP for a VM

The following tasks enable you to create, assign, and remove ILPIPs from VMs:

How to request an ILPIP during VM creation using PowerShell

The following PowerShell script creates a cloud service named *FTPService*, retrieves an image from Azure, creates a VM named *FTPInstance* using the retrieved image, sets the VM to use an ILPIP, and adds the VM to the new service:

```
New-AzureService -ServiceName FTSPService -Location "Central US"

$image = Get-AzureVMImage |?{$_._ImageName -like "*RightImage-Windows-2012R2-x64*"} ` 
New-AzureVMConfig -Name FTPInstance -InstanceSize Small -ImageName $image._ImageName ` 
| Add-AzureProvisioningConfig -Windows -AdminUsername adminuser -Password MyP@ssw0rd!! ` 
| Set-AzurePublicIP -PublicIPName ftppip | New-AzureVM -ServiceName FTSPService -Location "Central US"
```

How to retrieve ILPIP information for a VM

To view the ILPIP information for the VM created with the previous script, run the following PowerShell command and observe the values for *PublicIPAddress* and *PublicIPName*:

```
Get-AzureVM -Name FTPInstance -ServiceName FTSPService
```

Expected output:

```

DeploymentName      : FTPService
Name               : FTPInstance
Label              :
VM                : Microsoft.WindowsAzure.Commands.ServiceManagement.Model.PersistentVM
InstanceStateStatus : ReadyRole
IpAddress          : 100.74.118.91
InstanceStateDetails :
PowerState         : Started
InstanceErrorCode   :
InstanceFaultDomain : 0
InstanceName        : FTPInstance
InstanceUpgradeDomain : 0
InstanceSize        : Small
HostName            : FTPInstance
AvailabilitySetName :
DNSName            : http://ftpservice888.cloudapp.net/
Status              : ReadyRole
GuestAgentStatus    : Microsoft.WindowsAzure.Commands.ServiceManagement.Model.GuestAgentStatus
ResourceExtensionStatusList : {Microsoft.Compute.BGInfo}
PublicIPAddress     : 104.43.142.188
PublicIPName        : ftppip
NetworkInterfaces   : {}
ServiceName         : FTPService
OperationDescription : Get-AzureVM
OperationId         : 568d88d2be7c98f4bbb875e4d823718e
OperationStatus      : OK

```

How to remove an ILPIP from a VM

To remove the ILPIP added to the VM in the previous script, run the following PowerShell command:

```
Get-AzureVM -ServiceName FTPService -Name FTPInstance | Remove-AzurePublicIP | Update-AzureVM
```

How to add an ILPIP to an existing VM

To add an ILPIP to the VM created using the script previous, run the following command:

```
Get-AzureVM -ServiceName FTPService -Name FTPInstance | Set-AzurePublicIP -PublicIPName ftppip2 | Update-AzureVM
```

Manage an ILPIP for a Cloud Services role instance

To add an ILPIP to a Cloud Services role instance, complete the following steps:

1. Download the .cscfg file for the cloud service by completing the steps in the [How to Configure Cloud Services](#) article.
2. Update the .cscfg file by adding the `InstanceAddress` element. The following sample adds an ILPIP named `MyPublicIP` to a role instance named `WebRole1`:

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceConfiguration serviceName="ILPIPSample"
xmlns="http://schemas.microsoft.com/ServiceHosting/2008/10/ServiceConfiguration" osFamily="4"
osVersion="*" schemaVersion="2014-01.2.3">
  <Role name="WebRole1">
    <Instances count="1" />
    <ConfigurationSettings>
      <Setting name="Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"
value="UseDevelopmentStorage=true" />
    </ConfigurationSettings>
  </Role>
  <NetworkConfiguration>
    <AddressAssignments>
      <InstanceAddress roleName="WebRole1">
        <PublicIPs>
          <PublicIP name="MyPublicIP" domainNameLabel="MyPublicIP" />
        </PublicIPs>
      </InstanceAddress>
    </AddressAssignments>
  </NetworkConfiguration>
</ServiceConfiguration>
```

3. Upload the .cscfg file for the cloud service by completing the steps in the [How to Configure Cloud Services](#) article.

Next steps

- Understand how [IP addressing](#) works in the classic deployment model.
- Learn about [Reserved IPs](#).

How to set up endpoints on a classic Windows virtual machine in Azure

4/9/2018 • 4 min to read • [Edit Online](#)

All Windows virtual machines that you create in Azure using the classic deployment model can automatically communicate over a private network channel with other virtual machines in the same cloud service or virtual network. However, computers on the Internet or other virtual networks require endpoints to direct the inbound network traffic to a virtual machine. This article is also available for [Linux virtual machines](#).

IMPORTANT

Azure has two different deployment models for creating and working with resources: [Resource Manager and Classic](#). This article covers using the Classic deployment model. Microsoft recommends that most new deployments use the Resource Manager model.

Starting November 15, 2017, Virtual Machines will be available only in the [Azure portal](#).

In the **Resource Manager** deployment model, endpoints are configured using **Network Security Groups (NSGs)**. For more information, see [Allow external access to your VM using the Azure portal](#).

When you create a Windows virtual machine in the Azure portal, common endpoints like those for Remote Desktop and Windows PowerShell Remoting are typically created for you automatically. You can configure additional endpoints while creating the virtual machine or afterwards as needed.

Each endpoint has a *public port* and a *private port*:

- The public port is used by the Azure load balancer to listen for incoming traffic to the virtual machine from the Internet.
- The private port is used by the virtual machine to listen for incoming traffic, typically destined to an application or service running on the virtual machine.

Default values for the IP protocol and TCP or UDP ports for well-known network protocols are provided when you create endpoints with the Azure portal. For custom endpoints, you'll need to specify the correct IP protocol (TCP or UDP) and the public and private ports. To distribute incoming traffic randomly across multiple virtual machines, you'll need to create a load-balanced set consisting of multiple endpoints.

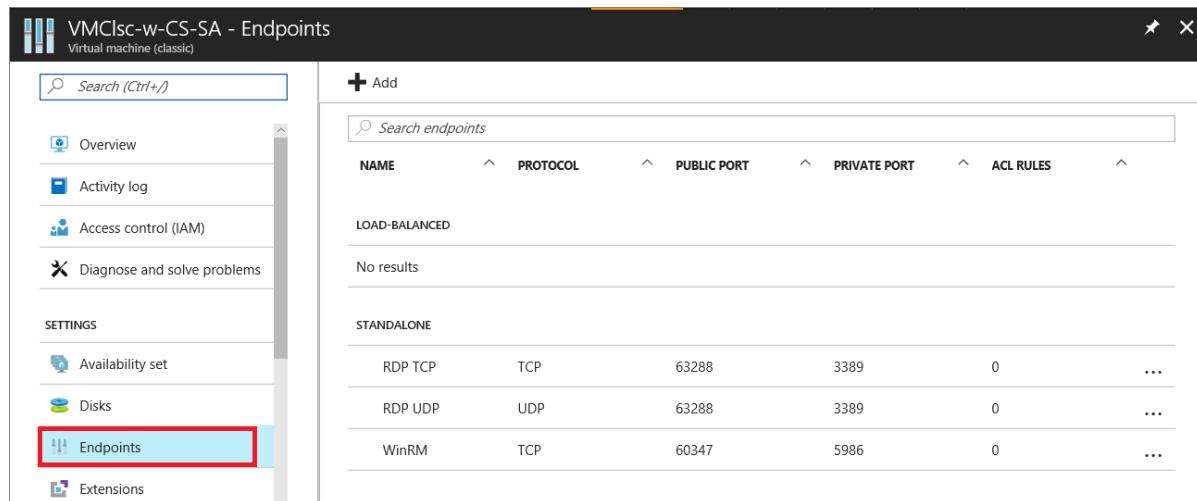
After you create an endpoint, you can use an access control list (ACL) to define rules that permit or deny the incoming traffic to the public port of the endpoint based on its source IP address. However, if the virtual machine is in an Azure virtual network, you should use network security groups instead. For details, see [About network security groups](#).

NOTE

Firewall configuration for Azure virtual machines is done automatically for ports associated with remote connectivity endpoints that Azure sets up automatically. For ports specified for all other endpoints, no configuration is done automatically to the firewall of the virtual machine. When you create an endpoint for the virtual machine, you'll need to ensure that the firewall of the virtual machine also allows the traffic for the protocol and private port corresponding to the endpoint configuration. To configure the firewall, see the documentation or on-line help for the operating system running on the virtual machine.

Create an endpoint

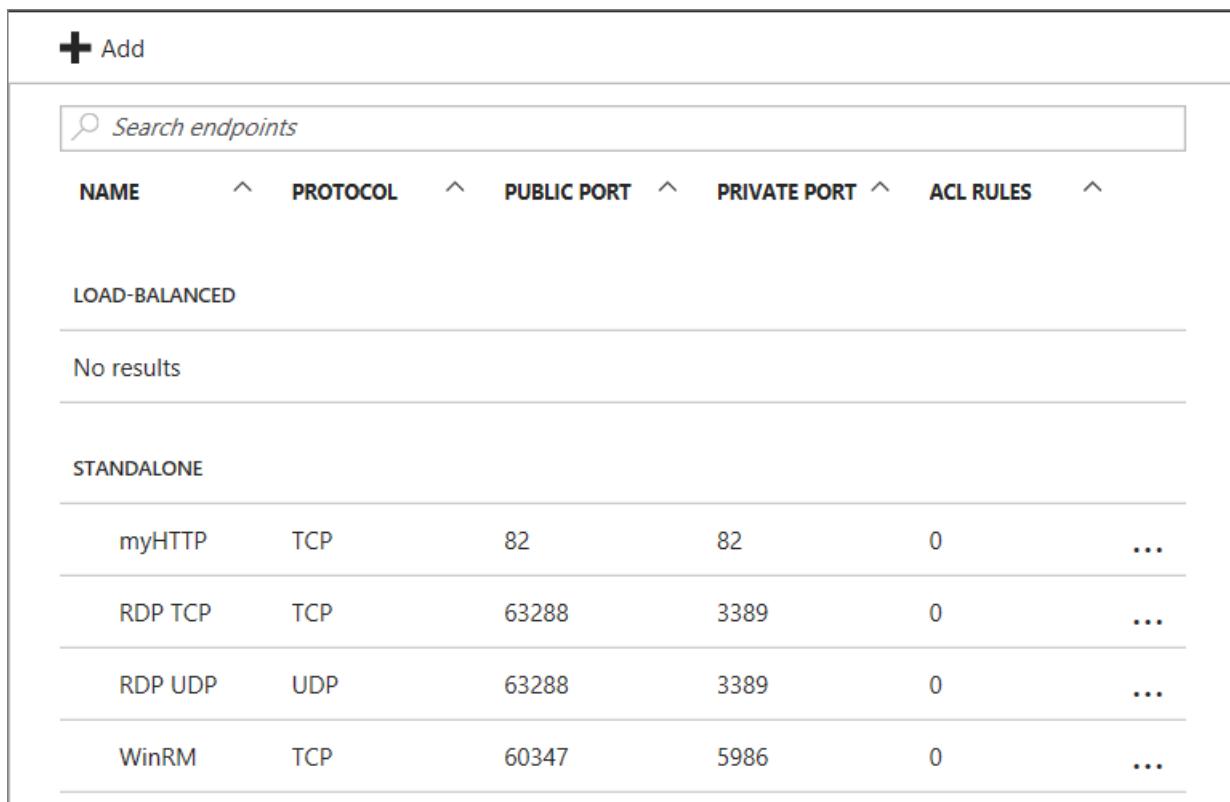
1. If you haven't already done so, sign in to the [Azure portal](#).
2. Click **Virtual Machines**, and then click the name of the virtual machine that you want to configure.
3. Click **Endpoints** in the **Settings** group. The **Endpoints** page lists all the current endpoints for the virtual machine. (This example is a Windows VM. A Linux VM will by default show an endpoint for SSH.)



The screenshot shows the Azure portal interface for a Windows VM named 'VMClsc-w-CS-SA'. The left sidebar has a 'SETTINGS' group with 'Endpoints' highlighted. The main area is titled 'Endpoints' and shows a table of existing endpoints. The table has columns: NAME, PROTOCOL, PUBLIC PORT, PRIVATE PORT, and ACL RULES. It lists three entries under 'STANDALONE': RDP TCP (TCP, 63288, 3389, 0), RDP UDP (UDP, 63288, 3389, 0), and WinRM (TCP, 60347, 5986, 0). A search bar at the top says 'Search endpoints'.

4. In the command bar above the endpoint entries, click **Add**.
5. On the **Add endpoint** page, type a name for the endpoint in **Name**.
6. In **Protocol**, choose either **TCP** or **UDP**.
7. In **Public Port**, type the port number for the incoming traffic from the Internet. In **Private Port**, type the port number on which the virtual machine is listening. These port numbers can be different. Ensure that the firewall on the virtual machine has been configured to allow the traffic corresponding to the protocol (in step 6) and private port.
8. Click **Ok**.

The new endpoint will be listed on the **Endpoints** page.



The screenshot shows the 'Add endpoint' page with a 'NAME' field containing 'myHTTP'. The table below shows the updated list of endpoints. The 'myHTTP' entry is added under 'STANDALONE' with TCP protocol, public port 82, and private port 82. Other entries remain the same: RDP TCP (TCP, 63288, 3389, 0), RDP UDP (UDP, 63288, 3389, 0), and WinRM (TCP, 60347, 5986, 0).

NAME	PROTOCOL	PUBLIC PORT	PRIVATE PORT	ACL RULES
myHTTP	TCP	82	82	0
RDP TCP	TCP	63288	3389	0
RDP UDP	UDP	63288	3389	0
WinRM	TCP	60347	5986	0

Manage the ACL on an endpoint

To define the set of computers that can send traffic, the ACL on an endpoint can restrict traffic based upon source IP address. Follow these steps to add, modify, or remove an ACL on an endpoint.

NOTE

If the endpoint is part of a load-balanced set, any changes you make to the ACL on an endpoint are applied to all endpoints in the set.

If the virtual machine is in an Azure virtual network, we recommend network security groups instead of ACLs. For details, see [About network security groups](#).

1. If you haven't already done so, sign in to the Azure portal.
2. Click **Virtual Machines**, and then click the name of the virtual machine that you want to configure.
3. Click **Endpoints**. From the list, select the appropriate endpoint. The ACL list is at the bottom of the page.

ORDER	NAME	ACTION	REMOTE SUBNET
1001	myServer	permit	10.1.0.0/8

4. Use rows in the list to add, delete, or edit rules for an ACL and change their order. The **Remote Subnet** value is an IP address range for incoming traffic from the Internet that the Azure load balancer uses to permit or deny the traffic based on its source IP address. Be sure to specify the IP address range in CIDR format, also known as address prefix format. An example is `10.1.0.0/8`.

ORDER	NAME	ACTION	REMOTE SUBNET
1001	myServer	permit	10.1.0.0/8

You can use rules to allow only traffic from specific computers corresponding to your computers on the Internet or to deny traffic from specific, known address ranges.

The rules are evaluated in order starting with the first rule and ending with the last rule. This means that rules should be ordered from least restrictive to most restrictive. For examples and more information, see [What is a Network Access Control List](#).

Next steps

- To use an Azure PowerShell cmdlet to set up a VM endpoint, see [Add-AzureEndpoint](#).
- To use an Azure PowerShell cmdlet to manage an ACL on an endpoint, see [Managing access control lists \(ACLs\) for endpoints by using PowerShell](#).
- If you created a virtual machine in the Resource Manager deployment model, you can use Azure PowerShell to [create network security groups](#) to control traffic to the VM.

Manage endpoint access control lists using PowerShell in the classic deployment model

6/27/2017 • 3 min to read • [Edit Online](#)

You can create and manage Network Access Control Lists (ACLs) for endpoints by using Azure PowerShell or in the Management Portal. In this topic, you'll find procedures for ACL common tasks that you can complete using PowerShell. For the list of Azure PowerShell cmdlets see [Azure Management Cmdlets](#). For more information about ACLs, see [What is a Network Access Control List \(ACL\)?](#). If you want to manage your ACLs by using the Management Portal, see [How to Set Up Endpoints to a Virtual Machine](#).

Manage Network ACLs by using Azure PowerShell

You can use Azure PowerShell cmdlets to create, remove, and configure (set) Network Access Control Lists (ACLs). We've included a few examples of some of the ways you can configure an ACL using PowerShell.

To retrieve a complete list of the ACL PowerShell cmdlets, you can use either of the following:

```
Get-Help *AzureACL*
Get-Command -Noun AzureACLConfig
```

Create a Network ACL with rules that permit access from a remote subnet

The example below illustrates a way to create a new ACL that contains rules. This ACL is then applied to a virtual machine endpoint. The ACL rules in the example below will allow access from a remote subnet. To create a new Network ACL with permit rules for a remote subnet, open an Azure PowerShell ISE. Copy and paste the script below, configuring the script with your own values, and then run the script.

1. Create the new network ACL object.

```
$acl1 = New-AzureAclConfig
```

2. Set a rule that permits access from a remote subnet. In the example below, you set rule 100 (which has priority over rule 200 and higher) to allow the remote subnet 10.0.0.0/8 access to the virtual machine endpoint. Replace the values with your own configuration requirements. The name "SharePoint ACL config" should be replaced with the friendly name that you want to call this rule.

```
Set-AzureAclConfig -AddRule -ACL $acl1 -Order 100 ` 
    -Action permit -RemoteSubnet "10.0.0.0/8" ` 
    -Description "SharePoint ACL config"
```

3. For additional rules, repeat the cmdlet, replacing the values with your own configuration requirements. Be sure to change the rule number Order to reflect the order in which you want the rules to be applied. The lower rule number takes precedence over the higher number.

```
Set-AzureAclConfig -AddRule -ACL $acl1 -Order 200 ` 
    -Action permit -RemoteSubnet "157.0.0.0/8" ` 
    -Description "web frontend ACL config"
```

4. Next, you can either create a new endpoint (Add) or set the ACL for an existing endpoint (Set). In this

example, we will add a new virtual machine endpoint called "web" and update the virtual machine endpoint with the ACL settings.

```
Get-AzureVM -ServiceName $serviceName -Name $vmName `| Add-AzureEndpoint -Name "web" -Protocol tcp -Localport 80 - PublicPort 80 -ACL $acl1 `| Update-AzureVM
```

5. Next, combine the cmdlets and run the script. For this example, the combined cmdlets would look like this:

```
$acl1 = New-AzureAclConfig  
Set-AzureAclConfig -AddRule -ACL $acl1 -Order 100 `  
-Action permit -RemoteSubnet "10.0.0.0/8" `  
-Description "Sharepoint ACL config"  
Set-AzureAclConfig -AddRule -ACL $acl1 -Order 200 `  
-Action permit -RemoteSubnet "157.0.0.0/8" `  
-Description "web frontend ACL config"  
Get-AzureVM -ServiceName $serviceName -Name $vmName `| Add-AzureEndpoint -Name "web" -Protocol tcp -Localport 80 - PublicPort 80 -ACL $acl1 `| Update-AzureVM
```

Remove a Network ACL rule that permits access from a remote subnet

The example below illustrates a way to remove a network ACL rule. To remove a Network ACL rule with permit rules for a remote subnet, open an Azure PowerShell ISE. Copy and paste the script below, configuring the script with your own values, and then run the script.

1. First step is to get the Network ACL object for the virtual machine endpoint. You'll then remove the ACL rule. In this case, we are removing it by rule ID. This will only remove the rule ID 0 from the ACL. It does not remove the ACL object from the virtual machine endpoint.

```
Get-AzureVM -ServiceName $serviceName -Name $vmName `| Get-AzureAclConfig -EndpointName "web" `| Set-AzureAclConfig -RemoveRule -ID 0 -ACL $acl1
```

2. Next, you must apply the Network ACL object to the virtual machine endpoint and update the virtual machine.

```
Get-AzureVM -ServiceName $serviceName -Name $vmName `| Set-AzureEndpoint -ACL $acl1 -Name "web" `| Update-AzureVM
```

Remove a Network ACL from a virtual machine endpoint

In certain scenarios, you might want to remove a Network ACL object from a virtual machine endpoint. To do that, open an Azure PowerShell ISE. Copy and paste the script below, configuring the script with your own values, and then run the script.

