

Table of Contents

[Overview](#)

[Virtual networks](#)

[User-defined routes and IP forwarding](#)

[Virtual network peering](#)

[Business continuity](#)

[FAQ](#)

[IP addressing](#)

[Resource Manager](#)

[Classic](#)

[Virtual machines](#)

[Network interfaces](#)

[Name resolution](#)

[Get Started](#)

[Create a virtual network](#)

[Deploy a VM to a virtual network](#)

[How To](#)

[Plan and design](#)

[Virtual networks](#)

[Network security groups](#)

[Deploy](#)

[Virtual networks](#)

[Network security groups](#)

[User-defined routes](#)

[Virtual network peering](#)

[Virtual machines](#)

[Connectivity scenarios](#)

[Security scenarios](#)

[Configure](#)

[Accelerated networking](#)

[Access control lists](#)

[Manage](#)

[Network security groups](#)

[Troubleshoot routes](#)

[Virtual machines](#)

[Reference](#)

[PowerShell \(Resource manager\)](#)

[PowerShell \(Classic\)](#)

[Azure CLI](#)

[Java](#)

[REST \(Resource Manager\)](#)

[REST \(Classic\)](#)

[Related](#)

[Virtual Machines](#)

[Application Gateway](#)

[Azure DNS](#)

[Traffic Manager](#)

[Load Balancer](#)

[VPN Gateway](#)

[ExpressRoute](#)

[Resources](#)

[Networking blog](#)

[Networking forum](#)

[Pricing](#)

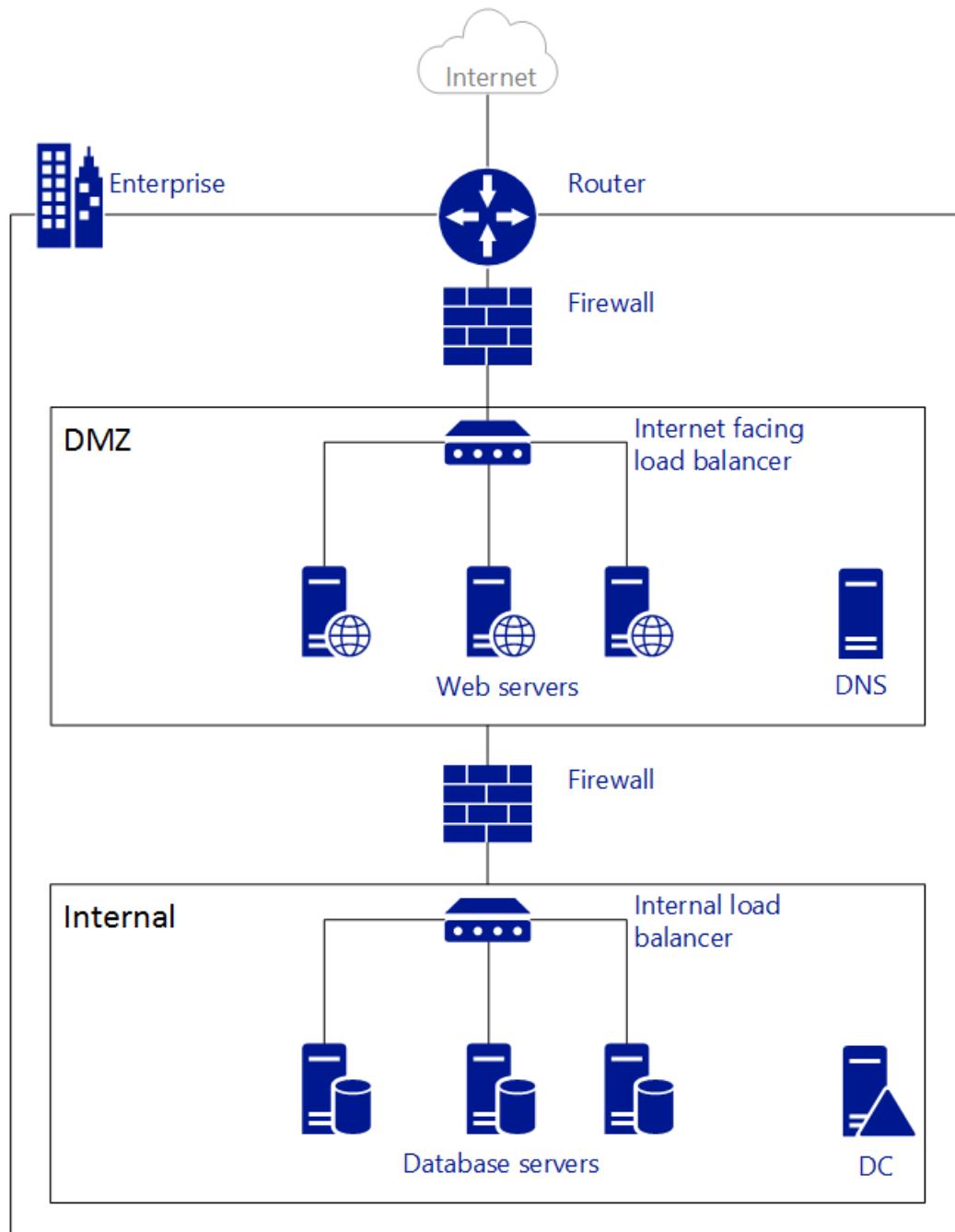
[Stack Overflow](#)

Virtual networks

1/17/2017 • 5 min to read • [Edit on GitHub](#)

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 fully control the IP address blocks, DNS settings, security policies, and route tables within this network. You can also further segment your VNet into subnets and launch Azure IaaS virtual machines (VMs) and/or [Cloud services \(PaaS role instances\)](#). Additionally, you can connect the virtual network to your on-premises network using one of the [connectivity options](#) available in Azure. In essence, you can expand your network to Azure, with complete control on IP address blocks with the benefit of enterprise scale Azure provides.

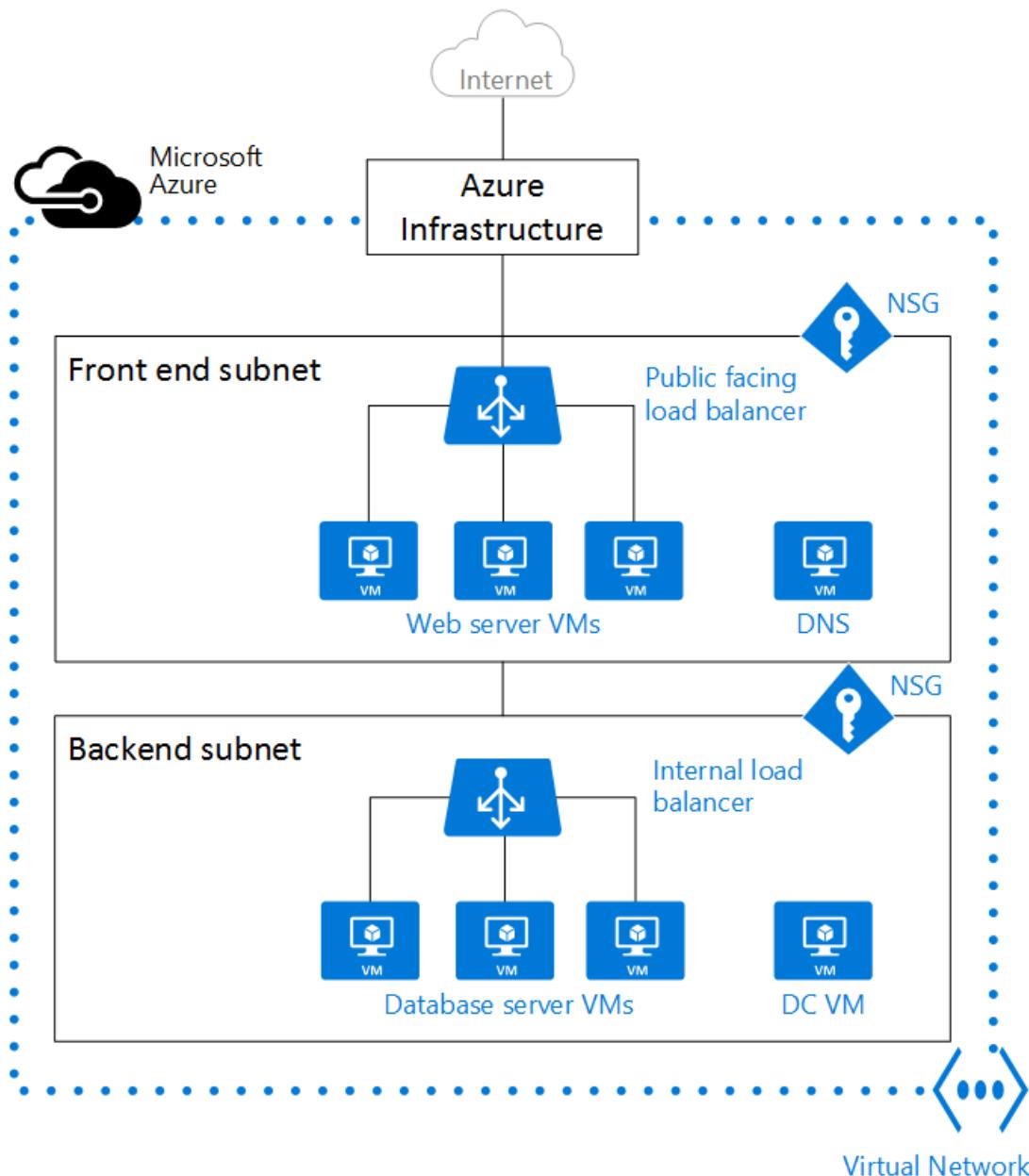
To better understand VNets, take a look at the figure below, which shows a simplified on-premises network.



The figure above shows an on-premises network connected to the public Internet through a router. You can also see a firewall between the router and a DMZ hosting a DNS server and a web server farm. The web server farm is

load balanced using a hardware load balancer that is exposed to the Internet, and consumes resources from the internal subnet. The internal subnet is separated from the DMZ by another firewall, and hosts Active Directory Domain Controller servers, database servers, and application servers.

The same network can be hosted in Azure as shown in the figure below.



Notice how the Azure infrastructure takes on the role of the router, allowing access from your VNet to the public Internet without the need of any configuration. Firewalls can be substituted by Network Security Groups (NSGs) applied to each individual subnet. And physical load balancers are substituted by internet facing and internal load balancers in Azure.

NOTE

There are two deployment modes in Azure: classic (also known as Service Management) and Azure Resource Manager (ARM). Classic VNets could be added to an affinity group, or created as a regional VNet. If you have a VNet in an affinity group, it is recommended to [migrate it to a regional VNet](#).

Benefits

- **Isolation.** VNets are completely isolated from one another. That allows you to create disjoint networks for

development, testing, and production that use the same CIDR address blocks.

- **Access to the public Internet.** All IaaS VMs and PaaS role instances in a VNet can access the public Internet by default. You can control access by using Network Security Groups (NSGs).
- **Access to VMs within the VNet.** PaaS role instances and IaaS VMs can be launched in the same virtual network and they can connect to each other using private IP addresses even if they are in different subnets without the need to configure a gateway or use public IP addresses.
- **Name resolution.** Azure provides internal name resolution for IaaS VMs and PaaS role instances deployed in your VNet. You can also deploy your own DNS servers and configure the VNet to use them.
- **Security.** Traffic entering and exiting the virtual machines and PaaS role instances in a VNet can be controlled using Network Security groups.
- **Connectivity.** VNets can be connected to each other using network gateways or VNet peering. VNets can be connected to on-premises data centers through site-to-site VPN networks or Azure ExpressRoute. To learn more about site-to-site VPN connectivity, visit [About VPN gateways](#). To learn more about ExpressRoute, visit [ExpressRoute technical overview](#). To learn more about VNet peering, visit [VNet peering](#).

NOTE

Make sure you create a VNet before deploying any IaaS VMs or PaaS role instances to your Azure environment. ARM based VMs require a VNet, and if you do not specify an existing VNet, Azure creates a default VNet that might have a CIDR address block clash with your on-premises network. Making it impossible for you to connect your VNet to your on-premises network.

Subnets

Subnet is a range of IP addresses in the VNet, you can divide a VNet into multiple subnets for organization and security. VMs and PaaS role instances deployed to subnets (same or different) within a VNet can communicate with each other without any extra configuration. You can also configure route tables and NSGs to a subnet.

IP addresses

There are two types of IP addresses assigned to resources in Azure: *public* and *private*. Public IP Addresses allow Azure resources to communicate with Internet and other Azure public-facing services like [Azure Redis Cache](#), [Azure Event Hubs](#). Private IP Addresses allows communication between resources in a virtual network, along with those connected through a VPN, without using an Internet-routable IP addresses.

To learn more about IP addresses in Azure, visit [IP addresses in virtual network](#)

Azure load balancers

Virtual machines and cloud services in a Virtual network can be exposed to Internet using Azure Load balancers. Line of Business applications that are internal facing only can be load balanced using Internal load balancer.

- **External load balancer.** You can use an external load balancer to provide high availability for IaaS VMs and PaaS role instances accessed from the public Internet.
- **Internal load balancer.** You can use an internal load balancer to provide high availability for IaaS VMs and PaaS role instances accessed from other services in your VNet.

To learn more about load balancing in Azure, visit [Load balancer overview](#).

Network Security Groups (NSG)

You can create NSGs to control inbound and outbound access to network interfaces (NICs), VMs, and subnets.

Each NSG contains one or more rules specifying whether or not traffic is approved or denied based on source IP address, source port, destination IP address, and destination port. To learn more about NSGs, visit [What is a Network Security Group](#).

Virtual appliances

A virtual appliance is just another VM in your VNet that runs a software based appliance function, such as firewall, WAN optimization, or intrusion detection. You can create a route in Azure to route your VNet traffic through a virtual appliance to use its capabilities.

For instance, NSGs can be used to provide security on your VNet. However, NSGs provide layer 4 Access Control List (ACL) to incoming and outgoing packets. If you want to use a layer 7 security model, you need to use a firewall appliance.

Virtual appliances depend on [user defined routes and IP forwarding](#).

Limits

There are limits on the number of Virtual Networks allowed in a subscription, please refer to [Azure Networking limits](#) for more information.

Pricing

There is no extra cost for using Virtual Networks in Azure. The compute instances launched within the Vnet will be charged the standard rates as described in [Azure VM Pricing](#). The [VPN Gateways](#) and [Public IP Addresses](#) used in the VNet will also be charged standard rates.

Next steps

- [Create a VNet](#) and subnets.
- [Create a VM in a VNet](#).
- Learn about NSGs.
- Learn about [user defined routes and IP forwarding](#).

What are User Defined Routes and IP Forwarding?

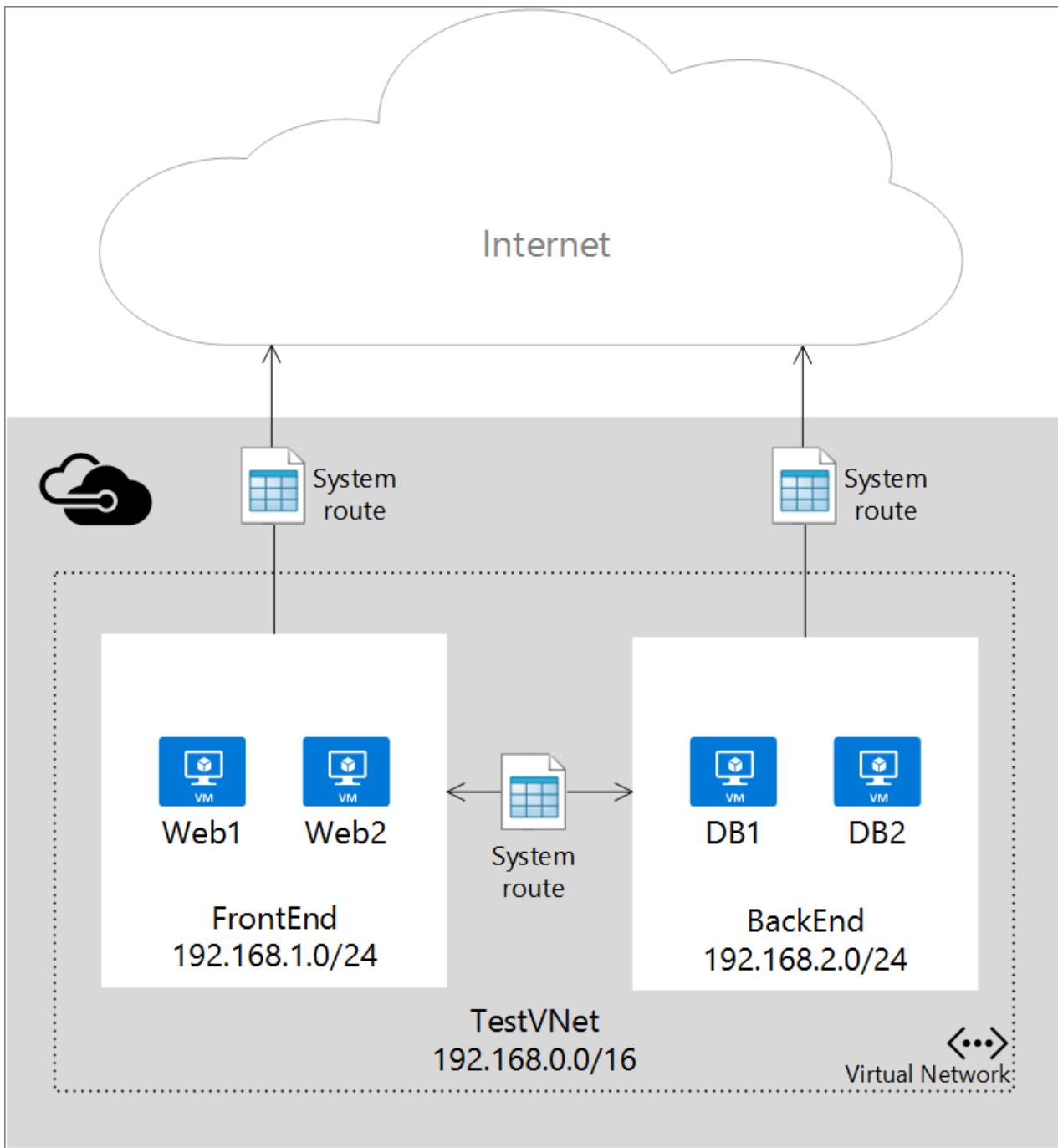
1/17/2017 • 6 min to read • [Edit on GitHub](#)

When you add virtual machines (VMs) to a virtual network (VNet) in Azure, you will notice that the VMs are able to communicate with each other over the network, automatically. You do not need to specify a gateway, even though the VMs are in different subnets. The same is true for communication from the VMs to the public Internet, and even to your on-premises network when a hybrid connection from Azure to your own datacenter is present.

This flow of communication is possible because Azure uses a series of system routes to define how IP traffic flows. System routes control the flow of communication in the following scenarios:

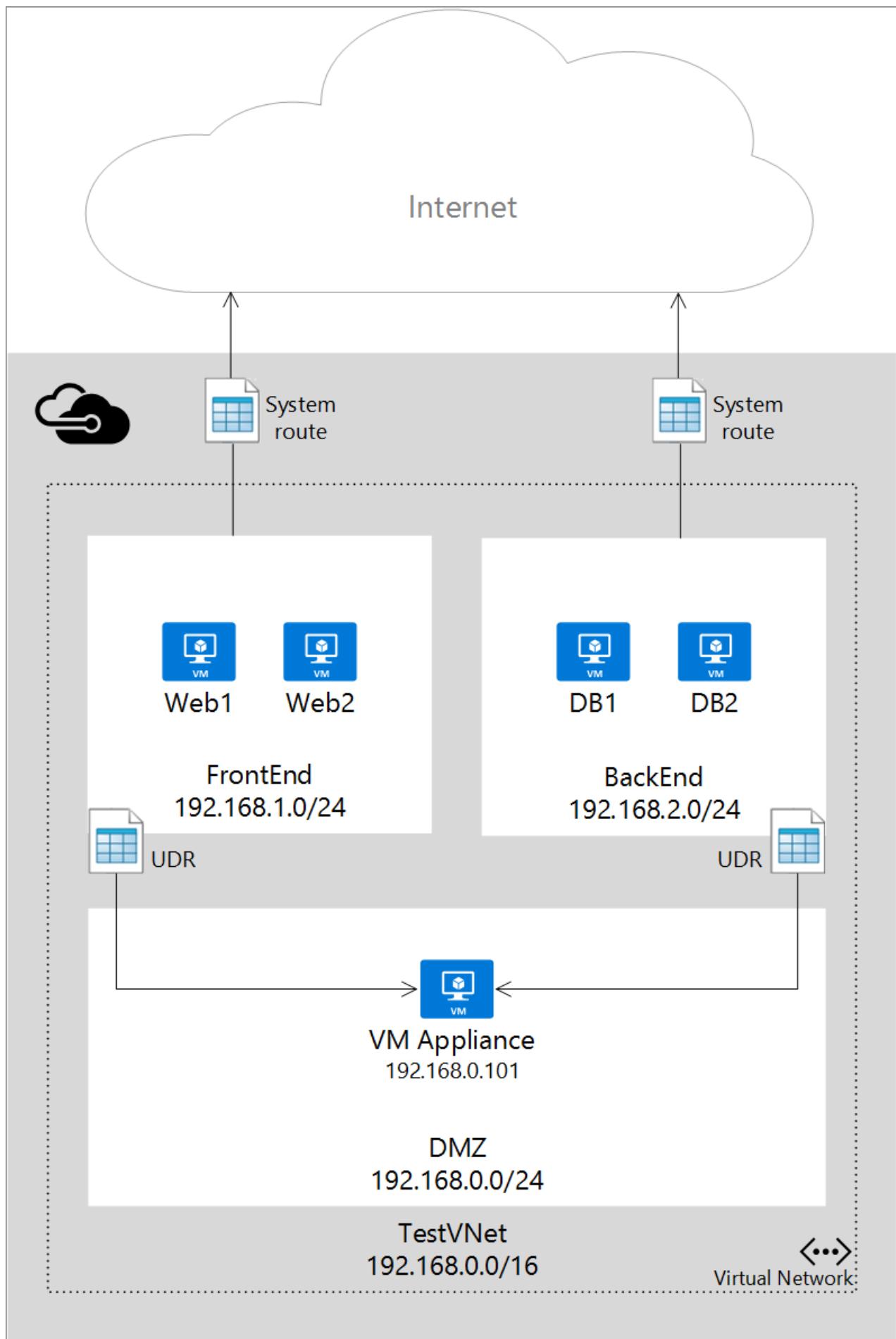
- From within the same subnet.
- From a subnet to another within a VNet.
- From VMs to the Internet.
- From a VNet to another VNet through a VPN gateway.
- From a VNet to another VNet through VNet Peering (Service Chaining).
- From a VNet to your on-premises network through a VPN gateway.

The figure below shows a simple setup with a VNet, two subnets, and a few VMs, along with the system routes that allow IP traffic to flow.



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.

The figure below shows an example of user defined routes and IP forwarding to force packets sent to one subnet from another to go through a virtual appliance on a third subnet.



IMPORTANT

User defined routes are only applied to traffic leaving a subnet. you cannot create routes to specify how traffic comes into a subnet from the Internet, for instance. Also, the appliance you are forwarding traffic to cannot be in the same subnet where the traffic originates. Always create a separate subnet for your appliances.

Route resource

Packets are routed over a TCP/IP network based on a route table defined at each node on the physical network. A route table is a collection of individual routes used to decide where to forward packets based on the destination IP address. A route consists of the following:

PROPERTY	DESCRIPTION	CONSTRAINTS	CONSIDERATIONS
Address Prefix	The destination CIDR to which the route applies, such as 10.1.0.0/16.	Must be a valid CIDR range representing addresses on the public Internet, Azure virtual network, or on-premises datacenter.	Make sure the Address prefix does not contain the address for the Next hop address , otherwise your packets will enter in a loop going from the source to the next hop without ever reaching the destination.
Next hop type	The type of Azure hop the packet should be sent to.	Must be one of the following values: Virtual Network. Represents the local virtual network. For instance, if you have two subnets, 10.1.0.0/16 and 10.2.0.0/16 in the same virtual network, the route for each subnet in the route table will have a next hop value of <i>Virtual Network</i> . Virtual Network Gateway. Represents an Azure S2S VPN Gateway. Internet Represents the default Internet gateway provided by the Azure Infrastructure. Virtual Appliance. Represents a virtual appliance you added to your Azure virtual network. None. Represents a black hole. Packets forwarded to a black hole will not be forwarded at all.	Consider using a None type to stop packets from flowing to a given destination.
Next hop address	The next hop address contains the IP address packets should be forwarded to. Next hop values are only allowed in routes where the next hop type is <i>Virtual Appliance</i> .	Must be an IP address that is reachable within the Virtual Network where the User Defined Route is applied.	If the IP address represents a VM, make sure you enable IP forwarding in Azure for the VM.

In Azure PowerShell some of the "NextHopType" values have different names:

- Virtual Network is VnetLocal
- Virtual Network Gateway is VirtualNetworkGateway
- Virtual Appliance is VirtualAppliance
- Internet is Internet
- None is None

System Routes

Every subnet created in a virtual network is automatically associated with a route table that contains the following system route rules:

- **Local Vnet Rule:** This rule is automatically created for every subnet in a virtual network. It specifies that there is a direct link between the VMs in the VNet and there is no intermediate next hop.
- **On-premises Rule:** This rule applies to all traffic destined to the on-premises address range and uses VPN gateway as the next hop destination.
- **Internet Rule:** This rule handles all traffic destined to the public Internet (address prefix 0.0.0.0/0) and uses the infrastructure internet gateway as the next hop for all traffic destined to the Internet.

User Defined Routes

For most environments you will only need the system routes already defined by Azure. However, you may need to create a route table and add one or more routes in specific cases, such as:

- Force tunneling to the Internet via your on-premises network.
- Use of virtual appliances in your Azure environment.

In the scenarios above, you will have to create a route table and add user defined routes to it. You can have multiple route tables, and the same route table can be associated to one or more subnets. And each subnet can only be associated to a single route table. All VMs and cloud services in a subnet use the route table associated to that subnet.

Subnets rely on system routes until a route table is associated to the subnet. Once an association exists, routing is done based on Longest Prefix Match (LPM) among both user defined routes 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:

1. User defined route
2. BGP route (when ExpressRoute is used)
3. System route

To learn how to create user defined routes, see [How to Create Routes and Enable IP Forwarding in Azure](#).

IMPORTANT

User defined routes are only applied to Azure VMs and cloud services. For instance, if you want to add a firewall virtual appliance between your on-premises network and Azure, you will have to create a user defined route for your Azure route tables that forwards all traffic going to the on-premises address space to the virtual appliance. You can also add a user defined route (UDR) to the GatewaySubnet to forward all traffic from on-premises to Azure through the virtual appliance. This is a recent addition.

BGP Routes

If you have an ExpressRoute connection between your on-premises network and Azure, you can enable BGP to propagate routes from your on-premises network to Azure. These BGP routes are used in the same way as system routes and user defined routes in each Azure subnet. For more information see [ExpressRoute Introduction](#).

IMPORTANT

You can configure your Azure environment to use force tunneling through your on-premises network by creating a user defined route for subnet 0.0.0.0/0 that uses the VPN gateway as the next hop. However, this only works if you are using a VPN gateway, not ExpressRoute. For ExpressRoute, forced tunneling is configured through BGP.