```
Get-AzureVM -ServiceName $serviceName -Name $vmName `| Remove-AzureAclConfig -EndpointName "web" `| Update-AzureVM
```

Next steps

[What is a Network Access Control List \(ACL\)?](#)

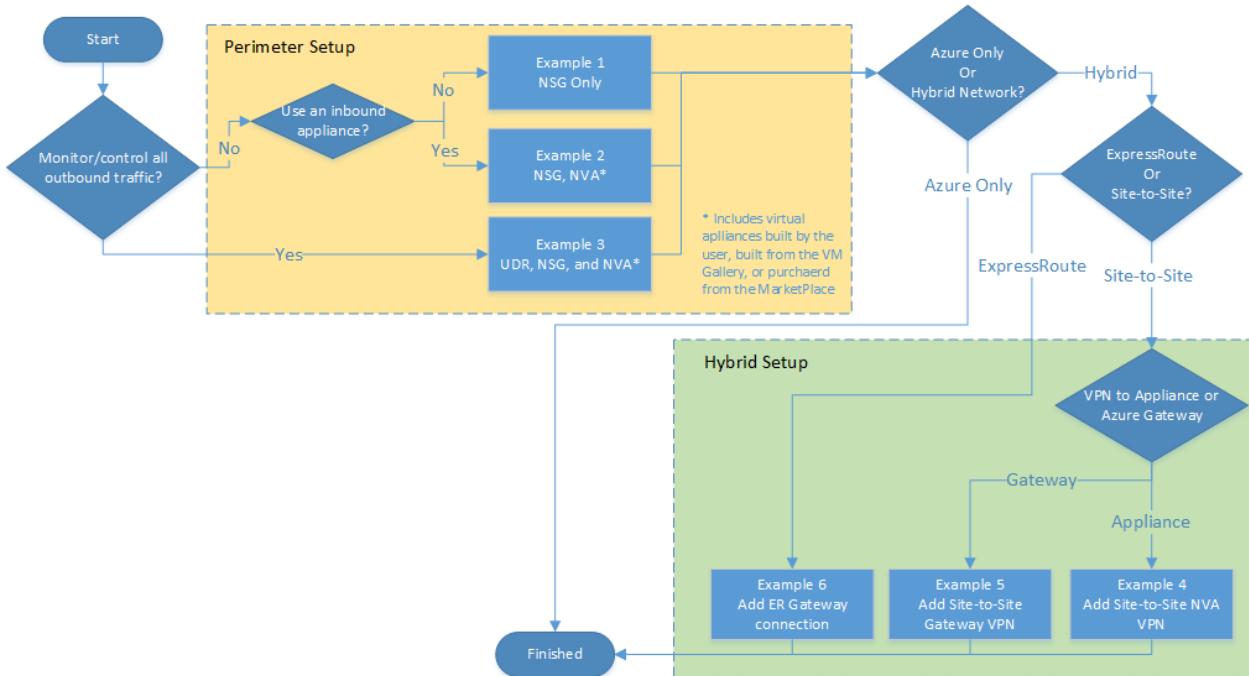
Microsoft cloud services and network security

6/27/2017 • 37 min to read • [Edit Online](#)

Microsoft cloud services deliver hyper-scale services and infrastructure, enterprise-grade capabilities, and many choices for hybrid connectivity. Customers can choose to access these services either via the Internet or with Azure ExpressRoute, which provides private network connectivity. The Microsoft Azure platform allows customers to seamlessly extend their infrastructure into the cloud and build multi-tier architectures. Additionally, third parties can enable enhanced capabilities by offering security services and virtual appliances. This white paper provides an overview of security and architectural issues that customers should consider when using Microsoft cloud services accessed via ExpressRoute. It also covers creating more secure services in Azure virtual networks.

Fast start

The following logic chart can direct you to a specific example of the many security techniques available with the Azure platform. For quick reference, find the example that best fits your case. For expanded explanations, continue reading through the paper.



Example 1: Build a perimeter network (also known as DMZ, demilitarized zone, or screened subnet) to help protect applications with network security groups (NSGs).

Example 2: Build a perimeter network to help protect applications with a firewall and NSGs.

Example 3: Build a perimeter network to help protect networks with a firewall, user-defined route (UDR), and NSG.

Example 4: Add a hybrid connection with a site-to-site, virtual appliance virtual private network (VPN).

Example 5: Add a hybrid connection with a site-to-site, Azure VPN gateway.

Example 6: Add a hybrid connection with ExpressRoute.

Examples for adding connections between virtual networks, high availability, and service chaining will be added to this document over the next few months.

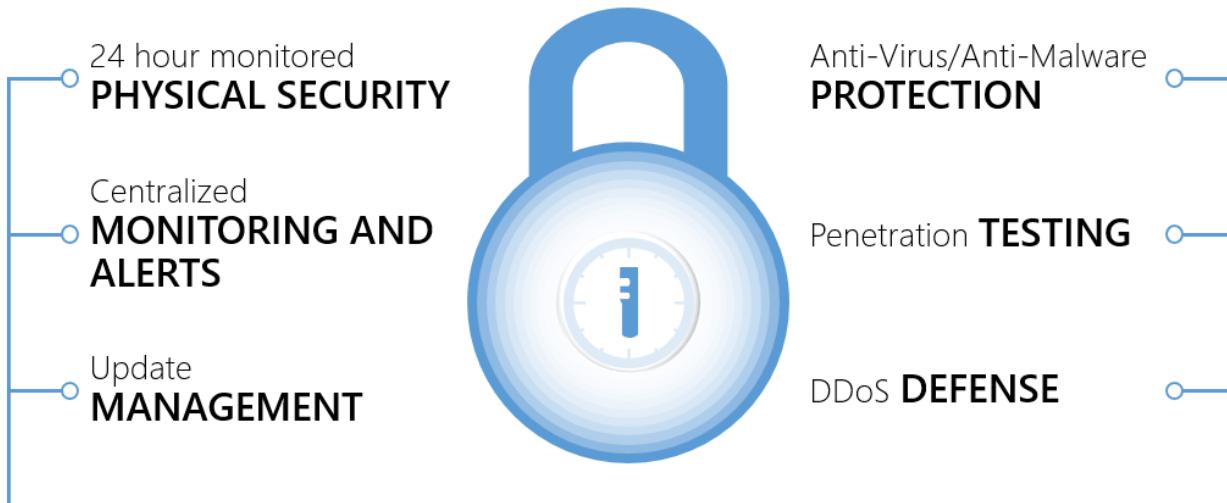
Microsoft compliance and infrastructure protection

To help organizations comply with national, regional, and industry-specific requirements governing the collection

and use of individuals' data, Microsoft offers over 40 certifications and attestations. The most comprehensive set of any cloud service provider.

For more information, see the compliance information on the [Microsoft Trust Center](#).

Microsoft has a comprehensive approach to protect cloud infrastructure needed to run hyper-scale global services. Microsoft cloud infrastructure includes hardware, software, networks, and administrative and operations staff, in addition to the physical data centers.

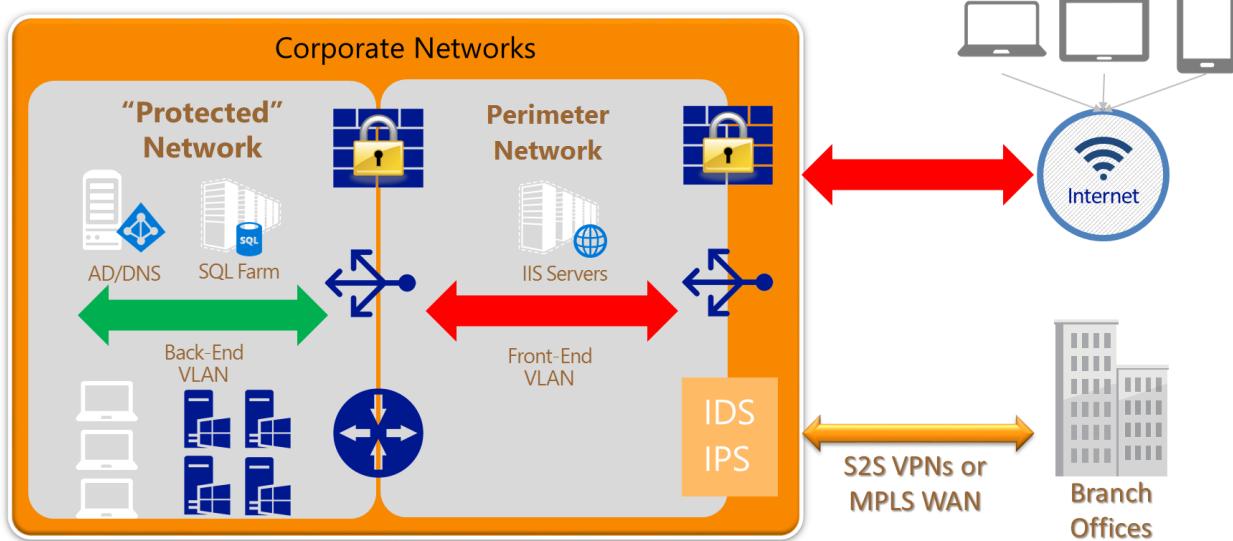


This approach provides a more secure foundation for customers to deploy their services in the Microsoft cloud. The next step is for customers to design and create a security architecture to protect these services.

Traditional security architectures and perimeter networks

Although Microsoft invests heavily in protecting the cloud infrastructure, customers must also protect their cloud services and resource groups. A multilayered approach to security provides the best defense. A perimeter network security zone protects internal network resources from an untrusted network. A perimeter network refers to the edges or parts of the network that sit between the Internet and the protected enterprise IT infrastructure.

In typical enterprise networks, the core infrastructure is heavily fortified at the perimeters, with multiple layers of security devices. The boundary of each layer consists of devices and policy enforcement points. Each layer can include a combination of the following network security devices: firewalls, Denial of Service (DoS) prevention, Intrusion Detection or Protection Systems (IDS/IPS), and VPN devices. Policy enforcement can take the form of firewall policies, access control lists (ACLs), or specific routing. The first line of defense in the network, directly accepting incoming traffic from the Internet, is a combination of these mechanisms to block attacks and harmful traffic while allowing legitimate requests further into the network. This traffic routes directly to resources in the perimeter network. That resource may then "talk" to resources deeper in the network, transiting the next boundary for validation first. The outermost layer is called the perimeter network because this part of the network is exposed to the Internet, usually with some form of protection on both sides. The following figure shows an example of a single subnet perimeter network in a corporate network, with two security boundaries.

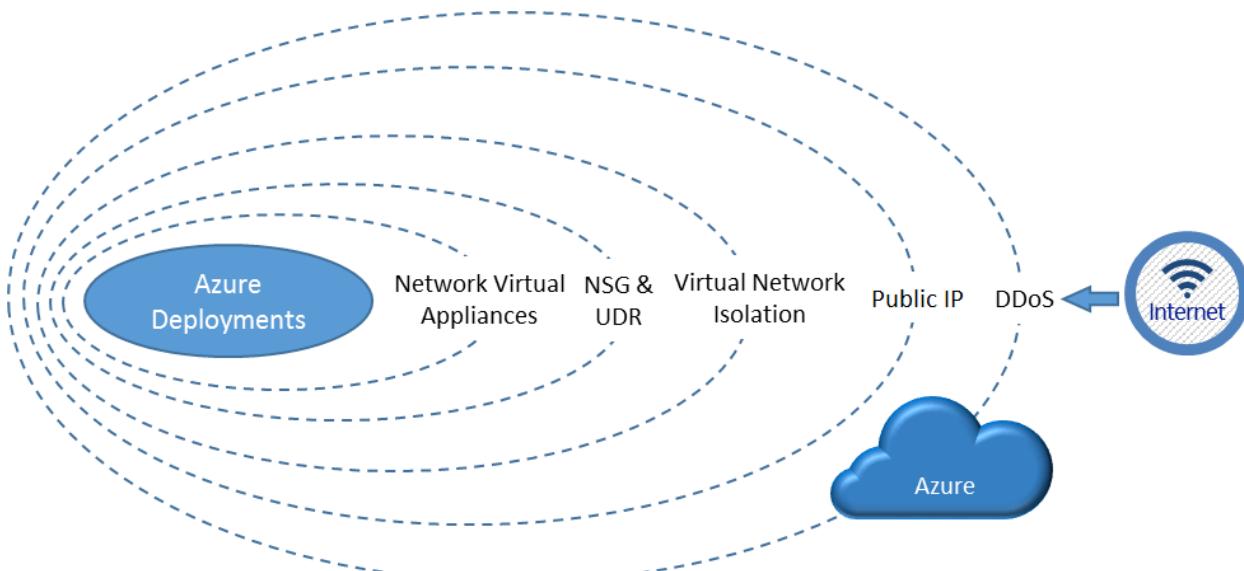


There are many architectures used to implement a perimeter network. These architectures can range from a simple load balancer to a multiple-subnet perimeter network with varied mechanisms at each boundary to block traffic and protect the deeper layers of the corporate network. How the perimeter network is built depends on the specific needs of the organization and its overall risk tolerance.

As customers move their workloads to public clouds, it is critical to support similar capabilities for perimeter network architecture in Azure to meet compliance and security requirements. This document provides guidelines on how customers can build a secure network environment in Azure. It focuses on the perimeter network, but also includes a comprehensive discussion of many aspects of network security. The following questions inform this discussion:

- How can a perimeter network in Azure be built?
- What are some of the Azure features available to build the perimeter network?
- How can back-end workloads be protected?
- How are Internet communications controlled to the workloads in Azure?
- How can the on-premises networks be protected from deployments in Azure?
- When should native Azure security features be used versus third-party appliances or services?

The following diagram shows various layers of security Azure provides to customers. These layers are both native in the Azure platform itself and customer-defined features:



Inbound from the Internet, Azure DDoS helps protect against large-scale attacks against Azure. The next layer is customer-defined public IP addresses (endpoints), which are used to determine which traffic can pass through

the cloud service to the virtual network. Native Azure virtual network isolation ensures complete isolation from all other networks and that traffic only flows through user configured paths and methods. These paths and methods are the next layer, where NSGs, UDR, and network virtual appliances can be used to create security boundaries to protect the application deployments in the protected network.

The next section provides an overview of Azure virtual networks. These virtual networks are created by customers, and are what their deployed workloads are connected to. Virtual networks are the basis of all the network security features required to establish a perimeter network to protect customer deployments in Azure.

Overview of Azure virtual networks

Before Internet traffic can get to the Azure virtual networks, there are two layers of security inherent to the Azure platform:

- DDoS protection:** DDoS protection is a layer of the Azure physical network that protects the Azure platform itself from large-scale Internet-based attacks. These attacks use multiple "bot" nodes in an attempt to overwhelm an Internet service. Azure has a robust DDoS protection mesh on all inbound, outbound, and cross-Azure region connectivity. This DDoS protection layer has no user configurable attributes and is not accessible to the customer. The DDoS protection layer protects Azure as a platform from large-scale attacks, it also monitors out-bound traffic and cross-Azure region traffic. Using network virtual appliances on the VNet, additional layers of resilience can be configured by the customer against a smaller scale attack that doesn't trip the platform level protection. An example of DDoS in action; if an internet facing IP address was attacked by a large-scale DDoS attack, Azure would detect the sources of the attacks and scrub the offending traffic before it reached its intended destination. In almost all cases, the attacked endpoint isn't affected by the attack. In the rare cases that an endpoint is affected, no traffic is affected to other endpoints, only the attacked endpoint. Thus other customers and services would see no impact from that attack. It's critical to note that Azure DDoS is only looking for large-scale attacks. It is possible that your specific service could be overwhelmed before the platform level protection thresholds are exceeded. For example, a web site on a single A0 IIS server, could be taken offline by a DDoS attack before Azure platform level DDoS protection registered a threat.
- Public IP Addresses:** Public IP addresses (enabled via service endpoints, Public IP addresses, Application Gateway, and other Azure features that present a public IP address to the internet routed to your resource) allow cloud services or resource groups to have public Internet IP addresses and ports exposed. The endpoint uses Network Address Translation (NAT) to route traffic to the internal address and port on the Azure virtual network. This path is the primary way for external traffic to pass into the virtual network. The Public IP addresses are configurable to determine which traffic is passed in, and how and where it's translated on to the virtual network.

Once traffic reaches the virtual network, there are many features that come into play. Azure virtual networks are the foundation for customers to attach their workloads and where basic network-level security applies. It is a private network (a virtual network overlay) in Azure for customers with the following features and characteristics:

- Traffic isolation:** A virtual network is the traffic isolation boundary on the Azure platform. Virtual machines (VMs) in one virtual network cannot communicate directly to VMs in a different virtual network, even if both virtual networks are created by the same customer. Isolation is a critical property that ensures customer VMs and communication remains private within a virtual network.

NOTE

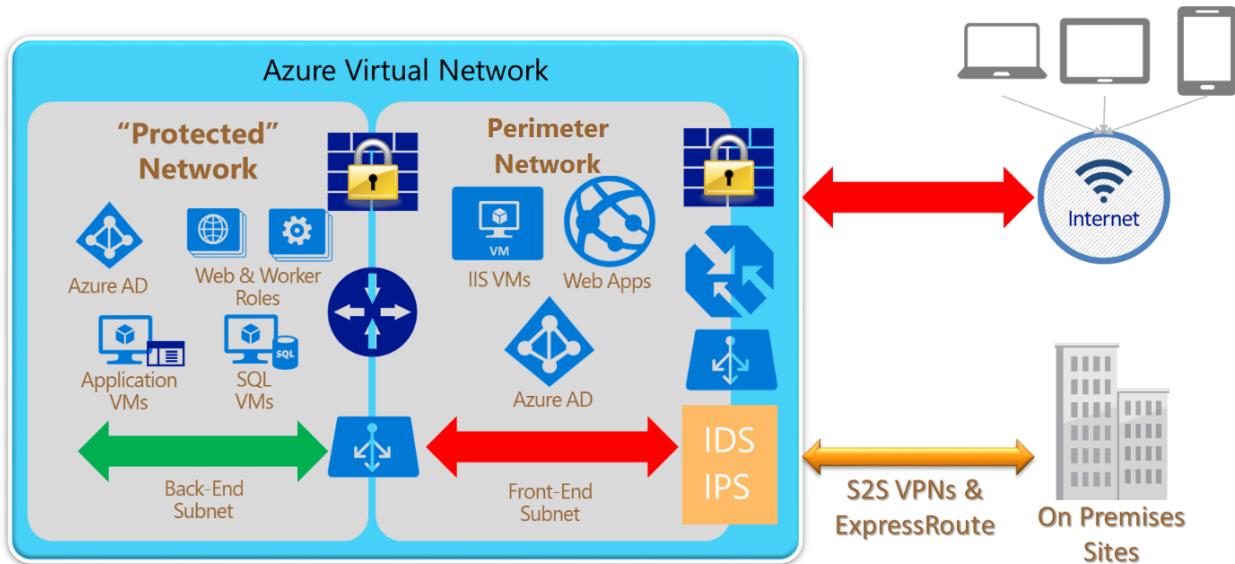
Traffic isolation refers only to traffic *inbound* to the virtual network. By default outbound traffic from the VNet to the internet is allowed, but can be prevented if desired by NSGs.

- Multi-tier topology:** Virtual networks allow customers to define multi-tier topology by allocating subnets

and designating separate address spaces for different elements or “tiers” of their workloads. These logical groupings and topologies enable customers to define different access policy based on the workload types, and also control traffic flows between the tiers.

- **Cross-premises connectivity:** Customers can establish cross-premises connectivity between a virtual network and multiple on-premises sites or other virtual networks in Azure. To construct a connection, customers can use VNet Peering, Azure VPN Gateways, third-party network virtual appliances, or ExpressRoute. Azure supports site-to-site (S2S) VPNs using standard IPsec/IKE protocols and ExpressRoute private connectivity.
- **NSG** allows customers to create rules (ACLs) at the desired level of granularity: network interfaces, individual VMs, or virtual subnets. Customers can control access by permitting or denying communication between the workloads within a virtual network, from systems on customer’s networks via cross-premises connectivity, or direct Internet communication.
- **UDR and IP Forwarding** allow customers to define the communication paths between different tiers within a virtual network. Customers can deploy a firewall, IDS/IPS, and other virtual appliances, and route network traffic through these security appliances for security boundary policy enforcement, auditing, and inspection.
- **Network virtual appliances** in the Azure Marketplace: Security appliances such as firewalls, load balancers, and IDS/IPS are available in the Azure Marketplace and the VM Image Gallery. Customers can deploy these appliances into their virtual networks, and specifically, at their security boundaries (including the perimeter network subnets) to complete a multi-tiered secure network environment.

With these features and capabilities, one example of how a perimeter network architecture could be constructed in Azure is the following diagram:



Perimeter network characteristics and requirements

The perimeter network is the front end of the network, directly interfacing communication from the Internet. The incoming packets should flow through the security appliances, such as the firewall, IDS, and IPS, before reaching the back-end servers. Internet-bound packets from the workloads can also flow through the security appliances in the perimeter network for policy enforcement, inspection, and auditing purposes, before leaving the network. Additionally, the perimeter network can host cross-premises VPN gateways between customer virtual networks and on-premises networks.

Perimeter network characteristics

Referencing the previous figure, some of the characteristics of a good perimeter network are as follows:

- Internet-facing:
 - The perimeter network subnet itself is Internet-facing, directly communicating with the Internet.

- Public IP addresses, VIPs, and/or service endpoints pass Internet traffic to the front-end network and devices.
- Inbound traffic from the Internet passes through security devices before other resources on the front-end network.
- If outbound security is enabled, traffic passes through security devices, as the final step, before passing to the Internet.
- Protected network:
 - There is no direct path from the Internet to the core infrastructure.
 - Channels to the core infrastructure must traverse through security devices such as NSGs, firewalls, or VPN devices.
 - Other devices must not bridge Internet and the core infrastructure.
 - Security devices on both the Internet-facing and the protected network facing boundaries of the perimeter network (for example, the two firewall icons shown in the previous figure) may actually be a single virtual appliance with differentiated rules or interfaces for each boundary. For example, one physical device, logically separated, handling the load for both boundaries of the perimeter network.
- Other common practices and constraints:
 - Workloads must not store business critical information.
 - Access and updates to perimeter network configurations and deployments are limited to only authorized administrators.

Perimeter network requirements

To enable these characteristics, follow these guidelines on virtual network requirements to implement a successful perimeter network:

- **Subnet architecture:** Specify the virtual network such that an entire subnet is dedicated as the perimeter network, separated from other subnets in the same virtual network. This separation ensures the traffic between the perimeter network and other internal or private subnet tiers flows through a firewall or IDS/IPS virtual appliance. User-defined routes on the boundary subnets are required to forward this traffic to the virtual appliance.
- **NSG:** The perimeter network subnet itself should be open to allow communication with the Internet, but that does not mean customers should be bypassing NSGs. Follow common security practices to minimize the network surfaces exposed to the Internet. Lock down the remote address ranges allowed to access the deployments or the specific application protocols and ports that are open. There may be circumstances, however, in which a complete lock-down is not possible. For example, if customers have an external website in Azure, the perimeter network should allow the incoming web requests from any public IP addresses, but should only open the web application ports: TCP on port 80 and/or TCP on port 443.
- **Routing table:** The perimeter network subnet itself should be able to communicate to the Internet directly, but should not allow direct communication to and from the back end or on-premises networks without going through a firewall or security appliance.
- **Security appliance configuration:** To route and inspect packets between the perimeter network and the rest of the protected networks, the security appliances such as firewall, IDS, and IPS devices may be multi-homed. They may have separate NICs for the perimeter network and the back-end subnets. The NICs in the perimeter network communicate directly to and from the Internet, with the corresponding NSGs and the perimeter network routing table. The NICs connecting to the back-end subnets have more restricted NSGs and routing tables of the corresponding back-end subnets.
- **Security appliance functionality:** The security appliances deployed in the perimeter network typically perform the following functionality:
 - Firewall: Enforcing firewall rules or access control policies for the incoming requests.
 - Threat detection and prevention: Detecting and mitigating malicious attacks from the Internet.
 - Auditing and logging: Maintaining detailed logs for auditing and analysis.

- Reverse proxy: Redirecting the incoming requests to the corresponding back-end servers. This redirection involves mapping and translating the destination addresses on the front-end devices, typically firewalls, to the back-end server addresses.
- Forward proxy: Providing NAT and performing auditing for communication initiated from within the virtual network to the Internet.
- Router: Forwarding incoming and cross-subnet traffic inside the virtual network.
- VPN device: Acting as the cross-premises VPN gateways for cross-premises VPN connectivity between customer on-premises networks and Azure virtual networks.
- VPN server: Accepting VPN clients connecting to Azure virtual networks.

TIP

Keep the following two groups separate: the individuals authorized to access the perimeter network security gear and the individuals authorized as application development, deployment, or operations administrators. Keeping these groups separate allows for a segregation of duties and prevents a single person from bypassing both applications security and network security controls.

Questions to be asked when building network boundaries

In this section, unless specifically mentioned, the term "networks" refers to private Azure virtual networks created by a subscription administrator. The term doesn't refer to the underlying physical networks within Azure.

Also, Azure virtual networks are often used to extend traditional on-premises networks. It is possible to incorporate either site-to-site or ExpressRoute hybrid networking solutions with perimeter network architectures. This hybrid link is an important consideration in building network security boundaries.

The following three questions are critical to answer when you're building a network with a perimeter network and multiple security boundaries.

1) How many boundaries are needed?

The first decision point is to decide how many security boundaries are needed in a given scenario:

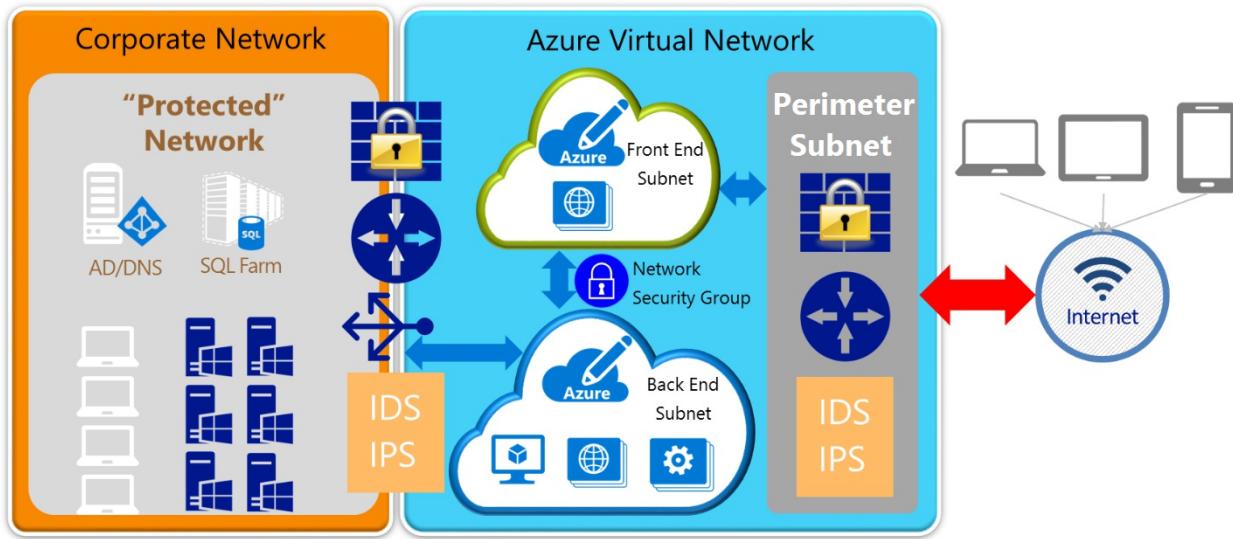
- A single boundary: One on the front-end perimeter network, between the virtual network and the Internet.
- Two boundaries: One on the Internet side of the perimeter network, and another between the perimeter network subnet and the back-end subnets in the Azure virtual networks.
- Three boundaries: One on the Internet side of the perimeter network, one between the perimeter network and back-end subnets, and one between the back-end subnets and the on-premises network.
- N boundaries: A variable number. Depending on security requirements, there is no limit to the number of security boundaries that can be applied in a given network.

The number and type of boundaries needed vary based on a company's risk tolerance and the specific scenario being implemented. This decision is often made together with multiple groups within an organization, often including a risk and compliance team, a network and platform team, and an application development team.

People with knowledge of security, the data involved, and the technologies being used should have a say in this decision to ensure the appropriate security stance for each implementation.

TIP

Use the smallest number of boundaries that satisfy the security requirements for a given situation. With more boundaries, operations and troubleshooting can be more difficult, as well as the management overhead involved with managing the multiple boundary policies over time. However, insufficient boundaries increase risk. Finding the balance is critical.



The preceding figure shows a high-level view of a three security boundary network. The boundaries are between the perimeter network and the Internet, the Azure front-end and back-end private subnets, and the Azure back-end subnet and the on-premises corporate network.

2) Where are the boundaries located?

Once the number of boundaries are decided, where to implement them is the next decision point. There are generally three choices:

- Using an Internet-based intermediary service (for example, a cloud-based Web application firewall, which is not discussed in this document)
- Using native features and/or network virtual appliances in Azure
- Using physical devices on the on-premises network

On purely Azure networks, the options are native Azure features (for example, Azure Load Balancers) or network virtual appliances from the rich partner ecosystem of Azure (for example, Check Point firewalls).

If a boundary is needed between Azure and an on-premises network, the security devices can reside on either side of the connection (or both sides). Thus a decision must be made on the location to place security gear.

In the previous figure, the Internet-to-perimeter network and the front-to-back-end boundaries are entirely contained within Azure, and must be either native Azure features or network virtual appliances. Security devices on the boundary between Azure (back-end subnet) and the corporate network could be either on the Azure side or the on-premises side, or even a combination of devices on both sides. There can be significant advantages and disadvantages to either option that must be seriously considered.

For example, using existing physical security gear on the on-premises network side has the advantage that no new gear is needed. It just needs reconfiguration. The disadvantage, however, is that all traffic must come back from Azure to the on-premises network to be seen by the security gear. Thus Azure-to-Azure traffic could incur significant latency, and affect application performance and user experience, if it was forced back to the on-premises network for security policy enforcement.

3) How are the boundaries implemented?

Each security boundary will likely have different capability requirements (for example, IDS and firewall rules on the Internet side of the perimeter network, but only ACLs between the perimeter network and back-end subnet). Deciding on which device (or how many devices) to use depends on the scenario and security requirements. In the following section, examples 1, 2, and 3 discuss some options that could be used. Reviewing the Azure native network features and the devices available in Azure from the partner ecosystem shows the myriad options available to solve virtually any scenario.

Another key implementation decision point is how to connect the on-premises network with Azure. Should you use the Azure virtual gateway or a network virtual appliance? These options are discussed in greater detail in the

following section (examples 4, 5, and 6).

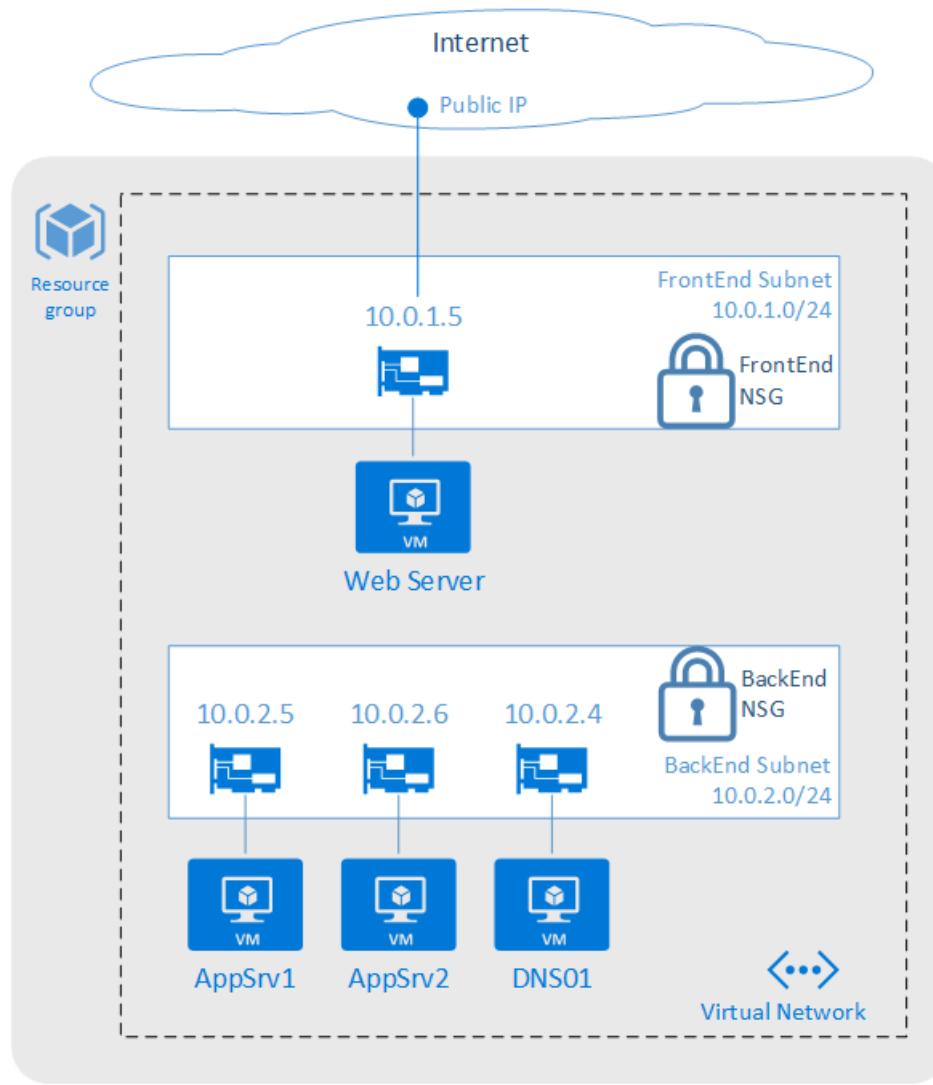
Additionally, traffic between virtual networks within Azure may be needed. These scenarios will be added in the future.

Once you know the answers to the previous questions, the [Fast Start](#) section can help identify which examples are most appropriate for a given scenario.

Examples: Building security boundaries with Azure virtual networks

Example 1 Build a perimeter network to help protect applications with NSGs

[Back to Fast start](#) | [Detailed build instructions for this example](#)



Environment description

In this example, there is a subscription that contains the following resources:

- A single resource group
- A virtual network with two subnets: "FrontEnd" and "BackEnd"
- A Network Security Group that is applied to both subnets
- A Windows server that represents an application web server ("IIS01")
- Two Windows servers that represent application back-end servers ("AppVM01", "AppVM02")
- A Windows server that represents a DNS server ("DNS01")
- A public IP associated with the application web server

For scripts and an Azure Resource Manager template, see the [detailed build instructions](#).

NSG description

In this example, an NSG group is built and then loaded with six rules.

TIP

Generally speaking, you should create your specific "Allow" rules first, followed by the more generic "Deny" rules. The given priority dictates which rules are evaluated first. Once traffic is found to apply to a specific rule, no further rules are evaluated. NSG rules can apply in either the inbound or outbound direction (from the perspective of the subnet).

Declaratively, the following rules are being built for inbound traffic:

1. Internal DNS traffic (port 53) is allowed.
2. RDP traffic (port 3389) from the Internet to any Virtual Machine is allowed.
3. HTTP traffic (port 80) from the Internet to web server (IIS01) is allowed.
4. Any traffic (all ports) from IIS01 to AppVM1 is allowed.
5. Any traffic (all ports) from the Internet to the entire virtual network (both subnets) is denied.
6. Any traffic (all ports) from the front-end subnet to the back-end subnet is denied.

With these rules bound to each subnet, if an HTTP request was inbound from the Internet to the web server, both rules 3 (allow) and 5 (deny) would apply. But because rule 3 has a higher priority, only it would apply, and rule 5 would not come into play. Thus the HTTP request would be allowed to the web server. If that same traffic was trying to reach the DNS01 server, rule 5 (deny) would be the first to apply, and the traffic would not be allowed to pass to the server. Rule 6 (deny) blocks the front-end subnet from talking to the back-end subnet (except for allowed traffic in rules 1 and 4). This rule-set protects the back-end network in case an attacker compromises the web application on the front end. The attacker would have limited access to the back-end "protected" network (only to resources exposed on the AppVM01 server).

There is a default outbound rule that allows traffic out to the Internet. For this example, we're allowing outbound traffic and not modifying any outbound rules. To lock down traffic in both directions, user-defined routing is required (see example 3).

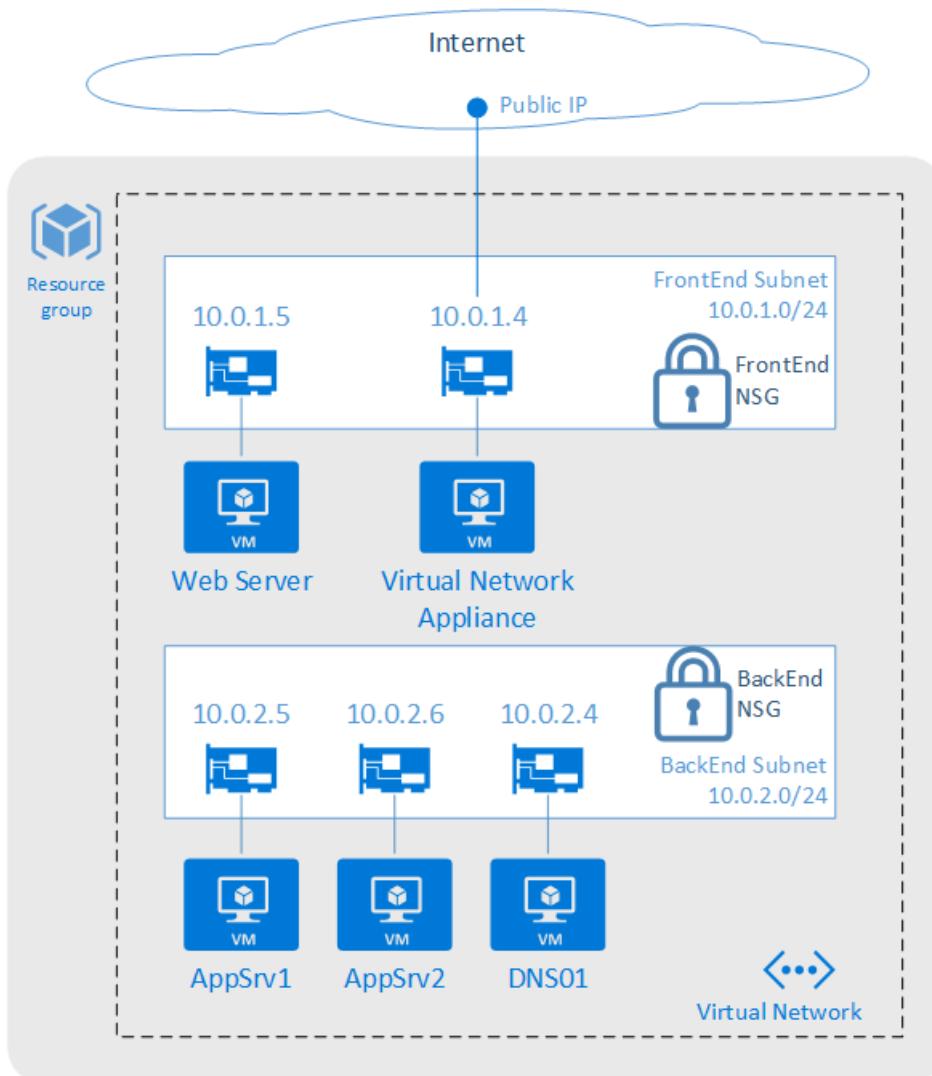
Conclusion

This example is a relatively simple and straightforward way of isolating the back-end subnet from inbound traffic. For more information, see the [detailed build instructions](#). These instructions include:

- How to build this perimeter network with classic PowerShell scripts.