IP Forwarding

As described above, one of the main reasons to create a user defined route is to forward traffic to a virtual appliance. 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.

Next Steps

- Learn how to [create routes in the Resource Manager deployment model](#) and associate them to subnets.
- Learn how to [create routes in the classic deployment model](#) and associate them to subnets.

VNet peering

1/17/2017 • 5 min to read • [Edit on GitHub](#)

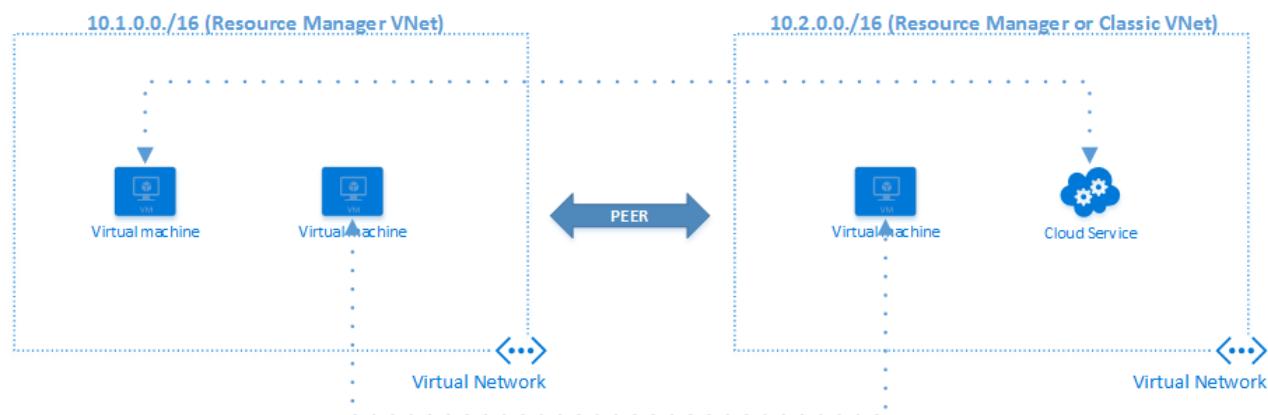
VNet peering is a mechanism that connects two virtual networks (VNets) in the same region through the Azure backbone network. Once peered, the two virtual networks appear as one for all connectivity purposes. They are still managed as separate resources, but virtual machines in these virtual networks can communicate with each other directly by using private IP addresses.

The traffic between virtual machines in the peered virtual networks is routed through the Azure infrastructure much like traffic is routed between VMs in the same virtual network. Some of the benefits of using VNet peering include:

- A low-latency, high-bandwidth connection between resources in different virtual networks.
- The ability to use resources such as network appliances and VPN gateways as transit points in a peered VNet.
- The ability to connect a virtual network that uses the Azure Resource Manager model to a virtual network that uses the classic deployment model and enable full connectivity between resources in these virtual networks.

Requirements and key aspects of VNet peering:

- The two virtual networks that are peered should be in the same Azure region.
- The virtual networks that are peered should have non-overlapping IP address spaces.
- VNet peering is between two virtual networks, and there is no derived transitive relationship. For example, if virtual network A is peered with virtual network B, and if virtual network B is peered with virtual network C, it does not translate to virtual network A being peered with virtual network C.
- Peering can be established between virtual networks in two different subscriptions as long a privileged user of both subscriptions authorizes the peering and the subscriptions are associated to the same Active Directory tenant.
- Peering between virtual network in resource manager model and classic deployment model requires that the VNets should be in the same subscription.
- A virtual network that uses the Resource Manager deployment model can be peered with another virtual network that uses this model, or with a virtual network that uses the classic deployment model. However, virtual networks that use the classic deployment model can't be peered to each other.
- Though the communication between virtual machines in peered virtual networks has no additional bandwidth restrictions, bandwidth cap based on VM size still applies.



Connectivity

After two virtual networks are peered, a virtual machine (web/worker role) in the virtual network can directly

connect with other virtual machines in the peered virtual network. These two networks have full IP-level connectivity.

The network latency for a round trip between two virtual machines in peered virtual networks is the same as for a round trip within a local 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.

The traffic between the virtual machines in peered virtual networks is routed directly through the Azure back-end infrastructure and not through a gateway.

Virtual machines in a virtual network can access the internal load-balanced (ILB) endpoints in the peered virtual network. Network security groups (NSGs) can be applied in either virtual network to block access to other virtual networks or subnets if desired.

When users configure peering, they can either open or close the NSG rules between the virtual networks. If the user chooses to open full connectivity between peered virtual networks (which is the default option), they can then use NSGs on specific subnets or virtual machines to block or deny specific access.

Azure-provided internal DNS name resolution for virtual machines doesn't work across peered virtual networks. Virtual machines have internal DNS names that are resolvable only within the local virtual network. However, users can configure virtual machines that are running in peered virtual networks as DNS servers for a virtual network.

Service chaining

Users can configure user-defined route tables that point to virtual machines in peered virtual networks as the "next hop" IP address, as shown in the diagram later in this article. This enables users to achieve service chaining, through which they can direct traffic from one virtual network to a virtual appliance that's running in a peered virtual network through user-defined route tables.

Users can also effectively build hub-and-spoke type environments where the hub can host infrastructure components such as a network virtual appliance. All the spoke virtual networks can then peer with it, as well as a subset of traffic to appliances that are running in the hub virtual network. In short, VNet peering enables the next hop IP address on the 'User defined route table' to be the IP address of a virtual machine in the peered virtual network.

Gateways and on-premises connectivity

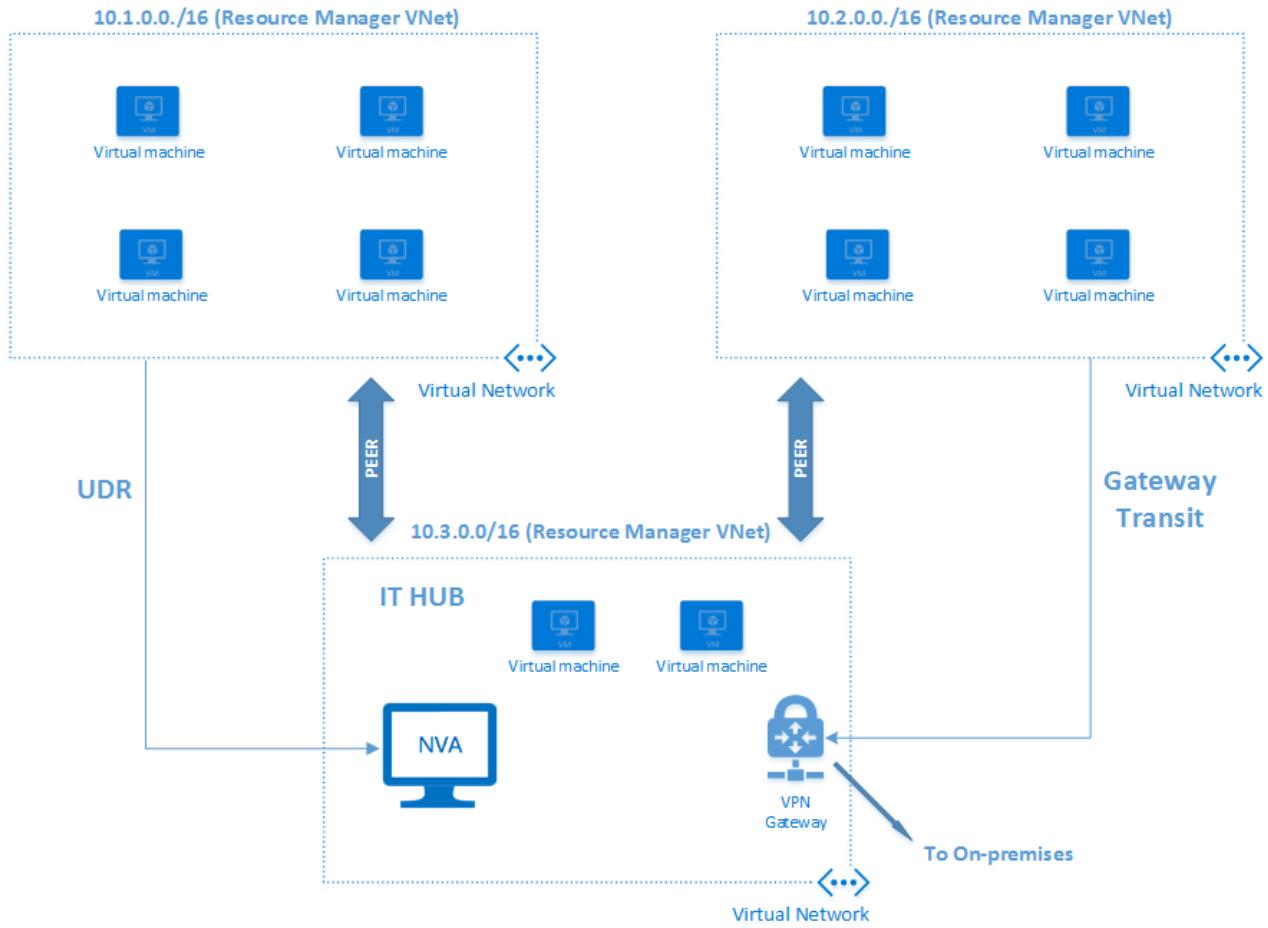
Each virtual network, regardless of whether it is peered with another virtual network, can still have its own gateway and use it to connect to on-premises. Users can also configure [VNet-to-VNet connections](#) by using gateways, even though the virtual networks are peered.

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, users can also configure the gateway in the peered virtual network as a transit point to on-premises. In this case, the virtual network that is using a remote gateway cannot have its own gateway. One virtual network can have only one gateway. It can either be a local gateway or a 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 using the Resource Manager model and those using the classic deployment model. Both virtual networks in the peering relationship need to use the Resource Manager deployment model for a gateway transit to work.

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). Users can still use local gateways in each virtual network to connect to the on-premises circuit. Alternatively, they can use a shared gateway and configure transit for on-premises connectivity.



Provisioning

VNet peering is a privileged operation. It's a separate function under the `VirtualNetworks` namespace. A user can be given specific rights to authorize peering. A user who has read-write access to the virtual network inherits these rights automatically.

A user who is either an admin or a privileged user of the peering ability can initiate a peering operation on another VNet. If there is a matching request for peering on the other side, and if other requirements are met, the peering will be established.

Refer to the articles in the "Next steps" section to learn more about how to establish VNet peering between two virtual networks.

Limits

There are limits on the number of peerings that are allowed for a single virtual network. Refer to [Azure networking limits](#) for more information.

Pricing

VNet peering will be free of charge during the review period. After it is released, there will be a nominal charge on ingress and egress traffic that utilizes the peering. For more information, refer to the [pricing page](#).

Next steps

- Set up peering between virtual networks.
- Learn about NSGs.
- Learn about user-defined routes and IP forwarding.

Virtual Network – Business Continuity

1/17/2017 • 2 min to read • [Edit on GitHub](#)

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. A Virtual Network can also be linked to an on-premises network through one of the hybrid options such as a VPN Gateway or ExpressRoute.

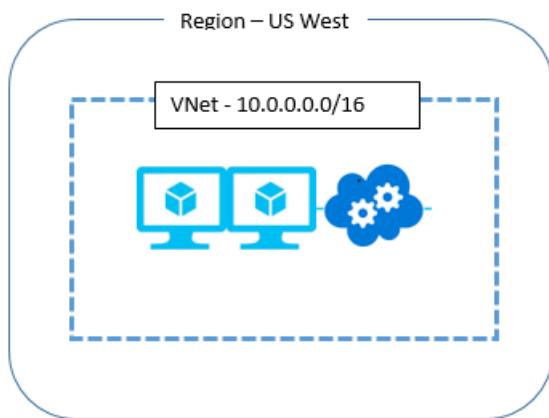
A VNet is created within the scope of a region. You can create VNets with same address space in two different regions (i.e. US East and US West but cannot connect them to one another directly).

Business Continuity

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

Q: What do you do in the event of an outage to an entire region? i.e. if a region is completely cutoff due to a natural disaster? What happens to the Virtual Networks hosted in the region?

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



Q: What can I do to re-create the same Virtual Network in a different region?

A: Virtual Network (VNet) is fairly lightweight resource. You can invoke Azure APIs to create a VNet with the same address space in a different region. To re-create the same environment that was present in the affected region, you have to make API calls to re-deploy your Cloud Services (web/worker roles) and Virtual Machines that you had. You will also have to spin up a VPN Gateway and connect to your on-premises network if you had on-premises connectivity (such as in a hybrid deployment).

The instructions for creating a VNet are found [here](#).

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 a customer was hosting internet facing services in the VNet, they could have set up Traffic Manager to geo-route traffic to the region that is active. However, a customer cannot connect two VNets with the

same address space to their on-premises network as it would cause routing issues. At the time of a disaster and loss of a VNet in one region, a customer can connect the other VNet in the available region with matching address space to their on-premises network.

The instructions for creating a VNet are found [here](#).

Virtual Network FAQ

1/17/2017 • 12 min to read • [Edit on GitHub](#)

Virtual Network Basics

What is an Azure Virtual network (VNet)?

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 collide. You also have controls of DNS server settings for VNets, and segmentation of the VNet into subnets.

Use VNets to:

- Create a dedicated private cloud-only virtual network

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. This keeps traffic securely within the VNet, but still allows you to 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 know if I need a virtual network?

Visit the [Virtual Network Overview](#) to see a decision table that will help you decide the best network design option for you.

How do I get started?

Visit [the Virtual Network documentation](#) to get started. This page has links to common configuration steps as well as information that will help you understand the things that you'll need to take into consideration when designing your virtual network.

What services can I use with VNets?

VNets can be used with a variety of different Azure services, such as Cloud Services (PaaS), Virtual Machines, and Web Apps. However, there are a few services that are not supported on a VNet. Please check the specific service you want to use and verify that it is compatible.

Can I use VNets without cross-premises connectivity?

Yes. You can use a VNet without using site-to-site connectivity. This is particularly useful if you want to run domain controllers and SharePoint farms in Azure.

Virtual Network Configuration

What tools do I use to create a VNet?

You can use the following tools to create or configure a virtual network:

- Azure Portal (for classic and Resource Manager VNets).
- A network configuration file (netcfg - for classic VNets only). See [Configure a virtual network using a network configuration file](#).
- PowerShell (for classic and Resource Manager VNets).
- Azure CLI (for classic and Resource Manager VNets).

What address ranges can I use in my VNets?

You can use public IP address ranges and any IP address range defined in [RFC 1918](#).

Can I have public IP addresses in my VNets?

Yes. For more information about public IP address ranges, see [Public IP address space in a Virtual Network \(VNet\)](#). Keep in mind that your public IPs will not be directly accessible from the Internet.

Is there a limit to the number of subnets in my virtual network?

There is no limit on the number of subnets you use within a VNet. All the subnets must be fully contained in the virtual network address space and should not overlap with one another.

Are there any restrictions on using IP addresses within these subnets?

Azure reserves some IP addresses within each subnet. The first and last IP addresses of the subnets are reserved for protocol conformance, along with 3 more addresses used for Azure services.

How small and how large can VNets and subnets be?

The smallest subnet we support is a /29 and the largest is a /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 use User Defined Routing (UDR). For more information about UDR, visit [User Defined Routes and IP Forwarding](#).

Do VNets support multicast or broadcast?

No. We do not support multicast or broadcast.

What protocols can I use within VNets?

You can use standard IP-based protocols within VNets. However, multicast, broadcast, IP-in-IP encapsulated packets and Generic Routing Encapsulation (GRE) packets are blocked within VNets. Standard protocols that work include:

- TCP
- UDP
- ICMP

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 is not part of another subnet in the VNet.

Can I modify the size of my subnet after I create it?

You can add, remove, expand or shrink a subnet if there are no VMs or services deployed within it by using PowerShell cmdlets or the NETCFG file. You can also add, remove, expand or shrink any prefixes as long as the subnets that contain VMs or services are not affected by the change.

Can I modify subnets after I created them?

Yes. You can add, remove, and modify the CIDR blocks used by a VNet.

Can I connect to the internet if I am running my services in a VNet?

Yes. All services deployed within a VNet can connect to the internet. Every cloud service deployed in Azure has a publicly addressable VIP assigned to it. You will have to 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.

Can a VNet span regions?

No. A VNet is limited to a single region.

Can I connect a VNet to another VNet in Azure?

Yes. You can create VNet to VNet communication by using REST APIs or Windows PowerShell. You can also connect VNets via VNet Peering. See more details about peering [here](#).

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. This will be applied as the default DNS server(s) for all VMs in the VNet.

How many DNS servers can I specify?

You can specify up to 12 DNS servers.

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 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.

NOTE

There is a limitation at this time to the first 100 cloud services in the virtual network 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 / service basis?

Yes. You can set DNS servers on a per-cloud service basis to override the default network settings. However, we recommend 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.

VNets and VMs

Can I deploy VMs to a VNet?

Yes.

Can I deploy Linux VMs to a VNet?

Yes. You can deploy any distro of Linux supported by Azure.

What is the difference between a public VIP and an internal IP address?

- An internal IP address is an IP address that is assigned to each VM within a VNet by DHCP. It's not public facing. If you have created a VNet, the internal IP address is assigned from the range that you specified in the subnet settings of your VNet. If you do not have a VNet, an internal IP address will still be assigned. The internal IP address will remain with the VM for its lifetime, unless that VM is deallocated.
- A public VIP is the public IP address that is assigned to your cloud service or load balancer. It is not assigned directly to your VM NIC. The VIP stays with the cloud service it is assigned to until all the VMs in that cloud service are deallocated or deleted. At that point, it is released.

What IP address will my VM receive?

- **Internal IP address** - If you deploy a VM to a VNet, the VM receives an internal IP address from a pool of internal IP addresses that you specify. VMs communicate within the VNets by using internal IP addresses. Although Azure assigns a dynamic internal IP address, you can request a static address for your VM. To learn more about static internal IP addresses, visit [How to Set a Static Internal IP](#).
- **VIP** - Your VM is also associated with a VIP, although a VIP is never assigned to the VM directly. A VIP is a public IP address that can be assigned to your cloud service. You can, optionally, reserve a VIP for your cloud service.
- **ILPIP** - You can also configure an instance-level public IP address (ILPIP). ILPIPs are directly associated with the VM, rather than the cloud service. To learn more about ILPIPs, visit [Instance-Level Public IP Overview](#).

Can I reserve an internal IP address for a VM that I will create at a later time?

No. You cannot reserve an internal IP address. If an internal IP address is available it will be assigned to a VM or role instance by the DHCP server. That VM may or may not be the one that you want the internal IP address to be assigned to. You can, however, change the internal IP address of an already created VM to any available internal IP address.

Do internal IP addresses change for VMs in a VNet?

Yes. Internal IP addresses remain with the VM for its lifetime unless the VM is deallocated. When a VM is deallocated, the internal IP address is released unless you defined a static internal IP address for your VM. If the VM is simply stopped (and not put in the status **Stopped (Deallocated)**) the IP address will remain assigned to the VM.

Can I manually assign IP addresses to NICs in VMs?

No. You must not change any interface properties of VMs. Any changes may lead to potentially losing connectivity to the VM.

What happens to my IP addresses if I shut down a VM?

Nothing. The IP addresses (both public VIP and internal IP address) will stay with your cloud service or VM.

NOTE

If you want to simply shut down the VM, don't use the Management Portal to do so. Currently, the shutdown button will deallocate the virtual machine.

Can I move VMs from one subnet to another subnet in a VNet without re-deploying?

Yes. You can find more information [here](#).

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 has been created?

Yes, the MAC address will remain the same for a VM even though the VM has been stopped (deallocated) and relaunched.

Can I connect to the internet from a VM in a VNet?

Yes. All services deployed within a VNet can connect to the Internet. Additionally, every cloud service deployed in Azure has a publicly addressable VIP assigned to it. You have to define input endpoints for PaaS roles and endpoints for VMs to enable these services to accept connections from the Internet.

VNets and Services

What services can I use with VNets?

You can only use compute services within VNets. Compute services are limited to Cloud Services (web and worker roles) and VMs.

Can I use Web Apps with Virtual Network?

Yes. You can deploy Web Apps inside a VNet using ASE (App Service Environment). Adding to that, Web Apps can securely connect and access resources in your Azure VNet if you have point-to-site configured for your VNet. For more information, see the following:

- [Creating Web Apps in an App Service Environment](#)
- [Web Apps Virtual Network Integration](#)
- [Using VNet Integration and Hybrid Connections with Web Apps](#)
- [Integrate a web app with an Azure Virtual Network](#)

Can I deploy cloud services with web and worker roles (PaaS) in a VNet?

Yes. You can deploy PaaS services within VNets.

How do I deploy PaaS roles to a VNet?

You can accomplish this by specifying 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 move my services in and out of VNets?

No. You cannot move services in and out of VNets. You will have to delete and re-deploy the service to move it to another VNet.

VNets and Security

What is the security model for VNets?

VNets are completely isolated from one another, and other services hosted in the Azure infrastructure. A VNet is a trust boundary.

Can I define ACLs or NSGs on my VNets?

No. You cannot associate ACLs or NSGs to VNets. However, ACLs can be defined on input endpoints for VMs that have been deployed to a VNets, and NSGs can be associated to subnets or NICs.

Is there a VNet security whitepaper?

Yes. You can download it [here](#).

APIs, Schemas, and Tools

Can I manage VNets from code?

Yes. You can use REST APIs to manage VNets and cross-premises connectivity. More information can be found [here](#).

Is there tooling support for VNets?

Yes. You can use PowerShell and command line tools for a variety of platforms. More information can be found [here](#).

Virtual Network Cross-premises Connectivity (VPNs)

For the latest FAQ on virtual network VPNs, see the [VPN Gateway FAQ](#).

IP addresses in Azure

1/17/2017 • 7 min to read • [Edit on GitHub](#)

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

In Azure Resource Manager, a [public IP](#) address is a resource that has its own properties. You can associate a public IP address resource with any of the following resources:

- Virtual machines (VM)
- Internet-facing load balancers
- VPN gateways
- Application gateways

Allocation method

There are two methods in which an IP address is allocated to a *public IP* resource - *dynamic* or *static*. The default allocation method is *dynamic*, where an IP address is **not** allocated at the time of its creation. Instead, the public IP address is allocated when you start (or create) the associated resource (like a VM or load balancer). The IP address is released when you stop (or delete) the resource. This causes the IP address to change when you stop and start a resource.

To ensure the IP address for the associated resource remains the same, you can set the allocation method explicitly to *static*. In this case an IP address is assigned immediately. It is released only when you delete the resource or change its allocation method to *dynamic*.

NOTE

Even when you set the allocation method to *static*, you cannot specify the actual IP address assigned to the *public IP resource*. Instead, it gets allocated 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:

- End-users need to 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

The list of IP ranges from which public IP addresses (dynamic/static) are allocated to Azure resources is published at [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** will resolve to the public IP address of the resource. You can use this FQDN to create a custom domain CNAME record pointing to the public IP address in Azure.

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](#) VM by assigning it to its **network interface**. In the case of a VM with multiple network interfaces, you can assign it to the *primary* network interface only. You can assign either a dynamic or a static public IP address to a VM.

Internet-facing load balancers

You can associate a public IP address with an [Azure Load Balancer](#), by assigning it to the load balancer **frontend** configuration. This 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.

VPN gateways

[Azure VPN Gateway](#) is used to connect an Azure virtual network (VNet) to other Azure VNets or to an on-premises network. You need to assign a public IP address to its **IP configuration** to enable it to communicate with the remote network. Currently, you can only assign a *dynamic* 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. Currently, you can only assign a *dynamic* public IP address to an application gateway frontend configuration.

At-a-glance

The table below 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
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:

- VMs
- Internal load balancers (ILBs)
- Application gateways

Allocation method

A private IP address is allocated from the address range of the subnet to which the resource is attached. The address range of the subnet itself is a part of the VNet's address range.

There are two methods in which a private IP address is allocated: *dynamic* or *static*. The default allocation method is *dynamic*, where the IP address is automatically allocated from the resource's subnet (using DHCP). This IP address can change when you stop and start the resource.

You can set the allocation method to *static* to ensure the IP address remains the same. In this case, you also need to provide a valid IP address that is part of the resource's subnet.

Static private IP addresses are commonly used for:

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

Virtual machines

A private IP address is assigned to the **network interface** of a [Windows](#) or [Linux](#) VM. In case of a multi-network interface VM, each interface gets a private IP address assigned. You can specify the allocation method as either dynamic or static for a network interface.

Internal DNS hostname resolution (for VMs)

All Azure VMs 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 that reside within the same VNet.

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-network interface VM, the hostname is mapped to the private IP address of the primary network interface.

VMs configured with Azure-managed DNS servers will be able to resolve the hostnames of all VMs within their VNet to their private IP addresses.

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.

At-a-glance

The table below 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. These 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 public IP using a template](#)
- [Deploy a VM with a static private IP address using the Azure portal](#)

IP addresses (classic) in Azure

1/17/2017 • 9 min to read • [Edit on GitHub](#)

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)	Yes	Yes	Yes
PaaS role instance (in a <i>standalone</i> cloud service)	Yes	No	Yes
VM or PaaS role instance (in a VNet)	Yes	Yes	Yes
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 classic portal.](#)

Network interfaces

1/17/2017 • 4 min to read • [Edit on GitHub](#)

A network interface (NIC) is the interconnection between a Virtual Machine (VM) and the underlying software network. This article explains what a network interface is and how it's used in the Azure Resource Manager deployment model.

Microsoft recommends deploying new resources using the Resource Manager deployment model, but you can also deploy VMs with network connectivity in the [classic](#) deployment model. If you're familiar with the classic model, there are important differences in VM networking in the Resource Manager deployment model. Learn more about the differences by reading the [Virtual machine networking - Classic](#) article.

In Azure, a network interface:

1. Is a resource that can be created, deleted, and has its own configurable settings.
2. Must be connected to one subnet in one Azure Virtual Network (VNet) when it's created. If you're not familiar with VNets, learn more about them by reading the [Virtual network overview](#) article. The NIC must be connected to a VNet that exists in the same Azure [location](#) and [subscription](#) as the NIC. After a NIC is created, you can change the subnet it's connected to, but you cannot change the VNet it's connected to.
3. Has a name assigned to it that cannot be changed after the NIC is created. The name must be unique within an Azure [resource group](#), but doesn't have to be unique within the subscription, the Azure location it's created in, or the VNet it's connected to. Several NICs are typically created within an Azure subscription. It's recommended that you devise a naming convention that makes managing multiple NICs easier than using default names. See the [Recommended naming conventions for Azure resources](#) article for suggestions.
4. May be attached to a VM, but can only be attached to a single VM that exists in the same location as the NIC.
5. Has a MAC address, which is persisted with the NIC for as long as it remains attached to a VM. The MAC address is persisted whether the VM is restarted (from within the operating system) or stopped (de-allocated) and started using the Azure Portal, Azure PowerShell, or the Azure Command-Line Interface. If it's detached from a VM and attached to a different VM, the NIC receives a different MAC address. If the NIC is deleted, the MAC address is assigned to other NICs.
6. Must have one primary **private IPv4** static or dynamic IP address assigned to it.
7. May have one public IP address resource associated to it.
8. Supports accelerated networking with single-root I/O virtualization (SR-IOV) for specific VM sizes running specific versions of the Microsoft Windows Server operating system. To learn more about this PREVIEW feature, read the [Accelerated networking for a virtual machine](#) article.
9. Can receive traffic not destined to private IP addresses assigned to it if IP forwarding is enabled for the NIC. If a VM is running firewall software for example, it routes packets not destined for its own IP addresses. The VM must still run software capable of routing or forwarding traffic, but to do so, IP forwarding must be enabled for a NIC.
10. Is often created in the same resource group as the VM it's attached to or the same VNet that it's connected to, though it isn't required to be.