- How to build this perimeter network with an Azure Resource Manager template.
- Detailed descriptions of each NSG command.
- Detailed traffic flow scenarios, showing how traffic is allowed or denied in each layer.

Example 2 Build a perimeter network to help protect applications with a firewall and NSGs

[Back to Fast start | Detailed build instructions for this example](#)



Environment description

In this example, there is a subscription that contains the following resources:

- A single resource group
- A virtual network with two subnets: "FrontEnd" and "BackEnd"
- A Network Security Group that is applied to both subnets
- A network virtual appliance, in this case a firewall, connected to the front-end subnet
- A Windows server that represents an application web server ("IIS01")
- Two Windows servers that represent application back-end servers ("AppVM01", "AppVM02")
- A Windows server that represents a DNS server ("DNS01")

For scripts and an Azure Resource Manager template, see the [detailed build instructions](#).

NSG description

In this example, an NSG group is built and then loaded with six rules.

TIP

Generally speaking, you should create your specific "Allow" rules first, followed by the more generic "Deny" rules. The given priority dictates which rules are evaluated first. Once traffic is found to apply to a specific rule, no further rules are evaluated. NSG rules can apply in either the inbound or outbound direction (from the perspective of the subnet).

Declaratively, the following rules are being built for inbound traffic:

1. Internal DNS traffic (port 53) is allowed.

2. RDP traffic (port 3389) from the Internet to any Virtual Machine is allowed.
3. Any Internet traffic (all ports) to the network virtual appliance (firewall) is allowed.
4. Any traffic (all ports) from IIS01 to AppVM1 is allowed.
5. Any traffic (all ports) from the Internet to the entire virtual network (both subnets) is denied.
6. Any traffic (all ports) from the front-end subnet to the back-end subnet is denied.

With these rules bound to each subnet, if an HTTP request was inbound from the Internet to the firewall, both rules 3 (allow) and 5 (deny) would apply. But because rule 3 has a higher priority, only it would apply, and rule 5 would not come into play. Thus the HTTP request would be allowed to the firewall. If that same traffic was trying to reach the IIS01 server, even though it's on the front-end subnet, rule 5 (deny) would apply, and the traffic would not be allowed to pass to the server. Rule 6 (deny) blocks the front-end subnet from talking to the back-end subnet (except for allowed traffic in rules 1 and 4). This rule-set protects the back-end network in case an attacker compromises the web application on the front end. The attacker would have limited access to the back-end "protected" network (only to resources exposed on the AppVM01 server).

There is a default outbound rule that allows traffic out to the Internet. For this example, we're allowing outbound traffic and not modifying any outbound rules. To lock down traffic in both directions, user-defined routing is required (see example 3).

Firewall rule description

On the firewall, forwarding rules should be created. Since this example only routes Internet traffic in-bound to the firewall and then to the web server, only one forwarding network address translation (NAT) rule is needed.

The forwarding rule accepts any inbound source address that hits the firewall trying to reach HTTP (port 80 or 443 for HTTPS). It's sent out of the firewall's local interface and redirected to the web server with the IP Address of 10.0.1.5.

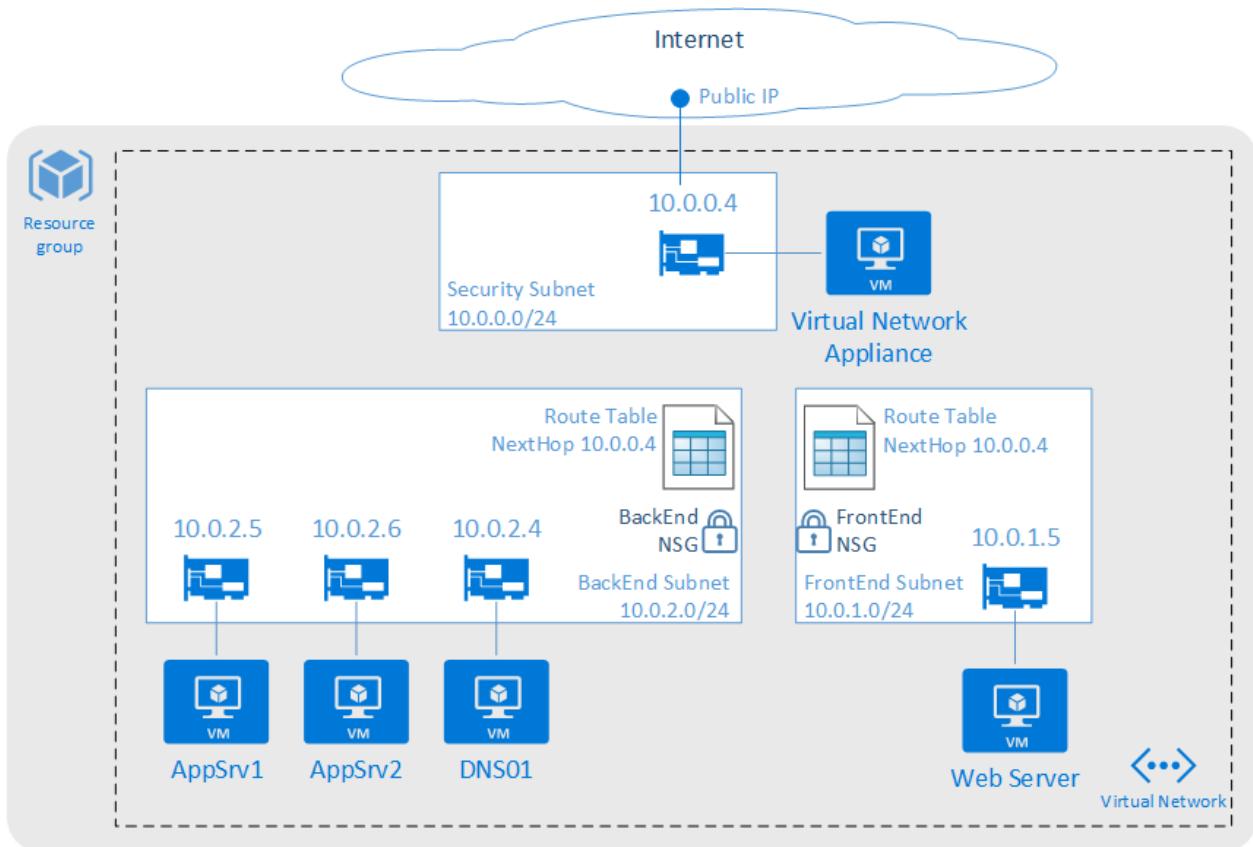
Conclusion

This example is a relatively straightforward way of protecting your application with a firewall and isolating the back-end subnet from inbound traffic. For more information, see the [detailed build instructions](#). These instructions include:

- How to build this perimeter network with classic PowerShell scripts.
- How to build this example with an Azure Resource Manager template.
- Detailed descriptions of each NSG command and firewall rule.
- Detailed traffic flow scenarios, showing how traffic is allowed or denied in each layer.

Example 3 Build a perimeter network to help protect networks with a firewall and UDR and NSG

[Back to Fast start | Detailed build instructions for this example](#)



Environment description

In this example, there is a subscription that contains the following resources:

- A single resource group
- A virtual network with three subnets: "SecNet", "FrontEnd", and "BackEnd"
- A network virtual appliance, in this case a firewall, connected to the SecNet subnet
- A Windows server that represents an application web server ("IIS01")
- Two Windows servers that represent application back-end servers ("AppVM01", "AppVM02")
- A Windows server that represents a DNS server ("DNS01")

For scripts and an Azure Resource Manager template, see the [detailed build instructions](#).

UDR description

By default, the following system routes are defined as:

Effective routes :					
Address Prefix	Next hop type	Next hop IP address	Status	Source	
{10.0.0.0/16}	VNETLocal		Active	Default	
{0.0.0.0/0}	Internet		Active	Default	
{10.0.0.0/8}	Null		Active	Default	
{100.64.0.0/10}	Null		Active	Default	
{172.16.0.0/12}	Null		Active	Default	
{192.168.0.0/16}	Null		Active	Default	

The VNETLocal is always one or more defined address prefixes that make up the virtual network for that specific network (that is, it changes from virtual network to virtual network, depending on how each specific virtual network is defined). The remaining system routes are static and default as indicated in the table.

In this example, two routing tables are created, one each for the front-end and back-end subnets. Each table is loaded with static routes appropriate for the given subnet. In this example, each table has three routes that direct all traffic (0.0.0.0/0) through the firewall (Next hop = Virtual Appliance IP address):

1. Local subnet traffic with no Next Hop defined to allow local subnet traffic to bypass the firewall.
2. Virtual network traffic with a Next Hop defined as firewall. This next hop overrides the default rule that allows local virtual network traffic to route directly.
3. All remaining traffic (0/0) with a Next Hop defined as the firewall.

TIP

Not having the local subnet entry in the UDR breaks local subnet communications.

- In our example, 10.0.1.0/24 pointing to VNETLocal is critical! Without it, packet leaving the Web Server (10.0.1.4) destined to another local server (for example) 10.0.1.25 will fail as they will be sent to the NVA. The NVA will send it to the subnet, and the subnet will resend it to the NVA in an infinite loop.
- The chances of a routing loop are typically higher on appliances with multiple NICs that are connected to separate subnets, which is often of traditional, on-premises appliances.

Once the routing tables are created, they must be bound to their subnets. The front-end subnet routing table, once created and bound to the subnet, would look like this output:

Effective routes :					
Address Prefix	Next hop type	Next hop IP address	Status	Source	
{10.0.1.0/24}	VNETLocal		Active		
{10.0.0.0/16}	VirtualAppliance	10.0.0.4	Active		
{0.0.0.0/0}	VirtualAppliance	10.0.0.4	Active		

NOTE

UDR can now be applied to the gateway subnet on which the ExpressRoute circuit is connected.

Examples of how to enable your perimeter network with ExpressRoute or site-to-site networking are shown in examples 3 and 4.

IP Forwarding description

IP Forwarding is a companion feature to UDR. IP Forwarding is a setting on a virtual appliance that allows it to receive traffic not specifically addressed to the appliance, and then forward that traffic to its ultimate destination.

For example, if AppVM01 makes a request to the DNS01 server, UDR would route this traffic to the firewall. With IP Forwarding enabled, the traffic for the DNS01 destination (10.0.2.4) is accepted by the appliance (10.0.0.4) and then forwarded to its ultimate destination (10.0.2.4). Without IP Forwarding enabled on the firewall, traffic would not be accepted by the appliance even though the route table has the firewall as the next hop. To use a virtual appliance, it's critical to remember to enable IP Forwarding along with UDR.

NSG description

In this example, an NSG group is built and then loaded with a single rule. This group is then bound only to the front-end and back-end subnets (not the SecNet). Declaratively the following rule is being built:

- Any traffic (all ports) from the Internet to the entire virtual network (all subnets) is denied.

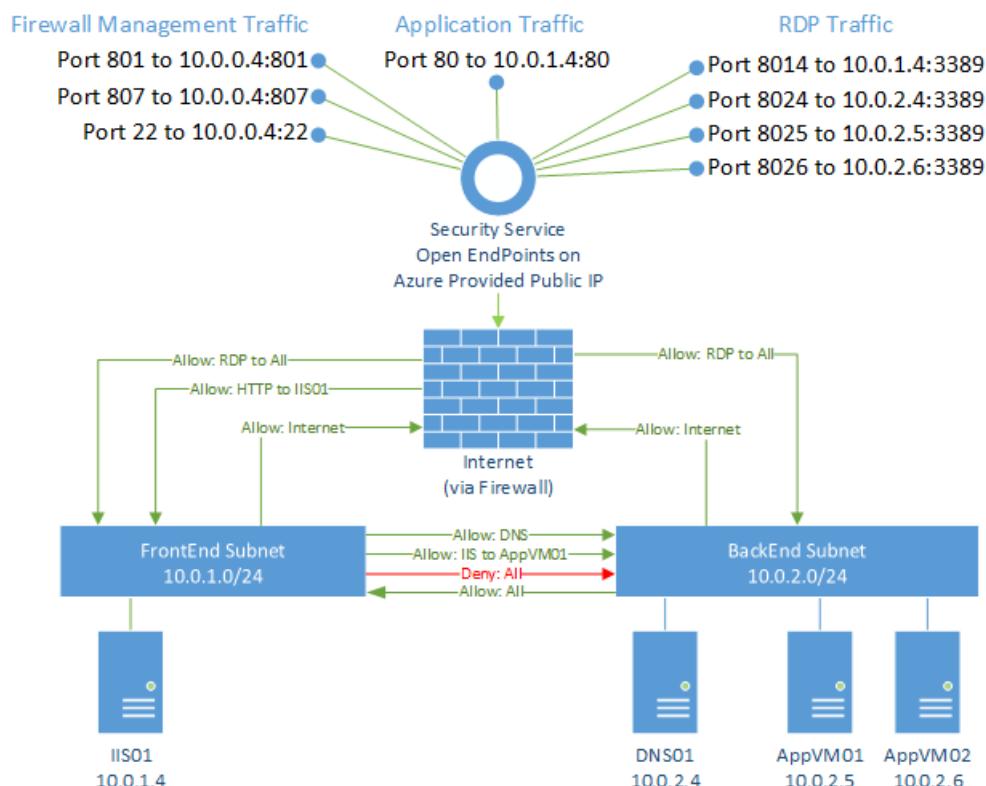
Although NSGs are used in this example, its main purpose is as a secondary layer of defense against manual misconfiguration. The goal is to block all inbound traffic from the Internet to either the front-end or back-end subnets. Traffic should only flow through the SecNet subnet to the firewall (and then, if appropriate, on to the front-end or back-end subnets). Plus, with the UDR rules in place, any traffic that did make it into the front-end or back-end subnets would be directed out to the firewall (thanks to UDR). The firewall would see this traffic as an asymmetric flow and would drop the outbound traffic. Thus there are three layers of security protecting the subnets:

- No Public IP addresses on any FrontEnd or BackEnd NICs.
- NSGs denying traffic from the Internet.
- The firewall dropping asymmetric traffic.

One interesting point regarding the NSG in this example is that it contains only one rule, which is to deny Internet traffic to the entire virtual network, including the Security subnet. However, since the NSG is only bound to the front-end and back-end subnets, the rule isn't processed on traffic inbound to the Security subnet. As a result, traffic flows to the Security subnet.

Firewall rules

On the firewall, forwarding rules should be created. Since the firewall is blocking or forwarding all inbound, outbound, and intra-virtual network traffic, many firewall rules are needed. Also, all inbound traffic hits the Security Service public IP address (on different ports), to be processed by the firewall. A best practice is to diagram the logical flows before setting up the subnets and firewall rules, to avoid rework later. The following figure is a logical view of the firewall rules for this example:



NOTE

Based on the Network Virtual Appliance used, the management ports vary. In this example, a Barracuda NextGen Firewall is referenced, which uses ports 22, 801, and 807. Consult the appliance vendor documentation to find the exact ports used for management of the device being used.

Firewall rules description

In the preceding logical diagram, the security subnet is not shown because the firewall is the only resource on that subnet. The diagram is showing the firewall rules and how they logically allow or deny traffic flows, not the actual routed path. Also, the external ports selected for the RDP traffic are higher ranged ports (8014 – 8026) and were selected to loosely align with the last two octets of the local IP address for easier readability (for example, local server address 10.0.1.4 is associated with external port 8014). Any higher non-conflicting ports, however, could be used.

For this example, we need seven types of rules:

- External rules (for inbound traffic):

1. Firewall management rule: This App Redirect rule allows traffic to pass to the management ports of the network virtual appliance.
 2. RDP rules (for each Windows server): These four rules (one for each server) allow management of the individual servers via RDP. The four RDP rules could also be collapsed into one rule, depending on the capabilities of the network virtual appliance being used.
 3. Application traffic rules: There are two of these rules, the first for the front-end web traffic, and the second for the back-end traffic (for example, web server to data tier). The configuration of these rules depends on the network architecture (where your servers are placed) and traffic flows (which direction the traffic flows, and which ports are used).
 - o The first rule allows the actual application traffic to reach the application server. While the other rules allow for security and management, application traffic rules are what allow external users or services to access the applications. For this example, there is a single web server on port 80. Thus a single firewall application rule redirects inbound traffic to the external IP, to the web servers internal IP address. The redirected traffic session would be translated via NAT to the internal server.
 - o The second rule is the back-end rule to allow the web server to talk to the AppVM01 server (but not AppVM02) via any port.
- Internal rules (for intra-virtual network traffic)
 1. Outbound to Internet rule: This rule allows traffic from any network to pass to the selected networks. This rule is usually a default rule already on the firewall, but in a disabled state. This rule should be enabled for this example.
 2. DNS rule: This rule allows only DNS (port 53) traffic to pass to the DNS server. For this environment, most traffic from the front end to the back end is blocked. This rule specifically allows DNS from any local subnet.
 3. Subnet to subnet rule: This rule is to allow any server on the back-end subnet to connect to any server on the front-end subnet (but not the reverse).
 - Fail-safe rule (for traffic that doesn't meet any of the previous):
 1. Deny all traffic rule: This deny rule should always be the final rule (in terms of priority), and as such if a traffic flow fails to match any of the preceding rules it is dropped by this rule. This rule is a default rule and usually in-place and active. No modifications are usually needed to this rule.

TIP

On the second application traffic rule, to simplify this example, any port is allowed. In a real scenario, the most specific port and address ranges should be used to reduce the attack surface of this rule.

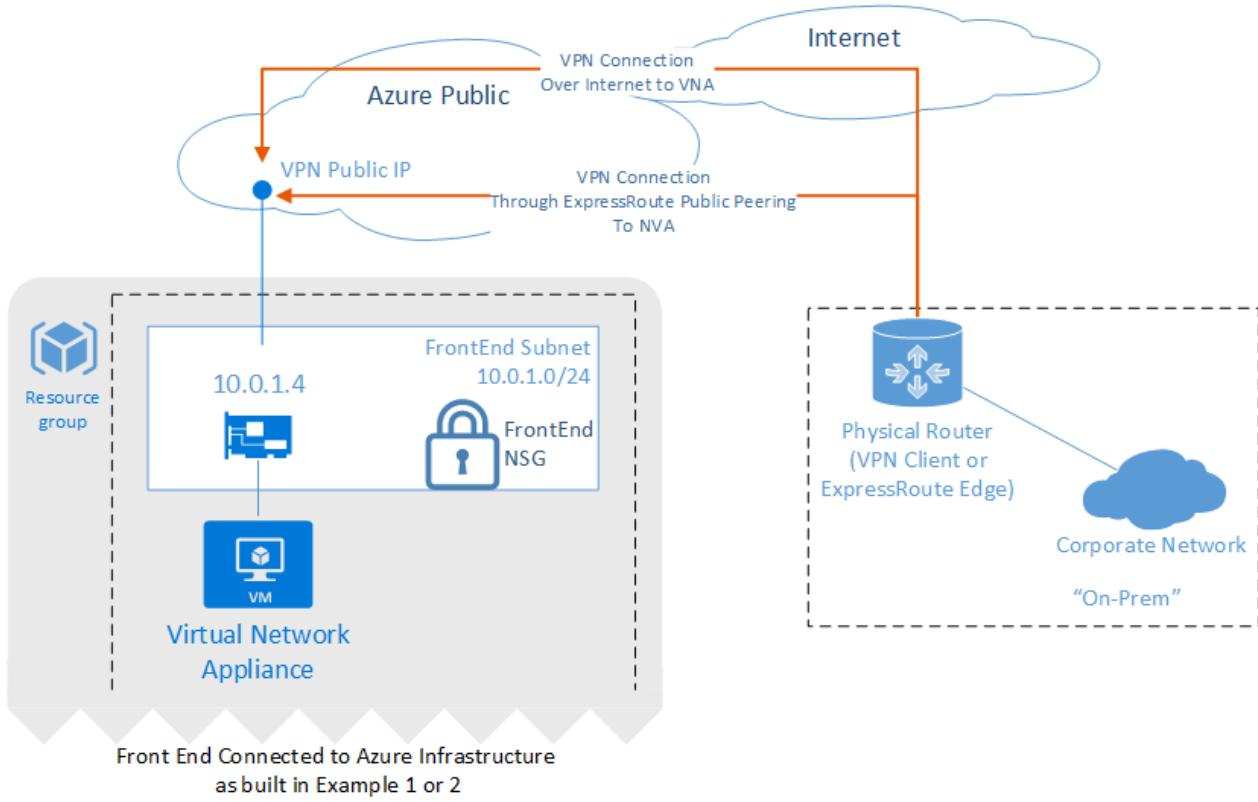
Once the previous rules are created, it's important to review the priority of each rule to ensure traffic is allowed or denied as desired. For this example, the rules are in priority order.

Conclusion

This example is a more complex but complete way of protecting and isolating the network than the previous examples. (Example 2 protects just the application, and Example 1 just isolates subnets). This design allows for monitoring traffic in both directions, and protects not just the inbound application server but enforces network security policy for all servers on this network. Also, depending on the appliance used, full traffic auditing and awareness can be achieved. For more information, see the [detailed build instructions](#). These instructions include:

- How to build this example perimeter network with classic PowerShell scripts.
- How to build this example with an Azure Resource Manager template.
- Detailed descriptions of each UDR, NSG command, and firewall rule.
- Detailed traffic flow scenarios, showing how traffic is allowed or denied in each layer.

Example 4 Add a hybrid connection with a site-to-site, virtual appliance VPN



Environment description

Hybrid networking using a network virtual appliance (NVA) can be added to any of the perimeter network types described in examples 1, 2, or 3.

As shown in the previous figure, a VPN connection over the Internet (site-to-site) is used to connect an on-premises network to an Azure virtual network via an NVA.

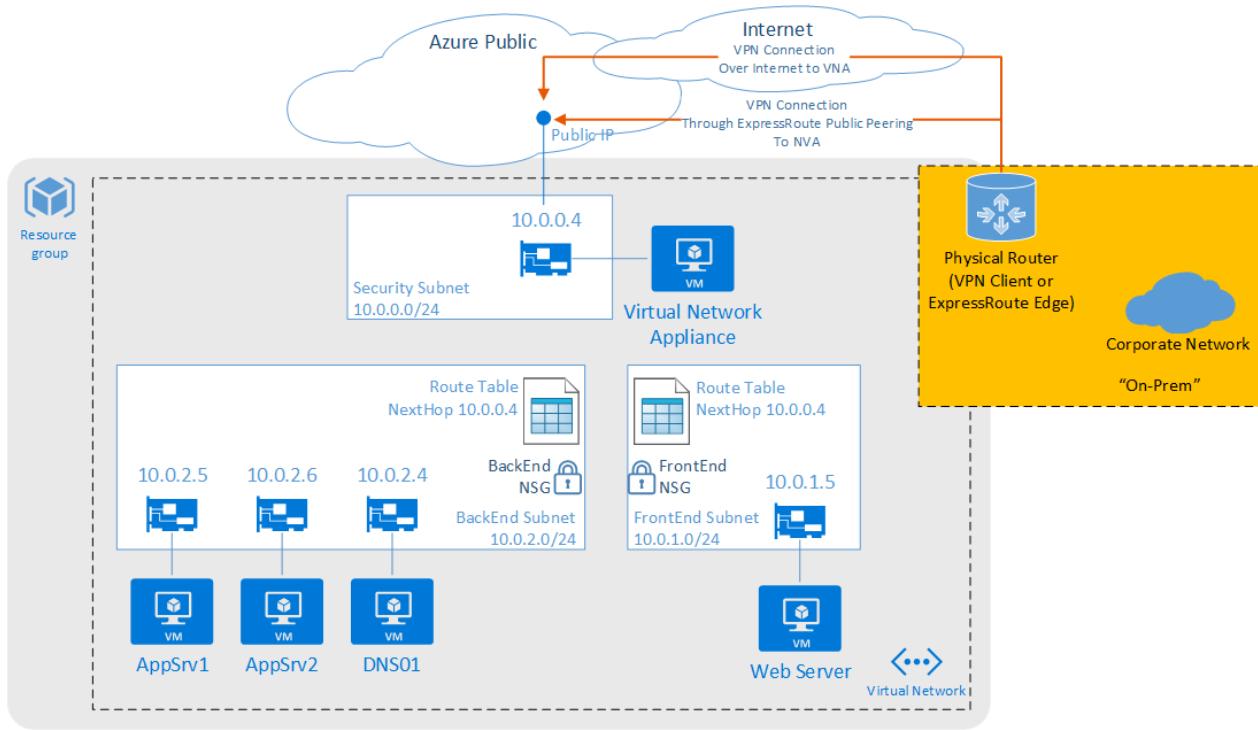
NOTE

If you use ExpressRoute with the Azure Public Peering option enabled, a static route should be created. This static route should route to the NVA VPN IP address out your corporate Internet and not via the ExpressRoute connection. The NAT required on the ExpressRoute Azure Public Peering option can break the VPN session.

Once the VPN is in place, the NVA becomes the central hub for all networks and subnets. The firewall forwarding rules determine which traffic flows are allowed, are translated via NAT, are redirected, or are dropped (even for traffic flows between the on-premises network and Azure).

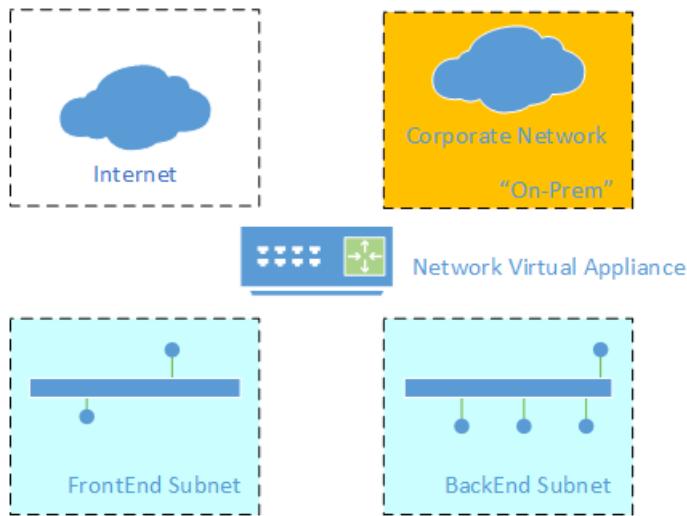
Traffic flows should be considered carefully, as they can be optimized or degraded by this design pattern, depending on the specific use case.

Using the environment built in example 3, and then adding a site-to-site VPN hybrid network connection, produces the following design:



The on-premises router, or any other network device that is compatible with your NVA for VPN, would be the VPN client. This physical device would be responsible for initiating and maintaining the VPN connection with your NVA.

Logically to the NVA, the network looks like four separate "security zones" with the rules on the NVA being the primary director of traffic between these zones:



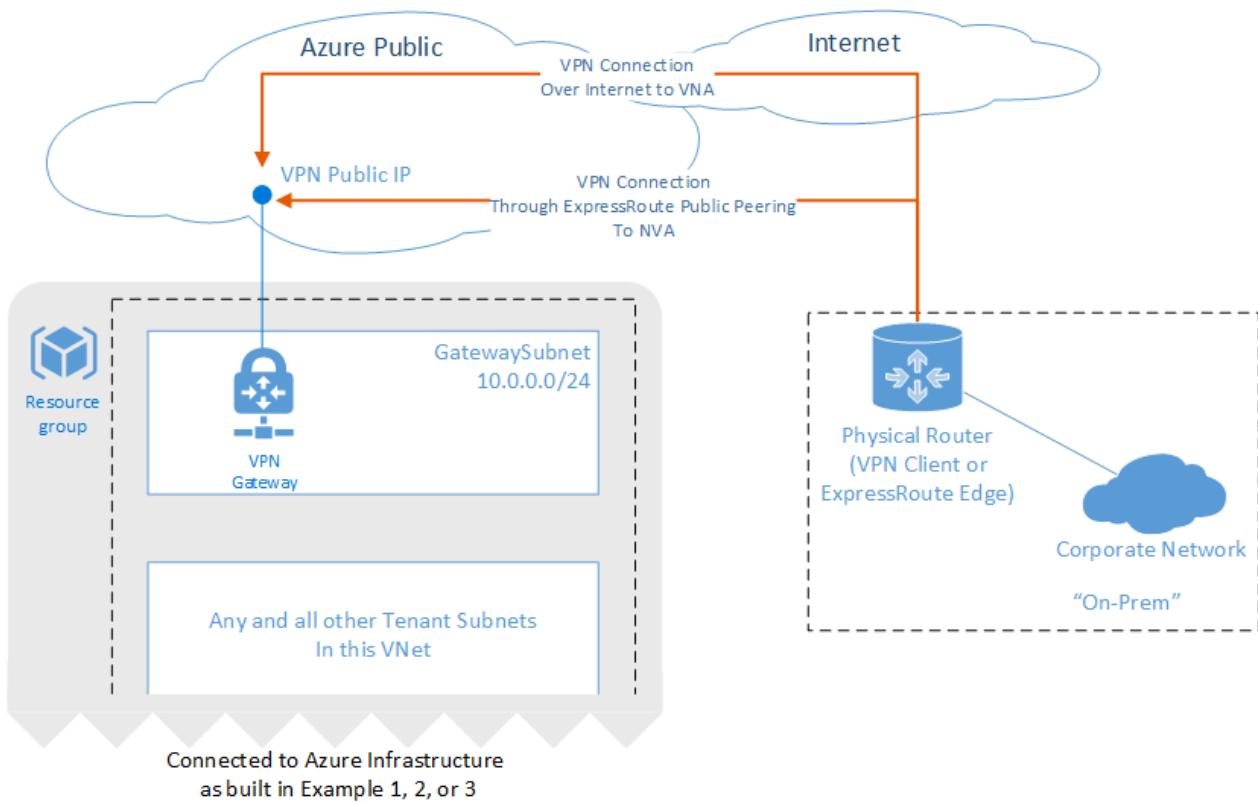
Conclusion

The addition of a site-to-site VPN hybrid network connection to an Azure virtual network can extend the on-premises network into Azure in a secure manner. In using a VPN connection, your traffic is encrypted and routes via the Internet. The NVA in this example provides a central location to enforce and manage the security policy. For more information, see the detailed build instructions (forthcoming). These instructions include:

- How to build this example perimeter network with PowerShell scripts.
- How to build this example with an Azure Resource Manager template.
- Detailed traffic flow scenarios, showing how traffic flows through this design.

Example 5 Add a hybrid connection with a site-to-site, Azure VPN gateway

[Back to Fast start](#) | Detailed build instructions available soon



Environment description

Hybrid networking using an Azure VPN gateway can be added to either perimeter network type described in examples 1 or 2.

As shown in the preceding figure, a VPN connection over the Internet (site-to-site) is used to connect an on-premises network to an Azure virtual network via an Azure VPN gateway.

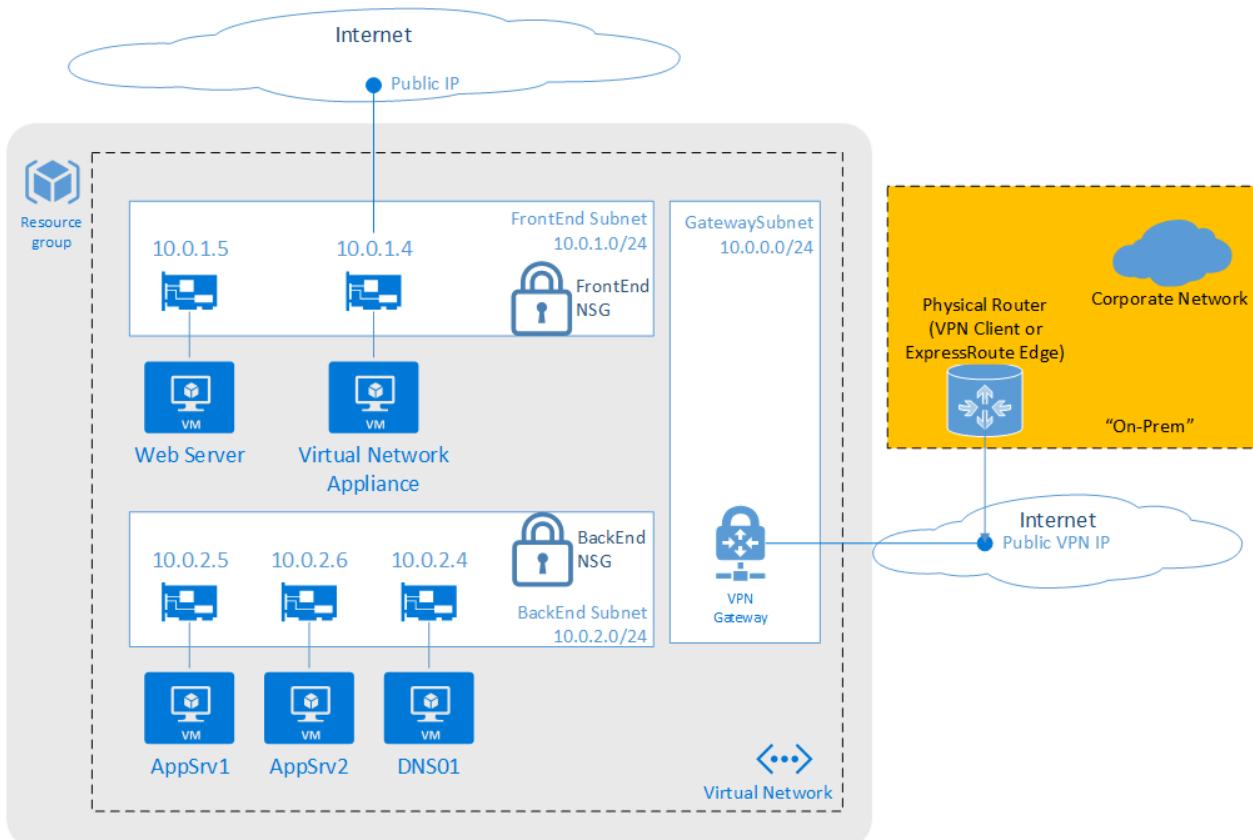
NOTE

If you use ExpressRoute with the Azure Public Peering option enabled, a static route should be created. This static route should route to the NVA VPN IP address out your corporate Internet and not via the ExpressRoute WAN. The NAT required on the ExpressRoute Azure Public Peering option can break the VPN session.

The following figure shows the two network edges in this example. On the first edge, the NVA and NSGs control traffic flows for intra-Azure networks and between Azure and the Internet. The second edge is the Azure VPN gateway, which is a separate and isolated network edge between on-premises and Azure.

Traffic flows should be considered carefully, as they can be optimized or degraded by this design pattern, depending on the specific use case.

Using the environment built in example 1, and then adding a site-to-site VPN hybrid network connection, produces the following design:



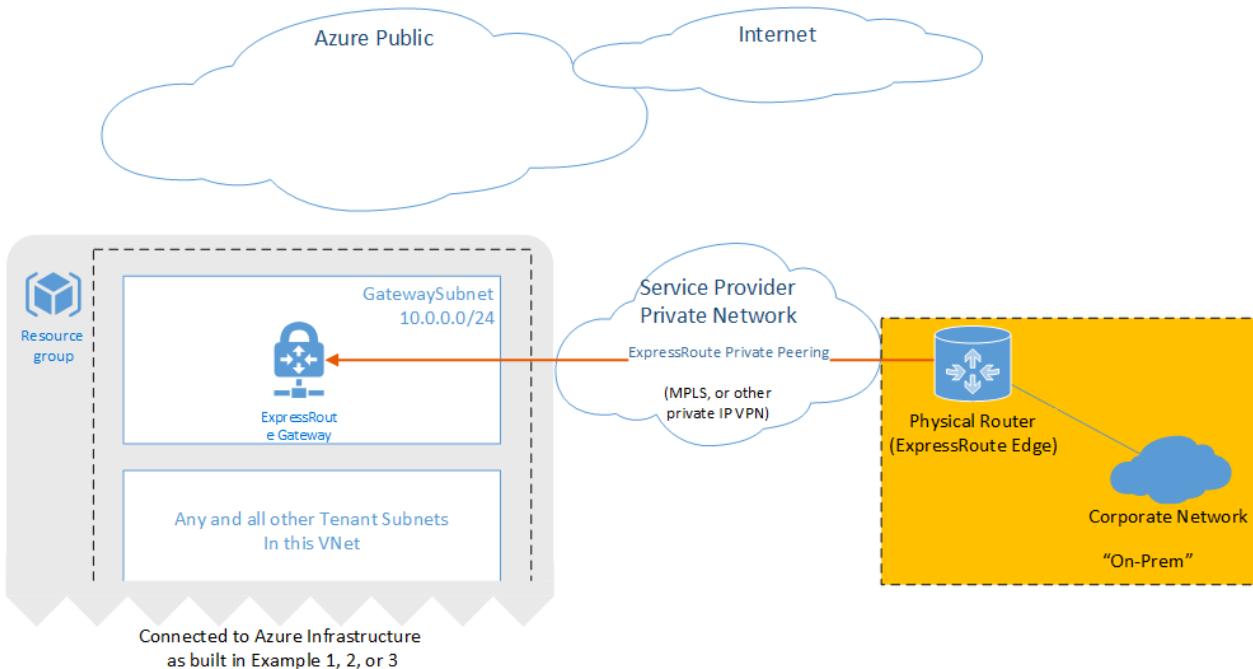
Conclusion

The addition of a site-to-site VPN hybrid network connection to an Azure virtual network can extend the on-premises network into Azure in a secure manner. Using the native Azure VPN gateway, your traffic is IPSec encrypted and routes via the Internet. Also, using the Azure VPN gateway can provide a lower-cost option (no additional licensing cost as with third-party NVAs). This option is most economical in example 1, where no NVA is used. For more information, see the detailed build instructions (forthcoming). These instructions include:

- How to build this example perimeter network with PowerShell scripts.
- How to build this example with an Azure Resource Manager template.
- Detailed traffic flow scenarios, showing how traffic flows through this design.

Example 6 Add a hybrid connection with ExpressRoute

[Back to Fast start](#) | Detailed build instructions available soon



Environment description

Hybrid networking using an ExpressRoute private peering connection can be added to either perimeter network type described in examples 1 or 2.

As shown in the preceding figure, ExpressRoute private peering provides a direct connection between your on-premises network and the Azure virtual network. Traffic transits only the service provider network and the Microsoft Azure network, never touching the Internet.

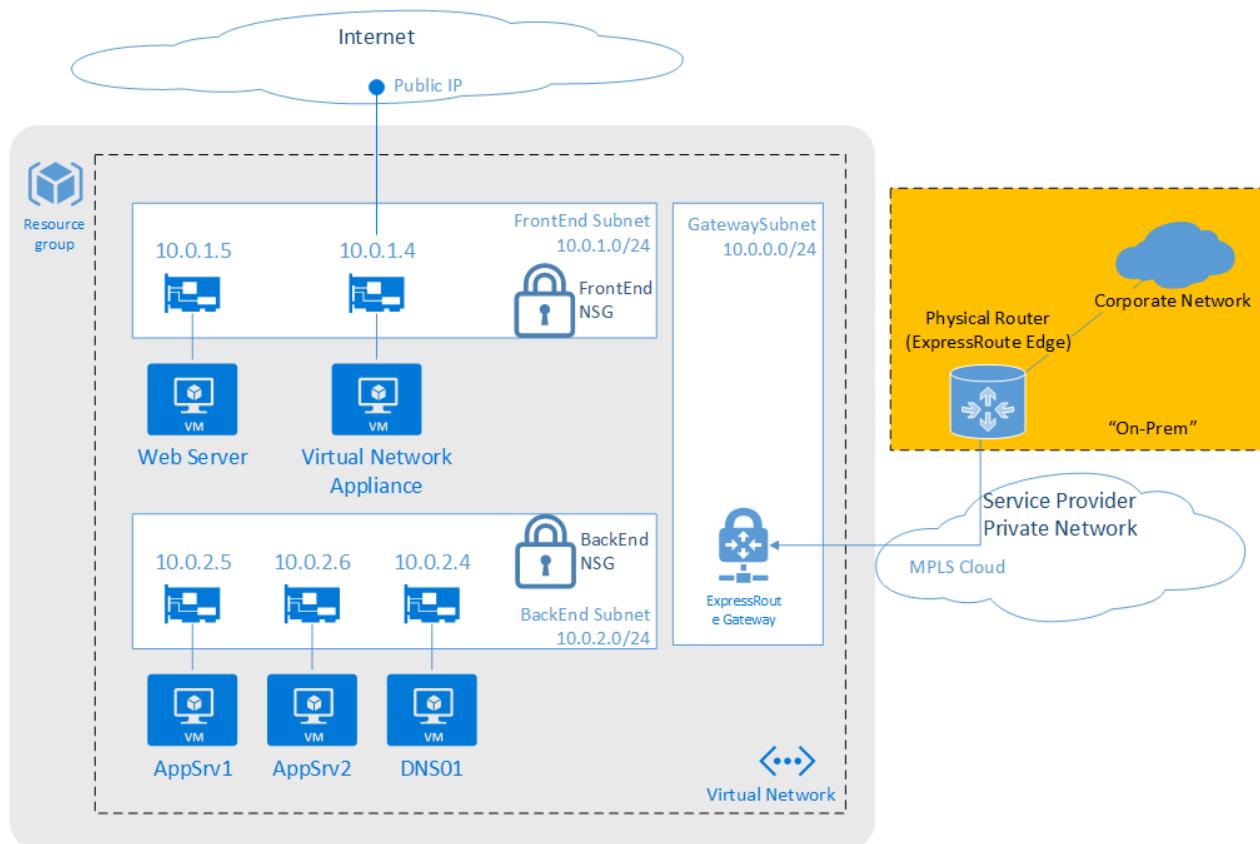
TIP

Using ExpressRoute keeps corporate network traffic off the Internet. It also allows for service level agreements from your ExpressRoute provider. The Azure Gateway can pass up to 10 Gbps with ExpressRoute, whereas with site-to-site VPNs, the Azure Gateway maximum throughput is 200 Mbps.

As seen in the following diagram, with this option the environment now has two network edges. The NVA and NSG control traffic flows for intra-Azure networks and between Azure and the Internet, while the gateway is a separate and isolated network edge between on-premises and Azure.

Traffic flows should be considered carefully, as they can be optimized or degraded by this design pattern, depending on the specific use case.

Using the environment built in example 1, and then adding an ExpressRoute hybrid network connection, produces the following design:



Conclusion

The addition of an ExpressRoute Private Peering network connection can extend the on-premises network into Azure in a secure, lower latency, higher performing manner. Also, using the native Azure Gateway, as in this example, provides a lower-cost option (no additional licensing as with third-party NVAs). For more information, see the detailed build instructions (forthcoming). These instructions include:

- How to build this example perimeter network with PowerShell scripts.
- How to build this example with an Azure Resource Manager template.

- Detailed traffic flow scenarios, showing how traffic flows through this design.

References

Helpful websites and documentation

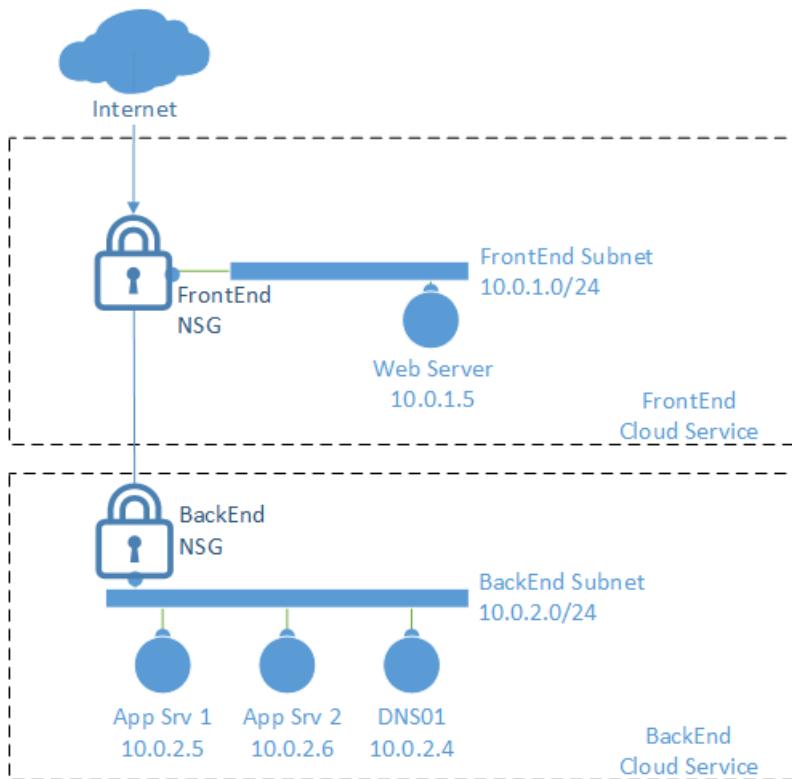
- Access Azure with Azure Resource Manager:
- Accessing Azure with PowerShell: <https://docs.microsoft.com/powershell/azureps-cmdlets-docs/>
- Virtual networking documentation: <https://docs.microsoft.com/azure/virtual-network/>
- Network security group documentation: <https://docs.microsoft.com/azure/virtual-network/virtual-networks-nsg>
- User-defined routing documentation: <https://docs.microsoft.com/azure/virtual-network/virtual-networks-udr-overview>
- Azure virtual gateways: <https://docs.microsoft.com/azure/vpn-gateway/>
- Site-to-Site VPNs: <https://docs.microsoft.com/azure/vpn-gateway/vpn-gateway-create-site-to-site-rm-powershell>
- ExpressRoute documentation (be sure to check out the "Getting Started" and "How To" sections):
<https://docs.microsoft.com/azure/expressroute/>

Example 1 – Build a simple DMZ using NSGs with classic PowerShell

6/27/2017 • 20 min to read • [Edit Online](#)

[Return to the Security Boundary Best Practices Page](#)

This example creates a primitive DMZ with four Windows servers and Network Security Groups. This example describes each of the relevant PowerShell commands to provide a deeper understanding of each step. There is also a Traffic Scenario section to provide an in-depth step-by-step how traffic proceeds through the layers of defense in the DMZ. Finally, in the references section is the complete code and instruction to build this environment to test and experiment with various scenarios.



Environment description

In this example a subscription contains the following resources:

- Two cloud services: "FrontEnd001" and "BackEnd001"
- A Virtual Network, "CorpNetwork", with two subnets; "FrontEnd" and "BackEnd"
- A Network Security Group that is applied to both subnets
- A Windows Server that represents an application web server ("IIS01")
- Two windows servers that represent application back-end servers ("AppVM01", "AppVM02")
- A Windows server that represents a DNS server ("DNS01")

In the references section, there is a PowerShell script that builds most of the environment described in this example. Building the VMs and Virtual Networks, although are done by the example script, are not described in detail in this document.

To build the environment;

1. Save the network config xml file included in the references section (updated with names, location, and IP addresses to match the given scenario)
2. Update the user variables in the script to match the environment the script is to be run against (subscriptions, service names, etc.)
3. Execute the script in PowerShell

NOTE

The region signified in the PowerShell script must match the region signified in the network configuration xml file.

Once the script runs successfully additional optional steps may be taken, in the references section are two scripts to set up the web server and app server with a simple web application to allow testing with this DMZ configuration.

The following sections provide a detailed description of Network Security Groups and how they function for this example by walking through key lines of the PowerShell script.

Network Security Groups (NSG)

For this example, an NSG group is built and then loaded with six rules.

TIP

Generally speaking, you should create your specific "Allow" rules first and then the more generic "Deny" rules last. The assigned priority dictates which rules are evaluated first. Once traffic is found to apply to a specific rule, no further rules are evaluated. NSG rules can apply in either in the inbound or outbound direction (from the perspective of the subnet).

Declaratively, the following rules are being built for inbound traffic:

1. Internal DNS traffic (port 53) is allowed
2. RDP traffic (port 3389) from the Internet to any VM is allowed
3. HTTP traffic (port 80) from the Internet to web server (IIS01) is allowed
4. Any traffic (all ports) from IIS01 to AppVM1 is allowed
5. Any traffic (all ports) from the Internet to the entire VNet (both subnets) is Denied
6. Any traffic (all ports) from the Frontend subnet to the Backend subnet is Denied

With these rules bound to each subnet, if an HTTP request was inbound from the Internet to the web server, both rules 3 (allow) and 5 (deny) would apply, but since rule 3 has a higher priority only it would apply and rule 5 would not come into play. Thus the HTTP request would be allowed to the web server. If that same traffic was trying to reach the DNS01 server, rule 5 (Deny) would be the first to apply and the traffic would not be allowed to pass to the server. Rule 6 (Deny) blocks the Frontend subnet from talking to the Backend subnet (except for allowed traffic in rules 1 and 4), this rule-set protects the Backend network in case an attacker compromises the web application on the Frontend, the attacker would have limited access to the Backend "protected" network (only to resources exposed on the AppVM01 server).

There is a default outbound rule that allows traffic out to the internet. For this example, we're allowing outbound traffic and not modifying any outbound rules. To lock down traffic in both directions, User Defined Routing is required and is explored in "Example 3" on the [Security Boundary Best Practices Page](#).

Each rule is discussed in more detail as follows (**Note**: any item in the following list beginning with a dollar sign (for example: \$NSGName) is a user-defined variable from the script in the reference section of this document):

1. First a Network Security Group must be built to hold the rules:

```

New-AzureNetworkSecurityGroup -Name $NSGName ` 
    -Location $DeploymentLocation ` 
    -Label "Security group for $VNetName subnets in $DeploymentLocation"

```

2. The first rule in this example allows DNS traffic between all internal networks to the DNS server on the backend subnet. The rule has some important parameters:

- “Type” signifies in which direction of traffic flow this rule takes effect. The direction is from the perspective of the subnet or Virtual Machine (depending on where this NSG is bound). Thus if Type is “Inbound” and traffic is entering the subnet, the rule would apply and traffic leaving the subnet would not be affected by this rule.
- “Priority” sets the order in which a traffic flow is evaluated. The lower the number the higher the priority. When a rule applies to a specific traffic flow, no further rules are processed. Thus if a rule with priority 1 allows traffic, and a rule with priority 2 denies traffic, and both rules apply to traffic then the traffic would be allowed to flow (since rule 1 had a higher priority it took effect and no further rules were applied).
- “Action” signifies if traffic affected by this rule is blocked or allowed.

```

Get-AzureNetworkSecurityGroup -Name $NSGName | ` 
    Set-AzureNetworkSecurityRule -Name "Enable Internal DNS" ` 
        -Type Inbound -Priority 100 -Action Allow ` 
        -SourceAddressPrefix VIRTUAL_NETWORK -SourcePortRange '*' ` 
        -DestinationAddressPrefix $VMIP[4] ` 
        -DestinationPortRange '53' ` 
        -Protocol *

```

3. This rule allows RDP traffic to flow from the internet to the RDP port on any server on the bound subnet. This rule uses two special types of address prefixes; “VIRTUAL_NETWORK” and “INTERNET”. These tags are an easy way to address a larger category of address prefixes.

```

Get-AzureNetworkSecurityGroup -Name $NSGName | ` 
    Set-AzureNetworkSecurityRule -Name "Enable RDP to $VNetName VNet" ` 
        -Type Inbound -Priority 110 -Action Allow ` 
        -SourceAddressPrefix INTERNET -SourcePortRange '*' ` 
        -DestinationAddressPrefix VIRTUAL_NETWORK ` 
        -DestinationPortRange '3389' ` 
        -Protocol *

```

4. This rule allows inbound internet traffic to hit the web server. This rule does not change the routing behavior. The rule only allows traffic destined for IIS01 to pass. Thus if traffic from the Internet had the web server as its destination this rule would allow it and stop processing further rules. (In the rule at priority 140 all other inbound internet traffic is blocked). If you’re only processing HTTP traffic, this rule could be further restricted to only allow Destination Port 80.

```

Get-AzureNetworkSecurityGroup -Name $NSGName | ` 
    Set-AzureNetworkSecurityRule -Name "Enable Internet to $VMName[0]" ` 
        -Type Inbound -Priority 120 -Action Allow ` 
        -SourceAddressPrefix Internet -SourcePortRange '*' ` 
        -DestinationAddressPrefix $VMIP[0] ` 
        -DestinationPortRange '*' ` 
        -Protocol *

```

5. This rule allows traffic to pass from the IIS01 server to the AppVM01 server, a later rule blocks all other Frontend to Backend traffic. To improve this rule, if the port is known that should be added. For example, if the IIS server is hitting only SQL Server on AppVM01, the Destination Port Range should be changed from “*” (Any) to 1433 (the SQL port) thus allowing a smaller inbound attack surface on AppVM01 should the

web application ever be compromised.

```
Get-AzureNetworkSecurityGroup -Name $NSGName | `  
Set-AzureNetworkSecurityRule -Name "Enable $VMName[1] to $VMName[2]" `  
-Type Inbound -Priority 130 -Action Allow `  
-SourceAddressPrefix $VMIP[1] -SourcePortRange '*' `  
-DestinationAddressPrefix $VMIP[2] `  
-DestinationPortRange '*' `  
-Protocol *
```

6. This rule denies traffic from the internet to any servers on the network. With the rules at priority 110 and 120, the effect is to allow only inbound internet traffic to the firewall and RDP ports on servers and blocks everything else. This rule is a "fail-safe" rule to block all unexpected flows.

```
Get-AzureNetworkSecurityGroup -Name $NSGName | `  
Set-AzureNetworkSecurityRule `  
-Name "Isolate the $VNetName VNet from the Internet" `  
-Type Inbound -Priority 140 -Action Deny `  
-SourceAddressPrefix INTERNET -SourcePortRange '*' `  
-DestinationAddressPrefix VIRTUAL_NETWORK `  
-DestinationPortRange '*' `  
-Protocol *
```

7. The final rule denies traffic from the Frontend subnet to the Backend subnet. Since this rule is an Inbound only rule, reverse traffic is allowed (from the Backend to the Frontend).

```
Get-AzureNetworkSecurityGroup -Name $NSGName | `  
Set-AzureNetworkSecurityRule `  
-Name "Isolate the $FESubnet subnet from the $BESubnet subnet" `  
-Type Inbound -Priority 150 -Action Deny `  
-SourceAddressPrefix $FEPPrefix -SourcePortRange '*' `  
-DestinationAddressPrefix $BEPrefix `  
-DestinationPortRange '*' `  
-Protocol *
```

Traffic scenarios

(Allowed) Internet to web server

1. An internet user requests an HTTP page from FrontEnd001.CloudApp.Net (Internet Facing Cloud Service)
2. Cloud service passes traffic through open endpoint on port 80 towards IIS01 (the web server)
3. Frontend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to IIS01) does apply, traffic is allowed, stop rule processing
4. Traffic hits internal IP address of the web server IIS01 (10.0.1.5)
5. IIS01 is listening for web traffic, receives this request and starts processing the request
6. IIS01 asks the SQL Server on AppVM01 for information
7. Since there are no outbound rules on Frontend subnet, traffic is allowed
8. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to Firewall) doesn't apply, move to next rule
 - d. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
9. AppVM01 receives the SQL Query and responds

10. Since there are no outbound rules on the Backend subnet, the response is allowed
11. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
12. The IIS server receives the SQL response and completes the HTTP response and sends to the requestor
13. Since there are no outbound rules on the Frontend subnet the response is allowed, and the internet User receives the web page requested.

(Allowed) RDP to backend

1. Server Admin on internet requests RDP session to AppVM01 on BackEnd001.CloudApp.Net:xxxxx where xxxx is the randomly assigned port number for RDP to AppVM01 (the assigned port can be found on the Azure portal or via PowerShell)
2. Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) does apply, traffic is allowed, stop rule processing
3. With no outbound rules, default rules apply and return traffic is allowed
4. RDP session is enabled
5. AppVM01 prompts for the user name and password

(Allowed) Web server DNS look-up on DNS server

1. Web Server, IIS01, needs a data feed at www.data.gov, but needs to resolve the address.
2. The network configuration for the VNet lists DNS01 (10.0.2.4 on the Backend subnet) as the primary DNS server, IIS01 sends the DNS request to DNS01
3. No outbound rules on Frontend subnet, traffic is allowed
4. Backend subnet begins inbound rule processing:
 - NSG Rule 1 (DNS) does apply, traffic is allowed, stop rule processing
5. DNS server receives the request
6. DNS server doesn't have the address cached and asks a root DNS server on the internet
7. No outbound rules on Backend subnet, traffic is allowed
8. Internet DNS server responds, since this session was initiated internally, the response is allowed
9. DNS server caches the response, and responds to the initial request back to IIS01
10. No outbound rules on Backend subnet, traffic is allowed
11. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed
12. IIS01 receives the response from DNS01

(Allowed) Web server access file on AppVM01

1. IIS01 asks for a file on AppVM01
2. No outbound rules on Frontend subnet, traffic is allowed
3. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to IIS01) doesn't apply, move to next rule
 - d. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
4. AppVM01 receives the request and responds with file (assuming access is authorized)
5. Since there are no outbound rules on the Backend subnet, the response is allowed

6. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
7. The IIS server receives the file

(Denied) Web to backend server

1. An internet user tries to access a file on AppVM01 through the BackEnd001.CloudApp.Net service
2. Since there are no endpoints open for file share, this traffic would not pass the Cloud Service and wouldn't reach the server
3. If the endpoints were open for some reason, NSG rule 5 (Internet to VNet) would block this traffic

(Denied) Web DNS look-up on DNS server

1. An internet user tries to look up an internal DNS record on DNS01 through the BackEnd001.CloudApp.Net service
2. Since there are no endpoints open for DNS, this traffic would not pass the Cloud Service and wouldn't reach the server
3. If the endpoints were open for some reason, NSG rule 5 (Internet to VNet) would block this traffic (Note: that Rule 1 (DNS) would not apply for two reasons, first the source address is the internet, this rule only applies to the local VNet as the source, also this rule is an Allow rule, so it would never deny traffic)

(Denied) Web to SQL access through firewall

1. An internet user requests SQL data from FrontEnd001.CloudApp.Net (Internet Facing Cloud Service)
2. Since there are no endpoints open for SQL, this traffic would not pass the Cloud Service and wouldn't reach the firewall
3. If endpoints were open for some reason, the Frontend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to IIS01) does apply, traffic is allowed, stop rule processing
4. Traffic hits internal IP address of the IIS01 (10.0.1.5)
5. IIS01 isn't listening on port 1433, so no response to the request

Conclusion

This example is a relatively simple and straight forward way of isolating the back-end subnet from inbound traffic.

More examples and an overview of network security boundaries can be found [here](#).

References

Main script and network config

Save the Full Script in a PowerShell script file. Save the Network Config into a file named "NetworkConf1.xml." Modify the user-defined variables as needed and run the script.

Full script

This script will, based on the user-defined variables;

1. Connect to an Azure subscription
2. Create a storage account
3. Create a VNet and two subnets as defined in the Network Config file
4. Build four windows server VMs
5. Configure NSG including:
 - Creating an NSG

- Populating it with rules
- Binding the NSG to the appropriate subnets

This PowerShell script should be run locally on an internet connected PC or server.

IMPORTANT

When this script is run, there may be warnings or other informational messages that pop in PowerShell. Only error messages in red are cause for concern.

```
<#
.SYNOPSIS
Example of Network Security Groups in an isolated network (Azure only, no hybrid connections)

.DESCRIPTION
This script will build out a sample DMZ setup containing:
- A default storage account for VM disks
- Two new cloud services
- Two Subnets (FrontEnd and BackEnd subnets)
- One server on the FrontEnd Subnet
- Three Servers on the BackEnd Subnet
- Network Security Groups to allow/deny traffic patterns as declared

Before running script, ensure the network configuration file is created in
the directory referenced by $NetworkConfigFile variable (or update the
variable to reflect the path and file name of the config file being used).

.Notes
Security requirements are different for each use case and can be addressed in a
myriad of ways. Please be sure that any sensitive data or applications are behind
the appropriate layer(s) of protection. This script serves as an example of some
of the techniques that can be used, but should not be used for all scenarios. You
are responsible to assess your security needs and the appropriate protections
needed, and then effectively implement those protections.

FrontEnd Service (FrontEnd subnet 10.0.1.0/24)
IIS01      - 10.0.1.5

BackEnd Service (BackEnd subnet 10.0.2.0/24)
DNS01      - 10.0.2.4
AppVM01    - 10.0.2.5
AppVM02    - 10.0.2.6

#>

# Fixed Variables
$LocalAdminPwd = Read-Host -Prompt "Enter Local Admin Password to be used for all VMs"
$VMName = @()
$ServiceName = @()
$VMFamily = @()
$img = @()
$size = @()
$SubnetName = @()
$VMIP = @()

# User-Defined Global Variables
# These should be changes to reflect your subscription and services
# Invalid options will fail in the validation section

# Subscription Access Details
$subID = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

# VM Account, Location, and Storage Details
$LocalAdmin = "theAdmin"
$DeploymentLocation = "Central US"
```

```

$StorageAccountName = "vmstore02"

# Service Details
$FrontEndService = "FrontEnd001"
$BackEndService = "BackEnd001"

# Network Details
$VNetName = "CorpNetwork"
$FESubnet = "FrontEnd"
$FEPrefix = "10.0.1.0/24"
$BESubnet = "BackEnd"
$BEPrefix = "10.0.2.0/24"
$NetworkConfigFile = "C:\Scripts\NetworkConf1.xml"

# VM Base Disk Image Details
$SrvImg = Get-AzureVMImage | Where {$_.ImageFamily -match 'Windows Server 2012 R2 Datacenter'} | sort PublishedDate -Descending | Select ImageName -First 1 | ForEach {$_.ImageName}

# NSG Details
$NSGName = "MyVNetSG"

# User-Defined VM Specific Configuration
# Note: To ensure proper NSG Rule creation later in this script:
#       - The Web Server must be VM 0
#       - The AppVM1 Server must be VM 1
#       - The DNS server must be VM 3
#
# Otherwise the NSG rules in the last section of this
# script will need to be changed to match the modified
# VM array numbers ($i) so the NSG Rule IP addresses
# are aligned to the associated VM IP addresses.