VMs with multiple network interfaces

Multiple NICs can be attached to a VM, but when doing so, be aware of the following:

- The VM size must support multiple NICs. To learn more about which VM sizes support multiple NICs, read the [Windows Server VM sizes](#) or [Linux VM sizes](#) articles.
- The VM must be created with at least two NICs. If the VM is created with only one NIC, even if the VM size

supports more than one, you cannot attach additional NICs to the VM after it's created. As long as the VM was created with at least two NICs, you can attach additional NICs to the VM after it's created, as long as the VM size supports more than two NICs.

- You can detach secondary NICs (the primary NIC cannot be detached) from a VM if the VM has at least three NICs attached to it. You cannot detach NICs if the VM has two or less NICs attached to it.
- The order of the NICs from inside the VM will be random, and could also change across Azure infrastructure updates. However, the IP addresses, and the corresponding ethernet MAC addresses, will remain the same. For example, assume the operating system identifies Azure NIC1 as Eth1. Eth1 has IP address 10.1.0.100 and MAC address 00-0D-3A-B0-39-0D. After an Azure infrastructure update and reboot, the operating system may now identify Azure NIC1 as Eth2, but the IP and MAC addresses will be the same as they were when the operating system identified Azure NIC1 as Eth1. When a restart is customer-initiated, the NIC order will remain the same within the operating system.
- If the VM is a member of an [availability set](#), all VMs within the availability set must have either a single NIC or multiple NICs. If the VMs have multiple NICs, the number they each have isn't required to be the same, as long as each VM has at least two NICs.

Next steps

- Learn how to create a VM with a single NIC by reading the [Create a VM](#) article.
- Learn how to create a VM with multiple NICs by reading the [Deploy a VM with multiple NIC](#) article.
- Learn how to create a NIC with multiple IP configurations by reading the [Multiple IP addresses for Azure virtual machines](#) article.

Name Resolution for VMs and Role Instances

1/17/2017 • 9 min to read • [Edit on GitHub](#)

Depending on how you use Azure to host IaaS, PaaS, and hybrid solutions, you may need to allow the VMs and role instances that you create to communicate with each other. Although this communication can be done by using IP addresses, it is much simpler to use names that can be easily remembered and do not change.

When role instances and VMs hosted in Azure need to resolve domain names to internal IP addresses, they can use one of two methods:

- [Azure-provided name resolution](#)
- [Name resolution using your own DNS server](#) (which may forward queries to the Azure-provided DNS servers)

The type of name resolution you use depends on how your VMs and role instances need to communicate with each other.

The following table illustrates scenarios and corresponding name resolution solutions:

SCENARIO	SOLUTION	SUFFIX
Name resolution between role instances or VMs located in the same cloud service or virtual network	Azure-provided name resolution	hostname or FQDN
Name resolution between role instances or VMs located in different virtual networks	Customer-managed DNS servers forwarding queries between vnets 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 role instances or VMs in Azure	Customer-managed DNS servers (e.g. on-premises domain controller, local read-only domain controller or a DNS secondary synced using zone transfers). 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 vnet, 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	n/a
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.	n/a

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/instances in a cloud service share the same DNS suffix (so the hostname alone is sufficient) but in classic virtual networks different cloud services have different DNS suffixes so the FQDN is needed to resolve names between different cloud services. In virtual networks in the Resource Manager deployment model, the DNS suffix is consistent across the virtual network (so the FQDN is not needed) and DNS names can be assigned to both NICs and VMs. Although Azure-provided name resolution does not require any configuration, it is not the appropriate choice for all deployment scenarios, as seen on the table above.

NOTE

In the case of web and worker roles, you can also access the internal IP addresses of role instances based on the role name and instance number using the Azure Service Management REST API. For more information, see [Service Management REST API Reference](#).

Features and Considerations

Features:

- Ease of use: No configuration is required in order to use Azure-provided name resolution.
- The Azure-provided name resolution service is highly available, saving you the need to create and manage clusters of your own DNS servers.
- Can be used in conjunction with your own DNS servers to resolve both on-premises and Azure hostnames.
- Name resolution is provided between role instances/VMs within the same cloud service without need for a FQDN.
- Name resolution is provided between VMs in virtual networks that use the Resource Manager deployment model, without need for the FQDN. Virtual networks in the classic deployment model require the FQDN when resolving names in different cloud services.
- You can use hostnames that best describe your deployments, rather than working with auto-generated names.

Considerations:

- 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.)
- Hostnames must be DNS-compatible (They must use only 0-9, a-z and '-', and cannot start or end with a '-'. See RFC 3696 section 2.)
- DNS query traffic is throttled for each VM. This shouldn't impact most applications. If request throttling is observed, ensure that client-side caching is enabled. For more details, see [Getting the most from Azure-provided name resolution](#).
- Only VMs in the first 180 cloud services are registered for each virtual network in a classic deployment model. This does not apply to virtual networks in Resource Manager deployment models.

Getting the most from Azure-provided name resolution

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) which allows the cache to store the record for as long as possible without impacting record freshness, so client-side caching is suitable for most situations.

The default Windows DNS Client has a DNS cache built-in. Some Linux distros do not include caching by default, it is recommended that one be added to each Linux VM (after checking that there isn't a local cache already).

There are a number of different DNS caching packages available, e.g. dnsmasq, here are the steps to install dnsmasq on the most common distros:

- **Ubuntu (uses resolvconf):**

- just install the dnsmasq package ("sudo apt-get install dnsmasq").

- **SUSE (uses netconf):**

- install the dnsmasq package ("sudo zypper install dnsmasq")
 - enable the dnsmasq service ("systemctl enable dnsmasq.service")
 - start the dnsmasq service ("systemctl start dnsmasq.service")
 - edit "/etc/sysconfig/network/config" and change NETCONFIG_DNS_FORWARDER="" to "dnsmasq"
 - update resolv.conf ("netconfig update") to set the cache as the local DNS resolver

- **OpenLogic (uses NetworkManager):**

- install the dnsmasq package ("sudo yum install dnsmasq")
 - enable the dnsmasq service ("systemctl enable dnsmasq.service")
 - start the dnsmasq service ("systemctl start dnsmasq.service")
 - add "prepend domain-name-servers 127.0.0.1;" to "/etc/dhclient-eth0.conf"
 - restart the network service ("service network restart") to set the cache as the local DNS resolver

NOTE

The 'dnsmasq' package is only one of the many DNS caches available for Linux. Before using it, please check its suitability for your particular needs and that no other cache is installed.

Client-side Retries:

DNS is primarily a UDP protocol. As 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 creators preference:

- Windows operating systems retry after 1 second and then again after another 2, 4 and another 4 seconds.
- The default Linux setup retries after 5 seconds. It is recommended to change this to retry 5 times at 1 second intervals.