# VM 0 - The Web Server
$VMName += "IIS01"
$ServiceName += $FrontEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $FESubnet
$VMIP += "10.0.1.5"

# VM 1 - The First Application Server
$VMName += "AppVM01"
$ServiceName += $BackEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.5"

# VM 2 - The Second Application Server
$VMName += "AppVM02"
$ServiceName += $BackEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.6"

# VM 3 - The DNS Server
$VMName += "DNS01"
$ServiceName += $BackEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.4"

# -----

```

```

# No User-Defined Variables or #
# Configuration past this point #
# ----- #

# Get your Azure accounts
Add-AzureAccount
Set-AzureSubscription -SubscriptionId $subID -ErrorAction Stop
Select-AzureSubscription -SubscriptionId $subID -Current -ErrorAction Stop

# Create Storage Account
If (Test-AzureName -Storage -Name $StorageAccountName) {
    Write-Host "Fatal Error: This storage account name is already in use, please pick a different name." -ForegroundColor Red
    Return}
Else {Write-Host "Creating Storage Account" -ForegroundColor Cyan
New-AzureStorageAccount -Location $DeploymentLocation -StorageAccountName $StorageAccountName}

# Update Subscription Pointer to New Storage Account
Write-Host "Updating Subscription Pointer to New Storage Account" -ForegroundColor Cyan
Set-AzureSubscription -SubscriptionId $subID -CurrentStorageAccountName $StorageAccountName -ErrorAction Stop

# Validation
$FatalError = $false

If (-Not (Get-AzureLocation | Where {$_.DisplayName -eq $DeploymentLocation})) {
    Write-Host "This Azure Location was not found or available for use" -ForegroundColor Yellow
    $FatalError = $true}

If (Test-AzureName -Service -Name $FrontEndService) {
    Write-Host "The FrontEndService service name is already in use, please pick a different service name." -ForegroundColor Yellow
    $FatalError = $true}
Else { Write-Host "The FrontEndService service name is valid for use." -ForegroundColor Green}

If (Test-AzureName -Service -Name $BackEndService) {
    Write-Host "The BackEndService service name is already in use, please pick a different service name." -ForegroundColor Yellow
    $FatalError = $true}
Else { Write-Host "The BackEndService service name is valid for use." -ForegroundColor Green}

If (-Not (Test-Path $NetworkConfigFile)) {
    Write-Host 'The network config file was not found, please update the $NetworkConfigFile variable to point to the network config xml file.' -ForegroundColor Yellow
    $FatalError = $true}
Else { Write-Host "The network configuration file was found" -ForegroundColor Green
    If (-Not (Select-String -Pattern $DeploymentLocation -Path $NetworkConfigFile)) {
        Write-Host 'The deployment location was not found in the network config file, please check the network config file to ensure the $DeploymentLocation variable is correct and the network config file matches.' -ForegroundColor Yellow
        $FatalError = $true}
    Else { Write-Host "The deployment location was found in the network config file." -ForegroundColor Green
        $FatalError = $false}
    }

If ($FatalError) {
    Write-Host "A fatal error has occurred, please see the above messages for more information." -ForegroundColor Red
    Return}
Else { Write-Host "Validation passed, now building the environment." -ForegroundColor Green}

# Create VNET
Write-Host "Creating VNET" -ForegroundColor Cyan
Set-AzureVNetConfig -ConfigurationPath $NetworkConfigFile -ErrorAction Stop

# Create Services
Write-Host "Creating Services" -ForegroundColor Cyan
New-AzureService -Location $DeploymentLocation -ServiceName $FrontEndService -ErrorAction Stop
New-AzureService -Location $DeploymentLocation -ServiceName $BackEndService -ErrorAction Stop

```

```

# Build VMs
$i=0
$VMName | Foreach {
    Write-Host "Building $($VMName[$i])" -ForegroundColor Cyan
    New-AzureVMConfig -Name $VMName[$i] -ImageName $img[$i] -InstanceSize $size[$i] | ` 
        Add-AzureProvisioningConfig -Windows -AdminUsername $LocalAdmin -Password $LocalAdminPwd | ` 
        Set-AzureSubnet -SubnetNames $SubnetName[$i] | ` 
        Set-AzureStaticVNetIP -IPAddress $VMIP[$i] | ` 
        Set-AzureVMMicrosoftAntimalwareExtension -AntimalwareConfiguration '{"AntimalwareEnabled" : true}' 
    | ` 
        Remove-AzureEndpoint -Name "PowerShell" | ` 
        New-AzureVM -ServiceName $ServiceName[$i] -VNetName $VNetName -Location $DeploymentLocation
    # Note: A Remote Desktop endpoint is automatically created when each VM is created.
    $i++
}
# Add HTTP Endpoint for IIS01
Get-AzureVM -ServiceName $ServiceName[0] -Name $VMName[0] | Add-AzureEndpoint -Name HTTP -Protocol tcp - 
LocalPort 80 -PublicPort 80 | Update-AzureVM

# Configure NSG
Write-Host "Configuring the Network Security Group (NSG)" -ForegroundColor Cyan

# Build the NSG
Write-Host "Building the NSG" -ForegroundColor Cyan
New-AzureNetworkSecurityGroup -Name $NSGName -Location $DeploymentLocation -Label "Security group for 
$VNetName subnets in $DeploymentLocation"

# Add NSG Rules
Write-Host "Writing rules into the NSG" -ForegroundColor Cyan
Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable Internal DNS" - 
Type Inbound -Priority 100 -Action Allow ` 
    -SourceAddressPrefix VIRTUAL_NETWORK -SourcePortRange '*' ` 
    -DestinationAddressPrefix $VMIP[3] -DestinationPortRange '53' ` 
    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable RDP to $VNetName 
VNet" -Type Inbound -Priority 110 -Action Allow ` 
    -SourceAddressPrefix INTERNET -SourcePortRange '*' ` 
    -DestinationAddressPrefix VIRTUAL_NETWORK -DestinationPortRange '3389' ` 
    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable Internet to 
 $($VMName[0])" -Type Inbound -Priority 120 -Action Allow ` 
    -SourceAddressPrefix Internet -SourcePortRange '*' ` 
    -DestinationAddressPrefix $VMIP[0] -DestinationPortRange '*' ` 
    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable $($VMName[0]) to 
 $($VMName[1])" -Type Inbound -Priority 130 -Action Allow ` 
    -SourceAddressPrefix $VMIP[0] -SourcePortRange '*' ` 
    -DestinationAddressPrefix $VMIP[1] -DestinationPortRange '*' ` 
    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Isolate the $VNetName 
VNet from the Internet" -Type Inbound -Priority 140 -Action Deny ` 
    -SourceAddressPrefix INTERNET -SourcePortRange '*' ` 
    -DestinationAddressPrefix VIRTUAL_NETWORK -DestinationPortRange '*' ` 
    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Isolate the $FESubnet 
subnet from the $BESubnet subnet" -Type Inbound -Priority 150 -Action Deny ` 
    -SourceAddressPrefix $FEPrefix -SourcePortRange '*' ` 
    -DestinationAddressPrefix $BEPrefix -DestinationPortRange '*' ` 
    -Protocol *

# Assign the NSG to the Subnets
Write-Host "Binding the NSG to both subnets" -ForegroundColor Cyan
Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName -SubnetName $FESubnet -VirtualNetworkName

```

```

$VNetName
    Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName -SubnetName $BESubnet -VirtualNetworkName
$VNetName

# Optional Post-script Manual Configuration
# Install Test Web App (Run Post-Build Script on the IIS Server)
# Install Backend resource (Run Post-Build Script on the AppVM01)
Write-Host
Write-Host "Build Complete!" -ForegroundColor Green
Write-Host
Write-Host "Optional Post-script Manual Configuration Steps" -ForegroundColor Gray
Write-Host "- Install Test Web App (Run Post-Build Script on the IIS Server)" -ForegroundColor Gray
Write-Host "- Install Backend resource (Run Post-Build Script on the AppVM01)" -ForegroundColor Gray
Write-Host

```

Network config file

Save this xml file with updated location and add the link to this file to the \$NetworkConfigFile variable in the preceding script.

```

<NetworkConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://schemas.microsoft.com/ServiceHosting/2011/07/NetworkConfiguration">
    <VirtualNetworkConfiguration>
        <Dns>
            <DnsServers>
                <DnsServer name="DNS01" IPAddress="10.0.2.4" />
                <DnsServer name="Level3" IPAddress="209.244.0.3" />
            </DnsServers>
        </Dns>
        <VirtualNetworkSites>
            <VirtualNetworkSite name="CorpNetwork" Location="Central US">
                <AddressSpace>
                    <AddressPrefix>10.0.0.0/16</AddressPrefix>
                </AddressSpace>
                <Subnets>
                    <Subnet name="FrontEnd">
                        <AddressPrefix>10.0.1.0/24</AddressPrefix>
                    </Subnet>
                    <Subnet name="BackEnd">
                        <AddressPrefix>10.0.2.0/24</AddressPrefix>
                    </Subnet>
                </Subnets>
            </VirtualNetworkSite>
        </VirtualNetworkSites>
    </VirtualNetworkConfiguration>
</NetworkConfiguration>

```

Sample application scripts

If you wish to install a sample application for this, and other DMZ Examples, one has been provided at the following link: [Sample Application Script](#)

Next steps

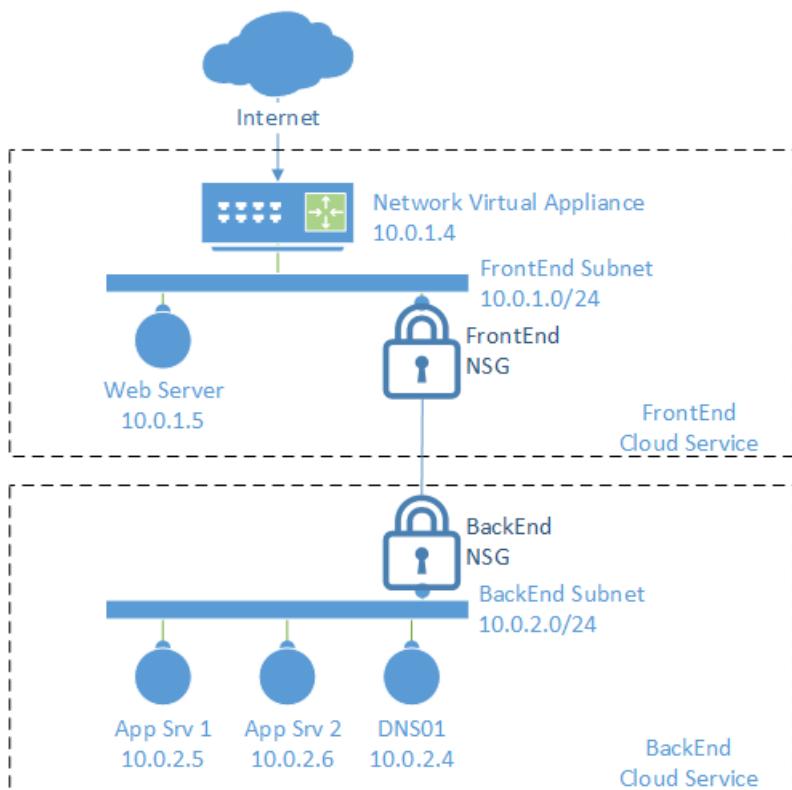
- Update and save XML file
- Run the PowerShell script to build the environment
- Install the sample application
- Test different traffic flows through this DMZ

Example 2 – Build a DMZ to protect applications with a Firewall and NSGs

6/27/2017 • 22 min to read • [Edit Online](#)

[Return to the Security Boundary Best Practices Page](#)

This example will create a DMZ with a firewall, four windows servers, and Network Security Groups. It will also walk through each of the relevant commands to provide a deeper understanding of each step. There is also a Traffic Scenario section to provide an in-depth step-by-step how traffic proceeds through the layers of defense in the DMZ. Finally, in the references section is the complete code and instruction to build this environment to test and experiment with various scenarios.



Environment Description

In this example there is a subscription that contains the following:

- Two cloud services: "FrontEnd001" and "BackEnd001"
- A Virtual Network "CorpNetwork", with two subnets: "FrontEnd" and "BackEnd"
- A single Network Security Group that is applied to both subnets
- A network virtual appliance, in this example a Barracuda NextGen Firewall, connected to the Frontend subnet
- A Windows Server that represents an application web server ("IIS01")
- Two windows servers that represent application back end servers ("AppVM01", "AppVM02")
- A Windows server that represents a DNS server ("DNS01")

NOTE

Although this example uses a Barracuda NextGen Firewall, many of the different Network Virtual Appliances could be used for this example.

In the references section below there is a PowerShell script that will build most of the environment described above. Building the VMs and Virtual Networks, although are done by the example script, are not described in detail in this document.

To build the environment:

1. Save the network config xml file included in the references section (updated with names, location, and IP addresses to match the given scenario)
2. Update the user variables in the script to match the environment the script is to be run against (subscriptions, service names, etc)
3. Execute the script in PowerShell

Note: The region signified in the PowerShell script must match the region signified in the network configuration xml file.

Once the script runs successfully the following post-script steps may be taken:

1. Set up the firewall rules, this is covered in the section below titled: Firewall Rules.
2. Optionally in the references section are two scripts to set up the web server and app server with a simple web application to allow testing with this DMZ configuration.

The next section explains most of the scripts statements relative to Network Security Groups.

Network Security Groups (NSG)

For this example, a NSG group is built and then loaded with six rules.

TIP

Generally speaking, you should create your specific "Allow" rules first and then the more generic "Deny" rules last. The assigned priority dictates which rules are evaluated first. Once traffic is found to apply to a specific rule, no further rules are evaluated. NSG rules can apply in either in the inbound or outbound direction (from the perspective of the subnet).

Declaratively, the following rules are being built for inbound traffic:

1. Internal DNS traffic (port 53) is allowed
2. RDP traffic (port 3389) from the Internet to any VM is allowed
3. HTTP traffic (port 80) from the Internet to the NVA (firewall) is allowed
4. Any traffic (all ports) from IIS01 to AppVM1 is allowed
5. Any traffic (all ports) from the Internet to the entire VNet (both subnets) is Denied
6. Any traffic (all ports) from the Frontend subnet to the Backend subnet is Denied

With these rules bound to each subnet, if a HTTP request was inbound from the Internet to the web server, both rules 3 (allow) and 5 (deny) would apply, but since rule 3 has a higher priority only it would apply and rule 5 would not come into play. Thus the HTTP request would be allowed to the firewall. If that same traffic was trying to reach the DNS01 server, rule 5 (Deny) would be the first to apply and the traffic would not be allowed to pass to the server. Rule 6 (Deny) blocks the Frontend subnet from talking to the Backend subnet (except for allowed traffic in rules 1 and 4), this protects the Backend network in case an attacker compromises the web application on the Frontend, the attacker would have limited access to the Backend "protected" network (only to resources exposed on

the AppVM01 server).

There is a default outbound rule that allows traffic out to the internet. For this example, we're allowing outbound traffic and not modifying any outbound rules. To lock down traffic in both directions, User Defined Routing is required, this is explored in a different example that can found in the [main security boundary document](#).

The above discussed NSG rules are very similar to the NSG rules in [Example 1 - Build a Simple DMZ with NSGs](#). Please review the NSG Description in that document for a detailed look at each NSG rule and its attributes.

Firewall Rules

A management client will need to be installed on a PC to manage the firewall and create the configurations needed. See the documentation from your firewall (or other NVA) vendor on how to manage the device. The remainder of this section will describe the configuration of the firewall itself, through the vendors management client (i.e. not the Azure portal or PowerShell).

Instructions for client download and connecting to the Barracuda used in this example can be found here:

[Barracuda NG Admin](#)

On the firewall, forwarding rules will need to be created. Since this example only routes internet traffic in-bound to the firewall and then to the web server, only one forwarding NAT rule is needed. On the Barracuda NextGen Firewall used in this example the rule would be a Destination NAT rule ("Dst NAT") to pass this traffic.

To create the following rule (or verify existing default rules), starting from the Barracuda NG Admin client dashboard, navigate to the configuration tab, in the Operational Configuration section click Ruleset. A grid called, "Main Rules" will show the existing active and deactivated rules on the firewall. In the upper right corner of this grid is a small, green "+" button, click this to create a new rule (Note: your firewall may be "locked" for changes, if you see a button marked "Lock" and you are unable to create or edit rules, click this button to "unlock" the ruleset and allow editing). If you wish to edit an existing rule, select that rule, right-click and select Edit Rule.

Create a new rule and provide a name, such as "WebTraffic".



The Destination NAT rule icon looks like this:

The rule itself would look something like this:

Source	Service	Destination
Any 0.0.0.0/0	HTTP +S Ref: HTTP Ref: HTTPS	DHCP1 Local IP Redirection Target List Reference <input type="checkbox"/> 10.0.1.5 Fallback List of Critical Ports

Here any inbound address that hits the Firewall trying to reach HTTP (port 80 or 443 for HTTPS) will be sent out the Firewall's "DHCP1 Local IP" interface and redirected to the Web Server with the IP Address of 10.0.1.5. Since the traffic is coming in on port 80 and going to the web server on port 80 no port change was needed. However, the Target List could have been 10.0.1.5:8080 if our Web Server listened on port 8080 thus translating the inbound port 80 on the firewall to inbound port 8080 on the web server.

A Connection Method should also be signified, for the Destination Rule from the Internet, "Dynamic SNAT" is most appropriate.

Although only one rule has been created it's important that its priority is set correctly. If in the grid of all rules on

the firewall this new rule is on the bottom (below the "BLOCKALL" rule) it will never come into play. Ensure the newly created rule for web traffic is above the BLOCKALL rule.

Once the rule is created, it must be pushed to the firewall and then activated, if this is not done the rule change will not take effect. The push and activation process is described in the next section.

Rule Activation

With the ruleset modified to add this rule, the ruleset must be uploaded to the firewall and activated.



In the upper right hand corner of the management client are a cluster of buttons. Click the "Send Changes" button to send the modified rules to the firewall, then click the "Activate" button.

With the activation of the firewall ruleset this example environment build is complete. Optionally, the post build scripts in the References section can be run to add an application to this environment to test the below traffic scenarios.

IMPORTANT

It is critical to realize that you will not hit the web server directly. When a browser requests an HTTP page from FrontEnd001.CloudApp.Net, the HTTP endpoint (port 80) passes this traffic to the firewall not the web server. The firewall then, according to the rule created above, NATs that request to the Web Server.

Traffic Scenarios

(Allowed) Web to Web Server through Firewall

1. Internet user requests HTTP page from FrontEnd001.CloudApp.Net (Internet Facing Cloud Service)
2. Cloud service passes traffic through open endpoint on port 80 to firewall local interface on 10.0.1.4:80
3. Frontend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to Firewall) does apply, traffic is allowed, stop rule processing
4. Traffic hits internal IP address of the firewall (10.0.1.4)
5. Firewall forwarding rule sees this is port 80 traffic, redirects it to the web server IIS01
6. IIS01 is listening for web traffic, receives this request and starts processing the request
7. IIS01 asks the SQL Server on AppVM01 for information
8. No outbound rules on Frontend subnet, traffic is allowed
9. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to Firewall) doesn't apply, move to next rule
 - d. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
10. AppVM01 receives the SQL Query and responds
11. Since there are no outbound rules on the Backend subnet the response is allowed
12. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply

- b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
- 13. The IIS server receives the SQL response and completes the HTTP response and sends to the requestor
- 14. Since this is a NAT session from the firewall, the response destination (initially) is for the Firewall
- 15. The firewall receives the response from the Web Server and forwards back to the Internet User
- 16. Since there are no outbound rules on the Frontend subnet the response is allowed, and the Internet User receives the web page requested.

(Allowed) RDP to Backend

1. Server Admin on internet requests RDP session to AppVM01 on BackEnd001.CloudApp.Net:xxxxx where xxxx is the randomly assigned port number for RDP to AppVM01 (the assigned port can be found on the Azure Portal or via PowerShell)
2. Since the Firewall is only listening on the FrontEnd001.CloudApp.Net address, it is not involved with this traffic flow
3. Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) does apply, traffic is allowed, stop rule processing
4. With no outbound rules, default rules apply and return traffic is allowed
5. RDP session is enabled
6. AppVM01 prompts for user name password

(Allowed) Web Server DNS lookup on DNS server

1. Web Server, IIS01, needs a data feed at www.data.gov, but needs to resolve the address.
2. The network configuration for the VNet lists DNS01 (10.0.2.4 on the Backend subnet) as the primary DNS server, IIS01 sends the DNS request to DNS01
3. No outbound rules on Frontend subnet, traffic is allowed
4. Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) does apply, traffic is allowed, stop rule processing
5. DNS server receives the request
6. DNS server doesn't have the address cached and asks a root DNS server on the internet
7. No outbound rules on Backend subnet, traffic is allowed
8. Internet DNS server responds, since this session was initiated internally, the response is allowed
9. DNS server caches the response, and responds to the initial request back to IIS01
10. No outbound rules on Backend subnet, traffic is allowed
11. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed
12. IIS01 receives the response from DNS01

(Allowed) Web Server access file on AppVM01

1. IIS01 asks for a file on AppVM01
2. No outbound rules on Frontend subnet, traffic is allowed
3. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 3 (Internet to Firewall) doesn't apply, move to next rule
 - d. NSG Rule 4 (IIS01 to AppVM01) does apply, traffic is allowed, stop rule processing
4. AppVM01 receives the request and responds with file (assuming access is authorized)
5. Since there are no outbound rules on the Backend subnet the response is allowed

6. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed.
7. The IIS server receives the file

(Denied) Web direct to Web Server

Since the Web Server, IIS01, and the Firewall are in the same Cloud Service they share the same public facing IP address. Thus any HTTP traffic would be directed to the firewall. While the request would be successfully served, it cannot go directly to the Web Server; it passed, as designed, through the Firewall first. See the first Scenario in this section for the traffic flow.

(Denied) Web to Backend Server

1. Internet user tries to access a file on AppVM01 through the BackEnd001.CloudApp.Net service
2. Since there are no endpoints open for file share, this would not pass the Cloud Service and wouldn't reach the server
3. If the endpoints were open for some reason, NSG rule 5 (Internet to VNet) would block this traffic

(Denied) Web DNS lookup on DNS server

1. Internet user tries to lookup an internal DNS record on DNS01 through the BackEnd001.CloudApp.Net service
2. Since there are no endpoints open for DNS, this would not pass the Cloud Service and wouldn't reach the server
3. If the endpoints were open for some reason, NSG rule 5 (Internet to VNet) would block this traffic (Note: that Rule 1 (DNS) would not apply for two reasons, first the source address is the internet, this rule only applies to the local VNet as the source, also this is an Allow rule, so it would never deny traffic)

(Denied) Web to SQL access through Firewall

1. Internet user requests SQL data from FrontEnd001.CloudApp.Net (Internet Facing Cloud Service)
2. Since there are no endpoints open for SQL, this would not pass the Cloud Service and wouldn't reach the firewall
3. If endpoints were open for some reason, the Frontend subnet begins inbound rule processing:
 - a. NSG Rule 1 (DNS) doesn't apply, move to next rule
 - b. NSG Rule 2 (RDP) doesn't apply, move to next rule
 - c. NSG Rule 2 (Internet to Firewall) does apply, traffic is allowed, stop rule processing
4. Traffic hits internal IP address of the firewall (10.0.1.4)
5. Firewall has no forwarding rules for SQL and drops the traffic

Conclusion

This is a relatively straight forward way of protecting your application with a firewall and isolating the back end subnet from inbound traffic.

More examples and an overview of network security boundaries can be found [here](#).

References

Main Script and Network Config

Save the Full Script in a PowerShell script file. Save the Network Config into a file named "NetworkConf2.xml". Modify the user defined variables as needed. Run the script, then follow the Firewall rule setup instruction above.

Full Script

This script will, based on the user defined variables:

1. Connect to an Azure subscription

2. Create a new storage account
3. Create a new VNet and two subnets as defined in the Network Config file
4. Build 4 windows server VMs
5. Configure NSG including:
 - Creating a NSG
 - Populating it with rules
 - Binding the NSG to the appropriate subnets

This PowerShell script should be run locally on an internet connected PC or server.

IMPORTANT

When this script is run, there may be warnings or other informational messages that pop in PowerShell. Only error messages in red are cause for concern.

```
<#
.SYNOPSIS
Example of DMZ and Network Security Groups in an isolated network (Azure only, no hybrid connections)

.DESCRIPTION
This script will build out a sample DMZ setup containing:
- A default storage account for VM disks
- Two new cloud services
- Two Subnets (FrontEnd and BackEnd subnets)
- A Network Virtual Appliance (NVA), in this case a Barracuda NextGen Firewall
- One server on the FrontEnd Subnet (plus the NVA on the FrontEnd subnet)
- Three Servers on the BackEnd Subnet
- Network Security Groups to allow/deny traffic patterns as declared

Before running script, ensure the network configuration file is created in
the directory referenced by $NetworkConfigFile variable (or update the
variable to reflect the path and file name of the config file being used).

.Notes
Security requirements are different for each use case and can be addressed in a
myriad of ways. Please be sure that any sensitive data or applications are behind
the appropriate layer(s) of protection. This script serves as an example of some
of the techniques that can be used, but should not be used for all scenarios. You
are responsible to assess your security needs and the appropriate protections
needed, and then effectively implement those protections.

FrontEnd Service (FrontEnd subnet 10.0.1.0/24)
myFirewall - 10.0.1.4
IIS01      - 10.0.1.5

BackEnd Service (BackEnd subnet 10.0.2.0/24)
DNS01      - 10.0.2.4
AppVM01    - 10.0.2.5
AppVM02    - 10.0.2.6

#>

# Fixed Variables
$LocalAdminPwd = Read-Host -Prompt "Enter Local Admin Password to be used for all VMs"
$VMName = @()
$ServiceName = @()
$VMFamily = @()
$img = @()
$size = @()
$SubnetName = @()
$VMIP = @()

# User Defined Global Variables
```

```

# User Defined Global Variables
# These should be changes to reflect your subscription and services
# Invalid options will fail in the validation section

# Subscription Access Details
$subID = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

# VM Account, Location, and Storage Details
$LocalAdmin = "theAdmin"
$DeploymentLocation = "Central US"
$StorageAccountName = "vmstore02"

# Service Details
$FrontEndService = "FrontEnd001"
$BackEndService = "BackEnd001"

# Network Details
$VNetName = "CorpNetwork"
$FESubnet = "FrontEnd"
$FEPrefix = "10.0.1.0/24"
$BESubnet = "BackEnd"
$BEPrefix = "10.0.2.0/24"
$NetworkConfigFile = "C:\Scripts\NetworkConf2.xml"

# VM Base Disk Image Details
$SrvImg = Get-AzureVMImage | Where {$_.ImageFamily -match 'Windows Server 2012 R2 Datacenter'} | sort PublishedDate -Descending | Select ImageName -First 1 | ForEach {$_.ImageName}
$FWImg = Get-AzureVMImage | Where {$_.ImageFamily -match 'Barracuda NextGen Firewall'} | sort PublishedDate -Descending | Select ImageName -First 1 | ForEach {$_.ImageName}

# NSG Details
$NSGName = "MyVNetSG"

# User Defined VM Specific Config
# Note: To ensure proper NSG Rule creation later in this script:
#       - The Web Server must be VM 1
#       - The AppVM1 Server must be VM 2
#       - The DNS server must be VM 4
#
#       Otherwise the NSG rules in the last section of this
#       script will need to be changed to match the modified
#       VM array numbers ($i) so the NSG Rule IP addresses
#       are aligned to the associated VM IP addresses.

# VM 0 - The Network Virtual Appliance (NVA)
$VMName += "myFirewall"
$ServiceName += $FrontEndService
$VMFamily += "Firewall"
$img += $FWImg
$size += "Small"
$SubnetName += $FESubnet
$VMIP += "10.0.1.4"

# VM 1 - The Web Server
$VMName += "IIS01"
$ServiceName += $FrontEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $FESubnet
$VMIP += "10.0.1.5"

# VM 2 - The First Application Server
$VMName += "AppVM01"
$ServiceName += $BackEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.5"

```

```

$VMIP += "10.0.2.5"

# VM 3 - The Second Application Server
$VMName += "AppVM02"
$ServiceName += $BackEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.6"

# VM 4 - The DNS Server
$VMName += "DNS01"
$ServiceName += $BackEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.4"

# ----- #
# No User Defined Variables or   #
# Configuration past this point #
# ----- #

# Get your Azure accounts
Add-AzureAccount
Set-AzureSubscription -SubscriptionId $subID -ErrorAction Stop
Select-AzureSubscription -SubscriptionId $subID -Current -ErrorAction Stop

# Create Storage Account
If (Test-AzureName -Storage -Name $StorageAccountName) {
    Write-Host "Fatal Error: This storage account name is already in use, please pick a different name." -ForegroundColor Red
    Return}
Else {Write-Host "Creating Storage Account" -ForegroundColor Cyan
      New-AzureStorageAccount -Location $DeploymentLocation -StorageAccountName $StorageAccountName}

# Update Subscription Pointer to New Storage Account
Write-Host "Updating Subscription Pointer to New Storage Account" -ForegroundColor Cyan
Set-AzureSubscription -SubscriptionId $subID -CurrentStorageAccountName $StorageAccountName -ErrorAction Stop

# Validation
$FatalError = $false

If (-Not (Get-AzureLocation | Where {$_.DisplayName -eq $DeploymentLocation})) {
    Write-Host "This Azure Location was not found or available for use" -ForegroundColor Yellow
    $FatalError = $true}

If (Test-AzureName -Service -Name $FrontEndService) {
    Write-Host "The FrontEndService service name is already in use, please pick a different service name." -ForegroundColor Yellow
    $FatalError = $true}
Else { Write-Host "The FrontEndService service name is valid for use." -ForegroundColor Green}

If (Test-AzureName -Service -Name $BackEndService) {
    Write-Host "The BackEndService service name is already in use, please pick a different service name." -ForegroundColor Yellow
    $FatalError = $true}
Else { Write-Host "The BackEndService service name is valid for use." -ForegroundColor Green}

If (-Not (Test-Path $NetworkConfigFile)) {
    Write-Host 'The network config file was not found, please update the $NetworkConfigFile variable to point to the network config xml file.' -ForegroundColor Yellow
    $FatalError = $true}
Else { Write-Host "The network config file was found" -ForegroundColor Green
      If (-Not (Select-String -Pattern $DeploymentLocation -Path $NetworkConfigFile)) {
          Write-Host 'The deployment location was not found in the network config file, please check the

```

```

network config file to ensure the $DeploymentLocation variable is correct and the netwrk config file matches.' 
-ForegroundColor Yellow
    $FatalError = $true}
    Else { Write-Host "The deployment location was found in the network config file." -ForegroundColor Green}

If ($FatalError) {
    Write-Host "A fatal error has occurred, please see the above messages for more information." - 
ForegroundColor Red
    Return}
Else { Write-Host "Validation passed, now building the environment." -ForegroundColor Green}

# Create VNET
    Write-Host "Creating VNET" -ForegroundColor Cyan
    Set-AzureVNetConfig -ConfigurationPath $NetworkConfigFile -ErrorAction Stop

# Create Services
    Write-Host "Creating Services" -ForegroundColor Cyan
    New-AzureService -Location $DeploymentLocation -ServiceName $FrontEndService -ErrorAction Stop
    New-AzureService -Location $DeploymentLocation -ServiceName $BackEndService -ErrorAction Stop

# Build VMs
    $i=0
    $VMName | Foreach {
        Write-Host "Building $($VMName[$i])" -ForegroundColor Cyan
        If ($VMFamily[$i] -eq "Firewall")
        {
            New-AzureVMConfig -Name $VMName[$i] -ImageName $img[$i] -InstanceSize $size[$i] | ` 
                Add-AzureProvisioningConfig -Linux -LinuxUser $LocalAdmin -Password $LocalAdminPwd | ` 
                Set-AzureSubnet -SubnetNames $SubnetName[$i] | ` 
                Set-AzureStaticVNetIP -IPAddress $VMIP[$i] | ` 
                New-AzureVM -ServiceName $ServiceName[$i] -VNetName $VNetName -Location $DeploymentLocation
            # Set up all the EndPoints we'll need once we're up and running
            # Note: Web traffic goes through the firewall, so we'll need to set up a HTTP endpoint.
            #       Also, the firewall will be redirecting web traffic to a new IP and Port in a 
            #       forwarding rule, so the HTTP endpoint here will have the same public and local 
            #       port and the firewall will do the NATing and redirection as declared in the 
            #       firewall rule.
            Add-AzureEndpoint -Name "MgmtPort1" -Protocol tcp -PublicPort 801 -LocalPort 801 -VM (Get-AzureVM 
-ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
            Add-AzureEndpoint -Name "MgmtPort2" -Protocol tcp -PublicPort 807 -LocalPort 807 -VM (Get-AzureVM 
-ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
            Add-AzureEndpoint -Name "HTTP" -Protocol tcp -PublicPort 80 -LocalPort 80 -VM (Get-AzureVM 
-ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
            # Note: A SSH endpoint is automatically created on port 22 when the appliance is created.
        }
        Else
        {
            New-AzureVMConfig -Name $VMName[$i] -ImageName $img[$i] -InstanceSize $size[$i] | ` 
                Add-AzureProvisioningConfig -Windows -AdminUsername $LocalAdmin -Password $LocalAdminPwd | ` 
                Set-AzureSubnet -SubnetNames $SubnetName[$i] | ` 
                Set-AzureStaticVNetIP -IPAddress $VMIP[$i] | ` 
                Set-AzureVMMicrosoftAntimalwareExtension -AntimalwareConfiguration '{"AntimalwareEnabled" : 
true}' | ` 
                Remove-AzureEndpoint -Name "PowerShell" | ` 
                New-AzureVM -ServiceName $ServiceName[$i] -VNetName $VNetName -Location $DeploymentLocation
            # Note: A Remote Desktop endpoint is automatically created when each VM is created.
        }
        $i++
    }

# Configure NSG
    Write-Host "Configuring the Network Security Group (NSG)" -ForegroundColor Cyan

# Build the NSG
    Write-Host "Building the NSG" -ForegroundColor Cyan
    New-AzureNetworkSecurityGroup -Name $NSGName -Location $DeploymentLocation -Label "Security group for 
$VNetName subnets in $DeploymentLocation"

```

```

# Add NSG Rules
Write-Host "Writing rules into the NSG" -ForegroundColor Cyan
Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable Internal DNS" -
Type Inbound -Priority 100 -Action Allow `

    -SourceAddressPrefix VIRTUAL_NETWORK -SourcePortRange '*' `

    -DestinationAddressPrefix $VMIP[4] -DestinationPortRange '53' `

    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable RDP to $VNetName
VNet" -Type Inbound -Priority 110 -Action Allow `

    -SourceAddressPrefix INTERNET -SourcePortRange '*' `

    -DestinationAddressPrefix VIRTUAL_NETWORK -DestinationPortRange '3389' `

    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable Internet to
$($VMName[0])" -Type Inbound -Priority 120 -Action Allow `

    -SourceAddressPrefix Internet -SourcePortRange '*' `

    -DestinationAddressPrefix $VMIP[0] -DestinationPortRange '*' `

    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Enable $($VMName[1]) to
$($VMName[2])" -Type Inbound -Priority 130 -Action Allow `

    -SourceAddressPrefix $VMIP[1] -SourcePortRange '*' `

    -DestinationAddressPrefix $VMIP[2] -DestinationPortRange '*' `

    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Isolate the $VNetName
VNet from the Internet" -Type Inbound -Priority 140 -Action Deny `

    -SourceAddressPrefix INTERNET -SourcePortRange '*' `

    -DestinationAddressPrefix VIRTUAL_NETWORK -DestinationPortRange '*' `

    -Protocol *

Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Isolate the $FESubnet
subnet from the $BESubnet subnet" -Type Inbound -Priority 150 -Action Deny `

    -SourceAddressPrefix $FEPrefix -SourcePortRange '*' `

    -DestinationAddressPrefix $BEPrefix -DestinationPortRange '*' `

    -Protocol *

# Assign the NSG to the Subnets
Write-Host "Binding the NSG to both subnets" -ForegroundColor Cyan
Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName -SubnetName $FESubnet -VirtualNetworkName
$VNetName
Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName -SubnetName $BESubnet -VirtualNetworkName
$VNetName

# Optional Post-script Manual Configuration
# Configure Firewall
# Install Test Web App (Run Post-Build Script on the IIS Server)
# Install Backend resource (Run Post-Build Script on the AppVM01)
Write-Host
Write-Host "Build Complete!" -ForegroundColor Green
Write-Host
Write-Host "Optional Post-script Manual Configuration Steps" -ForegroundColor Gray
Write-Host "- Configure Firewall" -ForegroundColor Gray
Write-Host "- Install Test Web App (Run Post-Build Script on the IIS Server)" -ForegroundColor Gray
Write-Host "- Install Backend resource (Run Post-Build Script on the AppVM01)" -ForegroundColor Gray
Write-Host

```

Network Config File

Save this xml file with updated location and add the link to this file to the \$NetworkConfigFile variable in the script above.