To check the current settings on a Linux VM, 'cat /etc/resolv.conf' and look at the 'options' line, e.g.:

```
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 distro:

- **Ubuntu (uses resolvconf):**

- add the options line to '/etc/resolvconf/resolv.conf.d/head'
 - run 'resolvconf -u' to update

- **SUSE (uses netconf):**

- add 'timeout:1 attempts:5' to the NETCONFIG_DNS_RESOLVER_OPTIONS="" parameter in '/etc/sysconfig/network/config'
 - run 'netconfig update' to update

- **OpenLogic (uses NetworkManager):**

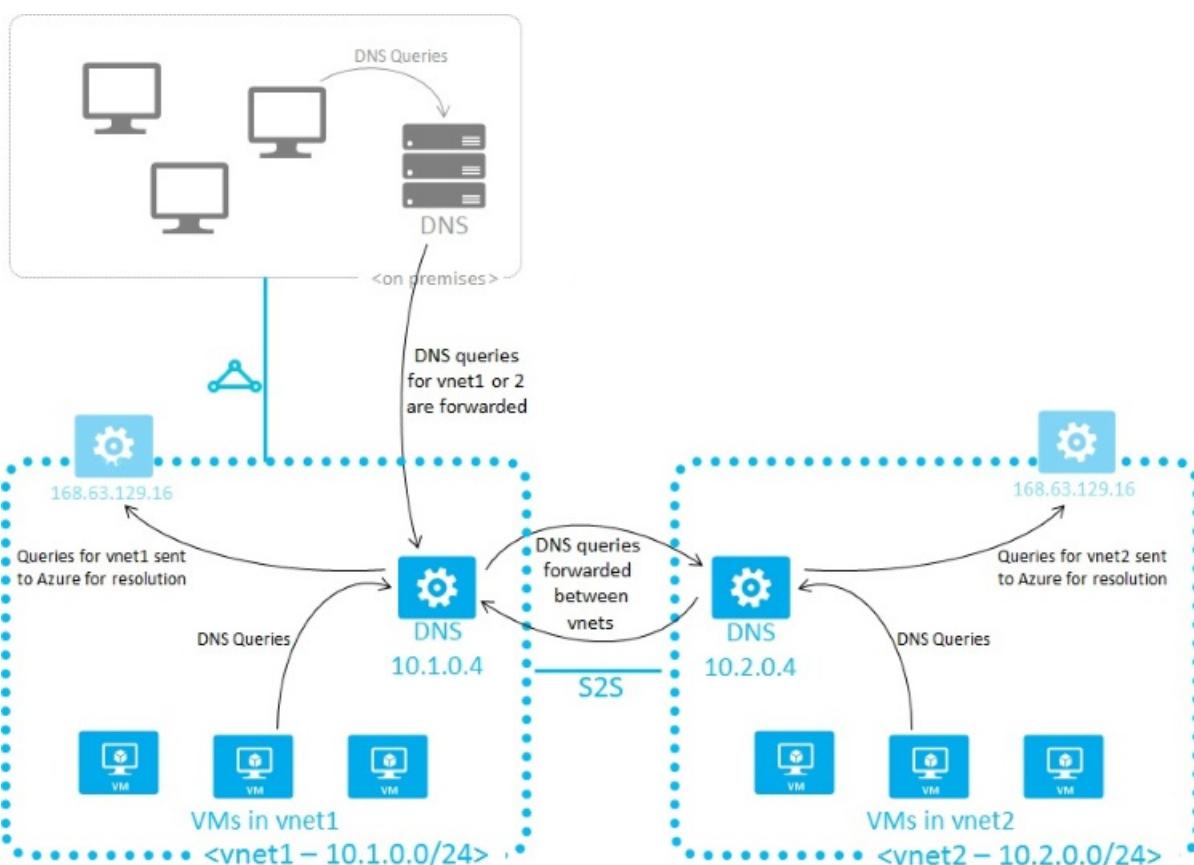
- add 'echo "options timeout:1 attempts:5"' to '/etc/NetworkManager/dispatcher.d/11-dhclient'
 - run 'service network restart' to update

Name resolution using your own DNS server

There are a number of situations where your name resolution needs may go beyond the features provided by Azure, for example when using Active Directory domains or when you require DNS resolution between virtual networks (vnets). 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 Azure's recursive resolvers to resolve hostnames 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. This allows VMs to see both your on-premises resources (via the DC) and Azure-provided hostnames (via the forwarder). Access to Azure's recursive resolvers is provided via the virtual IP 168.63.129.16.

DNS forwarding also enables inter-vnet DNS resolution and allows your on-premises machines to resolve Azure-provided hostnames. In order to resolve a VM's hostname, the DNS server VM must reside in the same virtual network and be configured to forward hostname queries to Azure. As the DNS suffix is different in each vnet, you can use conditional forwarding rules to send DNS queries to the correct vnet for resolution. The following image shows two vnets and an on-premises network doing inter-vnet DNS resolution using this method. An example DNS forwarder is available in the [Azure Quickstart Templates gallery](#) and [GitHub](#).



When using Azure-provided name resolution, an Internal DNS suffix (*.internal.cloudapp.net) is provided to each VM using DHCP. This enables hostname resolution as the hostname records are in the internal.cloudapp.net zone. When using your own name resolution solution, the IDNS suffix is not supplied to VMs because it interferes with other DNS architectures (like domain-joined scenarios). Instead we provide a non-functioning placeholder (reddog.microsoft.com).

If needed, the Internal DNS suffix can be determined using PowerShell or the API:

- For virtual networks in Resource Manager deployment models, the suffix is available via the [network interface card](#) resource or via the `Get-AzureRmNetworkInterface` cmdlet.
- In classic deployment models, the suffix is available via the `Get Deployment API` call or via the `Get-AzureVM -Debug` cmdlet.

If forwarding queries to Azure doesn't suit your needs, you will need to provide your own DNS solution. Your DNS solution will need to:

- Provide appropriate hostname resolution, e.g. via [DDNS](#). Note, if using DDNS you may need to disable DNS record scavenging as Azure's DHCP leases are very long and scavenging may 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 using Azure VMs as DNS servers, IPv6 should be disabled and an [Instance-Level Public IP](#) should be assigned to each DNS server VM. If you choose to use Windows Server as your DNS server, [this article](#) provides additional performance analysis and optimizations.

Specifying DNS servers

When using your own DNS servers, Azure provides the ability to specify multiple DNS servers per virtual network or per network interface (Resource Manager) / cloud service (classic). DNS servers specified for a cloud service/network interface get precedence over those specified for the virtual network.

NOTE

Network connection properties, such as DNS server IPs, should not be edited directly within Windows VMs as they may get erased during service heal when the virtual network adaptor gets replaced.

When using the Resource Manager deployment model, DNS servers can be specified in the Portal, API/Templates ([vnet](#), [nic](#)) or PowerShell ([vnet](#), [nic](#)).

When using the classic deployment model, DNS servers for the virtual network can be specified in the Portal or [the Network Configuration file](#). For cloud services, the DNS servers are specified via [the Service Configuration file](#) or in PowerShell ([New-AzureVM](#)).

NOTE

If you change the DNS settings for a virtual network/virtual machine that is already deployed, you need to restart each affected VM for the changes to take effect.

Next steps

Resource Manager deployment model:

- [Create or update a virtual network](#)
- [Create or update a network interface card](#)
- [New-AzureRmVirtualNetwork](#)
- [New-AzureRmNetworkInterface](#)

Classic deployment model:

- [Azure Service Configuration Schema](#)
- [Virtual Network Configuration Schema](#)
- [Configure a Virtual Network by Using a Network Configuration File](#)

Create a virtual network using the Azure portal

1/17/2017 • 3 min to read • [Edit on GitHub](#)

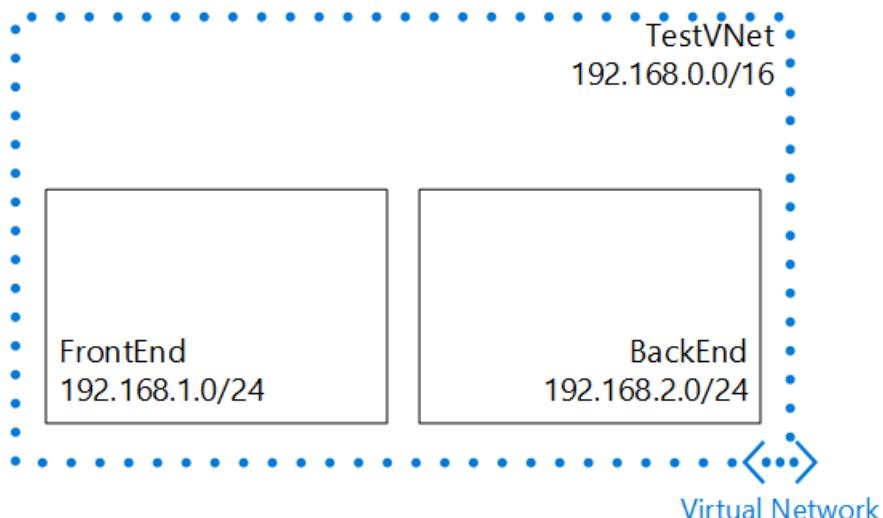
An Azure virtual network (VNet) is a representation of your own network in the cloud. You can control your Azure network settings and define DHCP address blocks, DNS settings, security policies, and routing. You can also further segment your VNet into subnets and deploy Azure IaaS virtual machines (VMs) 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.

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 explains how to create a VNet through the Resource Manager deployment model using the Azure portal. You can also create a VNet through Resource Manager using other tools or create a VNet through the classic deployment model by selecting a different option from the following list:

Scenario

To better illustrate how to create a VNet and subnets, this document will use the scenario below.



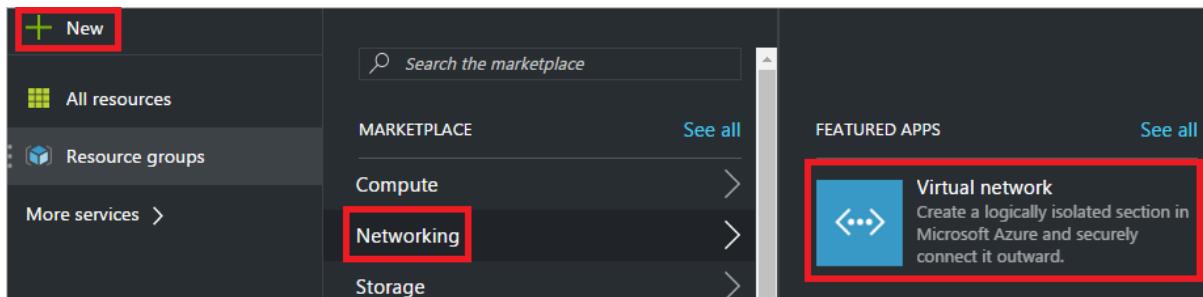
In this scenario you will create a VNet named **TestVNet** with a reserved CIDR block of **192.168.0.0/16**. Your VNet will contain the following subnets:

- **FrontEnd**, using **192.168.1.0/24** as its CIDR block.
- **BackEnd**, using **192.168.2.0/24** as its CIDR block.

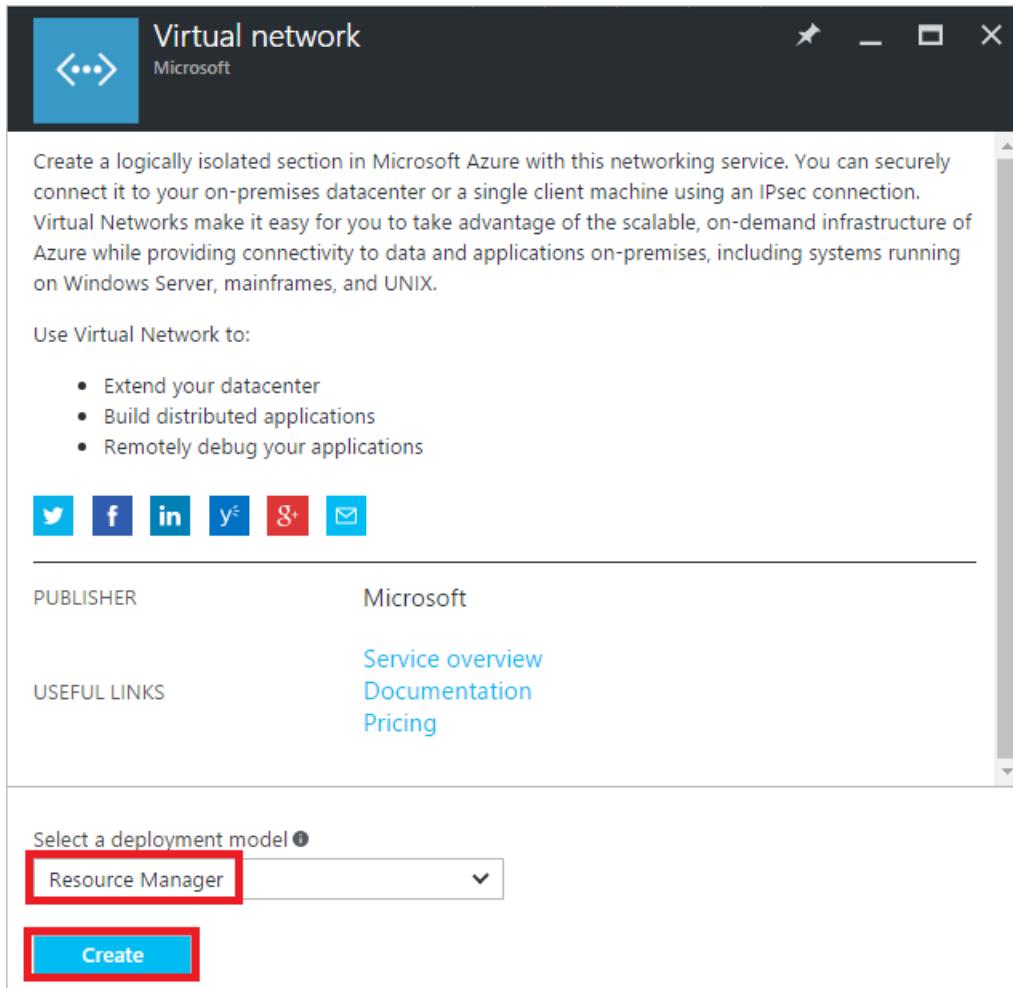
Create a virtual network

To create a virtual network using the Azure portal, complete the following steps:

1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Click **New > Networking > Virtual network**, as shown in the following picture:



3. In the **Virtual network** blade that appears, ensure that **Resource Manager** is selected and click **Create**, as shown in the following picture:



4. In the **Create virtual network** blade that appeared, enter **TestVNet** for **Name**, **192.168.0.0/16** for **Address space**, **FrontEnd** for **Subnet name** **192.168.1.0/24** for **Subnet address range**, **TestRG** for **Resource group**, select your **Subscription**, a **Location** and click the **Create** button, as shown in the following picture:

Create virtual network

* Name
TestVNet

* Address space
192.168.0.0/16
192.168.0.0 - 192.168.255.255 (65536 addresses)

* Subnet name
FrontEnd

* Subnet address range
192.168.1.0/24
192.168.1.0 - 192.168.1.255 (256 addresses)

* Subscription
[dropdown]

* Resource group
Create new Use existing
TestRG

* Location
West US

Pin to dashboard

Create [Automation options](#)

Alternatively, you can select an existing resource group. To learn more about resource groups, read the [Resource Manager overview](#) article. You can also select a different location. To learn more about Azure locations and regions, read the [Azure regions](#) article.

- The portal only enables you to create one subnet when creating a VNet. For this scenario, a second subnet must be created after the VNet is created. To create the second subnet, click **All resources**, then click **TestVNet** in the **All resources** blade, as shown in the following picture:

NAME	RESOURCE GROUP	LOCATION	TYPE
TestVNet	TestRG	West US	Virtual network

- In the **TestVNet** blade that appears, click **Subnet**, then click **+Subnet**, enter *BackEnd* for **Name**, 192.168.2.0/24 for **Address range** in the **Add subnet** blade, then click **OK**, as shown in the following picture:

The screenshot shows the Azure portal interface for managing subnets. On the left, there's a sidebar with options like Overview, Activity log, Access control (IAM), Tags, and Subnets (which is highlighted with a red box). The main area shows a list of existing subnets: 'FrontEnd' with address range 192.168.1.0/24. To the right, a modal window titled 'Add subnet' is open, allowing the creation of a new subnet. It has fields for 'Name' (set to 'BackEnd'), 'Address range (CIDR block)' (set to '192.168.2.0/24'), and other settings like Network security group (None) and Route table (None). The 'OK' button at the bottom right of the modal is also highlighted with a red box.

7. The two subnets are listed, as shown in the following picture:

This screenshot shows the list of subnets in the Azure portal. The 'Subnets' tab is selected in the top navigation bar. The table lists two subnets: 'FrontEnd' with address range 192.168.1.0/24 and 'BackEnd' with address range 192.168.2.0/24. Both subnets have 251 available addresses and are currently unassigned to a security group.

This article explained how to create a virtual network with two subnets for testing. Before creating a virtual network for production use, we recommend reading the [Virtual network overview](#) and [Virtual network plan and design](#) articles to fully understand virtual networks and all settings.

Next steps

Learn how to connect:

- A virtual machine (VM) to a virtual network by reading the [Create a Windows VM](#) or [Create a Linux VM](#) articles. Instead of creating a VNet and subnet in the steps of the articles, you can select an existing VNet and subnet to connect a VM to.
- The virtual network to other virtual networks by reading the [Connect VNets](#) article.
- The virtual network to an on-premises network using a site-to-site virtual private network (VPN) or ExpressRoute circuit. Learn how by reading the [Connect a VNet to an on-premises network using a site-to-site VPN](#) and [Link a VNet to an ExpressRoute circuit](#) articles.

Create your first Windows virtual machine in the Azure portal

1/17/2017 • 3 min to read • [Edit on GitHub](#)

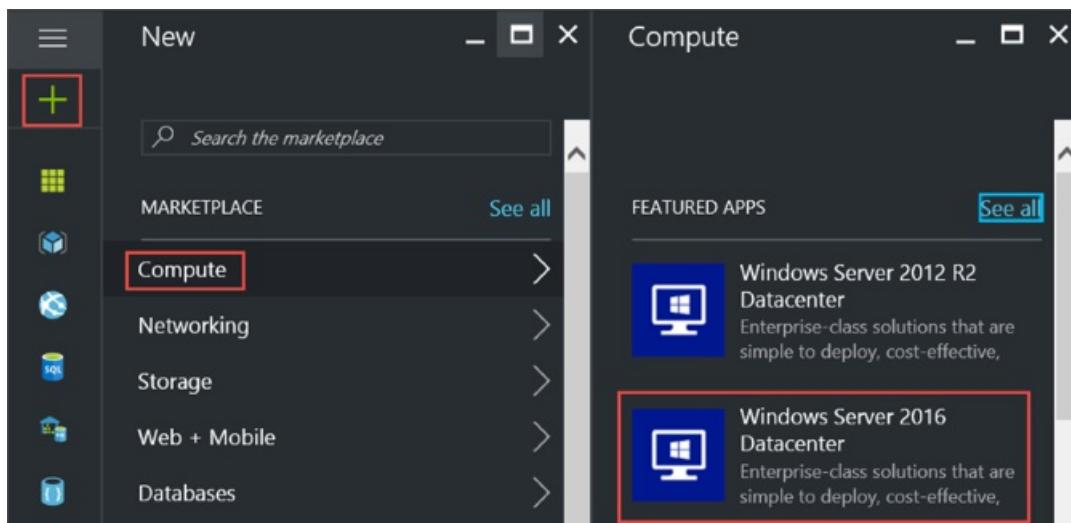
This tutorial shows you how easy it is to create a Windows virtual machine (VM) in just a few minutes, by using the Azure portal.

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

Choose the VM image from the marketplace

We use a Windows Server 2016 Datacenter image as an example, but that's just one of the many images Azure offers. Your image choices depend on your subscription. For example, some desktop images are available to [MSDN subscribers](#).

1. Sign in to the [Azure portal](#).
2. Starting in the upper-left, click **New > Compute > Windows Server 2016 Datacenter**.



3. On the **Windows Server 2016 Datacenter** blade, in **Select a deployment model**, verify that **Resource Manager** is selected. Click **Create**.



Create the Windows virtual machine

After you select the image, you can use the default settings and quickly create the virtual machine.

1. On the **Basics** blade, enter a **Name** for the virtual machine. In this example, *HeroVM* is the name of the virtual machine. The name must be 1-15 characters long and it cannot contain special characters.
2. Enter a **User name**, and a strong **Password** that will be used to create a local account on the VM. The local account is used to sign in to and manage the VM. In this example, *azureuser* is the user name.

The password must be 8-123 characters long and meet three out of the four following complexity requirements: one lower case character, one upper case character, one number, and one special character. See more about [username and password requirements](#).

3. Select an existing [Resource group](#) or type the name for a new one. In this example, *HeroVMRG* is the name of the resource group.
4. Select an Azure datacenter **Location**. In this example, *East US** is the location.
5. When you are done, click **OK** to continue to the next section.

Basics

* Name
HeroVM ✓

VM disk type ⓘ
SSD

* User name
azureuser ✓

* Password
***** ✓

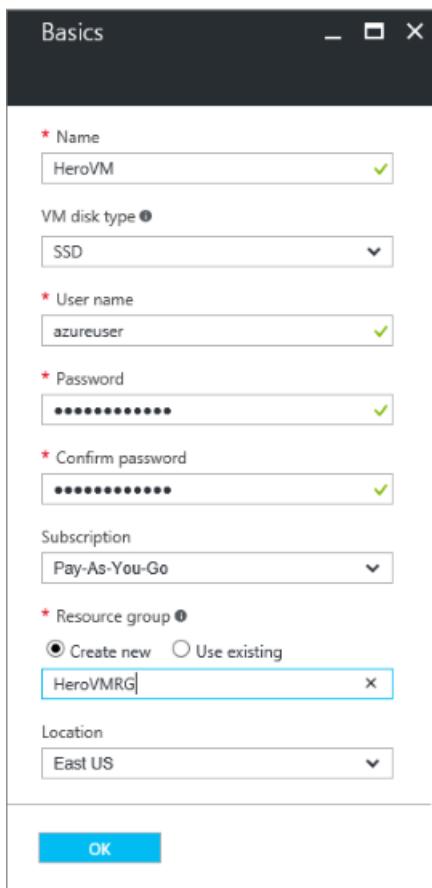
* Confirm password
***** ✓

Subscription
Pay-As-You-Go

* Resource group ⓘ
 Create new Use existing
HeroVMRG x

Location
East US

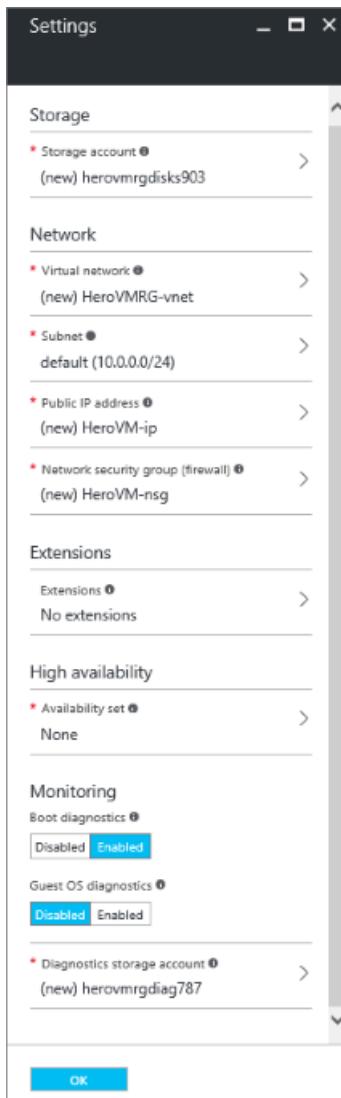
OK



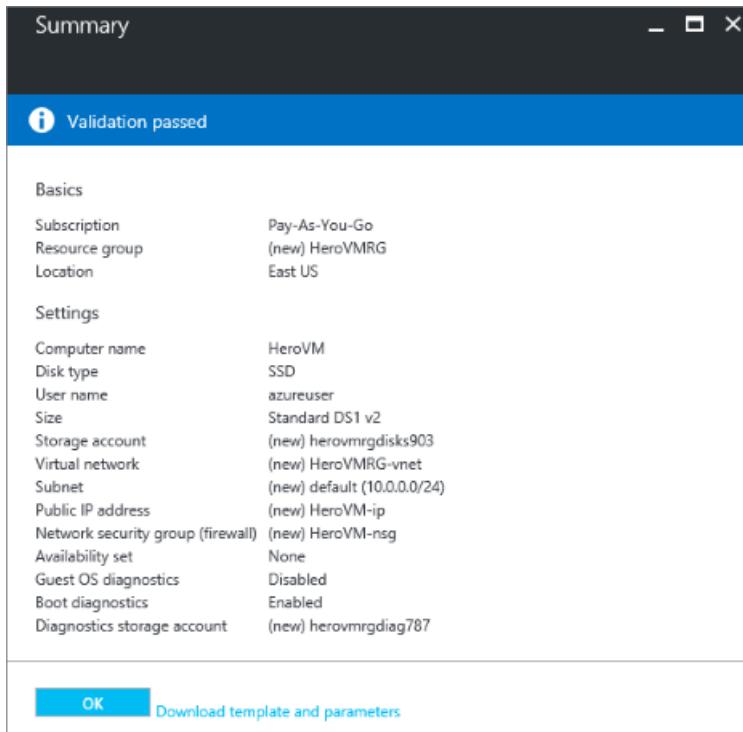
6. Choose a VM [size](#), and then click **Select** to continue. In this example, *DS1_V2 Standard* is the VM size.

Choose a size																													
Browse the available sizes and their features																													
Prices presented are estimates in your local currency that include only Azure infrastructure costs and any discounts for the subscription and location. The prices don't include any applicable software costs. Recommended sizes are determined by the publisher of the selected image based on hardware and software requirements.																													
★ Recommended View all																													
<table border="1"> <thead> <tr> <th>DS1_V2 Standard</th> <th>DS2_V2 Standard</th> <th>DS11_V2 Standard</th> </tr> </thead> <tbody> <tr> <td>1 Core</td> <td>2 Cores</td> <td>2 Cores</td> </tr> <tr> <td>3.5 GB</td> <td>7 GB</td> <td>14 GB</td> </tr> <tr> <td>2 Data disks</td> <td>4 Data disks</td> <td>4 Data disks</td> </tr> <tr> <td>3200 Max IOPS</td> <td>6400 Max IOPS</td> <td>6400 Max IOPS</td> </tr> <tr> <td>7 GB Local SSD</td> <td>14 GB Local SSD</td> <td>28 GB Local SSD</td> </tr> <tr> <td>Load balancing</td> <td>Load balancing</td> <td>Load balancing</td> </tr> <tr> <td>Premium disk support</td> <td>Premium disk support</td> <td>Premium disk support</td> </tr> <tr> <td>104.16 USD/MONTH (ESTIMATED)</td> <td>208.32 USD/MONTH (ESTIMATED)</td> <td>245.52 USD/MONTH (ESTIMATED)</td> </tr> </tbody> </table>			DS1_V2 Standard	DS2_V2 Standard	DS11_V2 Standard	1 Core	2 Cores	2 Cores	3.5 GB	7 GB	14 GB	2 Data disks	4 Data disks	4 Data disks	3200 Max IOPS	6400 Max IOPS	6400 Max IOPS	7 GB Local SSD	14 GB Local SSD	28 GB Local SSD	Load balancing	Load balancing	Load balancing	Premium disk support	Premium disk support	Premium disk support	104.16 USD/MONTH (ESTIMATED)	208.32 USD/MONTH (ESTIMATED)	245.52 USD/MONTH (ESTIMATED)
DS1_V2 Standard	DS2_V2 Standard	DS11_V2 Standard																											
1 Core	2 Cores	2 Cores																											
3.5 GB	7 GB	14 GB																											
2 Data disks	4 Data disks	4 Data disks																											
3200 Max IOPS	6400 Max IOPS	6400 Max IOPS																											
7 GB Local SSD	14 GB Local SSD	28 GB Local SSD																											
Load balancing	Load balancing	Load balancing																											
Premium disk support	Premium disk support	Premium disk support																											
104.16 USD/MONTH (ESTIMATED)	208.32 USD/MONTH (ESTIMATED)	245.52 USD/MONTH (ESTIMATED)																											
Select																													

7. On the **Settings** blade, you can change the storage and network options. For this tutorial, accept the default settings. If you selected a virtual machine size that supports it, you can try Azure Premium Storage by selecting **Premium (SSD)** in **Disk type**. When you're done making changes, click **OK**.



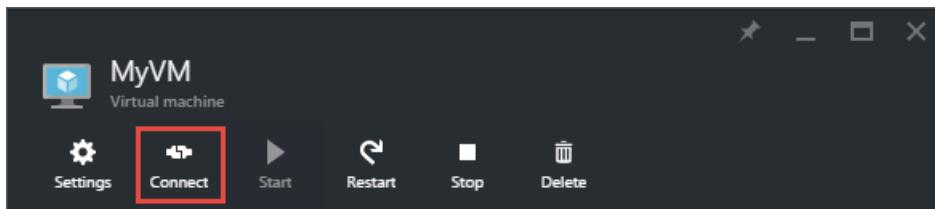
- Click **Summary** to review your choices. When you see the **Validation passed** message, click **OK**.



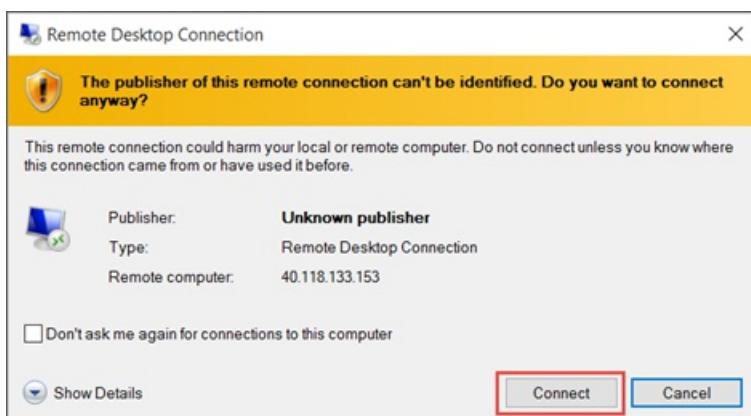
- While Azure creates the virtual machine, you can track the progress by clicking on **Virtual Machines** on left. When the VM has been created, the status will change to **Running**.

Connect to the virtual machine and sign on

- On the left, click **Virtual Machines**.
- Select the virtual machine from the list.
- On the blade for the virtual machine, click **Connect**. This creates and downloads a Remote Desktop Protocol file (.rdp file) that is like a shortcut to connect to your machine. You might want to save the file to your desktop for easy access. **Open** this file to connect to your VM.

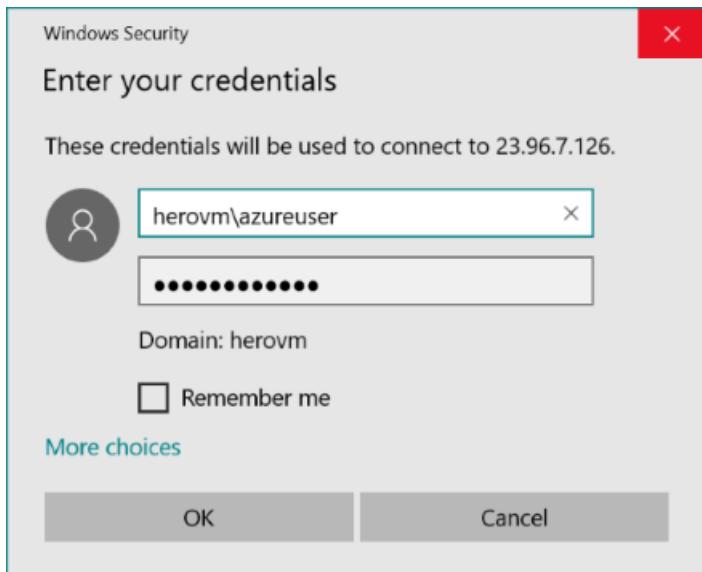


- You get a warning that the .rdp is from an unknown publisher. This is normal. In the Remote Desktop window, click **Connect** to continue.

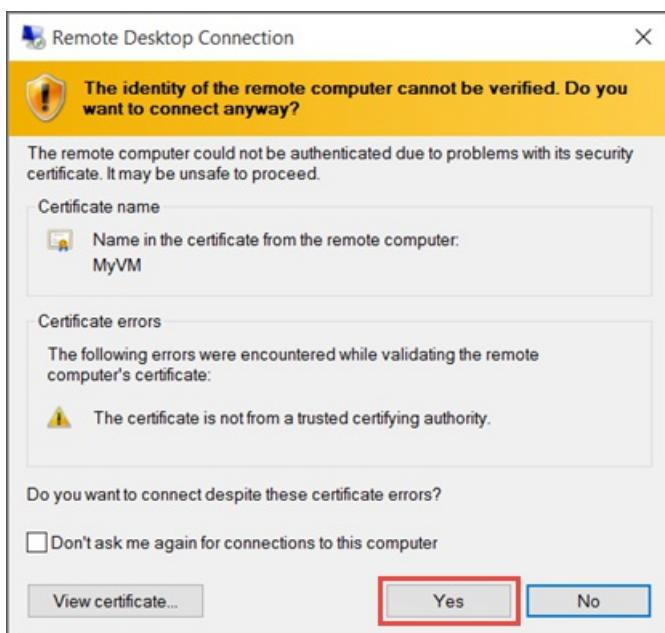


- In the Windows Security window, type the username and password for the local account that you created

when you created the VM. The username is entered as `vmname\username`, then click **OK**.



6. You get a warning that the certificate cannot be verified. This is normal. Click **Yes** to verify the identity of the virtual machine and finish logging on.

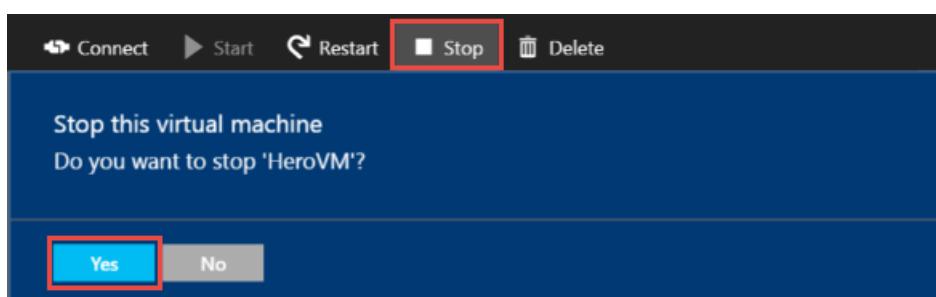


If you run in to trouble when you try to connect, see [Troubleshoot Remote Desktop connections to a Windows-based Azure Virtual Machine](#).

You can now work with the virtual machine as you would with any other server.

Optional: Stop the VM

It is a good idea to stop the VM so you don't incur charges when you aren't actually using it. Just click **Stop** and then click **Yes**.



Click the **Start** button to restart the VM when you're ready to use it again.

Next steps

- You can experiment with your new VM by [installing IIS](#). This tutorial also shows how to open port 80 to incoming web traffic using a network security group (NSG).
- You can also [create a Windows VM by using PowerShell](#) or [create a Linux virtual machine](#) by using the Azure CLI.
- If you're interested in automating deployments, see [Create a Windows virtual machine by using a Resource Manager template](#).

Plan and design Azure Virtual Networks

1/17/2017 • 14 min to read • [Edit on GitHub](#)

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. And resources can only be connected to a VNet that exists in the same region and subscription they are in.
- You can connect VNets to each other by using an [Azure VPN Gateway](#). You can also connect VNets across regions and subscriptions this way.
- 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.
networkSecurityGroup	NSG applied to the subnet	
routeTable	Route table applied to the subnet	

PROPERTY	DESCRIPTION	CONSTRAINTS
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 --- VNetProd1[VNet (production)]; Subscription1 --- VNetDev1[VNet (development)]; Enterprise --- Subscription2[Subscription 2 (app 2)]; Subscription2 --- VNetProd2[VNet (production)]; Subscription2 --- VNetDev2[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 --- VNetApp2Prod2[VNet app2 (production)]; Enterprise --- Subscription2[Subscription 2 (BU 2)]; Subscription2 --- VNetApp1Prod2[VNet app1 (production)]; Subscription2 --- VNetApp1Dev2[VNet app1 (development)]; Subscription2 --- VNetApp2Prod3[VNet app2 (development)]; Subscription2 --- VNetApp2Prod4[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 --- VNetProd1[VNet (production)]; Subscription1 --- VNetDev1[VNet (development)]; Enterprise --- Subscription2[Subscription 2 (BU 2)]; Subscription2 --- VNetProd2[VNet (production)]; Subscription2 --- VNetDev2[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)] --- NSG1Front1[NSG 1 (app1)] Subnet1 --- NSG2Front2[NSG 2 (app2)] Subnet2[Subnet 2 (Back end)] --- NSG1Back1[NSG 1 (app1)] Subnet2 --- NSG2Back2[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)] --- NSG1Front1[NSG 1] Subnet2[Subnet 2 (app1 back end)] --- NSG2Back1[NSG 2] Subnet3[Subnet 3 (app2 front end)] --- NSG3Front2[NSG 3] Subnet4[Subnet 2 (app2 back end)] --- NSG4Back2[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 and App3 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

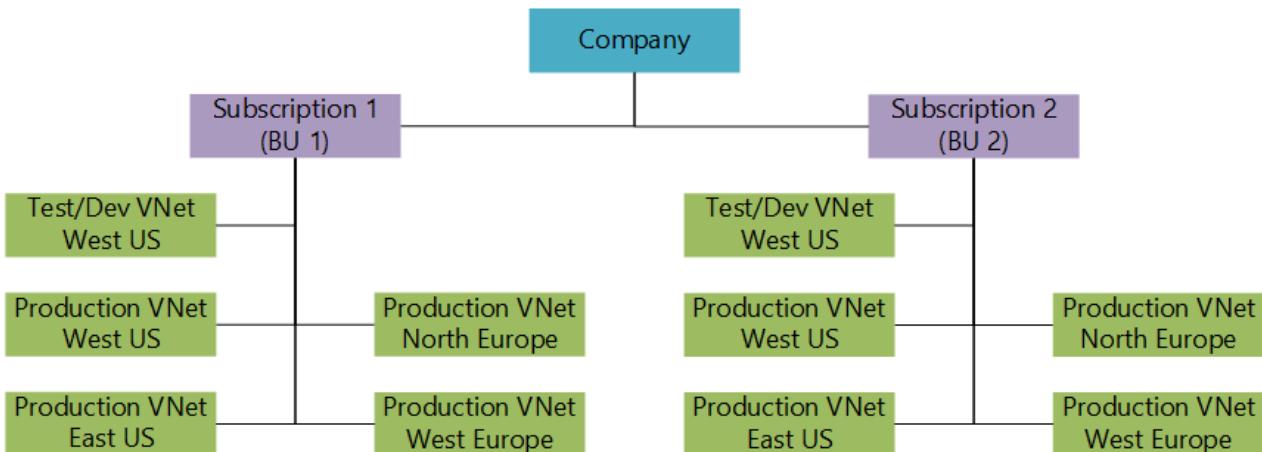
You should follow the design specifying subscriptions, VNets, subnets, and NSGs. We will discuss NSGs here, but you should learn more about [NSGs](#) before finishing your design.

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.

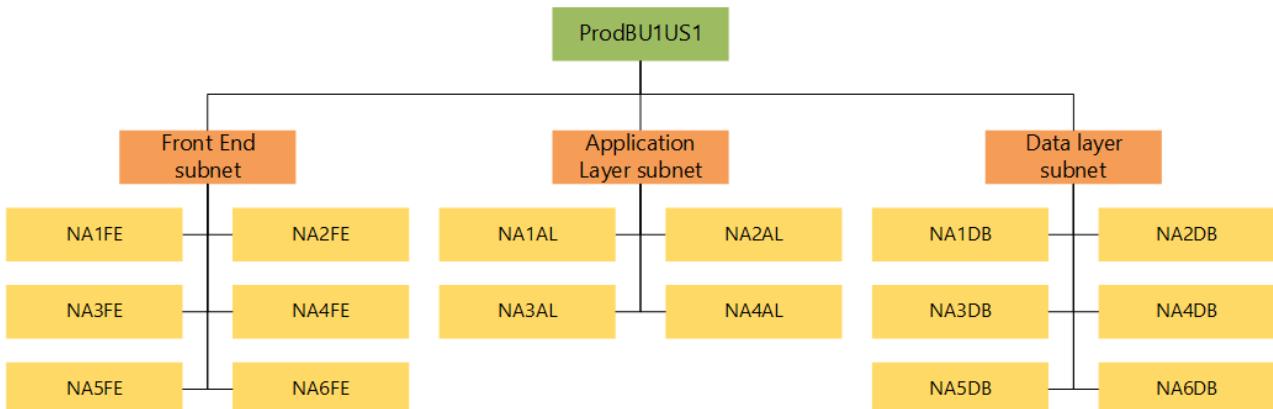
SUBSCRIPTION	VNET	AZURE REGION	ADDRESS SPACE
BU1	ProdBU1US1	West US	172.16.0.0/16
BU1	ProdBU1US2	East US	172.17.0.0/16
BU1	ProdBU1EU1	North Europe	172.18.0.0/16
BU1	ProdBU1EU2	West Europe	172.19.0.0/16
BU1	TestDevBU1	West US	172.20.0.0/16
BU2	TestDevBU2	West US	172.21.0.0/16
BU2	ProdBU2US1	West US	172.22.0.0/16
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](#) based on a scenario.
- Understand how to [load balance](#) IaaS VMs and [manage routing](#) over multiple Azure regions.
- Learn more about [NSGs](#) and [how to plan and design](#) an NSG solution.
- Learn more about your [cross-premises and VNet connectivity options](#).

Network security groups

1/17/2017 • 16 min to read • [Edit on GitHub](#)

A network security group (NSG) contains a list of access control list (ACL) rules that allow or deny network traffic to your VM instances in a Virtual Network. NSGs can be associated with either subnets or individual VM instances within that subnet. When a NSG is associated with a subnet, the ACL rules apply to all the VM instances in that subnet. In addition, traffic to an individual VM can be restricted further by associating a NSG directly to that VM.

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 Can have up to 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 hosted	NSGs can only be applied to resources within the region it is created	See limits below to understand how many NSGs you can have in a region
Resource group	Resource group the NSG belongs to	Although an NSG belongs to 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	Rules that define what traffic is allowed, or denied		See NSG rules below

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. For information about how to do this, see [Managing Access Control Lists \(ACLs\) for Endpoints by using PowerShell](#).

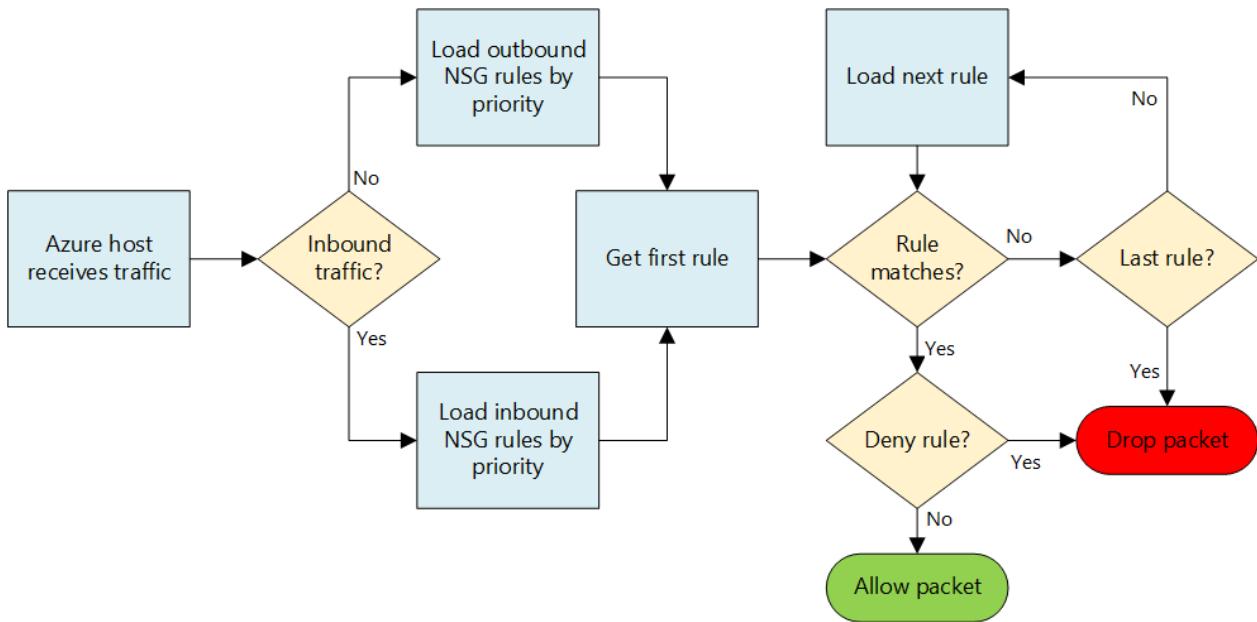
NSG rules

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 Can have up to 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 make sure you only use when really necessary
Source port range	Source port range to match for the rule	Single port number from 1 to 65535, port range (i.e. 1-65635), or * (for all ports)	Source ports could be ephemeral. Unless your client program is using a specific port, please 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 (i.e. 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 (i.e. 10.10.10.10), IP subnet (i.e. 192.168.1.0/24), default tag , or * (for all addresses)	Consider using ranges, default 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 (i.e. 10.10.10.10), IP subnet (i.e. 192.168.1.0/24), default tag , or * (for all addresses)	Consider using ranges, default 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 to come between existing rules
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 figure above shows how NSG rules are processed.

Default Tags

Default tags are system-provided identifiers to address a category of IP addresses. You can use default tags in the **source address prefix** and **destination address prefix** properties of any rule. There are three default tags you can use.

- **VIRTUAL_NETWORK:** This default tag denotes all of your network address space. It includes the virtual network address space (CIDR ranges defined in Azure) as well as all connected on-premises address spaces and connected Azure VNets (local networks).
- **AZURE_LOADBALANCER:** This default tag denotes Azure's Infrastructure load balancer. This will translate to an Azure datacenter IP where Azure's health probes originate.
- **INTERNET:** This default tag denotes the IP address space that is outside the virtual network and reachable by public Internet. This range includes [Azure owned public IP space](#) as well.

Default Rules

All NSGs contain a set of default 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.

As illustrated by the default rules below, traffic originating and ending in a virtual network is allowed both in Inbound and Outbound directions. While connectivity to the Internet is allowed for Outbound direction, it is by default blocked for Inbound direction. There is a default rule to allow Azure's load balancer to probe the health of your VMs and role instances. You can override this rule, if you are not using a load balanced set.

Inbound default rules

NAME	PRIORITY	SOURCE IP	SOURCE PORT	DESTINATION IP	DESTINATION PORT	PROTOCOL	ACCESS
ALLOW VNET INBOUND	65000	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*	*	ALLOW
ALLOW AZURE LOAD BALANCER INBOUND	65001	AZURE_LOADBALANCER	*	*	*	*	ALLOW
DENY ALL INBOUND	65500	*	*	*	*	*	DENY

Outbound default rules

NAME	PRIORITY	SOURCE IP	SOURCE PORT	DESTINATION IP	DESTINATION PORT	PROTOCOL	ACCESS
ALLOW VNET OUTBOUND	65000	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*	*	ALLOW
ALLOW INTERNET OUTBOUND	65001	*	*	INTERNET	*	*	ALLOW
DENY ALL OUTBOUND	65500	*	*	*	*	*	DENY

Associating NSGs

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

- **Associating an NSG to a VM (classic deployments only).** When you associate an NSG to a VM, the network access rules in the NSG are applied to all traffic that destined and leaving the VM.
- **Associating an NSG to a NIC (Resource Manager deployments only).** When you associate an NSG to a NIC, the network access rules in the NSG are applied only to that NIC. That means that in a multi-NIC VM, if an NSG is applied to a single NIC, it does not affect traffic bound to other NICs.
- **Associating an NSG to a subnet (all deployments).** When you associate an NSG to a subnet, the network access rules in the NSG are applied to all the IaaS and PaaS resources in 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 bound to. When that happens, all network access 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 will be dropped.
2. **NSG applied to NIC** (Resource Manager) or VM (classic): If VM\NIC NSG has a matching rule to deny traffic, packet will be dropped at VM\NIC, although subnet NSG has a matching rule to allow traffic.

- **Outbound traffic**

1. **NSG applied to NIC** (Resource Manager) or VM (classic): If VM\NIC NSG has a matching rule to deny traffic, the packet will be dropped.
2. **NSG applied to subnet:** If subnet NSG has a matching rule to deny traffic, packet will be dropped here, although VM\NIC NSG has a matching rule to allow 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 classic or Resource Manager deployment models using the different tools listed below.

DEPLOYMENT TOOL	CLASSIC	RESOURCE MANAGER
Classic portal	No	No
Azure portal	Yes	Yes
PowerShell	Yes	Yes
Azure CLI	Yes	Yes
ARM 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 (NICs in the same VM, VMs or other resources such as cloud services or application service environments connected to the same subnet, or between resources connected to different subnets)?
2. Are the resources you want to filter traffic to/from connected to subnets in existing VNets or will they be connected to new VNets or subnets?

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

Design considerations

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

Limits

You need to consider the following limits when designing your NSGs.

DESCRIPTION	DEFAULT LIMIT	IMPLICATIONS
Number of NSGs you can associate to a subnet, VM, or NIC	1	This means you cannot combine NSGs. Ensure all the rules needed for a given set of resources are included in a single NSG.
NSGs per region per subscription	100	By default, a new NSG is created for each VM you create in the Azure portal. If you allow this default behavior, you will run out of NSGs quickly. Make sure you keep this limit in mind during your design, and separate your resources into multiple regions or subscriptions if necessary.
NSG rules per NSG	200	Use a broad range of IP and ports to ensure you do not go over this limit.

IMPORTANT

Make sure you view all the [limits related to networking services in Azure](#) before designing your solution. Some limits can be increased by opening a support ticket.

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

You need to take into account the special rules listed below. Make sure you do not block traffic allowed by those rules, otherwise your infrastructure will not be able to 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 will be 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 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 this IP address should not be considered as an attack.
- Licensing (Key Management Service):** Windows images running in the virtual machines should be licensed. To do this, a licensing request is sent to the Key Management Service host servers that handle such queries. This will always be on outbound 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 Virtual Network by default through the Inbound VNet rule(Default rule 65000 inbound) that allows traffic from/to 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, make sure you do **NOT** apply an NSG to that subnet. If you do so, your cross VNet or cross premises connectivity will not work.
- If you need to implement a virtual appliance, make sure you deploy the virtual appliance on its own subnet, so that your User Defined Routes (UDRs) can work correctly. You can implement a subnet level NSG to filter traffic in and out of this subnet. Learn more about [how to control traffic flow and use virtual appliances](#).

Load balancers

- Consider the load balancing and NAT rules for each load balancer being used by each of your workloads. These rules are bound to a back end pool that contains NICs (Resource Manager deployments) or VMs/role instances (classic deployments). Consider creating an NSG for each back end pool, allowing only traffic mapped through the rules implemented in the load balancers. That guarantees that traffic coming to the backend pool directly, without passing 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 in a Resource Manager deployment. If you are restricting traffic to VMs and role instances that are part of a backend pool in a load balancer by using NSGs, keep in mind that the destination port for the incoming traffic is the actual port in the VM or role instance, not the port exposed by the load balancer. Also keep in mind that 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.
- Similar to public facing load balancers, when you create NSGs to filter traffic coming through an internal load balancer (ILB), you need to understand that the source port and address range applied are the ones from the computer originating the call, not the load balancer. And the destination port and address range are related to the computer receiving the traffic, not the load balancer.

Other

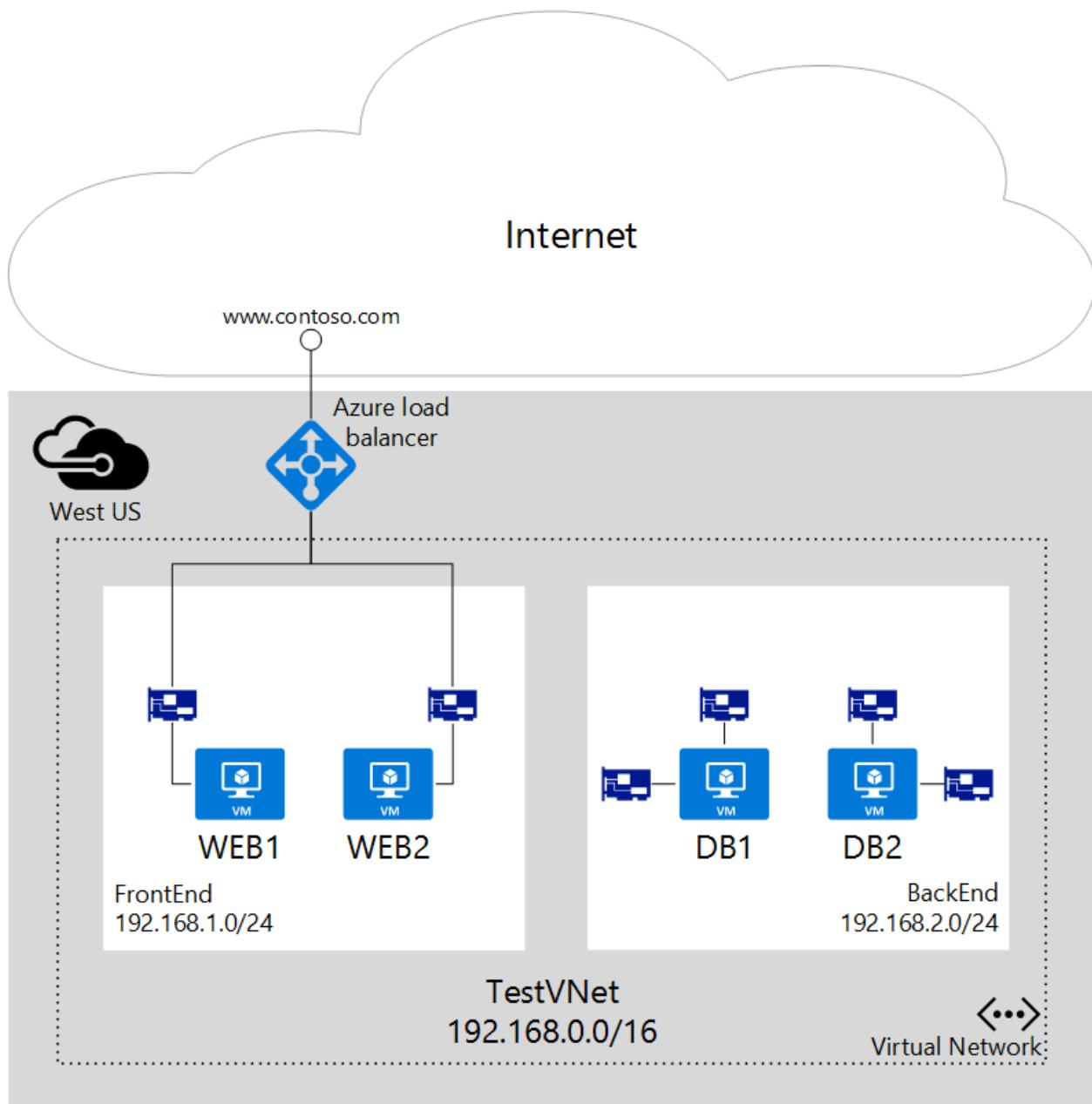
- Endpoint-based ACLs and NSGs are not supported on the same VM instance. If you want to use an NSG and have an endpoint ACL already in place, first remove the endpoint ACL. For information about how to do this, see [Manage endpoint ACLs](#).
- In the Resource Manager deployment model, you can use an NSG associated to a NIC for VMs with multiple NICs to enable management (remote access) by NIC, therefore segregating traffic.
- 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 Azure Virtual Networks and therefore, traffic to and from them cannot be filtered with NSGs. Read the documentation for the services you use to determine whether or not they can be connected to VNets.

Sample deployment

To illustrate the application of the information in this article, we'll define NSGs to filter network traffic for a two tier workload solution with the following requirements:

1. Separation of traffic between front end (Windows web servers) and back end (SQL database servers).
2. Load balancing rules forwarding traffic to the load balancer to all web servers on port 80.
3. NAT rules forwarding traffic coming in port 50001 on load balancer to port 3389 on only one VM in the front end.
4. No access to the front end or back end VMs from the Internet, with exception of requirement number 1.

5. No access from the front end or back end to the Internet.
6. Access to port 3389 to any web server in the front end, for traffic coming from the front end subnet itself.
7. Access to port 3389 to all SQL Server VMs in the back end from the front end subnet only.
8. Access to port 1433 to all SQL Server VMs in the back end from the front end subnet only.
9. Separation of management traffic (port 3389) and database traffic (1433) on different NICs in the back end VMs.



As seen in the diagram above, 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. All resources are assigned to the *West US* Azure region.

Requirements 1-6 (with exception of 3) above are all confined to subnet spaces. To minimize the number of rules required for each NSG, and to make it easy to add additional VMs to the subnets running the same workload types as the existing VMs, we can implement the following subnet level NSGs.

NSG for FrontEnd subnet

Incoming rules

RULE	ACCESS	PRIORITY	SOURCE ADDRESS RANGE	SOURCE PORT	DESTINATION ADDRESS RANGE	DESTINATION PORT	PROTOCOL
allow HTTP	Allow	100	INTERNET	*	*	80	TCP
allow RDP from FrontEnd	Allow	200	192.168.1.0 /24	*	*	3389	TCP
deny anything from Internet	Deny	300	INTERNET	*	*	*	TCP

Outgoing rules

RULE	ACCESS	PRIORITY	SOURCE ADDRESS RANGE	SOURCE PORT	DESTINATION ADDRESS RANGE	DESTINATION PORT	PROTOCOL
deny Internet	Deny	100	*	*	INTERNET	*	*

NSG for BackEnd subnet

Incoming rules

RULE	ACCESS	PRIORITY	SOURCE ADDRESS RANGE	SOURCE PORT	DESTINATION ADDRESS RANGE	DESTINATION PORT	PROTOCOL
deny Internet	Deny	100	INTERNET	*	*	*	*

Outgoing rules

RULE	ACCESS	PRIORITY	SOURCE ADDRESS RANGE	SOURCE PORT	DESTINATION ADDRESS RANGE	DESTINATION PORT	PROTOCOL
deny Internet	Deny	100	*	*	INTERNET	*	*

NSG for single VM (NIC) in FrontEnd for RDP from Internet

Incoming rules

RULE	ACCESS	PRIORITY	SOURCE ADDRESS RANGE	SOURCE PORT	DESTINATION ADDRESS RANGE	DESTINATION PORT	PROTOCOL
allow RDP from Internet	Allow	100	INTERNET	*	*	3389	TCP

NOTE

Notice how the source address range for this rule is **Internet**, and not the VIP for the load balancer; the source port is \ not 500001. *Do not get confused between NAT rules/load balancing rules and NSG rules. The NSG rules are always related to the original source and final destination of traffic, **NOT** the load balancer between the two.*

NSG for management NICs in BackEnd

Incoming rules

RULE	ACCESS	PRIORITY	SOURCE ADDRESS RANGE	SOURCE PORT	DESTINATION ADDRESS RANGE	DESTINATION PORT	PROTOCOL
allow RDP from front end	Allow	100	192.168.1.0 /24	*	*	3389	TCP

NSG for database access NICs in back end

Incoming rules

RULE	ACCESS	PRIORITY	SOURCE ADDRESS RANGE	SOURCE PORT	DESTINATION ADDRESS RANGE	DESTINATION PORT	PROTOCOL
allow SQL from front end	Allow	100	192.168.1.0 /24	*	*	1433	TCP

Since some of the NSGs above need to be associated to individual NICs, you need to deploy this scenario as a Resource Manager deployment. Notice how rules are combined for subnet and NIC level, depending on how they need to be applied.

Next steps

- [Deploy NSGs in the classic deployment model.](#)
- [Deploy NSGs in Resource Manager.](#)
- [Manage NSG logs.](#)

Create a virtual network using the Azure portal

1/17/2017 • 3 min to read • [Edit on GitHub](#)

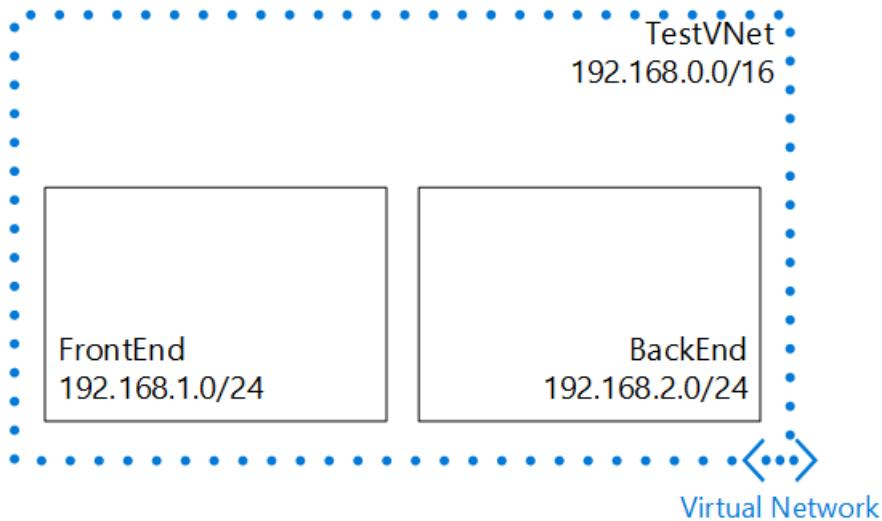
An Azure virtual network (VNet) is a representation of your own network in the cloud. You can control your Azure network settings and define DHCP address blocks, DNS settings, security policies, and routing. You can also further segment your VNet into subnets and deploy Azure IaaS virtual machines (VMs) 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.

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 explains how to create a VNet through the Resource Manager deployment model using the Azure portal. You can also create a VNet through Resource Manager using other tools or create a VNet through the classic deployment model by selecting a different option from the following list:

Scenario

To better illustrate how to create a VNet and subnets, this document will use the scenario below.



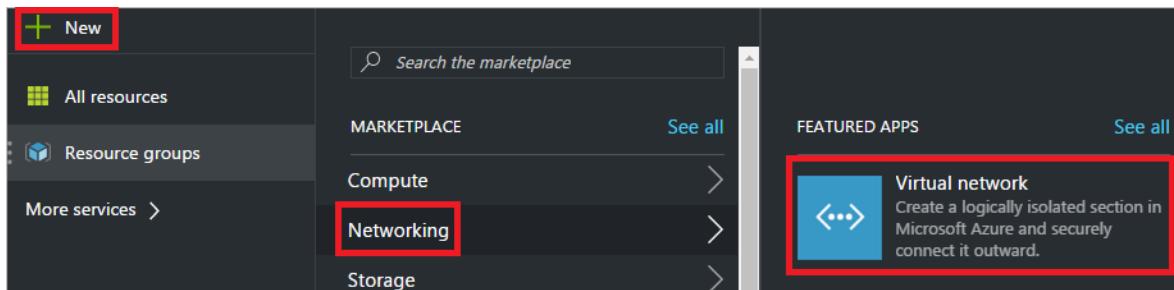
In this scenario you will create a VNet named **TestVNet** with a reserved CIDR block of **192.168.0.0/16**. Your VNet will contain the following subnets:

- **FrontEnd**, using **192.168.1.0/24** as its CIDR block.
- **BackEnd**, using **192.168.2.0/24** as its CIDR block.

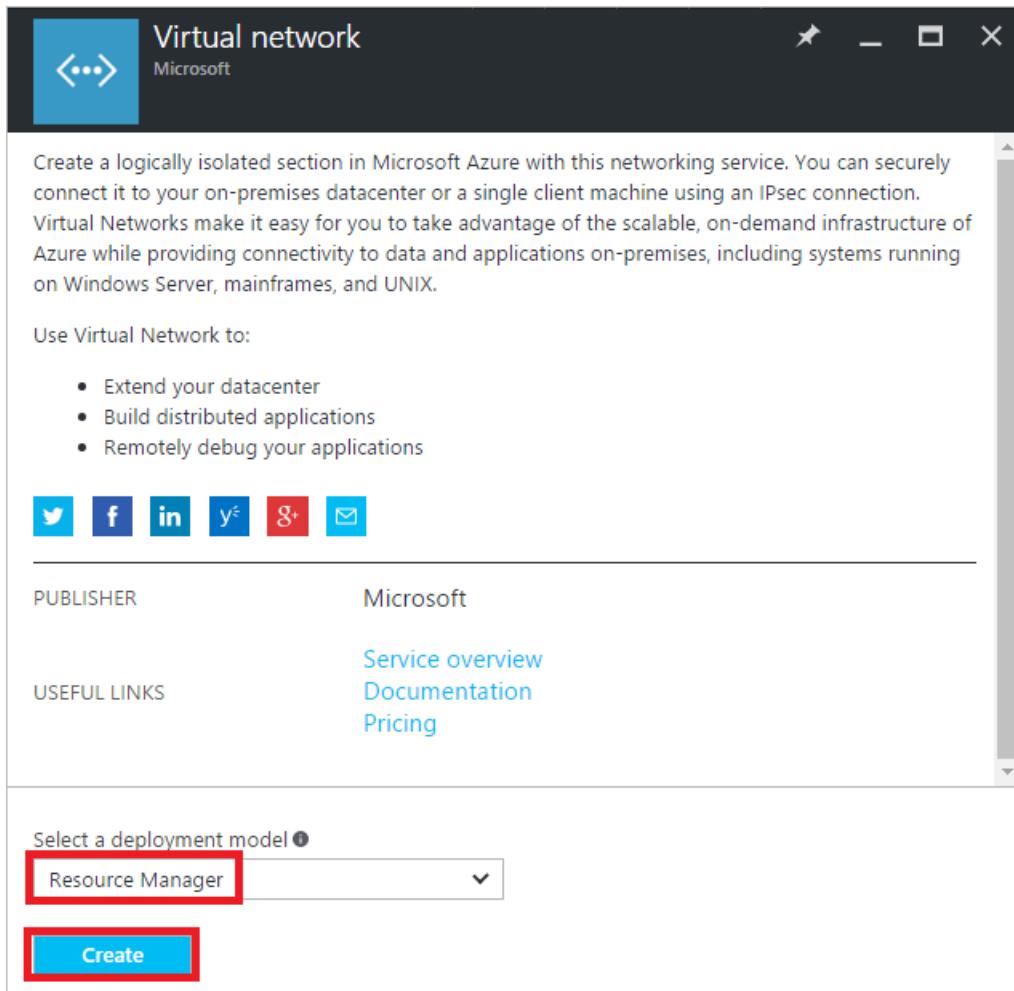
Create a virtual network

To create a virtual network using the Azure portal, complete the following steps:

1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Click **New > Networking > Virtual network**, as shown in the following picture:



3. In the **Virtual network** blade that appears, ensure that **Resource Manager** is selected and click **Create**, as shown in the following picture:



4. In the **Create virtual network** blade that appeared, enter **TestVNet** for **Name**, **192.168.0.0/16** for **Address space**, **FrontEnd** for **Subnet name** **192.168.1.0/24** for **Subnet address range**, **TestRG** for **Resource group**, select your **Subscription**, a **Location** and click the **Create** button, as shown in the following picture:

Create virtual network

* Name
TestVNet

* Address space
192.168.0.0/16
192.168.0.0 - 192.168.255.255 (65536 addresses)

* Subnet name
FrontEnd

* Subnet address range
192.168.1.0/24
192.168.1.0 - 192.168.1.255 (256 addresses)

* Subscription
[dropdown]

* Resource group
 Create new Use existing
TestRG

* Location
West US

Pin to dashboard

Create [Automation options](#)

Alternatively, you can select an existing resource group. To learn more about resource groups, read the [Resource Manager overview](#) article. You can also select a different location. To learn more about Azure locations and regions, read the [Azure regions](#) article.

- The portal only enables you to create one subnet when creating a VNet. For this scenario, a second subnet must be created after the VNet is created. To create the second subnet, click **All resources**, then click **TestVNet** in the **All resources** blade, as shown in the following picture:

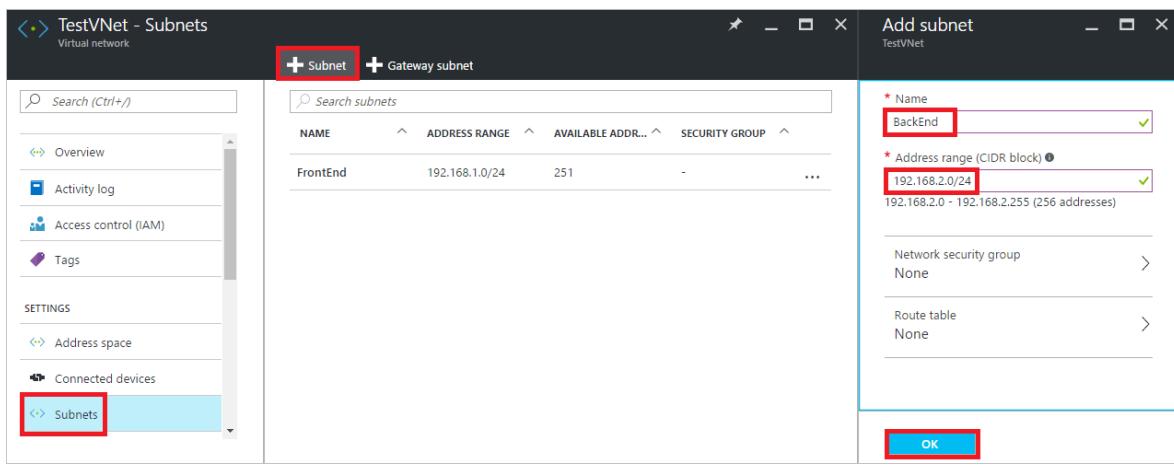
All resources

+ Add [Columns](#) Refresh

All resources

NAME	RESOURCE GROUP	LOCATION	TYPE
TestVNet	TestRG	West US	Virtual network

- In the **TestVNet** blade that appears, click **Subnet**, then click **+Subnet**, enter *BackEnd* for **Name**, 192.168.2.0/24 for **Address range** in the **Add subnet** blade, then click **OK**, as shown in the following picture:



7. The two subnets are listed, as shown in the following picture:

Subnets					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet + Gateway subnet					
+ Subnet +					

Create a virtual network using PowerShell

1/17/2017 • 3 min to read • [Edit on GitHub](#)

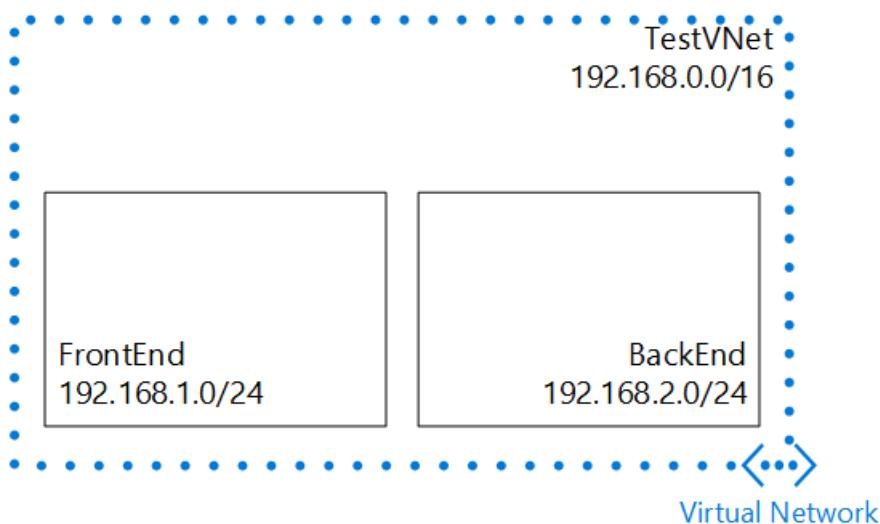
An Azure virtual network (VNet) is a representation of your own network in the cloud. You can control your Azure network settings and define DHCP address blocks, DNS settings, security policies, and routing. You can also further segment your VNet into subnets and deploy Azure IaaS virtual machines (VMs) 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.

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 explains how to create a VNet through the Resource Manager deployment model using PowerShell. You can also create a VNet through Resource Manager using other tools or create a VNet through the classic deployment model by selecting a different option from the following list:

Scenario

To better illustrate how to create a VNet and subnets, this document will use the scenario below.



In this scenario you will create a VNet named **TestVNet** with a reserved CIDR block of **192.168.0.0/16**. Your VNet will contain 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.

Create a virtual network

To create a virtual network using PowerShell, complete the following steps:

1. Install and configure Azure PowerShell, by following the steps in the [How to Install and Configure Azure PowerShell](#) article.
2. If necessary, create a new resource group, as shown below. For this scenario, create a resource group named *TestRG*. For more information about resource groups, visit [Azure Resource Manager Overview](#).