```
<NetworkConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/ServiceHosting/2011/07/NetworkConfiguration">
  <VirtualNetworkConfiguration>
    <Dns>
      <DnsServers>
        <DnsServer name="DNS01" IPAddress="10.0.2.4" />
        <DnsServer name="Level3" IPAddress="209.244.0.3" />
      </DnsServers>
    </Dns>
    <VirtualNetworkSites>
      <VirtualNetworkSite name="CorpNetwork" Location="Central US">
        <AddressSpace>
          <AddressPrefix>10.0.0.0/16</AddressPrefix>
        </AddressSpace>
        <Subnets>
          <Subnet name="FrontEnd">
            <AddressPrefix>10.0.1.0/24</AddressPrefix>
          </Subnet>
          <Subnet name="BackEnd">
            <AddressPrefix>10.0.2.0/24</AddressPrefix>
          </Subnet>
        </Subnets>
        <DnsServersRef>
          <DnsServerRef name="DNS01" />
          <DnsServerRef name="Level3" />
        </DnsServersRef>
      </VirtualNetworkSite>
    </VirtualNetworkSites>
  </VirtualNetworkConfiguration>
</NetworkConfiguration>
```

Sample Application Scripts

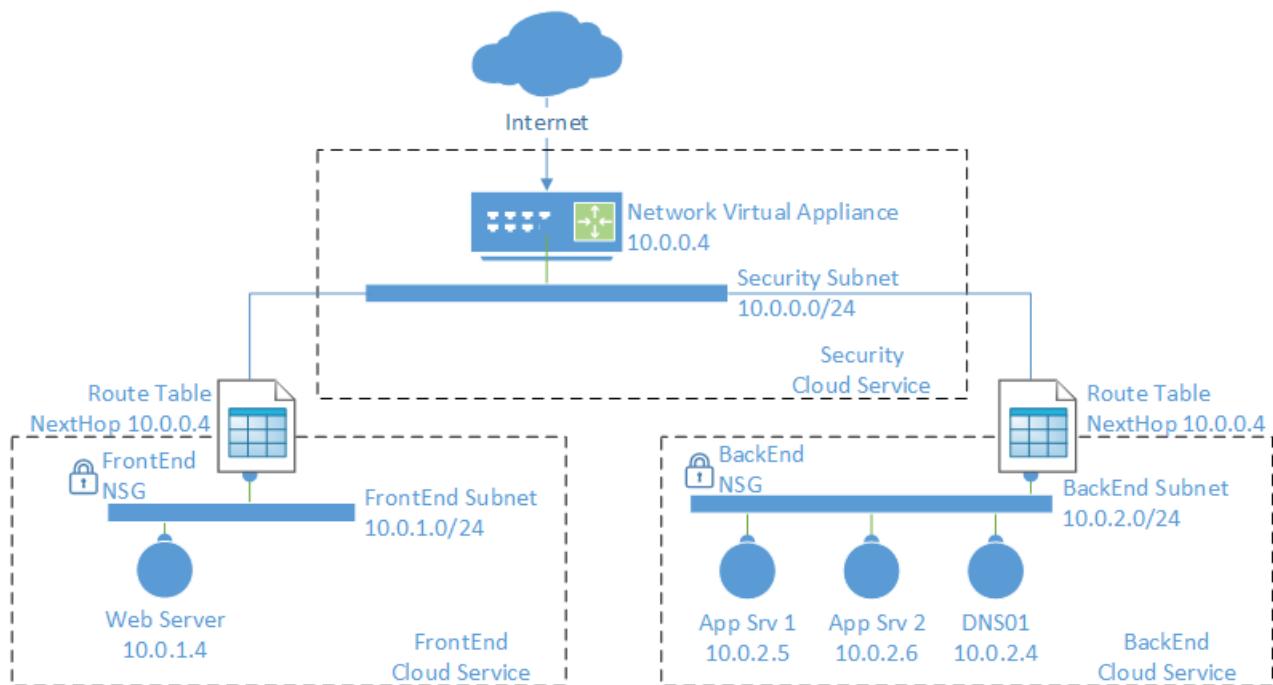
If you wish to install a sample application for this, and other DMZ Examples, one has been provided at the following link: [Sample Application Script](#)

Example 3 – Build a DMZ to Protect Networks with a Firewall, UDR, and NSG

7/10/2017 • 48 min to read • [Edit Online](#)

[Return to the Security Boundary Best Practices Page](#)

This example will create a DMZ with a firewall, four windows servers, User Defined Routing, IP Forwarding, and Network Security Groups. It will also walk through each of the relevant commands to provide a deeper understanding of each step. There is also a Traffic Scenario section to provide an in-depth step-by-step how traffic proceeds through the layers of defense in the DMZ. Finally, in the references section is the complete code and instruction to build this environment to test and experiment with various scenarios.



Environment Setup

In this example there is a subscription that contains the following:

- Three cloud services: "SecSvc001", "FrontEnd001", and "BackEnd001"
- A Virtual Network "CorpNetwork", with three subnets: "SecNet", "FrontEnd", and "BackEnd"
- A network virtual appliance, in this example a firewall, connected to the SecNet subnet
- A Windows Server that represents an application web server ("IIS01")
- Two windows servers that represent application back end servers ("AppVM01", "AppVM02")
- A Windows server that represents a DNS server ("DNS01")

In the references section below there is a PowerShell script that will build most of the environment described above. Building the VMs and Virtual Networks, although are done by the example script, are not described in detail in this document.

To build the environment:

1. Save the network config xml file included in the references section (updated with names, location, and IP addresses to match the given scenario)

2. Update the user variables in the script to match the environment the script is to be run against (subscriptions, service names, etc)
3. Execute the script in PowerShell

Note: The region signified in the PowerShell script must match the region signified in the network configuration xml file.

Once the script runs successfully the following post-script steps may be taken:

1. Set up the firewall rules, this is covered in the section below titled: Firewall Rule Description.
2. Optionally in the references section are two scripts to set up the web server and app server with a simple web application to allow testing with this DMZ configuration.

Once the script runs successfully the firewall rules will need to be completed, this is covered in the section titled: Firewall Rules.

User Defined Routing (UDR)

By default, the following system routes are defined as:

Effective routes :				
Address Prefix	Next hop type	Next hop IP address	Status	Source
{10.0.0.0/16}	VNETLocal		Active	Default
{0.0.0.0/0}	Internet		Active	Default
{10.0.0.0/8}	Null		Active	Default
{100.64.0.0/10}	Null		Active	Default
{172.16.0.0/12}	Null		Active	Default
{192.168.0.0/16}	Null		Active	Default

The VNETLocal is always the defined address prefix(es) of the VNet for that specific network (ie it will change from VNet to VNet depending on how each specific VNet is defined). The remaining system routes are static and default as above.

As for priority, routes are processed via the Longest Prefix Match (LPM) method, thus the most specific route in the table would apply to a given destination address.

Therefore, traffic (for example to the DNS01 server, 10.0.2.4) intended for the local network (10.0.0.0/16) would be routed across the VNet to its destination due to the 10.0.0.0/16 route. In other words, for 10.0.2.4, the 10.0.0.0/16 route is the most specific route, even though the 10.0.0.0/8 and 0.0.0.0/0 also could apply, but since they are less specific they don't affect this traffic. Thus the traffic to 10.0.2.4 would have a next hop of the local VNet and simply route to the destination.

If traffic was intended for 10.1.1.1 for example, the 10.0.0.0/16 route wouldn't apply, but the 10.0.0.0/8 would be the most specific, and this the traffic would be dropped ("black holed") since the next hop is Null.

If the destination didn't apply to any of the Null prefixes or the VNETLocal prefixes, then it would follow the least specific route, 0.0.0.0/0 and be routed out to the Internet as the next hop and thus out Azure's internet edge.

If there are two identical prefixes in the route table, the following is the order of preference based on the routes "source" attribute:

1. "VirtualAppliance" = A User Defined Route manually added to the table
2. "VPNGateway" = A Dynamic Route (BGP when used with hybrid networks), added by a dynamic network protocol, these routes may change over time as the dynamic protocol automatically reflects changes in peered network
3. "Default" = The System Routes, the local VNet and the static entries as shown in the route table above.

NOTE

You can now use User Defined Routing (UDR) with ExpressRoute and VPN Gateways to force outbound and inbound cross-premises traffic to be routed to a network virtual appliance (NVA).

Creating the local routes

In this example, two routing tables are needed, one each for the Frontend and Backend subnets. Each table is loaded with static routes appropriate for the given subnet. For the purpose of this example, each table has three routes:

1. Local subnet traffic with no Next Hop defined to allow local subnet traffic to bypass the firewall
2. Virtual Network traffic with a Next Hop defined as firewall, this overrides the default rule that allows local VNet traffic to route directly
3. All remaining traffic (0/0) with a Next Hop defined as the firewall

Once the routing tables are created they are bound to their subnets. For the Frontend subnet routing table, once created and bound to the subnet should look like this:

Effective routes :					
Address Prefix	Next hop type	Next hop IP address	Status	Source	
{10.0.1.0/24}	VNETLocal		Active		
{10.0.0.0/16}	VirtualAppliance	10.0.0.4	Active		
{0.0.0.0/0}	VirtualAppliance	10.0.0.4	Active		

For this example, the following commands are used to build the route table, add a user defined route, and then bind the route table to a subnet (Note; any items below beginning with a dollar sign (e.g.: \$BESubnet) are user defined variables from the script in the reference section of this document):

1. First the base routing table must be created. This snippet shows the creation of the table for the Backend subnet. In the script, a corresponding table is also created for the Frontend subnet.

```
New-AzureRouteTable -Name $BERouteTableName `
```

```
    -Location $DeploymentLocation `  
    -Label "Route table for $BESubnet subnet"
```

2. Once the route table is created, specific user defined routes can be added. In this snipped, all traffic (0.0.0.0/0) will be routed through the virtual appliance (a variable, \$VMIP[0], is used to pass in the IP address assigned when the virtual appliance was created earlier in the script). In the script, a corresponding rule is also created in the Frontend table.

```
Get-AzureRouteTable $BERouteTableName | `
```

```
Set-AzureRoute -RouteName "All traffic to FW" -AddressPrefix 0.0.0.0/0 `  
    -NextHopType VirtualAppliance `  
    -NextHopIpAddress $VMIP[0]
```

3. The above route entry will override the default "0.0.0.0/0" route, but the default 10.0.0.0/16 rule still existing which would allow traffic within the VNet to route directly to the destination and not to the Network Virtual Appliance. To correct this behavior the follow rule must be added.

```
Get-AzureRouteTable $BERouteTableName | ` 
    Set-AzureRoute -RouteName "Internal traffic to FW" -AddressPrefix $VNetPrefix ` 
        -NextHopType VirtualAppliance ` 
        -NextHopIpAddress $VMIP[0]
```

- At this point there is a choice to be made. With the above two routes all traffic will route to the firewall for assessment, even traffic within a single subnet. This may be desired, however to allow traffic within a subnet to route locally without involvement from the firewall a third, very specific rule can be added. This route states that any address destined for the local subnet can just route there directly (NextHopType = VNETLocal).

```
Get-AzureRouteTable $BERouteTableName | ` 
    Set-AzureRoute -RouteName "Allow Intra-Subnet Traffic" -AddressPrefix $BEPrefix ` 
        -NextHopType VNETLocal
```

- Finally, with the routing table created and populated with a user defined routes, the table must now be bound to a subnet. In the script, the front end route table is also bound to the Frontend subnet. Here is the binding script for the back end subnet.

```
Set-AzureSubnetRouteTable -VirtualNetworkName $VNetName `
```

```
-SubnetName $BESubnet ` 
-RouteTableName $BERouteTableName
```

IP Forwarding

A companion feature to UDR, is IP Forwarding. This is a setting on a Virtual Appliance that allows it to receive traffic not specifically addressed to the appliance and then forward that traffic to its ultimate destination.

As an example, if traffic from AppVM01 makes a request to the DNS01 server, UDR would route this to the firewall. With IP Forwarding enabled, the traffic for the DNS01 destination (10.0.2.4) will be accepted by the appliance (10.0.0.4) and then forwarded to its ultimate destination (10.0.2.4). Without IP Forwarding enabled on the Firewall, traffic would not be accepted by the appliance even though the route table has the firewall as the next hop.

IMPORTANT

It's critical to remember to enable IP Forwarding in conjunction with User Defined Routing.

Setting up IP Forwarding is a single command and can be done at VM creation time. For the flow of this example the code snippet is towards the end of the script and grouped with the UDR commands:

- Call the VM instance that is your virtual appliance, the firewall in this case, and enable IP Forwarding (Note; any item in red beginning with a dollar sign (e.g.: \$VMName[0]) is a user defined variable from the script in the reference section of this document. The zero in brackets, [0], represents the first VM in the array of VMs, for the example script to work without modification, the first VM (VM 0) must be the firewall):

```
Get-AzureVM -Name $VMName[0] -ServiceName $ServiceName[0] | `
```

```
Set-AzureIPForwarding -Enable
```

Network Security Groups (NSG)

In this example, a NSG group is built and then loaded with a single rule. This group is then bound only to the Frontend and Backend subnets (not the SecNet). Declaratively the following rule is being built:

1. Any traffic (all ports) from the Internet to the entire VNet (all subnets) is Denied

Although NSGs are used in this example, it's main purpose is as a secondary layer of defense against manual misconfiguration. We want to block all inbound traffic from the internet to either the Frontend or Backend subnets, traffic should only flow through the SecNet subnet to the firewall (and then if appropriate on to the Frontend or Backend subnets). Plus, with the UDR rules in place, any traffic that did make it into the Frontend or Backend subnets would be directed out to the firewall (thanks to UDR). The firewall would see this as an asymmetric flow and would drop the outbound traffic. Thus there are three layers of security protecting the Frontend and Backend subnets; 1) no open endpoints on the FrontEnd001 and BackEnd001 cloud services, 2) NSGs denying traffic from the Internet, 3) the firewall dropping asymmetric traffic.

One interesting point regarding the Network Security Group in this example is that it contains only one rule, shown below, which is to deny internet traffic to the entire virtual network which would include the Security subnet.

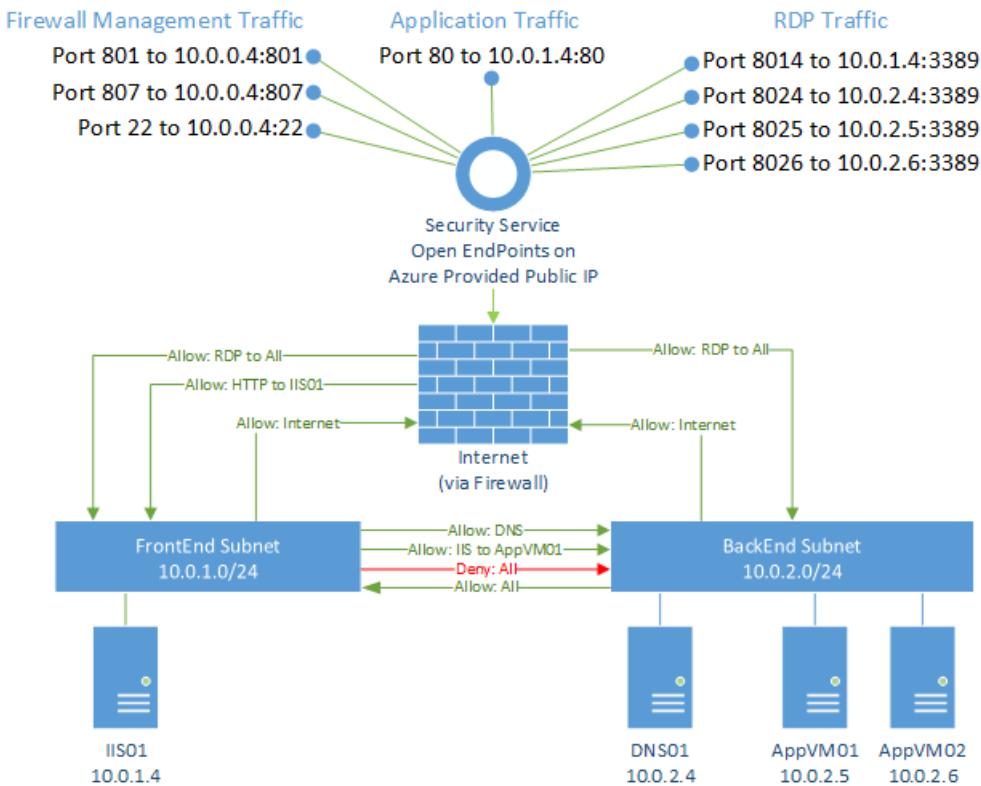
```
Get-AzureNetworkSecurityGroup -Name $NSGName | `  
Set-AzureNetworkSecurityRule -Name "Isolate the $VNetName VNet`  
from the Internet" `  
-Type Inbound -Priority 100 -Action Deny `  
-SourceAddressPrefix INTERNET -SourcePortRange '*' `  
-DestinationAddressPrefix VIRTUAL_NETWORK `  
-DestinationPortRange '*' `  
-Protocol *
```

However, since the NSG is only bound to the Frontend and Backend subnets, the rule isn't processed on traffic inbound to the Security subnet. As a result, even though the NSG rule says no Internet traffic to any address on the VNet, because the NSG was never bound to the Security subnet, traffic will flow to the Security subnet.

```
Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName `  
-SubnetName $FESubnet -VirtualNetworkName $VNetName  
  
Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName `  
-SubnetName $BESubnet -VirtualNetworkName $VNetName
```

Firewall Rules

On the firewall, forwarding rules will need to be created. Since the firewall is blocking or forwarding all inbound, outbound, and intra-VNet traffic many firewall rules are needed. Also, all inbound traffic will hit the Security Service public IP address (on different ports), to be processed by the firewall. A best practice is to diagram the logical flows before setting up the subnets and firewall rules to avoid rework later. The following figure is a logical view of the firewall rules for this example:



NOTE

Based on the Network Virtual Appliance used, the management ports will vary. In this example a Barracuda NextGen Firewall is referenced which uses ports 22, 801, and 807. Please consult the appliance vendor documentation to find the exact ports used for management of the device being used.

Logical Rule Description

In the logical diagram above, the security subnet is not shown since the firewall is the only resource on that subnet, and this diagram is showing the firewall rules and how they logically allow or deny traffic flows and not the actual routed path. Also, the external ports selected for the RDP traffic are higher ranged ports (8014 – 8026) and were selected to somewhat align with the last two octets of the local IP address for easier readability (e.g. local server address 10.0.1.4 is associated with external port 8014), however any higher non-conflicting ports could be used.

For this example, we need 7 types of rules, these rule types are described as follows:

- External Rules (for inbound traffic):
 1. Firewall Management Rule: This App Redirect rule allows traffic to pass to the management ports of the network virtual appliance.
 2. RDP Rules (for each windows server): These four rules (one for each server) will allow management of the individual servers via RDP. This could also be bundled into one rule depending on the capabilities of the Network Virtual Appliance being used.
 3. Application Traffic Rules: There are two Application Traffic Rules, the first for the front end web traffic, and the second for the back end traffic (eg web server to data tier). The configuration of these rules will depend on the network architecture (where your servers are placed) and traffic flows (which direction the traffic flows, and which ports are used).
 - The first rule will allow the actual application traffic to reach the application server. While the other rules allow for security, management, etc., Application Rules are what allow external users or services to access the application(s). For this example, there is a single web server on port 80, thus a single firewall application rule will redirect inbound traffic to the external IP, to the web servers internal IP address. The redirected traffic session would be NAT'd to the internal server.
 - The second Application Traffic Rule is the back end rule to allow the Web Server to talk to the

AppVM01 server (but not AppVM02) via any port.

- Internal Rules (for intra-VNet traffic)
 1. Outbound to Internet Rule: This rule will allow traffic from any network to pass to the selected networks.
This rule is usually a default rule already on the firewall, but in a disabled state. This rule should be enabled for this example.
 2. DNS Rule: This rule allows only DNS (port 53) traffic to pass to the DNS server. For this environment most traffic from the Frontend to the Backend is blocked, this rule specifically allows DNS from any local subnet.
 3. Subnet to Subnet Rule: This rule is to allow any server on the back end subnet to connect to any server on the front end subnet (but not the reverse).
- Fail-safe Rule (for traffic that doesn't meet any of the above):
 1. Deny All Traffic Rule: This should always be the final rule (in terms of priority), and as such if a traffic flows fails to match any of the preceding rules it will be dropped by this rule. This is a default rule and usually activated, no modifications are generally needed.

TIP

On the second application traffic rule, any port is allowed for easy of this example, in a real scenario the most specific port and address ranges should be used to reduce the attack surface of this rule.

IMPORTANT

Once all of the above rules are created, it's important to review the priority of each rule to ensure traffic will be allowed or denied as desired. For this example, the rules are in priority order. It's easy to be locked out of the firewall due to mis-ordered rules. At a minimum, ensure the management for the firewall itself is always the absolute highest priority rule.

Rule Prerequisites

One prerequisite for the Virtual Machine running the firewall are public endpoints. For the firewall to process traffic the appropriate public endpoints must be open. There are three types of traffic in this example; 1) Management traffic to control the firewall and firewall rules, 2) RDP traffic to control the windows servers, and 3) Application Traffic. These are the three columns of traffic types in the upper half of logical view of the firewall rules above.

IMPORTANT

A key takeaway here is to remember that **all** traffic will come through the firewall. So to remote desktop to the IIS01 server, even though it's in the Front End Cloud Service and on the Front End subnet, to access this server we will need to RDP to the firewall on port 8014, and then allow the firewall to route the RDP request internally to the IIS01 RDP Port. The Azure portal's "Connect" button won't work because there is no direct RDP path to IIS01 (as far as the portal can see). This means all connections from the internet will be to the Security Service and a Port, e.g. secscv001.cloudapp.net:xxxx (reference the above diagram for the mapping of External Port and Internal IP and Port).

An endpoint can be opened either at the time of VM creation or post build as is done in the example script and shown below in this code snippet (Note; any item beginning with a dollar sign (e.g.: \$VMName[\$i]) is a user defined variable from the script in the reference section of this document. The "\$i" in brackets, [\$i], represents the array number of a specific VM in an array of VMs):