```
New-AzureRmResourceGroup -Name TestRG -Location centralus
```

Expected output:

```
ResourceGroupName : TestRG
Location         : centralus
ProvisioningState : Succeeded
Tags             :
ResourceId       : /subscriptions/[Subscription Id]/resourceGroups/TestRG
```

3. Create a new VNet named *TestVNet*:

```
New-AzureRmVirtualNetwork -ResourceGroupName TestRG -Name TestVNet ` 
-AddressPrefix 192.168.0.0/16 -Location centralus
```

Expected output:

```
Name          : TestVNet
ResourceGroupName : TestRG
Location      : centralus
Id            : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet
Etag          : W/"[Id]"
ProvisioningState : Succeeded
Tags          :
AddressSpace   : {
    "AddressPrefixes": [
        "192.168.0.0/16"
    ]
}
DhcpOptions    : {}
Subnets        : []
VirtualNetworkPeerings : []
```

4. Store the virtual network object in a variable:

```
$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName TestRG -Name TestVNet
```

TIP

You can combine steps 3 and 4 by running

```
$vnet = New-AzureRmVirtualNetwork -ResourceGroupName TestRG -Name TestVNet -AddressPrefix
192.168.0.0/16 -Location centralus
```

.

5. Add a subnet to the new VNet variable:

```
Add-AzureRmVirtualNetworkSubnetConfig -Name FrontEnd ` 
-VirtualNetwork $vnet -AddressPrefix 192.168.1.0/24
```

Expected output:

```

Name : TestVNet
ResourceGroupName : TestRG
Location : centralus
Id : /subscriptions/[Subscription]
Etag : W/"[Id]"
ProvisioningState : Succeeded
Tags :
AddressSpace : {
    "AddressPrefixes": [
        "192.168.0.0/16"
    ]
}
DhcpOptions : {}
Subnets : [
    {
        "Name": "FrontEnd",
        "AddressPrefix": "192.168.1.0/24"
    }
]
VirtualNetworkPeerings : []

```

6. Repeat step 5 above for each subnet you want to create. The following command creates the *BackEnd* subnet for the scenario:

```

Add-AzureRmVirtualNetworkSubnetConfig -Name BackEnd ` 
-VirtualNetwork $vnet -AddressPrefix 192.168.2.0/24

```

7. Although you create subnets, they currently only exist in the local variable used to retrieve the VNet you create in step 4 above. To save the changes to Azure, run the following command:

```

Set-AzureRmVirtualNetwork -VirtualNetwork $vnet

```

Expected output:

```

Name : TestVNet
ResourceGroupName : TestRG
Location : centralus
Id : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet
Etag : W/"[Id]"
ProvisioningState : Succeeded
Tags :
AddressSpace : {
    "AddressPrefixes": [
        "192.168.0.0/16"
    ]
}
DhcpOptions : {
    "DnsServers": []
}
Subnets : [
{
    "Name": "FrontEnd",
    "Etag": "W/"[Id]\\"",
    "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd",
    "AddressPrefix": "192.168.1.0/24",
    "IpConfigurations": [],
    "ProvisioningState": "Succeeded"
},
{
    "Name": "BackEnd",
    "Etag": "W/"[Id]\\"",
    "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/BackEnd",
    "AddressPrefix": "192.168.2.0/24",
    "IpConfigurations": [],
    "ProvisioningState": "Succeeded"
}
]
VirtualNetworkPeerings : []

```

Next steps

Learn how to connect:

- A virtual machine (VM) to a virtual network by reading the [Create a Windows VM](#) article. Instead of creating a VNet and subnet in the steps of the articles, you can select an existing VNet and subnet to connect a VM to.
- The virtual network to other virtual networks by reading the [Connect VNets](#) article.
- The virtual network to an on-premises network using a site-to-site virtual private network (VPN) or ExpressRoute circuit. Learn how by reading the [Connect a VNet to an on-premises network using a site-to-site VPN](#) and [Link a VNet to an ExpressRoute circuit](#) articles.

Create a virtual network using the Azure CLI

1/17/2017 • 3 min to read • [Edit on GitHub](#)

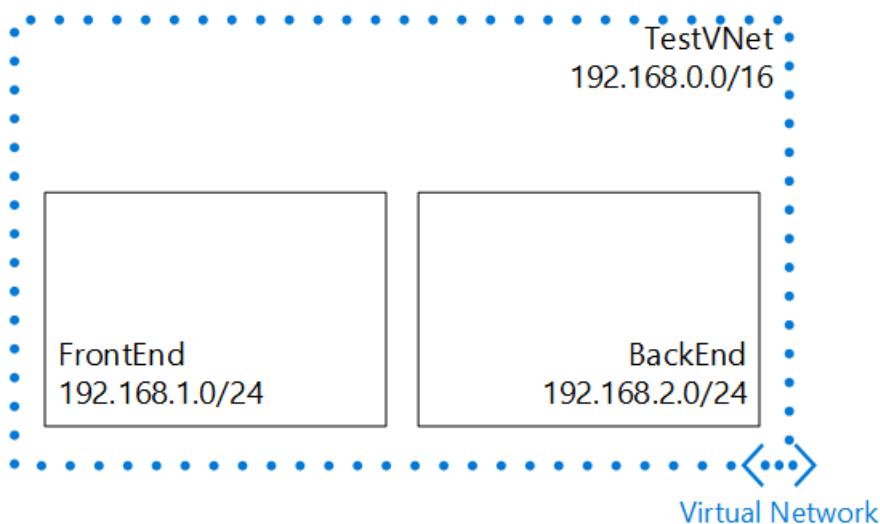
An Azure virtual network (VNet) is a representation of your own network in the cloud. You can control your Azure network settings and define DHCP address blocks, DNS settings, security policies, and routing. You can also further segment your VNet into subnets and deploy Azure IaaS virtual machines (VMs) 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.

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 explains how to create a VNet through the Resource Manager deployment model using the Azure command-line interface (CLI). You can also create a VNet through Resource Manager using other tools or create a VNet through the classic deployment model by selecting a different option from the following list:

Scenario

To better illustrate how to create a VNet and subnets, this document will use the scenario below.



In this scenario you will create a VNet named **TestVNet** with a reserved CIDR block of **192.168.0.0/16**. Your VNet will contain 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.

Create a virtual network

To create a virtual network using the Azure CLI, complete the following steps:

1. Install and configure the Azure CLI by following the steps in the [Install and Configure the Azure CLI](#) article.
2. Run the following command to create a VNet and a subnet:

```
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
```

Parameters used:

- **--vnet**. Name of the VNet to be created. For our scenario, *TestVNet*
- **-e (or --address-space)**. VNet address space. For our scenario, *192.168.0.0*
- **-i (or -cidr)**. Network mask in CIDR format. For our scenario, *16*.
- **-n (or --subnet-name)**. Name of the first subnet. For our scenario, *FrontEnd*.
- **-p (or --subnet-start-ip)**. Starting IP address for subnet, or subnet address space. For our scenario, *192.168.1.0*.
- **-r (or --subnet-cidr)**. Network mask in CIDR format for subnet. For our scenario, *24*.
- **-l (or --location)**. Azure region where the VNet will be created. For our scenario, *Central US*.

3. Run the following command to create a subnet:

```
azure network vnet subnet create -t TestVNet -n BackEnd -a 192.168.2.0/24
```

Expected output:

```
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
```

Parameters used:

- **-t (or --vnet-name)**. Name of the VNet where the subnet will be created. For our scenario, *TestVNet*.
- **-n (or --name)**. Name of the new subnet. For our scenario, *BackEnd*.
- **-a (or --address-prefix)**. Subnet CIDR block. For our scenario, *192.168.2.0/24*.

4. Run the following command to view the properties of the new VNet:

```
azure network vnet show
```

Expected output:

```
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
```

Next steps

Learn how to connect:

- A virtual machine (VM) to a virtual network by reading the [Create a Linux VM](#) article. Instead of creating a VNet and subnet in the steps of the articles, you can select an existing VNet and subnet to connect a VM to.
- The virtual network to other virtual networks by reading the [Connect VNets](#) article.
- The virtual network to an on-premises network using a site-to-site virtual private network (VPN) or ExpressRoute circuit. Learn how by reading the [Connect a VNet to an on-premises network using a site-to-site VPN](#) and [Link a VNet to an ExpressRoute circuit](#) articles.

Create a virtual network using a template

1/17/2017 • 6 min to read • [Edit on GitHub](#)

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.

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.

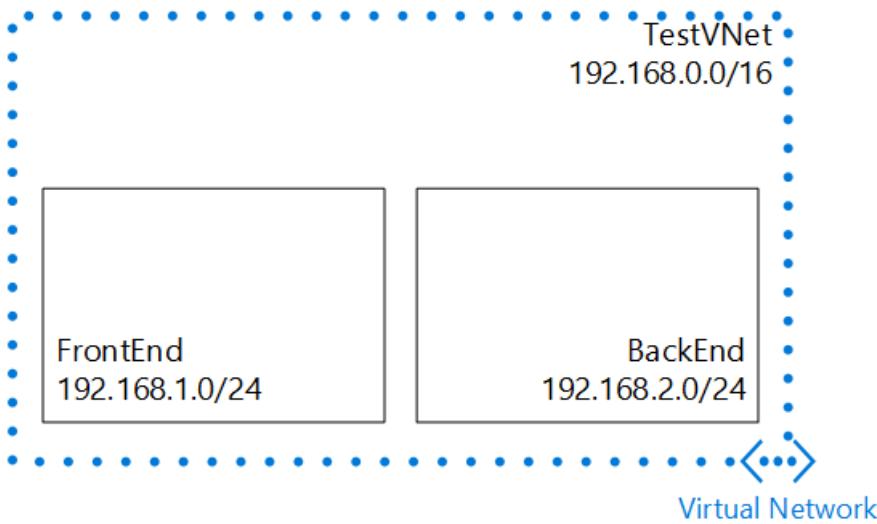
This article explains how to create a VNet through the Resource Manager deployment model using an Azure Resource Manager template. You can also create a VNet through Resource Manager using other tools or create a VNet through the classic deployment model by selecting a different option from the following list:

You will learn how to download and modify an existing ARM template from GitHub, and deploy the template from GitHub, PowerShell, and the Azure CLI.

If you are simply deploying the ARM template directly from GitHub, without any changes, skip to [deploy a template from github](#).

Scenario

To better illustrate how to create a VNet and subnets, this document will use the scenario below.



In this scenario you will create a VNet named **TestVNet** with a reserved CIDR block of **192.168.0.0/16**. Your VNet will contain 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.

Download and understand the Azure Resource Manager template

You can download the existing template for creating a VNet and two subnets from GitHub, make any changes you

might want, and reuse it. To do so, complete the following steps:

1. Navigate to [the sample template page](#).
2. Click **azuredeploy.json**, and then click **RAW**.
3. Save the file to a local folder on your computer.
4. If you are familiar with templates, skip to step 7.
5. Open the file you just saved and look at the contents under **parameters** in line 5. ARM template parameters provide a placeholder for values that can be filled out during deployment.

PARAMETER	DESCRIPTION
location	Azure region where the VNet will be created
vnetName	Name for the new VNet
addressPrefix	Address space for the VNet, in CIDR format
subnet1Name	Name for the first VNet
subnet1Prefix	CIDR block for the first subnet
subnet2Name	Name for the second VNet
subnet2Prefix	CIDR block for the second subnet

IMPORTANT

Azure Resource Manager templates maintained in GitHub can change over time. Make sure you check the template before using it.

6. Check the content under **resources** and notice the following:
 - **type**. Type of resource being created by the template. In this case, **Microsoft.Network/virtualNetworks**, which represent a VNet.
 - **name**. Name for the resource. Notice the use of **[parameters('vnetName')]**, which means the name will be provided as input by the user or a parameter file during deployment.
 - **properties**. List of properties for the resource. This template uses the address space and subnet properties during VNet creation.
7. Navigate back to [the sample template page](#).
8. Click **azuredeploy-parameters.json**, and then click **RAW**.
9. Save the file to a local folder on your computer.
10. Open the file you just saved and edit the values for the parameters. Use the following values below to deploy the VNet described in the scenario:

```
{  
    "location": {  
        "value": "Central US"  
    },  
    "vnetName": {  
        "value": "TestVNet"  
    },  
    "addressPrefix": {  
        "value": "192.168.0.0/16"  
    },  
    "subnet1Name": {  
        "value": "FrontEnd"  
    },  
    "subnet1Prefix": {  
        "value": "192.168.1.0/24"  
    },  
    "subnet2Name": {  
        "value": "BackEnd"  
    },  
    "subnet2Prefix": {  
        "value": "192.168.2.0/24"  
    }  
}
```

11. Save the file.

Deploy the template using PowerShell

Complete the following steps to deploy the template you downloaded by using PowerShell:

1. Install and configure Azure PowerShell by completing the steps in the [How to Install and Configure Azure PowerShell](#) article.
2. Run the following command to create a new resource group:

```
New-AzureRmResourceGroup -Name TestRG -Location centralus
```

The command creates a resource group named *TestRG* in the *Central US* azure region. For more information about resource groups, visit [Azure Resource Manager Overview](#).

Expected output:

```
ResourceGroupName : TestRG  
Location         : centralus  
ProvisioningState : Succeeded  
Tags             :  
Permissions       :  
                  Actions  NotActions  
                  =====  ======  
                  *  
ResourceId       : /subscriptions/[Id]/resourceGroups/TestRG
```

3. Run the following command to deploy the new VNet using the template and parameter files you downloaded and modified above:

```
New-AzureRmResourceGroupDeployment -Name TestVNetDeployment -ResourceGroupName TestRG `  
-TemplateFile C:\ARM\azuredeploy.json -TemplateParameterFile C:\ARM\azuredeploy-parameters.json
```

Expected output:

```

DeploymentName      : TestVNetDeployment
ResourceGroupName   : TestRG
ProvisioningState   : Succeeded
Timestamp          : [Date and time]
Mode                : Incremental
TemplateLink        :
Parameters          :
    Name           Type            Value
    ======         ======          =====
    location       String          Central US
    vnetName        String          TestVNet
    addressPrefix   String          192.168.0.0/16
    subnet1Prefix   String          192.168.1.0/24
    subnet1Name     String          FrontEnd
    subnet2Prefix   String          192.168.2.0/24
    subnet2Name     String          BackEnd

Outputs            :

```

4. Run the following command to view the properties of the new VNet:

```
Get-AzureRmVirtualNetwork -ResourceGroupName TestRG -Name TestVNet
```

Expected output:

```

Name : TestVNet
ResourceGroupName : TestRG
Location : centralus
Id :
/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet
Etag : W/"[Id]"
ProvisioningState : Succeeded
Tags :
AddressSpace : [
    "AddressPrefixes": [
        "192.168.0.0/16"
    ]
]
DhcpOptions : {
    "DnsServers": null
}
NetworkInterfaces : null
Subnets : [
    {
        "Name": "FrontEnd",
        "Etag": "W/"[Id]"",
        "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd",
        "AddressPrefix": "192.168.1.0/24",
        "IpConfigurations": [],
        "NetworkSecurityGroup": null,
        "RouteTable": null,
        "ProvisioningState": "Succeeded"
    },
    {
        "Name": "BackEnd",
        "Etag": "W/"[Id]"",
        "Id": "/subscriptions/[Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/BackEnd",
        "AddressPrefix": "192.168.2.0/24",
        "IpConfigurations": [],
        "NetworkSecurityGroup": null,
        "RouteTable": null,
        "ProvisioningState": "Succeeded"
    }
]

```

Deploy the template using click-to-deploy

You can reuse pre-defined Azure Resource Manager templates uploaded to a GitHub repository maintained by Microsoft and open to the community. These templates can be deployed straight out of GitHub, or downloaded and modified to fit your needs. To deploy a template that creates a VNet with two subnets, complete the following steps:

1. From a browser, navigate to <https://github.com/Azure/azure-quickstart-templates>.
2. Scroll down the list of templates, and click **101-vnet-two-subnets**. Check the **README.md** file, as shown below.

Virtual Network with two Subnets

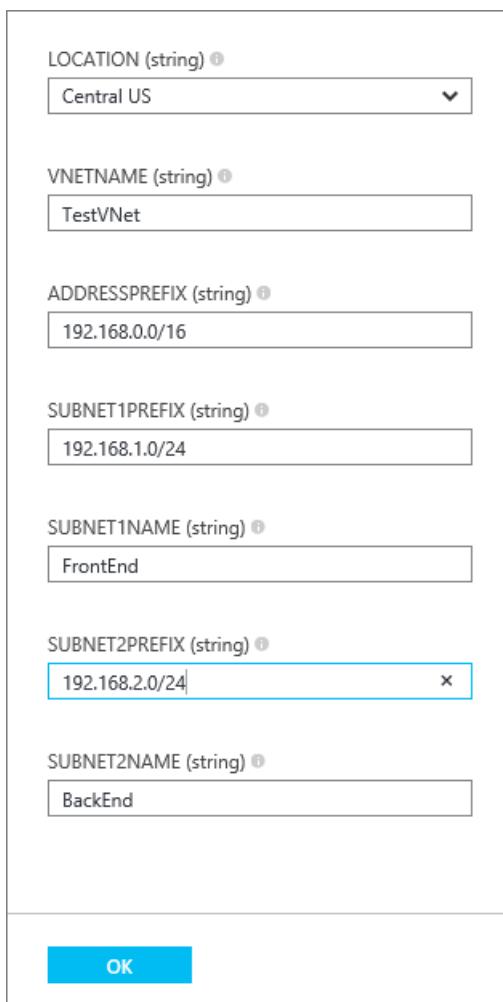
 Deploy to Azure

This template allows you to create a Virtual Network with two subnets.

Below are the parameters that the template expects

Name	Description
location	Region where the resources will be deployed
vnetName	Name for the new virtual network
addressPrefix	Address prefix for the Virtual Network specified in CIDR format
subnet1Name	Name for first subnet
subnet1Prefix	Prefix for the Subnet-1 specified in CIDR format
subnet2Name	Name for second subnet
subnet2Prefix	Prefix for the Subnet-2 specified in CIDR format

3. Click **Deploy to Azure**. If necessary, enter your Azure login credentials.
4. In the **Parameters** blade, enter the values you want to use to create your new VNet, and then click **OK**. The following figure shows the values for the scenario:

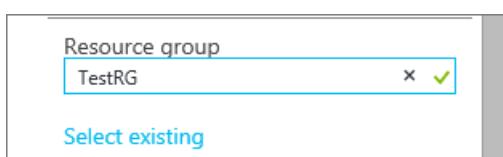


The screenshot shows the 'Parameters' blade with the following settings:

- LOCATION (string): Central US
- VNETNAME (string): TestVNet
- ADDRESSPREFIX (string): 192.168.0.0/16
- SUBNET1PREFIX (string): 192.168.1.0/24
- SUBNET1NAME (string): FrontEnd
- SUBNET2PREFIX (string): 192.168.2.0/24
- SUBNET2NAME (string): BackEnd

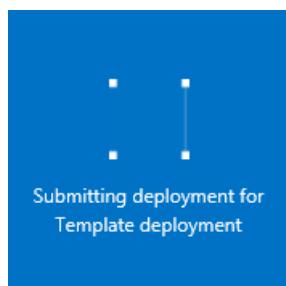
At the bottom is a blue 'OK' button.

5. Click **Resource group** and select a resource group to add the VNet to, or click **Create new** to add the VNet to a new resource group. The following figure shows the resource group settings for a new resource group called **TestRG**:



The screenshot shows the 'Resource group' selection dialog with 'TestRG' selected. There is also a 'Select existing' link.

6. If necessary, change the **Subscription** and **Location** settings for your VNet.
7. If you do not want to see the VNet as a tile in the **Startboard**, disable **Pin to Startboard**.
8. Click **Legal terms**, read the terms, and click **Buy** to agree.
9. Click **Create** to create the VNet.



10. Once the deployment is complete, in the Azure portal click **More services**, type *virtual networks* in the filter box that appears, then click **Virtual networks** to see the Virtual networks blade. In the blade, click *TestVNet*. In the *TestVNet* blade, click **Subnets** to see the created subnets, as shown in the following picture:

NAME	ADDRESS RANGE	AVAILABLE ADDR...	SECURITY GROUP
FrontEnd	192.168.1.0/24	251	-
BackEnd	192.168.2.0/24	251	-

Next steps

Learn how to connect:

- A virtual machine (VM) to a virtual network by reading the [Create a Windows VM](#) or [Create a Linux VM](#) articles. Instead of creating a VNet and subnet in the steps of the articles, you can select an existing VNet and subnet to connect a VM to.
- The virtual network to other virtual networks by reading the [Connect VNets](#) article.
- The virtual network to an on-premises network using a site-to-site virtual private network (VPN) or ExpressRoute circuit. Learn how by reading the [Connect a VNet to an on-premises network using a site-to-site VPN](#) and [Link a VNet to an ExpressRoute circuit](#) articles.

Create a virtual network (classic) by using the Azure preview portal

1/17/2017 • 2 min to read • [Edit on GitHub](#)

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.

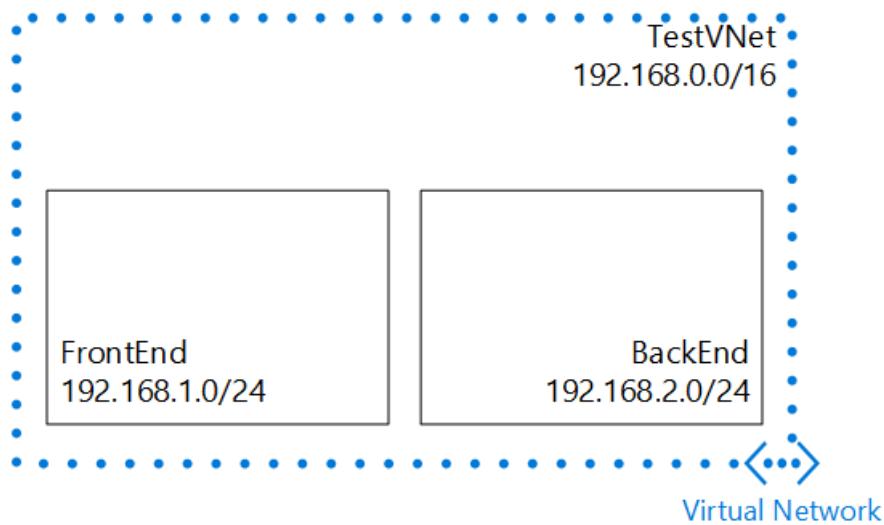
IMPORTANT

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 preview portal](#).

Scenario

To better illustrate how to create a VNet and subnets, this document will use the scenario below.



In this scenario you will create a VNet named **TestVNet** with a reserved CIDR block of **192.168.0.0/16**. Your VNet will contain the following subnets:

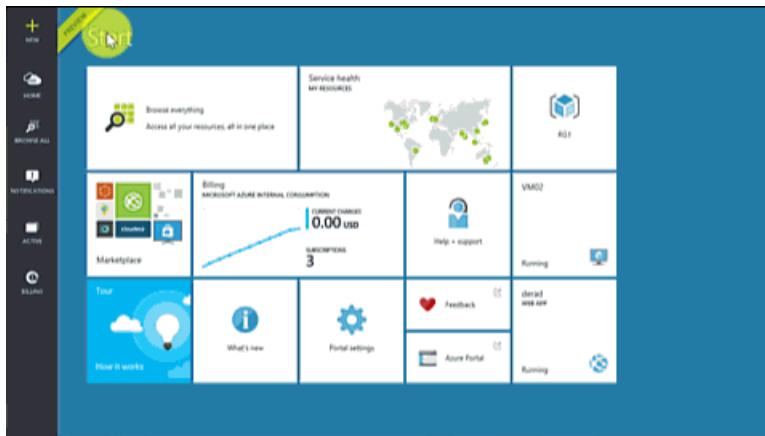
- **FrontEnd**, using **192.168.1.0/24** as its CIDR block.
- **BackEnd**, using **192.168.2.0/24** as its CIDR block.

How to create a classic VNet in the Azure portal

To create a classic VNet based on the scenario above, follow the steps below.

1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.

2. Click **NEW > Networking > Virtual network**, notice that the **Select a deployment model** list already shows **Classic**, and then click **Create**, as seen in the figure below.



3. On the **Virtual network** blade, 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 figure below shows the CIDR block settings for our scenario.

The screenshot shows two overlapping Azure portal windows. The top window is titled "Virtual network" and contains fields for "Name" (TestVNet), "Address space" (10.0.0.0/16), "Resource Group" (Group), "Subscription" (Microsoft Azure Internal Consu...), and "Location" (East US). The bottom window is titled "Address space" and contains fields for "Address space CIDR block" (192.168.0.0/16) and "Subnet name" (FrontEnd).

Virtual network

Name
TestVNet

Address space
10.0.0.0/16

Resource Group
Group

Subscription
Microsoft Azure Internal Consu...

Location
East US

Address space

Address space CIDR block
192.168.0.0/16

Subnet name
FrontEnd

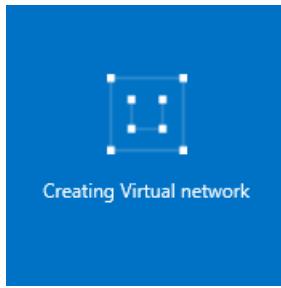
Subnet CIDR block
192.168.1.0/24

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 figure below shows the resource group settings for a new resource group called **TestRG**. For more information about resource groups, visit [Azure Resource Manager Overview](#).

The screenshot shows the Azure portal interface with three windows open:

- Virtual network**: Shows settings for a virtual network named "TestVNet" with an address space of "192.168.0.0/16". It also lists "Resource Group", "Subscription", and "Location".
- Resource group**: Describes what resource groups are and provides links to "Create a new resource group" or "Use an existing resource group". It lists existing resource groups: "appgw1-rg" (Microsoft Azure Internal Consumption) and "appgw1-rg" (Microsoft Azure Internal Consumption).
- Create resource group**: A modal window for naming a new resource group, with the name "TestRG" entered.

- If necessary, change the **Subscription** and **Location** settings for your VNet.
 - If you do not want to see the VNet as a tile in the **Startboard**, disable **Pin to Startboard**.
 - Click **Create** and notice the tile named **Creating Virtual network** as shown in the figure below.



8. Wait for the VNet to be created, and when you see the tile below, click it to add more subnets.



9. You should see the **Configuration** for your VNet as shown below.

A screenshot of the Azure portal showing the configuration of a virtual network named "TestVNet".

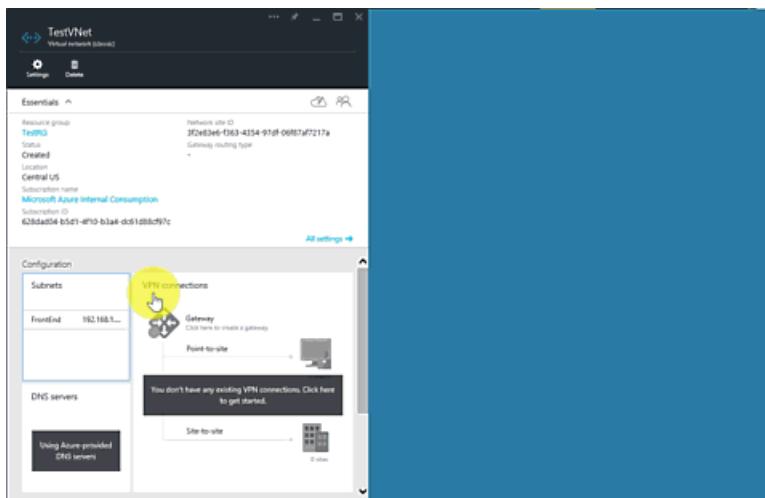
The left sidebar shows the "Configuration" section with the following details:

- Subnets:** A table with one entry: FrontEnd 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 button to "Click here to create a gateway".
- Point-to-site:** An icon showing a computer monitor connected to a central point. Below it is a button: "You don't have any existing VPN connections. Click here to get started."
- Site-to-site:** An icon showing two buildings connected by a line. Below it is a button: "0 sites".

10. Click **Subnets > Add**, then type a **Name** and specify an **Address range (CIDR block)** for your subnet, and then click **OK**. The figure below shows the settings for our current scenario.



Create a virtual network (classic) by using PowerShell

1/17/2017 • 3 min to read • [Edit on GitHub](#)

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.

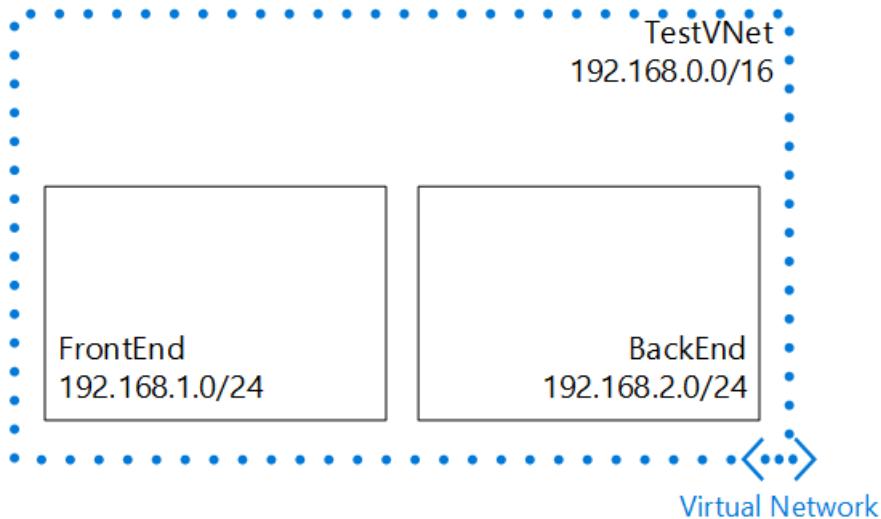
IMPORTANT

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 better illustrate how to create a VNet and subnets, this document will use the scenario below.



In this scenario you will create a VNet named **TestVNet** with a reserved CIDR block of **192.168.0.0/16**. Your VNet will contain 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 VNet using a network config file from PowerShell

Azure uses an xml file to define all VNets available to a subscription. You can download this file, and edit it to modify or delete existing VNets, and create new ones. In this document, you will learn how to download this file, referred to as network configuration (or netcfg) file, and edit it to create a new VNet. Check the [Azure virtual network configuration schema](#) to learn more about the network configuration file.

To create a VNet using a netcfg file using PowerShell, follow the steps below.

1. If you have never used Azure PowerShell, see [How to Install and Configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.
2. From the Azure PowerShell console, use the **Get-AzureVnetConfig** cmdlet to download the network configuration file by running the command below.

```
Get-AzureVNetConfig -ExportToFile c:\NetworkConfig.xml
```

Expected output:

```
XMLConfiguration
-----
<?xml version="1.0" encoding="utf-8"?>...
```

3. Open the file you saved in step 2 above using any XML or text editor application, and look for the element. If you have any networks already created, each network will be displayed as its own element.
4. To create the virtual network described in this scenario, add the following XML just under the element:

```
<VirtualNetworkSite name="TestVNet" Location="Central 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>
```

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 command below. Notice the output under the command, you should see **Succeeded** under **OperationStatus**. If that is not the case, check the xml file for errors.

```
Set-AzureVNetConfig -ConfigurationPath c:\NetworkConfig.xml
```

Here is the expected output for the command above:

OperationDescription	OperationId	OperationStatus
-----	-----	-----
Set-AzureVNetConfig	49579cb9-3f49-07c3-ada2-7abd0e28c4e4	Succeeded

7. From the Azure PowerShell console, use the **Get-AzureVnetSite** cmdlet to verify that the new network was added by running the command below.

```
Get-AzureVNetSite -VNetName TestVNet
```

Here is the expected output for the command above:

```
AddressSpacePrefixes : {192.168.0.0/16}
Location           : Central US
AffinityGroup      :
DnsServers        : {}
GatewayProfile    :
GatewaySites      :
Id                : b953f47b-fad9-4075-8cfe-73ff9c98278f
InUse              : False
Label              :
Name               : TestVNet
State              : Created
Subnets            : {FrontEnd, BackEnd}
OperationDescription: Get-AzureVNetSite
OperationId        : 3f35d533-1f38-09c0-b286-3d07cd0904d8
OperationStatus    : Succeeded
```

Create a virtual network (classic) by using the Azure CLI

1/17/2017 • 3 min to read • [Edit on GitHub](#)

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.

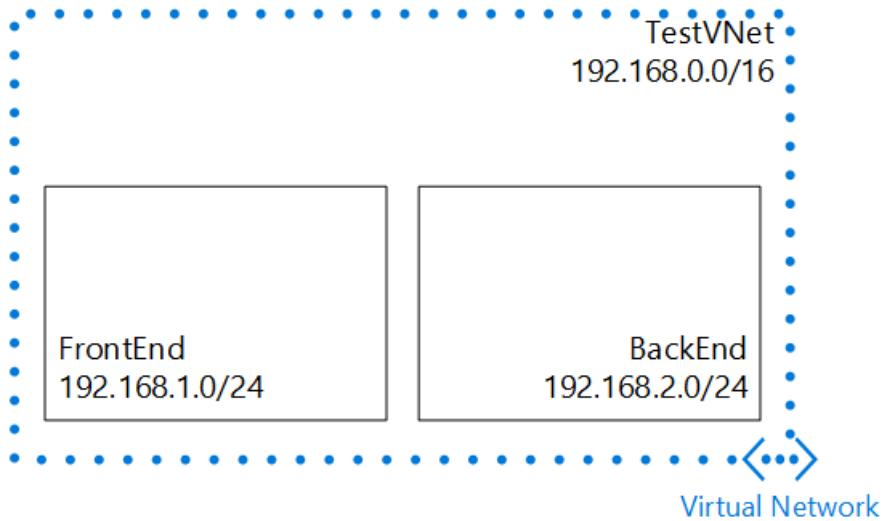
IMPORTANT

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 better illustrate how to create a VNet and subnets, this document will use the scenario below.



In this scenario you will create a VNet named **TestVNet** with a reserved CIDR block of **192.168.0.0/16**. Your VNet will contain 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. To create a VNet by using the Azure CLI, follow the steps below.

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 network vnet create** command to create a VNet and a subnet, as shown below. The list shown after the output explains the parameters used.

```
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 our scenario, *TestVNet*
 - **-e (or --address-space)**. VNet address space. For our scenario, *192.168.0.0*
 - **-i (or -cidr)**. Network mask in CIDR format. For our scenario, *16*.
 - **-n (or --subnet-name)**. Name of the first subnet. For our scenario, *FrontEnd*.
 - **-p (or --subnet-start-ip)**. Starting IP address for subnet, or subnet address space. For our scenario, *192.168.1.0*.
 - **-r (or --subnet-cidr)**. Network mask in CIDR format for subnet. For our scenario, *24*.
 - **-l (or --location)**. Azure region where the VNet will be created. For our scenario, *Central US*.
3. Run the **azure network vnet subnet create** command to create a subnet as shown below. The list shown after the output explains the parameters used.

```
azure network vnet subnet create -t TestVNet -n BackEnd -a 192.168.2.0/24
```

Here is the expected output for the command above:

```
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 our scenario, *TestVNet*.
 - **-n (or --name)**. Name of the new subnet. For our scenario, *BackEnd*.
 - **-a (or --address-prefix)**. Subnet CIDR block. For our scenario, *192.168.2.0/24*.
4. Run the **azure network vnet show** command to view the properties of the new vnet, as shown below.

```
azure network vnet show
```

Here is the expected output for the command above:

```
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
```

How to manage NSGs using the Azure portal

1/17/2017 • 3 min to read • [Edit on GitHub](#)

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](#).

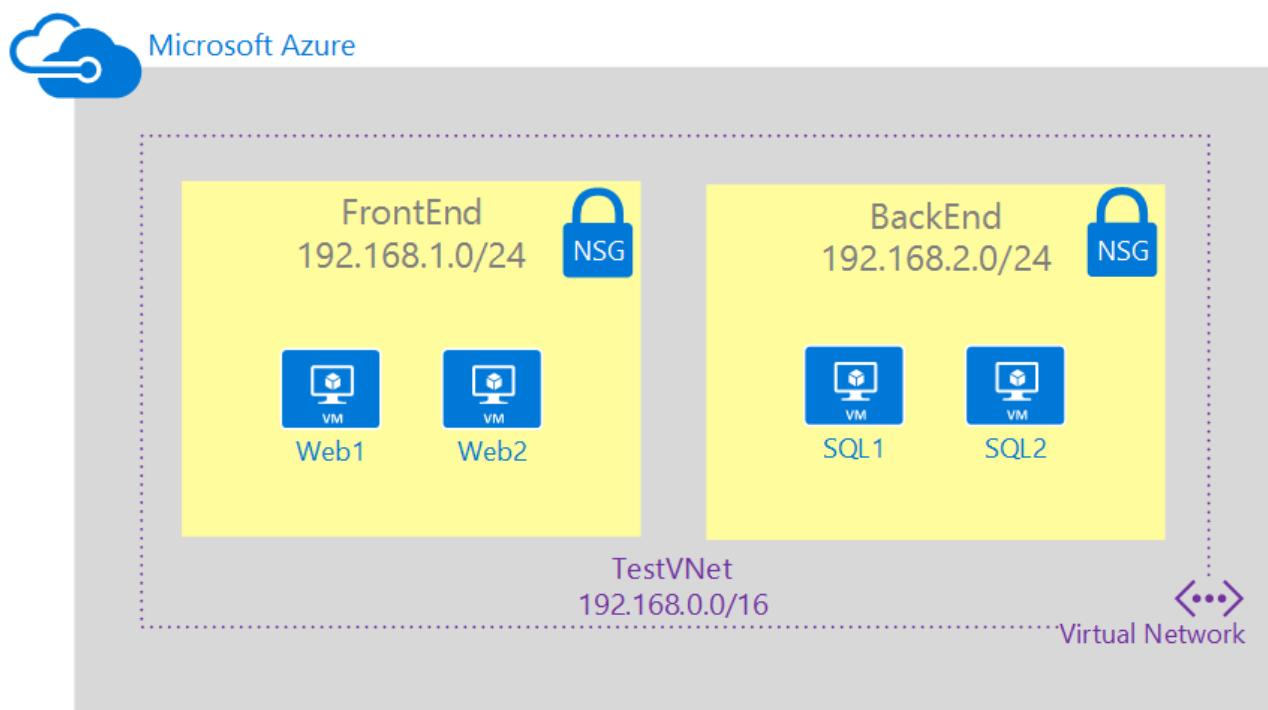
IMPORTANT

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 will use the scenario below.



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

- **NSG-FrontEnd.** The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule.** This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule.** This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd.** The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule.** This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule.** This rule denies all internet bound traffic from the *BackEnd* subnet.

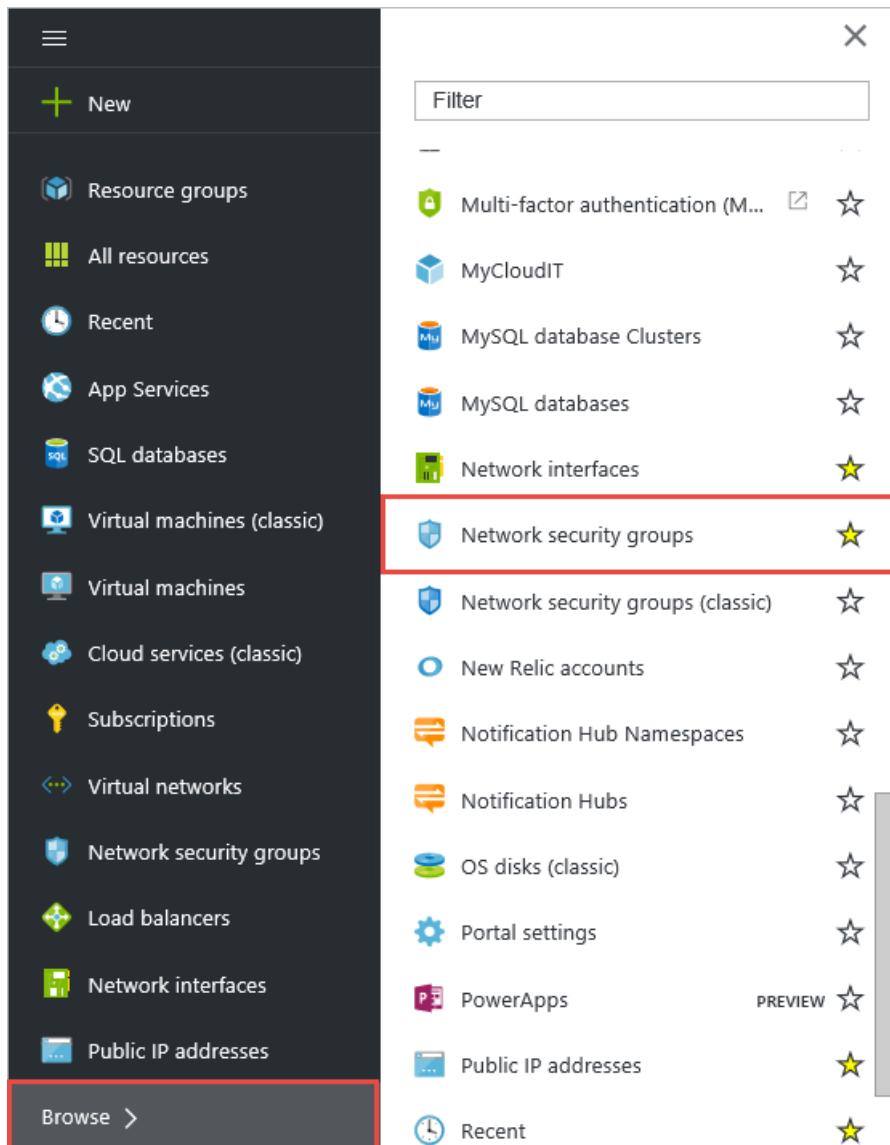
The combination of these rules create a DMZ-like scenario, where the back end subnet can only receive incoming traffic for SQL from the front end subnet, and has no access to the Internet, while the front end subnet can communicate with the Internet, and receive incoming HTTP requests only.

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 deploying [this template](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal. The steps below use **RG-NSG** as the name of the resource group the template was deployed to.

Create the NSG-FrontEnd NSG

To create the **NSG-FrontEnd** NSG as shown in the scenario above, follow the steps below.

1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Click **Browse > > Network Security Groups**.



3. In the **Network security groups** blade, click **Add**.

Network security groups

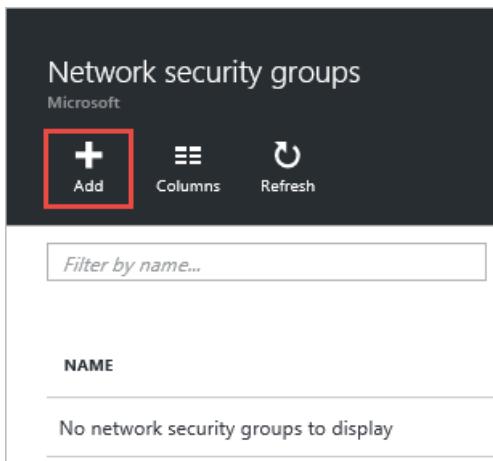
Microsoft

Add Columns Refresh

Filter by name...

NAME

No network security groups to display



4. In the **Create network security group** blade, create an NSG named *NSG-FrontEnd* in the *RG-NSG* resource group, and then click **Create**.

Create network security group

* Name
NSG-FrontEnd ✓

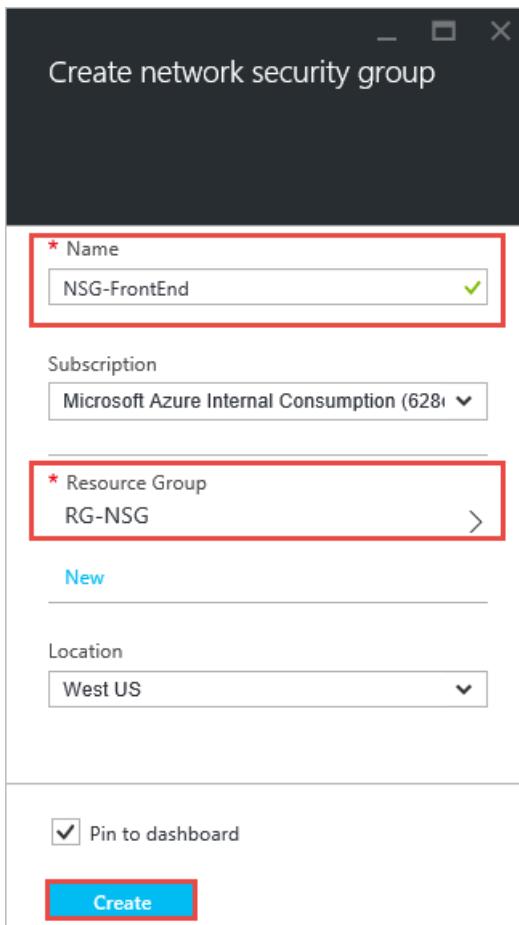
Subscription
Microsoft Azure Internal Consumption (628)

* Resource Group
RG-NSG >
New

Location
West US

Pin to dashboard

Create



Create rules in an existing NSG

To create rules in an existing NSG from the Azure portal, follow the steps below.

1. Click **Browse > > Network security groups**.
2. In the list of NSGs, click **NSG-FrontEnd > Inbound security rules**

The screenshot shows three windows from the Azure portal:

- Network security groups**: A list of network security groups. One group, "NSG-FrontEnd", is selected and highlighted with a red border.
- NSG-FrontEnd - Network security group**: The details for the selected NSG. It shows it's associated with "TestRG", "West US", and "Microsoft Azure Internal Consumption". It has 1 inbound and 1 outbound security rule, and is associated with 1 subnet and 0 network interfaces.
- Settings - NSG-FrontEnd**: The settings blade for the NSG. The "Inbound security rules" section is highlighted with a red border.

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

The screenshot shows the "Inbound security rules" blade for the "NSG-FrontEnd" settings. The "Add" button is highlighted with a red border.

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

4. In the **Add inbound security rule** blade, 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 click **OK**. Notice that most of these settings are default values already.

Add inbound security rule

NSG-FrontEnd

* Name
web-rule

* Priority
200

Source
Any CIDR block Tag

Protocol
Any TCP UDP

* Source port range
*

Destination
Any CIDR block Tag

* Destination port range
80

Action
Allow Deny

OK

5. After a few seconds you will 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

1. Click **Browse > > Resource groups > RG-NSG**.
2. In the **RG-NSG** blade, click **... > TestVNet**.

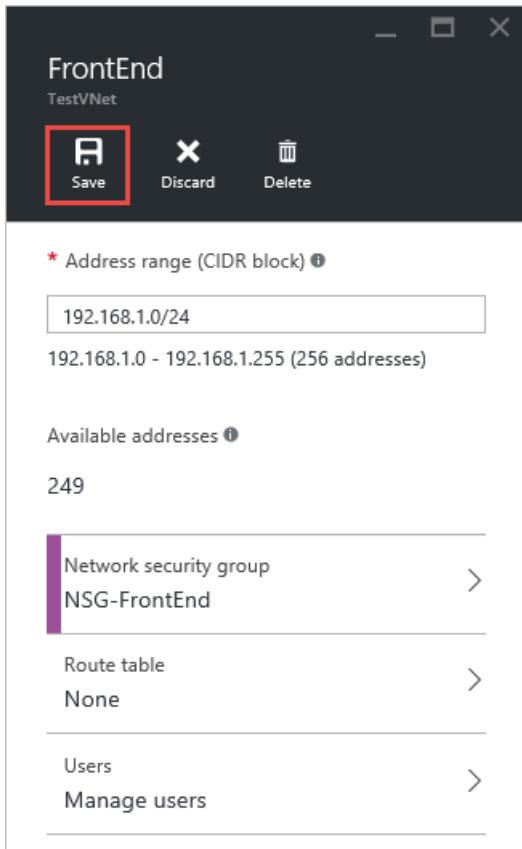
The screenshot shows the Azure portal interface. On the left, the 'RG-NSG' resource group blade displays subscription details and a summary of resources. A red box highlights the 'Resources' section, which lists various Azure services. On the right, the 'Resources' blade shows a table of resources with their names and resource groups. A red box highlights the 'TestVNet' entry under the 'RG-NSG' resource 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. In the **Settings** blade, click **Subnets** > **FrontEnd** > **Network security group** > **NSG-FrontEnd**.

The screenshot shows the Azure portal interface with three open blades. The 'Settings' blade on the left has 'Subnets' selected. The 'Subnets' blade in the center shows two subnets: 'FrontEnd' and 'BackEnd'. The 'FrontEnd' row is highlighted with a red box. The 'FrontEnd' blade on the right shows the configuration for the 'FrontEnd' subnet, including the address range '192.168.1.0/24' and the 'NSG-FrontEnd' network security group, which is also highlighted with a red box.

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



Create the NSG-BackEnd NSG

To create the **NSG-BackEnd** NSG and associate it to the **BackEnd** subnet, follow the steps below.

1. Repeat the steps in [Create the NSG-FrontEnd NSG](#) to create an NSG named *NSG-BackEnd*
2. Repeat the steps in [Create rules in an existing NSG](#) to create the **inbound** rules in the table below.

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>

3. Repeat the steps in [Associate the NSG to the FrontEnd subnet](#) to associate the **NSG-Backend** NSG to the **BackEnd** subnet.

Next Steps

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

Create NSGs using PowerShell

1/17/2017 • 4 min to read • [Edit on GitHub](#)

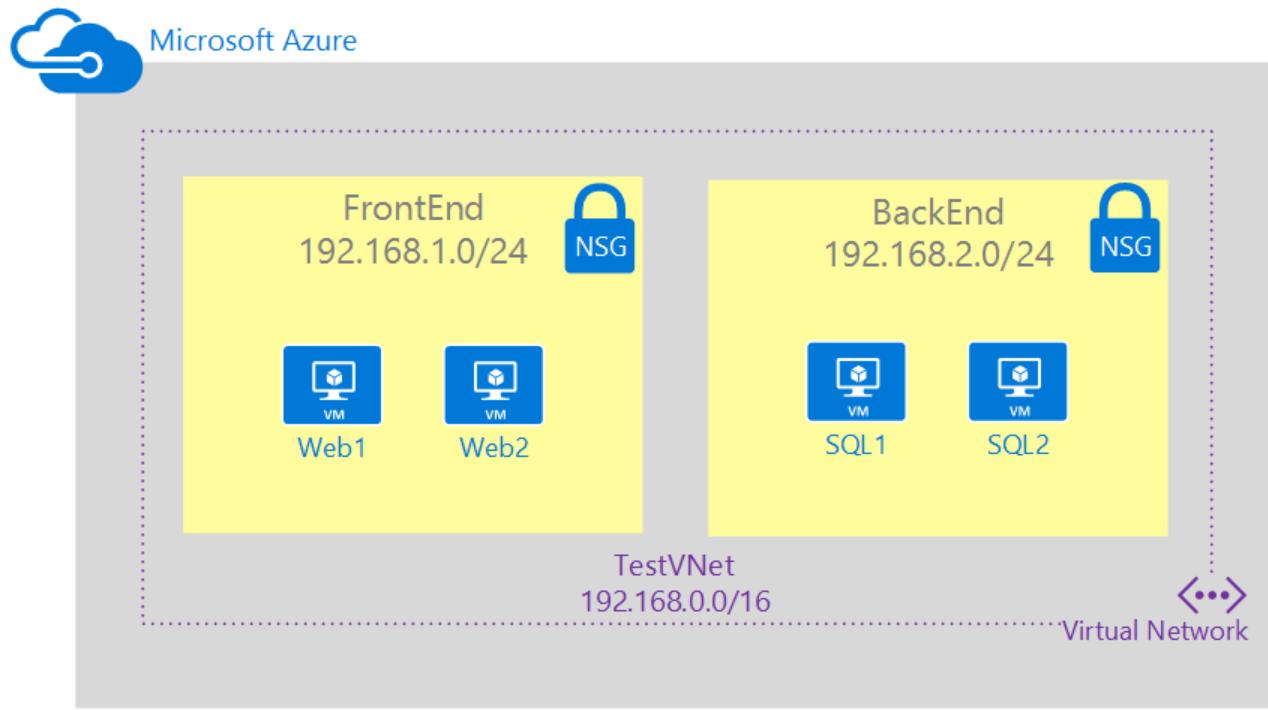
You can use an NSG to control traffic to one or more virtual machines (VMs), role instances, network adapters (NICs), or subnets in your virtual network. An NSG contains access control rules that allow or deny traffic based on traffic direction, protocol, source address and port, and destination address 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](#).

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

Scenario

To better illustrate how to create NSGs, this document will use the scenario below.



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

- **NSG-FrontEnd**. The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule**. This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule**. This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd**. The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule**. This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule**. This 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 deploying [this template](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal.

How to create the NSG for the front end subnet

To create an NSG named *NSG-FrontEnd* based on the scenario, complete the following steps:

1. If you have never used Azure PowerShell, see [How to Install and Configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.
2. Create a security rule allowing access from the Internet to port 3389.

```
$rule1 = New-AzureRmNetworkSecurityRuleConfig -Name rdp-rule -Description "Allow RDP" `  
-Access Allow -Protocol Tcp -Direction Inbound -Priority 100 `  
-SourceAddressPrefix Internet -SourcePortRange * `  
-DestinationAddressPrefix * -DestinationPortRange 3389
```

3. Create a security rule allowing access from the Internet to port 80.

```
$rule2 = New-AzureRmNetworkSecurityRuleConfig -Name web-rule -Description "Allow HTTP" `  
-Access Allow -Protocol Tcp -Direction Inbound -Priority 101 `  
-SourceAddressPrefix Internet -SourcePortRange * -DestinationAddressPrefix * `  
-DestinationPortRange 80
```

4. Add the rules created above to a new NSG named **NSG-FrontEnd**.

```
$nsg = New-AzureRmNetworkSecurityGroup -ResourceGroupName TestRG -Location westus `  
-Name "NSG-FrontEnd" -SecurityRules $rule1,$rule2
```

5. Check the rules created in the NSG by typing the following:

```
$nsg
```

Output showing just the security rules:

```

SecurityRules      : [
    {
        "Name": "rdp-rule",
        "Etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
        "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-
FrontEnd/securityRules/rdp-rule",
        "Description": "Allow RDP",
        "Protocol": "Tcp",
        "SourcePortRange": "*",
        "DestinationPortRange": "3389",
        "SourceAddressPrefix": "Internet",
        "DestinationAddressPrefix": "*",
        "Access": "Allow",
        "Priority": 100,
        "Direction": "Inbound",
        "ProvisioningState": "Succeeded"
    },
    {
        "Name": "web-rule",
        "Etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
        "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-
FrontEnd/securityRules/web-rule",
        "Description": "Allow HTTP",
        "Protocol": "Tcp",
        "SourcePortRange": "*",
        "DestinationPortRange": "80",
        "SourceAddressPrefix": "Internet",
        "DestinationAddressPrefix": "*",
        "Access": "Allow",
        "Priority": 101,
        "Direction": "Inbound",
        "ProvisioningState": "Succeeded"
    }
]

```

6. Associate the NSG created above to the *FrontEnd* subnet.

```

$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName TestRG -Name TestVNet
Set-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet -Name FrontEnd ` 
-AddressPrefix 192.168.1.0/24 -NetworkSecurityGroup $nsg

```

Output showing only the *FrontEnd* subnet settings, notice the value for the **NetworkSecurityGroup** property:

```

Subnets      : [
    {
        "Name": "FrontEnd",
        "Etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
        "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd"
    ,
        "AddressPrefix": "192.168.1.0/24",
        "IpConfigurations": [
            {
                "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNICWeb2/ipConfigurat-
ions/ipconfig1"
            ,
                "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNICWeb1/ipConfigurat-
ions/ipconfig1"
            }
        ],
        "NetworkSecurityGroup": {
            "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd"
        },
        "RouteTable": null,
        "ProvisioningState": "Succeeded"
    }
]

```

WARNING

The output for the command above shows the content for the virtual network configuration object, which only exists on the computer where you are running PowerShell. You need to run the `Set-AzureRmVirtualNetwork` cmdlet to save these settings to Azure.

- Save the new VNet settings to Azure.