```
Add-AzureEndpoint -Name "HTTP" -Protocol tcp -PublicPort 80 -LocalPort 80 `  
-VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | `  
Update-AzureVM
```

Although not clearly shown here due to the use of variables, but endpoints are **only** opened on the Security Cloud Service. This is to ensure that all inbound traffic is handled (routed, NAT'd, dropped) by the firewall.

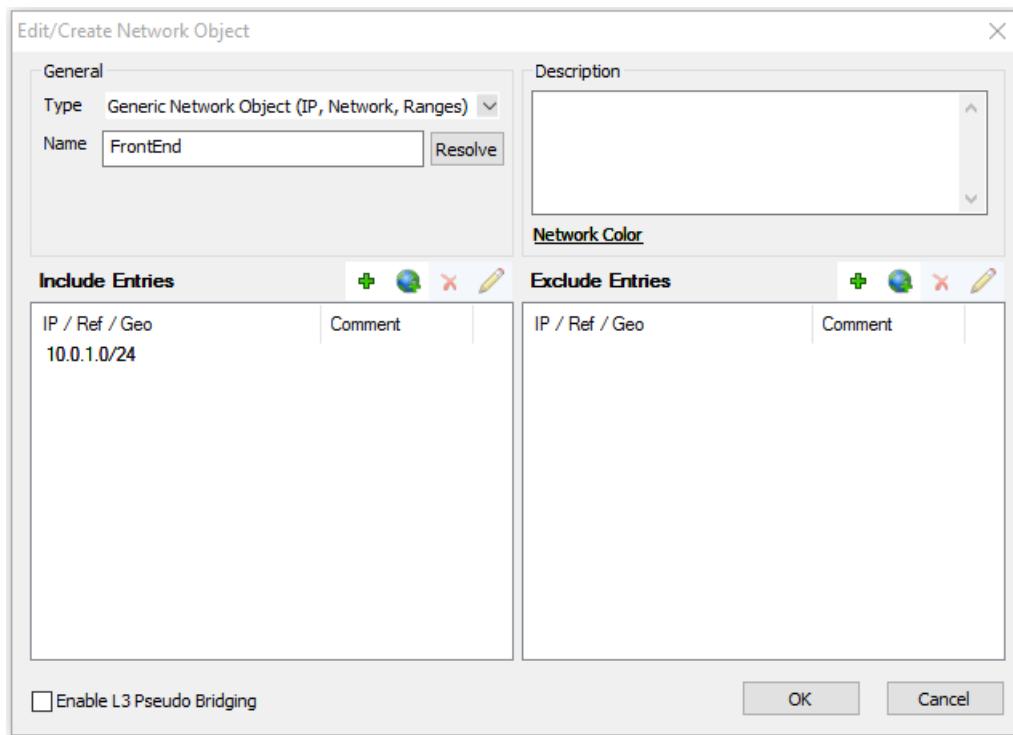
A management client will need to be installed on a PC to manage the firewall and create the configurations needed. See the documentation from your firewall (or other NVA) vendor on how to manage the device. The remainder of this section and the next section, Firewall Rules Creation, will describe the configuration of the firewall itself, through the vendors management client (i.e. not the Azure portal or PowerShell).

Instructions for client download and connecting to the Barracuda used in this example can be found here:

[Barracuda NG Admin](#)

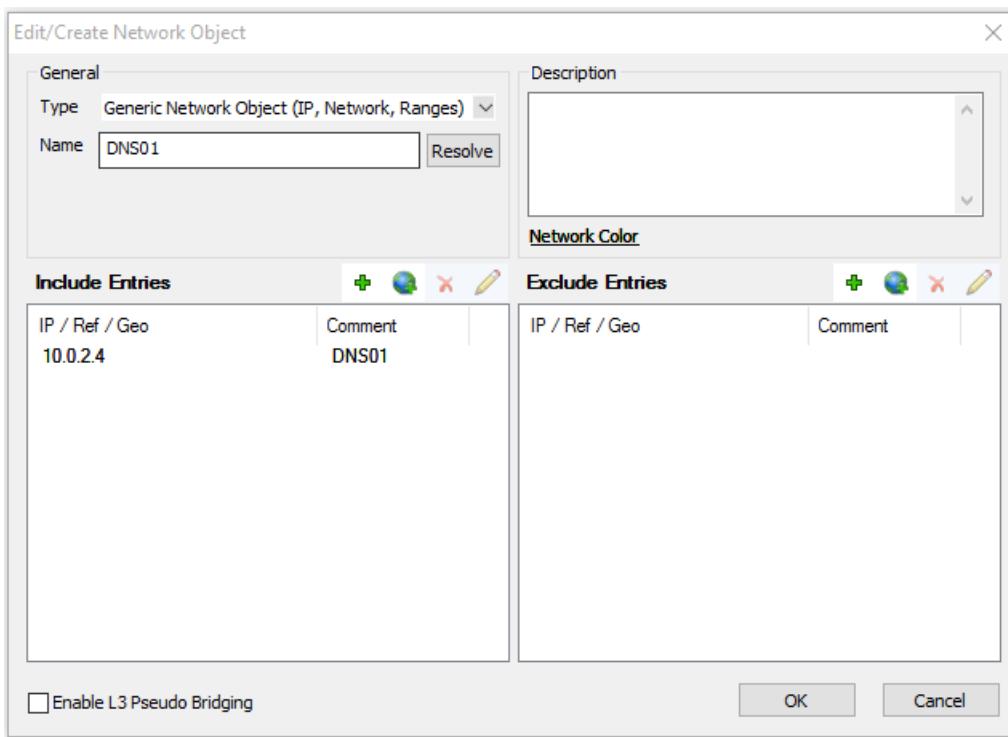
Once logged onto the firewall but before creating firewall rules, there are two prerequisite object classes that can make creating the rules easier; Network & Service objects.

For this example, three named network objects should be defined (one for the Frontend subnet and the Backend subnet, also a network object for the IP address of the DNS server). To create a named network; starting from the Barracuda NG Admin client dashboard, navigate to the configuration tab, in the Operational Configuration section click Ruleset, then click "Networks" under the Firewall Objects menu, then click New in the Edit Networks menu. The network object can now be created, by adding the name and the prefix:



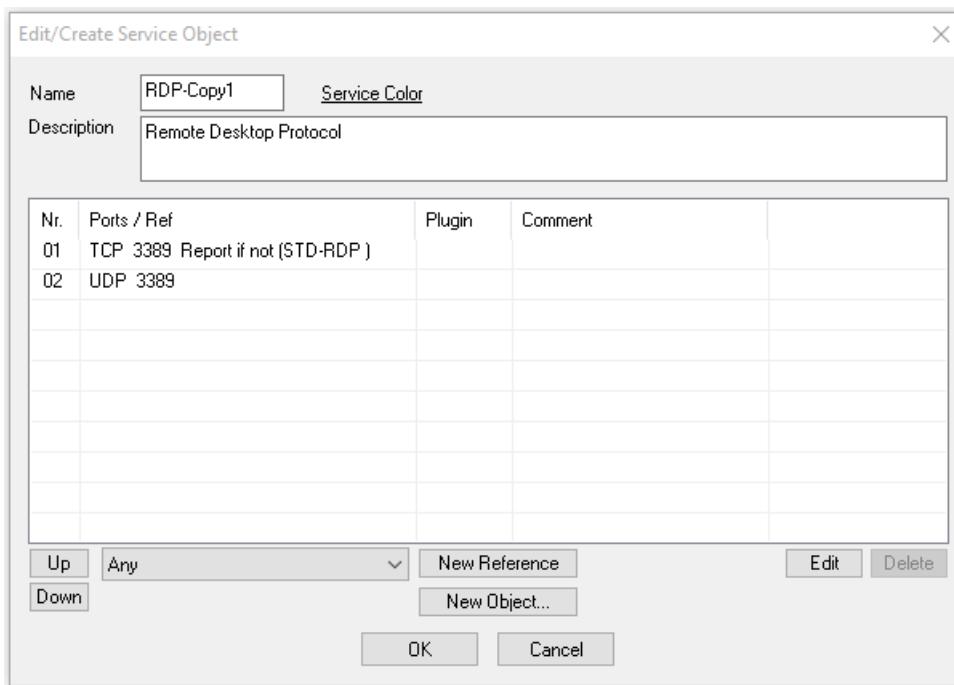
This will create a named network for the FrontEnd subnet, a similar object should be created for the BackEnd subnet as well. Now the subnets can be more easily referenced by name in the firewall rules.

For the DNS Server Object:

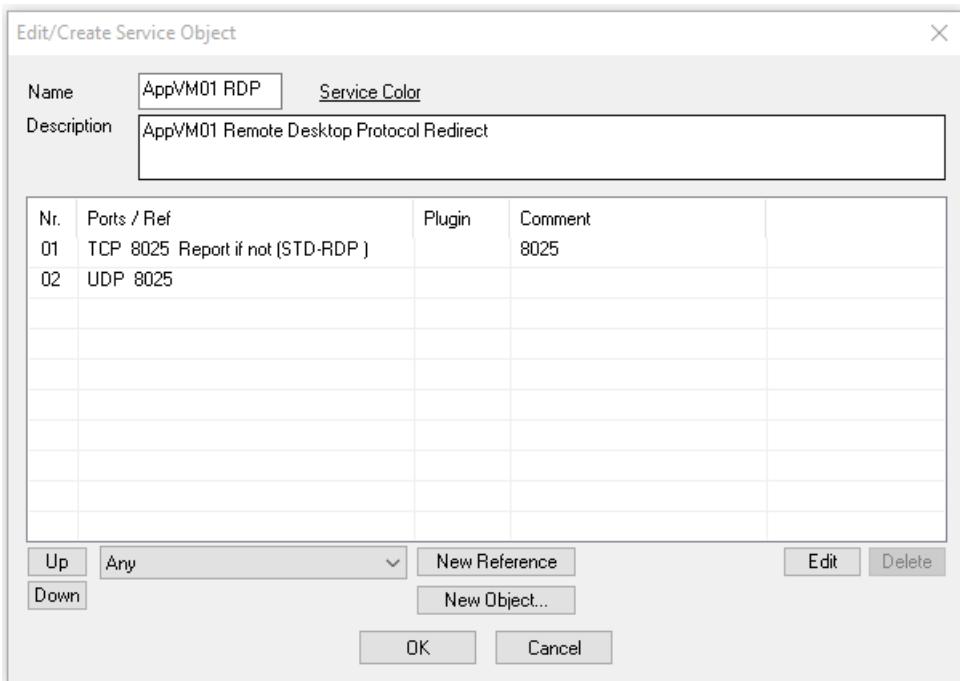


This single IP address reference will be utilized in a DNS rule later in the document.

The second prerequisite objects are Services objects. These will represent the RDP connection ports for each server. Since the existing RDP service object is bound to the default RDP port, 3389, new Services can be created to allow traffic from the external ports (8014-8026). The new ports could also be added to the existing RDP service, but for ease of demonstration, an individual rule for each server can be created. To create a new RDP rule for a server; starting from the Barracuda NG Admin client dashboard, navigate to the configuration tab, in the Operational Configuration section click Ruleset, then click "Services" under the Firewall Objects menu, scroll down the list of services and select the "RDP" service. Right-click and select copy, then right-click and select Paste. There is now a RDP-Copy1 Service Object that can be edited. Right-click RDP-Copy1 and select Edit, the Edit Service Object window will pop up as shown here:



The values can then be edited to represent the RDP service for a specific server. For AppVM01 the above default RDP rule should be modified to reflect a new Service Name, Description, and external RDP Port used in the Figure 8 diagram (Note: the ports are changed from the RDP default of 3389 to the external port being used for this specific server, in the case of AppVM01 the external Port is 8025) the modified service is shown below:



This process must be repeated to create RDP Services for the remaining servers; AppVM02, DNS01, and IIS01. The creation of these Services will make the Rule creation simpler and more obvious in the next section.

NOTE

An RDP service for the Firewall is not needed for two reasons; 1) first the firewall VM is a Linux based image so SSH would be used on port 22 for VM management instead of RDP, and 2) port 22, and two other management ports are allowed in the first management rule described below to allow for management connectivity.

Firewall Rules Creation

There are three types of firewall rules used in this example, they all have distinct icons:



The Application Redirect rule:



The Destination NAT rule:



The Pass rule:

More information on these rules can be found at the Barracuda web site.

To create the following rules (or verify existing default rules), starting from the Barracuda NG Admin client dashboard, navigate to the configuration tab, in the Operational Configuration section click Ruleset. A grid called, "Main Rules" will show the existing active and deactivated rules on this firewall. In the upper right corner of this grid is a small, green "+" button, click this to create a new rule (Note: your firewall may be "locked" for changes, if you see a button marked "Lock" and you are unable to create or edit rules, click this button to "unlock" the rule set and allow editing). If you wish to edit an existing rule, select that rule, right-click and select Edit Rule.

Once your rules are created and/or modified, they must be pushed to the firewall and then activated, if this is not done the rule changes will not take effect. The push and activation process is described below the details rule descriptions.

The specifics of each rule required to complete this example are described as follows:

- Firewall Management Rule:** This App Redirect rule allows traffic to pass to the management ports of the network virtual appliance, in this example a Barracuda NextGen Firewall. The management ports are 801, 807 and optionally 22. The external and internal ports are the same (i.e. no port translation). This rule, SETUP-MGMT-ACCESS, is a default rule and enabled by default (in Barracuda NextGen Firewall version 6.1).

The screenshot shows the 'App Redirect' configuration screen. The rule name is 'SETUP-MGMT-ACCESS'. Under the 'Source' section, 'Any' is selected, and the IP '0.0.0.0/0' is listed. In the 'Service' section, 'NGF-MGMT-BOX' is selected, with a dropdown menu showing various management services: Ref: NGF-MGMT-CONF, Ref: NGF-MGMT-CTRL, Ref: NGF-MGMT-DHCP, Ref: NGF-MGMT-EVENT, Ref: NGF-MGMT-FW, Ref: NGF-MGMT-LOG, Ref: NGF-MGMT-MAIL, and Ref: NGF-MGMT-STAT. The 'Destination' section shows 'DHCP1 Local IP'. In the 'Redirection' section, 'Local Address' is set to '127.0.0.2'.

TIP

The source address space in this rule is Any, if the management IP address ranges are known, reducing this scope would also reduce the attack surface to the management ports.

- RDP Rules:** These Destination NAT rules will allow management of the individual servers via RDP. There are four critical fields needed to create this rule:
 1. Source – to allow RDP from anywhere, the reference “Any” is used in the Source field.
 2. Service – use the appropriate Service Object created earlier, in this case “AppVM01 RDP”, the external ports redirect to the servers local IP address and to port 3386 (the default RDP port). This specific rule is for RDP access to AppVM01.
 3. Destination – should be the *local port on the firewall*, “DCHP 1 Local IP” or eth0 if using static IPs. The ordinal number (eth0, eth1, etc) may be different if your network appliance has multiple local interfaces. This is the port the firewall is sending out from (may be the same as the receiving port), the actual routed destination is in the Target List field.
 4. Redirection – this section tells the virtual appliance where to ultimately redirect this traffic. The simplest redirection is to place the IP and Port (optional) in the Target List field. If no port is used the destination port on the inbound request will be used (ie no translation), if a port is designated the port will also be NAT'd along with the IP address.

The screenshot shows the 'Dst NAT' configuration screen. The rule name is 'RDP-to-AppVM01'. Under the 'Source' section, 'Any' is selected, and the IP '0.0.0.0/0' is listed. In the 'Service' section, 'AppVM01 RDP' is selected, with a dropdown menu showing 'TCP 8025 Report if not (STD-RDP)' and 'UDP 8025'. The 'Destination' section shows 'DHCP1 Local IP'. In the 'Redirection' section, the 'Target List' is set to '10.0.2.5:3389'. The 'Fallback' section is set to 'List of Critical Ports'.

A total of four RDP rules will need to be created:

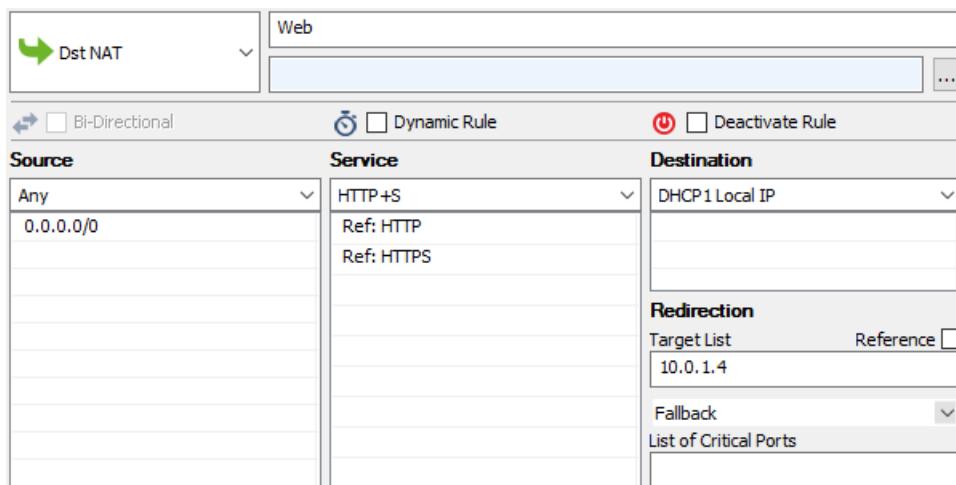
RULE NAME	SERVER	SERVICE	TARGET LIST
RDP-to-IIS01	IIS01	IIS01 RDP	10.0.1.4:3389
RDP-to-DNS01	DNS01	DNS01 RDP	10.0.2.4:3389
RDP-to-AppVM01	AppVM01	AppVM01 RDP	10.0.2.5:3389
RDP-to-AppVM02	AppVM02	AppVm02 RDP	10.0.2.6:3389

TIP

Narrowing down the scope of the Source and Service fields will reduce the attack surface. The most limited scope that will allow functionality should be used.

- **Application Traffic Rules:** There are two Application Traffic Rules, the first for the front end web traffic, and the second for the back end traffic (eg web server to data tier). These rules will depend on the network architecture (where your servers are placed) and traffic flows (which direction the traffic flows, and which ports are used).

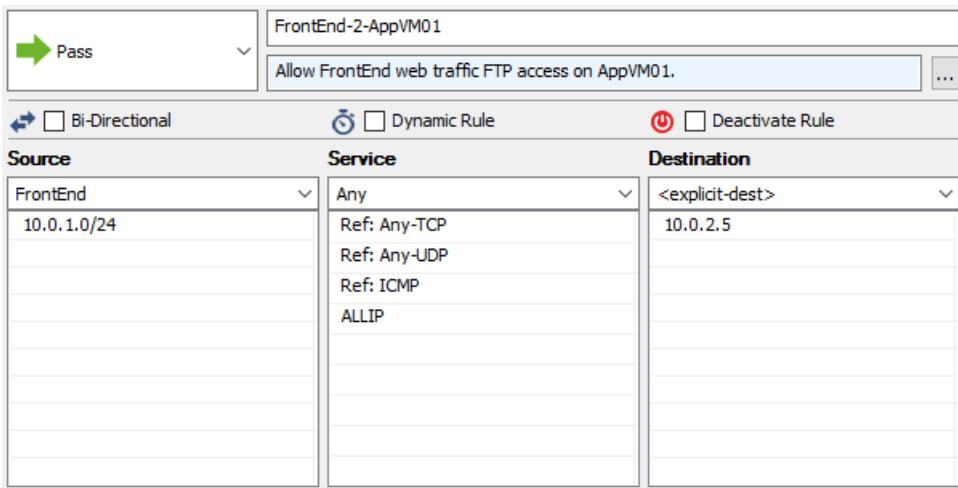
First discussed is the front end rule for web traffic:



This Destination NAT rule allows the actual application traffic to reach the application server. While the other rules allow for security, management, etc., Application Rules are what allow external users or services to access the application(s). For this example, there is a single web server on port 80, thus the single firewall application rule will redirect inbound traffic to the external IP, to the web servers internal IP address.

Note: that there is no port assigned in the Target List field, thus the inbound port 80 (or 443 for the Service selected) will be used in the redirection of the web server. If the web server is listening on a different port, for example port 8080, the Target List field could be updated to 10.0.1.4:8080 to allow for the Port redirection as well.

The next Application Traffic Rule is the back end rule to allow the Web Server to talk to the AppVM01 server (but not AppVM02) via Any service:



This Pass rule allows any IIS server on the Frontend subnet to reach the AppVM01 (IP Address 10.0.2.5) on Any port, using any Protocol to access data needed by the web application.

In this screen shot an "<explicit-dest>" is used in the Destination field to signify 10.0.2.5 as the destination. This could be either explicit as shown or a named Network Object (as was done in the prerequisites for the DNS server). This is up to the administrator of the firewall as to which method will be used. To add 10.0.2.5 as an Explicit Destination, double-click on the first blank row under <explicit-dest> and enter the address in the window that pops up.

With this Pass Rule, no NAT is needed since this is internal traffic, so the Connection Method can be set to "No SNAT".

Note: The Source network in this rule is any resource on the FrontEnd subnet, if there will only be one, or a known specific number of web servers, a Network Object resource could be created to be more specific to those exact IP addresses instead of the entire Frontend subnet.

TIP

This rule uses the service "Any" to make the sample application easier to setup and use, this will also allow ICMPv4 (ping) in a single rule. However, this is not a recommended practice. The ports and protocols ("Services") should be narrowed to the minimum possible that allows application operation to reduce the attack surface across this boundary.

TIP

Although this rule shows an explicit-dest reference being used, a consistent approach should be used throughout the firewall configuration. It is recommended that the named Network Object be used throughout for easier readability and supportability. The explicit-dest is used here only to show an alternative reference method and is not generally recommended (especially for complex configurations).

- **Outbound to Internet Rule:** This Pass rule will allow traffic from any Source network to pass to the selected Destination networks. This rule is a default rule usually already on the Barracuda NextGen firewall, but is in a disabled state. Right-clicking on this rule can access the Activate Rule command. The rule shown here has been modified to add the two local subnets that were created as references in the prerequisite section of this document to the Source attribute of this rule.

 Pass	LAN-2-INTERNET	
Allows internet access from Trusted LAN for typical applications.		
<input type="checkbox"/> Bi-Directional		<input type="checkbox"/> Dynamic Rule
Source	Service	Destination
<explicit-src>	Any	<explicit-dest>
Ref: Backend Ref: FrontEnd	Ref: Any-TCP Ref: Any-UDP Ref: ICMP ALLIP	Ref: Internet

- **DNS Rule:** This Pass rule allows only DNS (port 53) traffic to pass to the DNS server. For this environment most traffic from the FrontEnd to the BackEnd is blocked, this rule specifically allows DNS.

 Pass	DNS	
Allow local DNS traffic from the Front End subnet to the DNS server on the Back End		
<input type="checkbox"/> Bi-Directional		<input type="checkbox"/> Dynamic Rule
Source	Service	Destination
FrontEnd 10.0.1.0/24	DNS UDP 53 dns Report if not (STD-D... TCP 53 dns Report if not (STD-D...	DNS01 10.0.2.4
Authenticated User	Policy	Connection Method
Any	IPS Policy Default Policy Application Policy AppControl, URL.Fil Schedule Always QoS Band (Fwd) VoIP (ID 2) QoS Band (Reply) Like-Fwd	No SNAT Std Client
OK Cancel		

Note: In this screen shot the Connection Method is included. Because this rule is for internal IP to internal IP address traffic, no NATing is required, this the Connection Method is set to "No SNAT" for this Pass rule.

- **Subnet to Subnet Rule:** This Pass rule is a default rule that has been activated and modified to allow any server on the back end subnet to connect to any server on the front end subnet. This rule is all internal traffic so the Connection Method can be set to No SNAT.

Pass	LAN-2-LAN	Allows unrestricted communication between hosts on the Trusted LAN networks.	...
<input type="checkbox"/> Bi-Directional <input type="checkbox"/> Dynamic Rule <input type="checkbox"/> Deactivate Rule			
Source	Service	Destination	
Backend 10.0.2.0/24	Any Ref: Any-TCP Ref: Any-UDP Ref: ICMP ALLIP	FrontEnd 10.0.1.0/24	
Authenticated User	Policy	Connection Method	
Any	IPS Policy Default Policy Application Policy AppControl, URL.Fil Schedule Always QoS Band (Fwd) Business (ID 3) QoS Band (Reply) Like-Fwd	No SNAT Std Client	

Note: The Bi-directional checkbox is not checked (nor is it checked in most rules), this is significant for this rule in that it makes this rule "one directional", a connection can be initiated from the back end subnet to the front end network, but not the reverse. If that checkbox was checked, this rule would enable bi-directional traffic, which from our logical diagram is not desired.

- **Deny All Traffic Rule:** This should always be the final rule (in terms of priority), and as such if a traffic flows fails to match any of the preceding rules it will be dropped by this rule. This is a default rule and usually activated, no modifications are generally needed.

Block	BLOCKALL	Blocks all IP traffic.	...
<input type="checkbox"/> Bi-Directional <input type="checkbox"/> Dynamic Rule <input type="checkbox"/> Deactivate Rule			
Source	Service	Destination	
Any 0.0.0.0/0	Any Ref: Any-TCP Ref: Any-UDP Ref: ICMP ALLIP	Any 0.0.0.0/0	

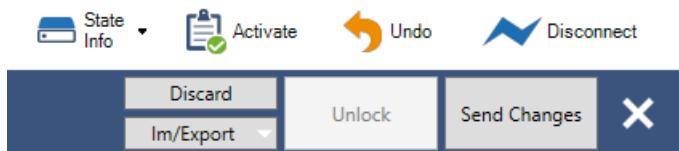
IMPORTANT

Once all of the above rules are created, it's important to review the priority of each rule to ensure traffic will be allowed or denied as desired. For this example, the rules are in the order they should appear in the Main Grid of forwarding rules in the Barracuda Management Client.

Rule Activation

With the ruleset modified to the specification of the logic diagram, the ruleset must be uploaded to the firewall and

then activated.



In the upper right hand corner of the management client are a cluster of buttons. Click the "Send Changes" button to send the modified rules to the firewall, then click the "Activate" button.

With the activation of the firewall ruleset this example environment build is complete.

Traffic Scenarios

IMPORTANT

A key takeaway is to remember that **all** traffic will come through the firewall. So to remote desktop to the IIS01 server, even though it's in the Front End Cloud Service and on the Front End subnet, to access this server we will need to RDP to the firewall on port 8014, and then allow the firewall to route the RDP request internally to the IIS01 RDP Port. The Azure portal's "Connect" button won't work because there is no direct RDP path to IIS01 (as far as the portal can see). This means all connections from the internet will be to the Security Service and a Port, e.g. secscv001.cloudapp.net:xxxx.

For these scenarios, the following firewall rules should be in place:

1. Firewall Management
2. RDP to IIS01
3. RDP to DNS01
4. RDP to AppVM01
5. RDP to AppVM02
6. App Traffic to the Web
7. App Traffic to AppVM01
8. Outbound to the Internet
9. Frontend to DNS01
10. Intra-Subnet Traffic (back end to front end only)
11. Deny All

The actual firewall ruleset will most likely have many other rules in addition to these, the rules on any given firewall will also have different priority numbers than the ones listed here. This list and associated numbers are to provide relevance between just these eleven rules and the relative priority amongst them. In other words; on the actual firewall, the "RDP to IIS01" may be rule number 5, but as long as it's below the "Firewall Management" rule and above the "RDP to DNS01" rule it would align with the intention of this list. The list will also aid in the below scenarios allowing brevity; e.g. "FW Rule 9 (DNS)". Also for brevity, the four RDP rules will be collectively called, "the RDP rules" when the traffic scenario is unrelated to RDP.

Also recall that Network Security Groups are in-place for inbound internet traffic on the Frontend and Backend subnets.

(Allowed) Internet to Web Server

1. Internet user requests HTTP page from SecSvc001.CloudApp.Net (Internet Facing Cloud Service)
2. Cloud service passes traffic through open endpoint on port 80 to firewall interface on 10.0.0.4:80
3. No NSG assigned to Security subnet, so system NSG rules allow traffic to firewall
4. Traffic hits internal IP address of the firewall (10.0.1.4)
5. Firewall begins rule processing:

- a. FW Rule 1 (FW Mgmt) doesn't apply, move to next rule
 - b. FW Rules 2 - 5 (RDP Rules) don't apply, move to next rule
 - c. FW Rule 6 (App: Web) does apply, traffic is allowed, firewall NATs it to 10.0.1.4 (IIS01)
6. The Frontend subnet begins inbound rule processing:
- a. NSG Rule 1 (Block Internet) doesn't apply (this traffic was NAT'd by the firewall, thus the source address is now the firewall which is on the Security subnet and seen by the Frontend subnet NSG to be "local" traffic and is thus allowed), move to next rule
 - b. Default NSG Rules allow subnet to subnet traffic, traffic is allowed, stop NSG rule processing
7. IIS01 is listening for web traffic, receives this request and starts processing the request
8. IIS01 attempts to initiates an FTP session to AppVM01 on Backend subnet
9. The UDR route on Frontend subnet makes the firewall the next hop
10. No outbound rules on Frontend subnet, traffic is allowed
11. Firewall begins rule processing:
- a. FW Rule 1 (FW Mgmt) doesn't apply, move to next rule
 - b. FW Rule 2 - 5 (RDP Rules) don't apply, move to next rule
 - c. FW Rule 6 (App: Web) doesn't apply, move to next rule
 - d. FW Rule 7 (App: Backend) does apply, traffic is allowed, firewall forwards traffic to 10.0.2.5 (AppVM01)
12. The Backend subnet begins inbound rule processing:
- a. NSG Rule 1 (Block Internet) doesn't apply, move to next rule
 - b. Default NSG Rules allow subnet to subnet traffic, traffic is allowed, stop NSG rule processing
13. AppVM01 receives the request and initiates the session and responds
14. The UDR route on Backend subnet makes the firewall the next hop
15. Since there are no outbound NSG rules on the Backend subnet the response is allowed
16. Because this is returning traffic on an established session the firewall passes the response back to the web server (IIS01)
17. Frontend subnet begins inbound rule processing:
- a. NSG Rule 1 (Block Internet) doesn't apply, move to next rule
 - b. Default NSG Rules allow subnet to subnet traffic, traffic is allowed, stop NSG rule processing
18. The IIS server receives the response, completes the transaction with AppVM01, and then completes building the HTTP response, this HTTP response is sent to the requestor
19. Since there are no outbound NSG rules on the Frontend subnet the response is allowed
20. The HTTP response hits the firewall, and because this is the response to an established NAT session is accepted by the firewall
21. The firewall then redirects the response back to the Internet User
22. Since there are no outbound NSG rules or UDR hops on the Frontend subnet the response is allowed, and the Internet User receives the web page requested.

(Allowed) Internet RDP to Backend

1. Server Admin on internet requests RDP session to AppVM01 via SecSvc001.CloudApp.Net:8025, where 8025 is the user assigned port number for the "RDP to AppVM01" firewall rule
2. The cloud service passes traffic through the open endpoint on port 8025 to firewall interface on 10.0.0.4:8025
3. No NSG assigned to Security subnet, so system NSG rules allow traffic to firewall
4. Firewall begins rule processing:
 - a. FW Rule 1 (FW Mgmt) doesn't apply, move to next rule
 - b. FW Rule 2 (RDP IIS) doesn't apply, move to next rule
 - c. FW Rule 3 (RDP DNS01) doesn't apply, move to next rule
 - d. FW Rule 4 (RDP AppVM01) does apply, traffic is allowed, firewall NATs it to 10.0.2.5:3386 (RDP port on AppVM01)

5. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (Block Internet) doesn't apply (this traffic was NAT'd by the firewall, thus the source address is now the firewall which is on the Security subnet and seen by the Backend subnet NSG to be "local" traffic and is thus allowed), move to next rule
 - b. Default NSG Rules allow subnet to subnet traffic, traffic is allowed, stop NSG rule processing
6. AppVM01 is listening for RDP traffic and responds
7. With no outbound NSG rules, default rules apply and return traffic is allowed
8. UDR routes outbound traffic to the firewall as the next hop
9. Because this is returning traffic on an established session the firewall passes the response back to the internet user
10. RDP session is enabled
11. AppVM01 prompts for user name password

(Allowed) Web Server DNS lookup on DNS server

1. Web Server, IIS01, needs a data feed at www.data.gov, but needs to resolve the address.
2. The network configuration for the VNet lists DNS01 (10.0.2.4 on the Backend subnet) as the primary DNS server, IIS01 sends the DNS request to DNS01
3. UDR routes outbound traffic to the firewall as the next hop
4. No outbound NSG rules are bound to the Frontend subnet, traffic is allowed
5. Firewall begins rule processing:
 - a. FW Rule 1 (FW Mgmt) doesn't apply, move to next rule
 - b. FW Rule 2 - 5 (RDP Rules) don't apply, move to next rule
 - c. FW Rules 6 & 7 (App Rules) don't apply, move to next rule
 - d. FW Rule 8 (To Internet) doesn't apply, move to next rule
 - e. FW Rule 9 (DNS) does apply, traffic is allowed, firewall forwards traffic to 10.0.2.4 (DNS01)
6. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (Block Internet) doesn't apply, move to next rule
 - b. Default NSG Rules allow subnet to subnet traffic, traffic is allowed, stop NSG rule processing
7. DNS server receives the request
8. DNS server doesn't have the address cached and asks a root DNS server on the internet
9. UDR routes outbound traffic to the firewall as the next hop
10. No outbound NSG rules on Backend subnet, traffic is allowed
11. Firewall begins rule processing:
 - a. FW Rule 1 (FW Mgmt) doesn't apply, move to next rule
 - b. FW Rule 2 - 5 (RDP Rules) don't apply, move to next rule
 - c. FW Rules 6 & 7 (App Rules) don't apply, move to next rule
 - d. FW Rule 8 (To Internet) does apply, traffic is allowed, session is SNAT out to root DNS server on the Internet
12. Internet DNS server responds, since this session was initiated from the firewall, the response is accepted by the firewall
13. As this is an established session, the firewall forwards the response to the initiating server, DNS01
14. The Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (Block Internet) doesn't apply, move to next rule
 - b. Default NSG Rules allow subnet to subnet traffic, traffic is allowed, stop NSG rule processing
15. The DNS server receives and caches the response, and then responds to the initial request back to IIS01
16. The UDR route on backend subnet makes the firewall the next hop
17. No outbound NSG rules exist on the Backend subnet, traffic is allowed
18. This is an established session on the firewall, the response is forwarded by the firewall back to the IIS server

19. Frontend subnet begins inbound rule processing:
 - a. There is no NSG rule that applies to Inbound traffic from the Backend subnet to the Frontend subnet, so none of the NSG rules apply
 - b. The default system rule allowing traffic between subnets would allow this traffic so the traffic is allowed
20. IIS01 receives the response from DNS01

(Allowed) Backend server to Frontend server

1. An administrator logged on to AppVM02 via RDP requests a file directly from the IIS01 server via windows file explorer
2. The UDR route on Backend subnet makes the firewall the next hop
3. Since there are no outbound NSG rules on the Backend subnet the response is allowed
4. Firewall begins rule processing:
 - a. FW Rule 1 (FW Mgmt) doesn't apply, move to next rule
 - b. FW Rule 2 - 5 (RDP Rules) don't apply, move to next rule
 - c. FW Rules 6 & 7 (App Rules) don't apply, move to next rule
 - d. FW Rule 8 (To Internet) doesn't apply, move to next rule
 - e. FW Rule 9 (DNS) doesn't apply, move to next rule
 - f. FW Rule 10 (Intra-Subnet) does apply, traffic is allowed, firewall passes traffic to 10.0.1.4 (IIS01)
5. Frontend subnet begins inbound rule processing:
 - a. NSG Rule 1 (Block Internet) doesn't apply, move to next rule
 - b. Default NSG Rules allow subnet to subnet traffic, traffic is allowed, stop NSG rule processing
6. Assuming proper authentication and authorization, IIS01 accepts the request and responds
7. The UDR route on Frontend subnet makes the firewall the next hop
8. Since there are no outbound NSG rules on the Frontend subnet the response is allowed
9. As this is an existing session on the firewall this response is allowed and the firewall returns the response to AppVM02
10. Backend subnet begins inbound rule processing:
 - a. NSG Rule 1 (Block Internet) doesn't apply, move to next rule
 - b. Default NSG Rules allow subnet to subnet traffic, traffic is allowed, stop NSG rule processing
11. AppVM02 receives the response

(Denied) Internet direct to Web Server

1. Internet user tries to access the web server, IIS01, through the FrontEnd001.CloudApp.Net service
2. Since there are no endpoints open for HTTP traffic, this would not pass through the Cloud Service and wouldn't reach the server
3. If the endpoints were open for some reason, the NSG (Block Internet) on the Frontend subnet would block this traffic
4. Finally, the Frontend subnet UDR route would send any outbound traffic from IIS01 to the firewall as the next hop, and the firewall would see this as asymmetric traffic and drop the outbound response Thus there are at least three independent layers of defense between the internet and IIS01 via its cloud service preventing unauthorized/inappropriate access.

(Denied) Internet to Backend Server

1. Internet user tries to access a file on AppVM01 through the BackEnd001.CloudApp.Net service
2. Since there are no endpoints open for file share, this would not pass the Cloud Service and wouldn't reach the server
3. If the endpoints were open for some reason, the NSG (Block Internet) would block this traffic
4. Finally, the UDR route would send any outbound traffic from AppVM01 to the firewall as the next hop, and the firewall would see this as asymmetric traffic and drop the outbound response Thus there are at least three independent layers of defense between the internet and AppVM01 via its cloud service preventing

unauthorized/inappropriate access.

(Denied) Frontend server to Backend Server

1. Assume IIS01 was compromised and is running malicious code trying to scan the Backend subnet servers.
2. The Frontend subnet UDR route would send any outbound traffic from IIS01 to the firewall as the next hop. This is not something that can be altered by the compromised VM.
3. The firewall would process the traffic, if the request was to AppVM01, or to the DNS server for DNS lookups that traffic could potentially be allowed by the firewall (due to FW Rules 7 and 9). All other traffic would be blocked by FW Rule 11 (Deny All).
4. If advanced threat detection was enabled on the firewall (which is not covered in this document, see the vendor documentation for your specific network appliance advanced threat capabilities), even traffic that would be allowed by the basic forwarding rules discussed in this document could be prevented if the traffic contained known signatures or patterns that flag an advanced threat rule.

(Denied) Internet DNS lookup on DNS server

1. Internet user tries to lookup an internal DNS record on DNS01 through BackEnd001.CloudApp.Net service
2. Since there are no endpoints open for DNS traffic, this would not pass through the Cloud Service and wouldn't reach the server
3. If the endpoints were open for some reason, the NSG rule (Block Internet) on the Frontend subnet would block this traffic
4. Finally, the Backend subnet UDR route would send any outbound traffic from DNS01 to the firewall as the next hop, and the firewall would see this as asymmetric traffic and drop the outbound response. Thus there are at least three independent layers of defense between the internet and DNS01 via its cloud service preventing unauthorized/inappropriate access.

(Denied) Internet to SQL access through Firewall

1. Internet user requests SQL data from SecSvc001.CloudApp.Net (Internet Facing Cloud Service)
2. Since there are no endpoints open for SQL, this would not pass the Cloud Service and wouldn't reach the firewall
3. If SQL endpoints were open for some reason, the firewall would begin rule processing:
 - a. FW Rule 1 (FW Mgmt) doesn't apply, move to next rule
 - b. FW Rules 2 - 5 (RDP Rules) don't apply, move to next rule
 - c. FW Rule 6 & 7 (Application Rules) don't apply, move to next rule
 - d. FW Rule 8 (To Internet) doesn't apply, move to next rule
 - e. FW Rule 9 (DNS) doesn't apply, move to next rule
 - f. FW Rule 10 (Intra-Subnet) doesn't apply, move to next rule
 - g. FW Rule 11 (Deny All) does apply, traffic is blocked, stop rule processing

References

Main Script and Network Config

Save the Full Script in a PowerShell script file. Save the Network Config into a file named "NetworkConf2.xml". Modify the user defined variables as needed. Run the script, then follow the Firewall rule setup instruction above.

Full Script

This script will, based on the user defined variables:

1. Connect to an Azure subscription
2. Create a new storage account
3. Create a new VNet and three subnets as defined in the Network Config file
4. Build five virtual machines; 1 firewall and 4 windows server VMs
5. Configure UDR including:

- a. Creating two new route tables
 - b. Add routes to the tables
 - c. Bind tables to appropriate subnets
6. Enable IP Forwarding on the NVA
7. Configure NSG including:
- a. Creating a NSG
 - b. Adding a rule
 - c. Binding the NSG to the appropriate subnets

This PowerShell script should be run locally on an internet connected PC or server.

IMPORTANT

When this script is run, there may be warnings or other informational messages that pop in PowerShell. Only error messages in red are cause for concern.

```
<#
.SYNOPSIS
Example of DMZ and User Defined Routing in an isolated network (Azure only, no hybrid connections)

.DESCRIPTION
This script will build out a sample DMZ setup containing:
- A default storage account for VM disks
- Three new cloud services
- Three Subnets (SecNet, FrontEnd, and BackEnd subnets)
- A Network Virtual Appliance (NVA), in this case a Barracuda NextGen Firewall
- One server on the FrontEnd Subnet
- Three Servers on the BackEnd Subnet
- IP Forwading from the FireWall out to the internet
- User Defined Routing FrontEnd and BackEnd Subnets to the NVA

Before running script, ensure the network configuration file is created in
the directory referenced by $NetworkConfigFile variable (or update the
variable to reflect the path and file name of the config file being used).

.Notes
Everyone's security requirements are different and can be addressed in a myriad of ways.
Please be sure that any sensitive data or applications are behind the appropriate
layer(s) of protection. This script serves as an example of some of the techniques
that can be used, but should not be used for all scenarios. You are responsible to
assess your security needs and the appropriate protections needed, and then effectively
implement those protections.

Security Service (SecNet subnet 10.0.0.0/24)
myFirewall - 10.0.0.4

FrontEnd Service (FrontEnd subnet 10.0.1.0/24)
IIS01      - 10.0.1.4

BackEnd Service (BackEnd subnet 10.0.2.0/24)
DNS01      - 10.0.2.4
AppVM01    - 10.0.2.5
AppVM02    - 10.0.2.6

#>

# Fixed Variables
$LocalAdminPwd = Read-Host -Prompt "Enter Local Admin Password to be used for all VMs"
$VMName = @()
$ServiceName = @()
$VMFamily = @()
$img = @()
```

```

$size = @()
$SubnetName = @()
$VMIP = @()

# User Defined Global Variables
# These should be changes to reflect your subscription and services
# Invalid options will fail in the validation section

# Subscription Access Details
$subID = "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"

# VM Account, Location, and Storage Details
$LocalAdmin = "theAdmin"
$DeploymentLocation = "Central US"
$StorageAccountName = "vmstore02"

# Service Details
$SecureService = "SecSvc001"
$FrontEndService = "FrontEnd001"
$BackEndService = "BackEnd001"

# Network Details
$VNetName = "CorpNetwork"
$VNetPrefix = "10.0.0.0/16"
$SecNet = "SecNet"
$FESubnet = "FrontEnd"
$FEPrefix = "10.0.1.0/24"
$BESubnet = "BackEnd"
$BEPrefix = "10.0.2.0/24"
$NetworkConfigFile = "C:\Scripts\NetworkConf3.xml"

# VM Base Disk Image Details
$SrvImg = Get-AzureVMImage | Where {$_.ImageFamily -match 'Windows Server 2012 R2 Datacenter'} | sort PublishedDate -Descending | Select ImageName -First 1 | ForEach {$_.ImageName}
$FWImg = Get-AzureVMImage | Where {$_.ImageFamily -match 'Barracuda NextGen Firewall'} | sort PublishedDate -Descending | Select ImageName -First 1 | ForEach {$_.ImageName}

# UDR Details
$FERouteTableName = "FrontEndSubnetRouteTable"
$BERouteTableName = "BackEndSubnetRouteTable"

# NSG Details
$NSGName = "MyVNetSG"

# User Defined VM Specific Config
# Note: To ensure UDR and IP forwarding is setup
# properly this script requires VM 0 be the NVA.

# VM 0 - The Network Virtual Appliance (NVA)
$VMName += "myFirewall"
$ServiceName += $SecureService
$VMFamily += "Firewall"
$img += $FWImg
$size += "Small"
$SubnetName += $SecNet
$VMIP += "10.0.0.4"

# VM 1 - The Web Server
$VMName += "IIS01"
$ServiceName += $FrontEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $FESubnet
$VMIP += "10.0.1.4"

# VM 2 - The First Application Server
$VMName += "AppVM01"
$ServiceName += $BackEndService

```

```

$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.5"

# VM 3 - The Second Application Server
$VMName += "AppVM02"
$ServiceName += $BackEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.6"

# VM 4 - The DNS Server
$VMName += "DNS01"
$ServiceName += $BackEndService
$VMFamily += "Windows"
$img += $SrvImg
$size += "Standard_D3"
$SubnetName += $BESubnet
$VMIP += "10.0.2.4"

# ----- #
# No User Defined Variables or   #
# Configuration past this point #
# ----- #

# Get your Azure accounts
Add-AzureAccount
Set-AzureSubscription -SubscriptionId $subID -ErrorAction Stop
Select-AzureSubscription -SubscriptionId $subID -Current -ErrorAction Stop

# Create Storage Account
If (Test-AzureName -Storage -Name $StorageAccountName) {
    Write-Host "Fatal Error: This storage account name is already in use, please pick a different name." -ForegroundColor Red
    Return}
Else {Write-Host "Creating Storage Account" -ForegroundColor Cyan
      New-AzureStorageAccount -Location $DeploymentLocation -StorageAccountName $StorageAccountName}

# Update Subscription Pointer to New Storage Account
Write-Host "Updating Subscription Pointer to New Storage Account" -ForegroundColor Cyan
Set-AzureSubscription -SubscriptionId $subID -CurrentStorageAccountName $StorageAccountName -ErrorAction Stop

# Validation
$FatalError = $false

If (-Not (Get-AzureLocation | Where {$_.DisplayName -eq $DeploymentLocation})) {
    Write-Host "This Azure Location was not found or available for use" -ForegroundColor Yellow
    $FatalError = $true}

If (Test-AzureName -Service -Name $SecureService) {
    Write-Host "The SecureService service name is already in use, please pick a different service name." -ForegroundColor Yellow
    $FatalError = $true}
Else { Write-Host "The FrontEndService service name is valid for use." -ForegroundColor Green}

If (Test-AzureName -Service -Name $FrontEndService) {
    Write-Host "The FrontEndService service name is already in use, please pick a different service name." -ForegroundColor Yellow
    $FatalError = $true}
Else { Write-Host "The FrontEndService service name is valid for use" -ForegroundColor Green}

If (Test-AzureName -Service -Name $BackEndService) {
    Write-Host "The BackEndService service name is already in use, please pick a different service name." -ForegroundColor Yellow
    $FatalError = $true}

```

```

$FatalError = $true
Else { Write-Host "The BackEndService service name is valid for use." -ForegroundColor Green}

If (-Not (Test-Path $NetworkConfigFile)) {
    Write-Host 'The network config file was not found, please update the $NetworkConfigFile variable to point to the network config xml file.' -ForegroundColor Yellow
    $FatalError = $true
}
Else { Write-Host "The network config file was found" -ForegroundColor Green
    If (-Not (Select-String -Pattern $DeploymentLocation -Path $NetworkConfigFile)) {
        Write-Host 'The deployment location was not found in the network config file, please check the network config file to ensure the $DeploymentLocation varible is correct and the netowrk config file matches.' -ForegroundColor Yellow
        $FatalError = $true
    }
    Else { Write-Host "The deployment location was found in the network config file." -ForegroundColor Green
}
}

If ($FatalError) {
    Write-Host "A fatal error has occured, please see the above messages for more information." -ForegroundColor Red
    Return
}
Else { Write-Host "Validation passed, now building the environment." -ForegroundColor Green}

# Create VNET
Write-Host "Creating VNET" -ForegroundColor Cyan
Set-AzureVNetConfig -ConfigurationPath $NetworkConfigFile -ErrorAction Stop

# Create Services
Write-Host "Creating Services" -ForegroundColor Cyan
New-AzureService -Location $DeploymentLocation -ServiceName $SecureService -ErrorAction Stop
New-AzureService -Location $DeploymentLocation -ServiceName $FrontEndService -ErrorAction Stop
New-AzureService -Location $DeploymentLocation -ServiceName $BackEndService -ErrorAction Stop

# Build VMs
$i=0
$VMName | Foreach {
    Write-Host "Building $($VMName[$i])" -ForegroundColor Cyan
    If ($VMFamily[$i] -eq "Firewall")
    {
        New-AzureVMConfig -Name $VMName[$i] -ImageName $img[$i] -InstanceSize $size[$i] | ` 
            Add-AzureProvisioningConfig -Linux -LinuxUser $LocalAdmin -Password $LocalAdminPwd | ` 
            Set-AzureSubnet -SubnetNames $SubnetName[$i] | ` 
            Set-AzureStaticVNetIP -IPAddress $VMIP[$i] | ` 
            New-AzureVM -ServiceName $ServiceName[$i] -VNetName $VNetName -Location $DeploymentLocation
        # Set up all the EndPoints we'll need once we're up and running
        # Note: All traffic goes through the firewall, so we'll need to set up all ports here.
        #       Also, the firewall will be redirecting traffic to a new IP and Port in a forwarding rule, so all of these endpoint have the same public and local port and the firewall will do the mapping, NATing, and/or redirection as declared in the firewall rules.
        Add-AzureEndpoint -Name "MgmtPort1" -Protocol tcp -PublicPort 801 -LocalPort 801 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
        Add-AzureEndpoint -Name "MgmtPort2" -Protocol tcp -PublicPort 807 -LocalPort 807 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
        Add-AzureEndpoint -Name "HTTP" -Protocol tcp -PublicPort 80 -LocalPort 80 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
        Add-AzureEndpoint -Name "RDPWeb" -Protocol tcp -PublicPort 8014 -LocalPort 8014 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
        Add-AzureEndpoint -Name "RDPAppl" -Protocol tcp -PublicPort 8025 -LocalPort 8025 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
        Add-AzureEndpoint -Name "RDPAppl2" -Protocol tcp -PublicPort 8026 -LocalPort 8026 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
        Add-AzureEndpoint -Name "RDPDNS01" -Protocol tcp -PublicPort 8024 -LocalPort 8024 -VM (Get-AzureVM -ServiceName $ServiceName[$i] -Name $VMName[$i]) | Update-AzureVM
        # Note: A SSH endpoint is automatically created on port 22 when the appliance is created.
    }
}
Else {
    New-AzureVMConfig -Name $VMName[$i] -ImageName $img[$i] -InstanceSize $size[$i] | ` 
        Add-AzureProvisioningConfig -Windows -AdminUsername $LocalAdmin -Password $LocalAdminPwd | ` 
        Set-AzureSubnet -SubnetNames $SubnetName[$i] | ` 
}

```

```

Set-AzureSubnet -SubnetNames $SubnetName[$i] |
Set-AzureStaticVNetIP -IPAddress $VMIP[$i] |
Set-AzureVMMicrosoftAntimalwareExtension -AntimalwareConfiguration '{"AntimalwareEnabled" :
true}' | `

Remove-AzureEndpoint -Name "RemoteDesktop" | `

Remove-AzureEndpoint -Name "PowerShell" | `

New-AzureVM -ServiceName $ServiceName[$i] -VNetName $VNetName -Location $DeploymentLocation
}

$i++
}

# Configure UDR and IP Forwarding
Write-Host "Configuring UDR" -ForegroundColor Cyan

# Create the Route Tables
Write-Host "Creating the Route Tables" -ForegroundColor Cyan
New-AzureRouteTable -Name $BERouteTableName `

-Location $DeploymentLocation `

-Label "Route table for $BESubnet subnet"
New-AzureRouteTable -Name $FERouteTableName `

-Location $DeploymentLocation `

-Label "Route table for $FESubnet subnet"

# Add Routes to Route Tables
Write-Host "Adding Routes to the Route Tables" -ForegroundColor Cyan
Get-AzureRouteTable $BERouteTableName `

|Set-AzureRoute -RouteName "All traffic to FW" -AddressPrefix 0.0.0.0/0 `

-NextHopType VirtualAppliance `

-NextHopIpAddress $VMIP[0]
Get-AzureRouteTable $BERouteTableName `

|Set-AzureRoute -RouteName "Internal traffic to FW" -AddressPrefix $VNetPrefix `

-NextHopType VirtualAppliance `

-NextHopIpAddress $VMIP[0]
Get-AzureRouteTable $BERouteTableName `

|Set-AzureRoute -RouteName "Allow Intra-Subnet Traffic" -AddressPrefix $BEPrefix `

-NextHopType VNETLocal
Get-AzureRouteTable $FERouteTableName `

|Set-AzureRoute -RouteName "All traffic to FW" -AddressPrefix 0.0.0.0/0 `

-NextHopType VirtualAppliance `

-NextHopIpAddress $VMIP[0]
Get-AzureRouteTable $FERouteTableName `

|Set-AzureRoute -RouteName "Internal traffic to FW" -AddressPrefix $VNetPrefix `

-NextHopType VirtualAppliance `

-NextHopIpAddress $VMIP[0]
Get-AzureRouteTable $FERouteTableName `

|Set-AzureRoute -RouteName "Allow Intra-Subnet Traffic" -AddressPrefix $FEPPrefix `

-NextHopType VNETLocal

# Associate the Route Tables with the Subnets
Write-Host "Binding Route Tables to the Subnets" -ForegroundColor Cyan
Set-AzureSubnetRouteTable -VirtualNetworkName $VNetName `

-SubnetName $BESubnet `

-RouteTableName $BERouteTableName
Set-AzureSubnetRouteTable -VirtualNetworkName $VNetName `

-SubnetName $FESubnet `

-RouteTableName $FERouteTableName

# Enable IP Forwarding on the Virtual Appliance
Get-AzureVM -Name $VMName[0] -ServiceName $ServiceName[0] `

|Set-AzureIPForwarding -Enable

# Configure NSG
Write-Host "Configuring the Network Security Group (NSG)" -ForegroundColor Cyan

# Build the NSG
Write-Host "Building the NSG" -ForegroundColor Cyan
New-AzureNetworkSecurityGroup -Name $NSGName -Location $DeploymentLocation -Label "Security group for
$VNetName subnets in $DeploymentLocation"

```

```

# Add NSG Rule
Write-Host "Writing rules into the NSG" -ForegroundColor Cyan
Get-AzureNetworkSecurityGroup -Name $NSGName | Set-AzureNetworkSecurityRule -Name "Isolate the $VNetName
VNet from the Internet" -Type Inbound -Priority 100 -Action Deny `

    -SourceAddressPrefix INTERNET -SourcePortRange '*' `

    -DestinationAddressPrefix VIRTUAL_NETWORK -DestinationPortRange '*' `

    -Protocol *

# Assign the NSG to two Subnets
# The NSG is *not* bound to the Security Subnet. The result
# is that internet traffic flows only to the Security subnet
# since the NSG bound to the Frontend and Backback subnets
# will Deny internet traffic to those subnets.
Write-Host "Binding the NSG to two subnets" -ForegroundColor Cyan
Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName -SubnetName $FESubnet -VirtualNetworkName $VNetName
Set-AzureNetworkSecurityGroupToSubnet -Name $NSGName -SubnetName $BESubnet -VirtualNetworkName $VNetName

# Optional Post-script Manual Configuration
# Configure Firewall
# Install Test Web App (Run Post-Build Script on the IIS Server)
# Install Backend resource (Run Post-Build Script on the AppVM01)
Write-Host
Write-Host "Build Complete!" -ForegroundColor Green
Write-Host
Write-Host "Optional Post-script Manual Configuration Steps" -ForegroundColor Gray
Write-Host "- Configure Firewall" -ForegroundColor Gray
Write-Host "- Install Test Web App (Run Post-Build Script on the IIS Server)" -ForegroundColor Gray
Write-Host "- Install Backend resource (Run Post-Build Script on the AppVM01)" -ForegroundColor Gray
Write-Host

```

Network Config File

Save this xml file with updated location and add the link to this file to the \$NetworkConfigFile variable in the script above.

```

<NetworkConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/ServiceHosting/2011/07/NetworkConfiguration">
    <VirtualNetworkConfiguration>
        <Dns>
            <DnsServers>
                <DnsServer name="DNS01" IPAddress="10.0.2.4" />
                <DnsServer name="Level3" IPAddress="209.244.0.3" />
            </DnsServers>
        </Dns>
        <VirtualNetworkSites>
            <VirtualNetworkSite name="CorpNetwork" Location="Central US">
                <AddressSpace>
                    <AddressPrefix>10.0.0.0/16</AddressPrefix>
                </AddressSpace>
                <Subnets>
                    <Subnet name="SecNet">
                        <AddressPrefix>10.0.0.0/24</AddressPrefix>
                    </Subnet>
                    <Subnet name="FrontEnd">
                        <AddressPrefix>10.0.1.0/24</AddressPrefix>
                    </Subnet>
                    <Subnet name="BackEnd">
                        <AddressPrefix>10.0.2.0/24</AddressPrefix>
                    </Subnet>
                </Subnets>
                <DnsServersRef>
                    <DnsServerRef name="DNS01" />
                    <DnsServerRef name="Level3" />
                </DnsServersRef>
            </VirtualNetworkSite>
        </VirtualNetworkSites>
    </VirtualNetworkConfiguration>
</NetworkConfiguration>

```

Sample Application Scripts

If you wish to install a sample application for this, and other DMZ Examples, one has been provided at the following link: [Sample Application Script](#)

Sample application for use with DMZs

6/27/2017 • 5 min to read • [Edit Online](#)

[Return to the Security Boundary Best Practices Page](#)

These PowerShell scripts can be run locally on the IIS01 and AppVM01 servers to install and set up a simple web application that displays an html page from the front-end IIS01 server with content from the back-end AppVM01 server.

This application provides a simple testing environment for many of the DMZ Examples and how changes on the Endpoints, NSGs, UDR, and Firewall rules can affect traffic flows.

Firewall rule to allow ICMP

This simple PowerShell statement can be run on any Windows VM to allow ICMP (Ping) traffic. This firewall update allows for easier testing and troubleshooting by allowing the ping protocol to pass through the windows firewall (for most Linux distros ICMP is on by default).

```
# Turn On ICMPv4
New-NetFirewallRule -Name Allow_ICMPv4 -DisplayName "Allow ICMPv4" ` 
    -Protocol ICMPv4 -Enabled True -Profile Any -Action Allow
```

If you use the following scripts, this firewall rule addition is the first statement.

IIS01 - Web application installation script

This script will:

1. Open IMCPv4 (Ping) on the local server windows firewall for easier testing
2. Install IIS and the .Net Framework v4.5
3. Create an ASP.NET web page and a Web.config file
4. Change the Default application pool to make file access easier
5. Set the Anonymous user to your admin account and password

This PowerShell script should be run locally while RDP'd into IIS01.

```
# IIS Server Post Build Config Script
# Get Admin Account and Password
Write-Host "Please enter the admin account information used to create this VM:" -ForegroundColor Cyan
$theAdmin = Read-Host -Prompt "The Admin Account Name (no domain or machine name)"
$password = Read-Host -Prompt "The Admin Password"

# Turn On ICMPv4
Write-Host "Creating ICMP Rule in Windows Firewall" -ForegroundColor Cyan
New-NetFirewallRule -Name Allow_ICMPv4 -DisplayName "Allow ICMPv4" -Protocol ICMPv4 -Enabled True -Profile Any -Action Allow

# Install IIS
Write-Host "Installing IIS and .Net 4.5, this can take some time, like 15+ minutes..." -ForegroundColor Cyan
add-windowsfeature Web-Server, Web-WebServer, Web-Common-Http, Web-Default-Doc, Web-Dir-Browsing, Web-Http-Errors, Web-Static-Content, Web-Health, Web-Http-Logging, Web-Performance, Web-Stat-Compression, Web-Security, Web-Filtering, Web-App-Dev, Web-ISAPI-Ext, Web-ISAPI-Filter, Web-Net-Ext, Web-Net-Ext45, Web-AspNet45, Web-Mgmt-Tools, Web-Mgmt-Console
```

```

# Create Web App Pages
Write-Host "Creating Web page and Web.Config file" -ForegroundColor Cyan
$MainPage = '<%@ Page Language="vb" AutoEventWireup="false" %>
<%@ Import Namespace="System.IO" %>
<script language="vb" runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        Dim FILENAME As String = "\\\\" + $env:COMPUTERNAME + "\\WebShare\\Rand.txt"
        Dim objStreamReader As StreamReader
        objStreamReader = File.OpenText(FILENAME)
        Dim contents As String = objStreamReader.ReadToEnd()
        lblOutput.Text = contents
        objStreamReader.Close()
        lblTime.Text = Now()
    End Sub
</script>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>DMZ Example App</title>
</head>
<body style="font-family: Optima,Segoe,Segoe UI,Candara,Calibri,Arial,sans-serif;">
    <form id="frmMain" runat="server">
        <div>
            <h1>Looks like you made it!</h1>
            This is a page from the inside (a web server on a private network),<br />
            and it is making its way to the outside! (If you are viewing this from the internet)<br />
            <br />
            The following sections show:
            <ul style="margin-top: 0px;">
                <li> Local Server Time - Shows if this page is or isn't cached anywhere</li>
                <li> File Output - Shows that the web server is reaching AppVM01 on the backend subnet and
                    successfully returning content</li>
                <li> Image from the Internet - Doesn't really show anything, but it made me happy to see this when
                    the app worked</li>
            </ul>
            <div style="border: 2px solid #8AC007; border-radius: 25px; padding: 20px; margin: 10px; width:
650px;">
                <b>Local Web Server Time</b>: <asp:Label runat="server" ID="lblTime" /></div>
                <div style="border: 2px solid #8AC007; border-radius: 25px; padding: 20px; margin: 10px; width:
650px;">
                    <b>File Output from AppVM01</b>: <asp:Label runat="server" ID="lblOutput" /></div>
                    <div style="border: 2px solid #8AC007; border-radius: 25px; padding: 20px; margin: 10px; width:
650px;">
                        <b>Image File Linked from the Internet</b>:<br />
                        <br />
                        
                        <br />
                    </div>
                </form>
            </body>
        </html>'
```

\$WebConfig = '<?xml version="1.0" encoding="utf-8"?>
<configuration>
 <system.web>
 <compilation debug="true" strict="false" explicit="true" targetFramework="4.5" />
 <httpRuntime targetFramework="4.5" />
 <identity impersonate="true" />
 <customErrors mode="Off"/>
 </system.web>
 <system.webServer>
 <defaultDocument>
 <files>
 <add value="Home.aspx" />
 </files>
 </defaultDocument>
 </system.webServer>
 <!-- Other configuration sections -->
</configuration>'

```

</configuration>

$MainPage | Out-File -FilePath "C:\inetpub\wwwroot\Home.aspx" -Encoding ascii
$WebConfig | Out-File -FilePath "C:\inetpub\wwwroot\Web.config" -Encoding ascii

# Set App Pool to Clasic Pipeline to remote file access will work easier
Write-Host "Updaing IIS Settings" -ForegroundColor Cyan
c:\windows\system32\inetsrv\appcmd.exe set app "Default Web Site/" /applicationPool:".NET v4.5 Classic"
c:\windows\system32\inetsrv\appcmd.exe set config "Default Web Site/"
/section:system.webServer/security/authentication/anonymousAuthentication /userName:$theAdmin
/password:$thePassword /commit:apphost

# Make sure the IIS settings take
Write-Host "Restarting the W3SVC" -ForegroundColor Cyan
Restart-Service -Name W3SVC

Write-Host
Write-Host "Web App Creation Successfull!" -ForegroundColor Green
Write-Host

```

AppVM01 - File server installation script

This script sets up the back-end for this simple application. This script will:

1. Open IMCPv4 (Ping) on the firewall for easier testing
2. Create a directory for the web site
3. Create a text file to be remotely access by the web page
4. Set permissions on the directory and file to Anonymous to allow access
5. Turn off IE Enhanced Security to allow easier browsing from this server

IMPORTANT

Best Practice: Never turn off IE Enhanced Security on a production server, plus it's generally a bad idea to surf the web from a production server. Also, opening up file shares for anonymous access is a bad idea, but done here for simplicity.

This PowerShell script should be run locally while RDP'd into AppVM01. PowerShell is required to be run as Administrator to ensure successful execution.

```

# AppVM01 Server Post Build Config Script
# PowerShell must be run as Administrator for Net Share commands to work

# Turn On ICMPv4
New-NetFirewallRule -Name Allow_ICMPv4 -DisplayName "Allow ICMPv4" -Protocol ICMPv4 -Enabled True -Profile Any -Action Allow

# Create Directory
New-Item "C:\WebShare" -ItemType Directory

# Write out Rand.txt
$FileContent = "Hello, I'm the contents of a remote file on AppVM01."
$FileContent | Out-File -FilePath "C:\WebShare\Rand.txt" -Encoding ascii

# Set Permissions on share
$Acl = Get-Acl "C:\WebShare"
$AccessRule = New-Object system.security.accesscontrol.filesystemaccessrule("Everyone", "ReadAndExecute, Synchronize", "ContainerInherit, ObjectInherit", "InheritOnly", "Allow")
$Acl.SetAccessRule($AccessRule)
Set-Acl "C:\WebShare" $Acl

# Create network share
Net Share WebShare=C:\WebShare "/grant:Everyone,READ"

# Turn Off IE Enhanced Security Configuration for Admins
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\Active Setup\Installed Components\{A509B1A7-37EF-4b3f-8CFC-4F3A74704073}" -Name "IsInstalled" -Value 0

Write-Host
Write-Host "File Server Set up Successfull!" -ForegroundColor Green
Write-Host

```

DNS01 - DNS server installation script

There is no script included in this sample application to set up the DNS server. If testing of the firewall rules, NSG, or UDR needs to include DNS traffic, the DNS01 server needs to be set up manually. The Network Configuration xml file and Resource Manager Template for both examples includes DNS01 as the primary DNS server and the public DNS server hosted by Level 3 as the backup DNS server. The Level 3 DNS server would be the actual DNS server used for non-local traffic, and with DNS01 not setup, no local network DNS would occur.

Next steps

- Run the IIS01 script on an IIS server
- Run File Server script on AppVM01
- Browse to the Public IP on IIS01 to validate your build

Azure Virtual Network frequently asked questions (FAQ)

3/8/2018 • 12 min 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.

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 some IP addresses within each subnet. The first and last IP addresses of each subnet are reserved for protocol conformance, along with the x.x.x.1-x.x.x.3 addresses of each subnet, which are used for Azure services.

How small and how large can VNets and subnets be?

The smallest supported subnet is /29, and the largest is /8 (using CIDR subnet definitions).

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?

No. You cannot use IPv6 with VNets at this time. You can however, assign IPv6 addresses to Azure load balancers to load balance virtual machines. For details, see [Overview of IPv6 for Azure Load Balancer](#).

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 will need to restart each of the VMs in your VNet in order for them to pick up the new DNS server.

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.

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). If you have a point-to-site connection configured for your VNet, all Web Apps can securely connect and access resources in the VNet. For more information, see the following articles:

- [Creating Web Apps in an App Service Environment](#)
- [Integrate your app with an Azure Virtual Network](#)
- [Using VNet Integration and Hybrid Connections with Web Apps](#)

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 (VMSS) to a VNet?

Yes. You must connect a VMSS 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](#).

Which Azure PaaS resources can I restrict access to from a VNet?

Resources deployed through some Azure PaaS services (such as Azure Storage and Azure SQL Database), can restrict network access to only resources in a VNet through the use of virtual network service endpoints. For details, see [Virtual network service endpoints 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 \(Service Management\)](#) 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.