```
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

Output showing just the NSG portion:

```

"NetworkSecurityGroup": {
    "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd"
}

```

How to create the NSG for the back-end subnet

To create an NSG named *NSG-BackEnd* based on the scenario above, complete the following steps:

- Create a security rule allowing access from the front-end subnet to port 1433 (default port used by SQL Server).

```
$rule1 = New-AzureRmNetworkSecurityRuleConfig -Name frontend-rule `-
-Description "Allow FE subnet" `-
-Access Allow -Protocol Tcp -Direction Inbound -Priority 100 `-
-SourceAddressPrefix 192.168.1.0/24 -SourcePortRange * `-
-DestinationAddressPrefix * -DestinationPortRange 1433
```

- Create a security rule blocking access to the Internet.

```
$rule2 = New-AzureRmNetworkSecurityRuleConfig -Name web-rule `  
-Description "Block Internet" `  
-Access Deny -Protocol * -Direction Outbound -Priority 200 `  
-SourceAddressPrefix * -SourcePortRange * `  
-DestinationAddressPrefix Internet -DestinationPortRange *
```

3. Add the rules created above to a new NSG named **NSG-BackEnd**.

```
$nsg = New-AzureRmNetworkSecurityGroup -ResourceGroupName TestRG `  
-Location westus -Name "NSG-BackEnd" `  
-SecurityRules $rule1,$rule2
```

4. Associate the NSG created above to the *BackEnd* subnet.

```
Set-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet -Name BackEnd `  
-AddressPrefix 192.168.2.0/24 -NetworkSecurityGroup $nsg
```

Output showing only the *BackEnd* subnet settings, notice the value for the **NetworkSecurityGroup** property:

```
Subnets : [  
{  
    "Name": "BackEnd",  
    "Etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",  
    "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/BackEnd",  
    "AddressPrefix": "192.168.2.0/24",  
    "IpConfigurations": [...],  
    "NetworkSecurityGroup": {  
        "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-BackEnd"  
    },  
    "RouteTable": null,  
    "ProvisioningState": "Succeeded"  
}]
```

5. Save the new VNet settings to Azure.

```
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

How to remove an NSG

To delete an existing NSG, called *NSG-Frontend* in this case, follow the step below:

Run the **Remove-AzureRmNetworkSecurityGroup** shown below and be sure to include the resource group the NSG is in.

```
Remove-AzureRmNetworkSecurityGroup -Name "NSG-FrontEnd" -ResourceGroupName "TestRG"
```

How to create NSGs in the Azure CLI

1/17/2017 • 8 min to read • [Edit on GitHub](#)

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](#).

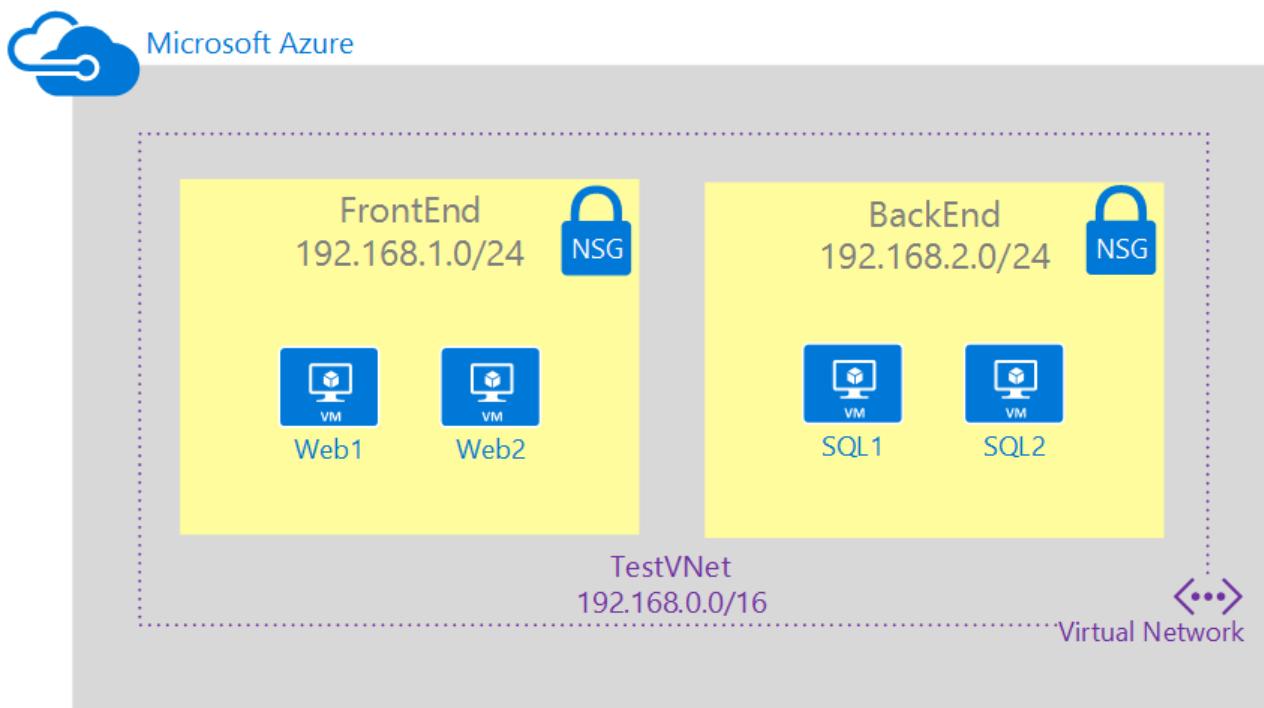
IMPORTANT

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 will use the scenario below.



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

- **NSG-FrontEnd**. The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule**. This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule**. This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd**. The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule**. This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule**. This 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 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, first build the test environment by deploying [this template](#), click **Deploy to Azure**, replace the default parameter values if necessary, and then follow the instructions in the portal.

How to create the NSG for the front end subnet

To create an NSG named named *NSG-FrontEnd* based on the scenario above, follow the steps below.

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 config mode** command to switch to Resource Manager mode, as shown below.

```
azure config mode arm
```

Expected output:

```
info: New mode is arm
```

3. Run the **azure network nsg create** command to create an NSG.

```
azure network nsg create -g TestRG -l westus -n NSG-FrontEnd
```

Expected output:

```
info: Executing command network nsg create
info: Looking up the network security group "NSG-FrontEnd"
info: Creating a network security group "NSG-FrontEnd"
info: Looking up the network security group "NSG-FrontEnd"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd
data: Name : NSG-FrontEnd
data: Type : Microsoft.Network/networkSecurityGroups
data: Location : westus
data: Provisioning state : Succeeded
data: Security group rules:
data: Name Source IP Source Port Destination IP Destination Port
Protocol Direction Access Priority
data: -----
----- -----
data: AllowVnetInBound VirtualNetwork * VirtualNetwork *
* Inbound Allow 65000
data: AllowAzureLoadBalancerInBound AzureLoadBalancer * * *
* Inbound Allow 65001
data: DenyAllInBound * * * *
* Inbound Deny 65500
data: AllowVnetOutBound VirtualNetwork * VirtualNetwork *
* Outbound Allow 65000
data: AllowInternetOutBound * * Internet *
* Outbound Allow 65001
data: DenyAllOutBound * * * *
* Outbound Deny 65500
info: network nsg create command OK
```

Parameters:

- **-g (or --resource-group)**. Name of the resource group where the NSG will be created. For our scenario, *TestRG*.
- **-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*.

4. Run the **azure network nsg rule create** command to create a rule that allows access to port 3389 (RDP) from the Internet.

```
azure network nsg rule create -g TestRG -a NSG-FrontEnd -n rdp-rule -c Allow -p Tcp -r Inbound -y 100 -f
Internet -o * -e * -u 3389
```

Expected output:

```
info: Executing command network nsg rule create
warn: Using default direction: Inbound
info: Looking up the network security rule "rdp-rule"
info: Creating a network security rule "rdp-rule"
info: Looking up the network security group "NSG-FrontEnd"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-
FrontEnd/securityRules/rdp
-rule
data: Name : rdp-rule
data: Type : Microsoft.Network/networkSecurityGroups/securityRules
data: Provisioning state : Succeeded
data: Source IP : Internet
data: Source Port : *
data: Destination IP : *
data: Destination Port : 3389
data: Protocol : Tcp
data: Direction : Inbound
data: Access : Allow
data: Priority : 100
info: network nsg rule create command OK
```

Parameters:

- **-a (or --nsg-name)**. Name of the NSG in which the rule will be created. For our scenario, *NSG-FrontEnd*.
- **-n (or --name)**. Name for the new rule. For our scenario, *rdp-rule*.
- **-c (or --access)**. Access level for the rule (Deny or Allow).
- **-p (or --protocol)**. Protocol (Tcp, Udp, or *) for the rule.
- **-r (or --direction)**. Direction of connection (Inbound or Outbound).
- **-y (or --priority)**. Priority for the rule.
- **-f (or --source-address-prefix)**. Source address prefix in CIDR or using default tags.
- **-o (or --source-port-range)**. Source port, or port range.
- **-e (or --destination-address-prefix)**. Destination address prefix in CIDR or using default tags.
- **-u (or --destination-port-range)**. Destination port, or port range.

5. Run the **azure network nsg rule create** command to create a rule that allows access to port 80 (HTTP) from the Internet.

```
azure network nsg rule create -g TestRG -a NSG-FrontEnd -n web-rule -c Allow -p Tcp -r Inbound -y 200 -f
Internet -o * -e * -u 80
```

Expected output:

```
info: Executing command network nsg rule create
info: Looking up the network security rule "web-rule"
info: Creating a network security rule "web-rule"
info: Looking up the network security group "NSG-FrontEnd"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/
networkSecurityGroups/NSG-FrontEnd/securityRules/web-rule
data: Name : web-rule
data: Type : Microsoft.Network/networkSecurityGroups/securityRules
data: Provisioning state : Succeeded
data: Source IP : Internet
data: Source Port : *
data: Destination IP : *
data: Destination Port : 80
data: Protocol : Tcp
data: Direction : Inbound
data: Access : Allow
data: Priority : 200
info: network nsg rule create command OK
```

- Run the **azure network vnet subnet set** command to link the NSG to the front end subnet.

```
azure network vnet subnet set -g TestRG -e TestVNet -n FrontEnd -o NSG-FrontEnd
```

Expected output:

```
info: Executing command network vnet subnet set
info: Looking up the subnet "FrontEnd"
info: Looking up the network security group "NSG-FrontEnd"
info: Setting subnet "FrontEnd"
info: Looking up the subnet "FrontEnd"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/
virtualNetworks/TestVNet/subnets/FrontEnd
data: Type : Microsoft.Network/virtualNetworks/subnets
data: ProvisioningState : Succeeded
data: Name : FrontEnd
data: Address prefix : 192.168.1.0/24
data: Network security group : [object Object]
data: IP configurations:
data:   /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNICWeb2/ip
Configurations/ipconfig1
data:   /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNICWeb1/ip
Configurations/ipconfig1
data:
info: network vnet subnet set command OK
```

How to create the NSG for the back end subnet

To create an NSG named named *NSG-BackEnd* based on the scenario above, follow the steps below.

- Run the **azure network nsg create** command to create an NSG.

```
azure network nsg create -g TestRG -l westus -n NSG-BackEnd
```

Expected output:

```

info: Executing command network nsg create
info: Looking up the network security group "NSG-BackEnd"
info: Creating a network security group "NSG-BackEnd"
info: Looking up the network security group "NSG-BackEnd"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/
networkSecurityGroups/NSG-BackEnd
data: Name : NSG-BackEnd
data: Type : Microsoft.Network/networkSecurityGroups
data: Location : westus
data: Provisioning state : Succeeded
data: Security group rules:
data: Name Source IP Source Port Destination IP Destination Port
Protocol Direction Access Priority
data: -----
-----
data: AllowVnetInBound VirtualNetwork * VirtualNetwork *
* Inbound Allow 65000
data: AllowAzureLoadBalancerInBound AzureLoadBalancer * * *
* Inbound Allow 65001
data: DenyAllInBound * * * *
* Inbound Deny 65500
data: AllowVnetOutBound VirtualNetwork * VirtualNetwork *
* Outbound Allow 65000
data: AllowInternetOutBound * * Internet *
* Outbound Allow 65001
data: DenyAllOutBound * * * *
* Outbound Deny 65500
info: network nsg create command OK

```

- Run the **azure network nsg rule create** command to create a rule that allows access to port 1433 (SQL) from the front end subnet.

```

azure network nsg rule create -g TestRG -a NSG-BackEnd -n sql-rule -c Allow -p Tcp -r Inbound -y 100 -f
192.168.1.0/24 -o * -e * -u 1433

```

Expected output:

```

info: Executing command network nsg rule create
info: Looking up the network security rule "sql-rule"
info: Creating a network security rule "sql-rule"
info: Looking up the network security group "NSG-BackEnd"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/
networkSecurityGroups/NSG-BackEnd/securityRules/sql-rule
data: Name : sql-rule
data: Type : Microsoft.Network/networkSecurityGroups/securityRules
data: Provisioning state : Succeeded
data: Source IP : 192.168.1.0/24
data: Source Port : *
data: Destination IP : *
data: Destination Port : 1433
data: Protocol : Tcp
data: Direction : Inbound
data: Access : Allow
data: Priority : 100
info: network nsg rule create command OK

```

- Run the **azure network nsg rule create** command to create a rule that denies access to the Internet from.

```

azure network nsg rule create -g TestRG -a NSG-BackEnd -n web-rule -c Deny -p * -r Outbound -y 200 -f *
-o * -e Internet -u *

```

Expected output:

```
info: Executing command network nsg rule create
info: Looking up the network security rule "web-rule"
info: Creating a network security rule "web-rule"
info: Looking up the network security group "NSG-BackEnd"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/
networkSecurityGroups/NSG-BackEnd/securityRules/web-rule
data: Name : web-rule
data: Type : Microsoft.Network/networkSecurityGroups/securityRules
data: Provisioning state : Succeeded
data: Source IP : *
data: Source Port : *
data: Destination IP : Internet
data: Destination Port : *
data: Protocol : *
data: Direction : Outbound
data: Access : Deny
data: Priority : 200
info: network nsg rule create command OK
```

4. Run the **azure network vnet subnet set** command to link the NSG to the back end subnet.

```
azure network vnet subnet set -g TestRG -e TestVNet -n BackEnd -o NSG-BackEnd
```

Expected output:

```
info: Executing command network vnet subnet set
info: Looking up the subnet "BackEnd"
info: Looking up the network security group "NSG-BackEnd"
info: Setting subnet "BackEnd"
info: Looking up the subnet "BackEnd"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/
virtualNetworks/TestVNet/subnets/BackEnd
data: Type : Microsoft.Network/virtualNetworks/subnets
data: ProvisioningState : Succeeded
data: Name : BackEnd
data: Address prefix : 192.168.2.0/24
data: Network security group : [object Object]
data: IP configurations:
data: /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNICSQ1/ip
Configurations/ipconfig1
data: /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNICSQ2/ip
Configurations/ipconfig1
data:
info: network vnet subnet set command OK
```

How to create NSGs using a template

1/17/2017 • 4 min to read • [Edit on GitHub](#)

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](#).

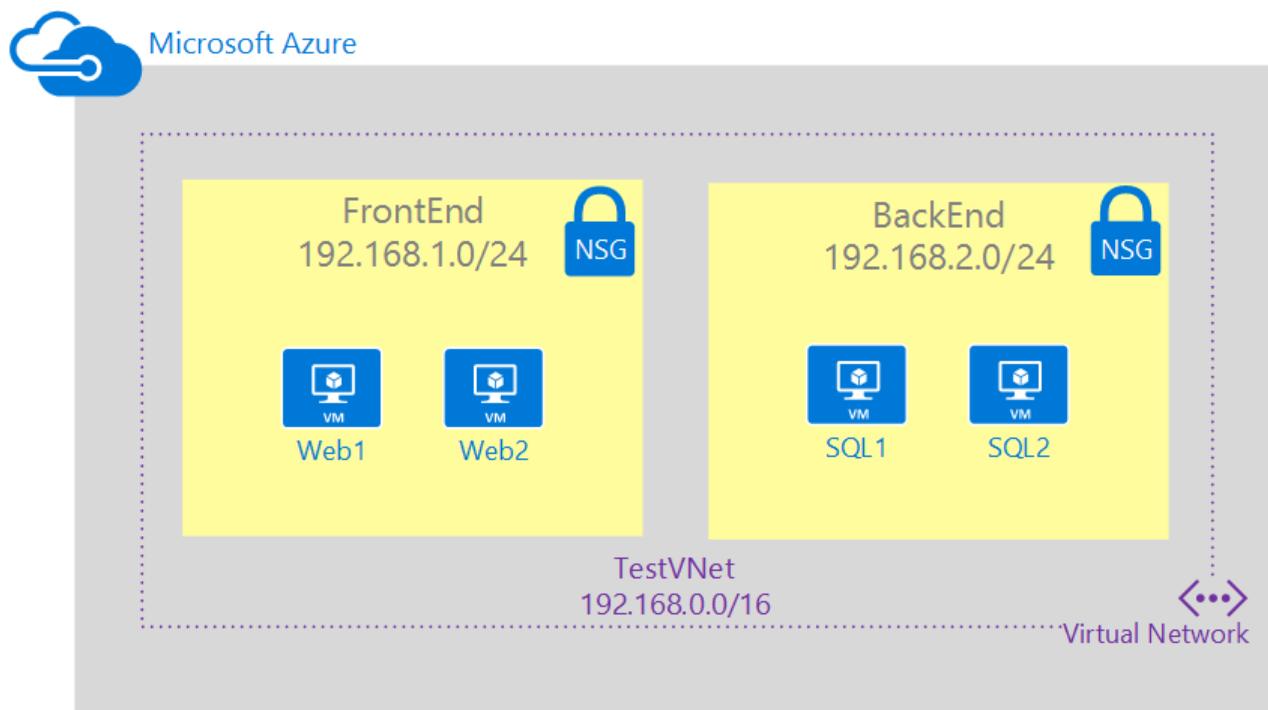
IMPORTANT

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 will use the scenario below.



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

- **NSG-FrontEnd**. The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule**. This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule**. This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd**. The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule**. This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule**. This 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.

NSG resources in a template file

You can view and download the [sample template](#).

The following section shows the definition of the front-end NSG, based on the scenario.

```
"apiVersion": "2015-06-15",
"type": "Microsoft.Network/networkSecurityGroups",
"name": "[parameters('frontEndNSGName')]",
"location": "[resourceGroup().location]",
"tags": {
    "displayName": "NSG - Front End"
},
"properties": {
    "securityRules": [
        {
            "name": "rdp-rule",
            "properties": {
                "description": "Allow RDP",
                "protocol": "Tcp",
                "sourcePortRange": "*",
                "destinationPortRange": "3389",
                "sourceAddressPrefix": "Internet",
                "destinationAddressPrefix": "*",
                "access": "Allow",
                "priority": 100,
                "direction": "Inbound"
            }
        },
        {
            "name": "web-rule",
            "properties": {
                "description": "Allow WEB",
                "protocol": "Tcp",
                "sourcePortRange": "*",
                "destinationPortRange": "80",
                "sourceAddressPrefix": "Internet",
                "destinationAddressPrefix": "*",
                "access": "Allow",
                "priority": 101,
                "direction": "Inbound"
            }
        }
    ]
}
```

To associate the NSG to the front-end subnet, you have to change the subnet definition in the template, and use the reference id for the NSG.

```
"subnets": [
{
    "name": "[parameters('frontEndSubnetName')]",
    "properties": {
        "addressPrefix": "[parameters('frontEndSubnetPrefix')]",
        "networkSecurityGroup": {
            "id": "[resourceId('Microsoft.Network/networkSecurityGroups', parameters('frontEndNSGName'))]"
        }
    }
},
```

Notice the same being done for the back-end NSG and the back-end subnet in the template.

Deploy the ARM template by using click to deploy

The sample template available in the public repository uses a parameter file containing the default values used to generate the scenario described above. To deploy this template using click to deploy, follow [this link](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal.

Deploy the ARM template by using PowerShell

To deploy the ARM template you downloaded by using PowerShell, follow the steps below.

1. If you have never used Azure PowerShell, follow the instructions in the [How to Install and Configure Azure PowerShell](#) to install and configure it.
2. Run the `New-AzureRmResourceGroup` cmdlet to create a resource group using the template.

```
New-AzureRmResourceGroup -Name TestRG -Location uswest `  
-TemplateFile 'https://raw.githubusercontent.com/telmosampaio/azure-templates/master/201-IaaS-`  
WebFrontEnd-SQLBackEnd/azuredeploy.json' `  
-TemplateParameterFile 'https://raw.githubusercontent.com/telmosampaio/azure-templates/master/201-IaaS-`  
WebFrontEnd-SQLBackEnd/azuredeploy.parameters.json'
```

Expected output:

```
ResourceGroupName : TestRG  
Location         : westus  
ProvisioningState : Succeeded  
Tags             :  
Permissions       :  
                  Actions  NotActions  
                  =====  ======  
                  *  
  
Resources        :  
                  Name          Type          Location  
                  ======  ======  =====  
                  sqlAvSet     Microsoft.Compute/availabilitySets  westus  
                  webAvSet     Microsoft.Compute/availabilitySets  westus  
                  SQL1         Microsoft.Compute/virtualMachines  westus  
                  SQL2         Microsoft.Compute/virtualMachines  westus  
                  Web1         Microsoft.Compute/virtualMachines  westus  
                  Web2         Microsoft.Compute/virtualMachines  westus  
                  TestNICSQL1 Microsoft.Network/networkInterfaces  westus  
                  TestNICSQL2 Microsoft.Network/networkInterfaces  westus  
                  TestNICWeb1 Microsoft.Network/networkInterfaces  westus  
                  TestNICWeb2 Microsoft.Network/networkInterfaces  westus  
                  NSG-BackEnd  Microsoft.Network/networkSecurityGroups  westus  
                  NSG-FrontEnd Microsoft.Network/networkSecurityGroups  westus  
                  TestPIPSQL1 Microsoft.Network/publicIPAddresses  westus  
                  TestPIPSQL2 Microsoft.Network/publicIPAddresses  westus  
                  TestPIPWeb1 Microsoft.Network/publicIPAddresses  westus  
                  TestPIPWeb2 Microsoft.Network/publicIPAddresses  westus  
                  TestVNet      Microsoft.Network/virtualNetworks   westus  
                  testvnetstorageprm Microsoft.Storage/storageAccounts  westus  
                  testvnetstoragestd Microsoft.Storage/storageAccounts  westus  
  
ResourceId       : /subscriptions/[Subscription Id]/resourceGroups/TestRG
```

Deploy the ARM template by using the Azure CLI

To deploy the ARM template by using the Azure CLI, follow the steps below.

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 config mode` command to switch to Resource Manager mode, as shown below.

```
azure config mode arm
```

The following is the expected output for the command:

```
info: New mode is arm
```

3. Run the `azure group deployment create` cmdlet to deploy the new VNet by using the template and parameter files you downloaded and modified above. The list shown after the output explains the parameters used.

```
azure group create -n TestRG -l westus -f 'https://raw.githubusercontent.com/telmosampaio/azure-templates/master/201-IaaS-WebFrontEnd-SQLBackEnd/azuredetect.json' -e 'https://raw.githubusercontent.com/telmosampaio/azure-templates/master/201-IaaS-WebFrontEnd-SQLBackEnd/azuredetect.parameters.json'
```

Expected output:

```
info: Executing command group create
info: Getting resource group TestRG
info: Creating resource group TestRG
info: Created resource group TestRG
info: Initializing template configurations and parameters
info: Creating a deployment
info: Created template deployment "azuredetect"
data: Id: /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG
data: Name: TestRG
data: Location: westus
data: Provisioning State: Succeeded
data: Tags: null
data:
info: group create command OK
```

- **-n (or --name)**. Name of the resource group to be created.
- **-l (or --location)**. Azure region where the resource group will be created.
- **-f (or --template-file)**. Path to your ARM template file.
- **-e (or --parameters-file)**. Path to your ARM parameters file.

How to create NSGs (classic) in PowerShell

1/17/2017 • 6 min to read • [Edit on GitHub](#)

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](#).

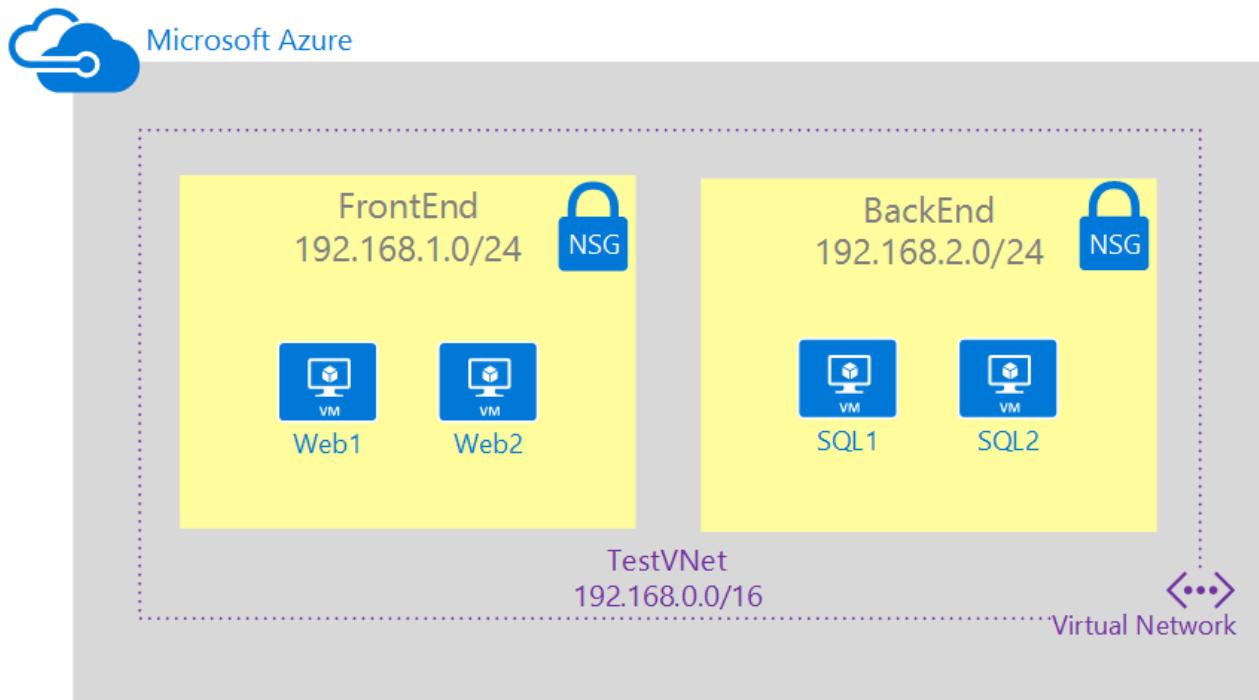
IMPORTANT

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 will use the scenario below.



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

- **NSG-FrontEnd**. The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule**. This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule**. This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd**. The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule**. This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule**. This 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](#).

How to create the NSG for the front end subnet

To create an NSG named **NSG-FrontEnd** based on the scenario above, follow the steps below:

1. If you have never used Azure PowerShell, see [How to Install and Configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.
2. Create a network security group named **NSG-FrontEnd**.

```
New-AzureNetworkSecurityGroup -Name "NSG-FrontEnd" -Location uswest `  
-Label "Front end subnet NSG"
```

Expected output:

Name	Location	Label
---	-----	-----
NSG-FrontEnd	West US	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'
```

Expected output:

Name : NSG-FrontEnd
 Location : Central US
 Label : Front end subnet NSG
 Rules :

Type: Inbound

Destination	Protocol	Name	Priority	Action	Source Address	Source Port	Destination	
					Prefix	Range	Address Prefix	Port
Range	---	---	---	---	-----	-----	-----	-----
	-----	rdp-rule	100	Allow	INTERNET	*	*	3389
TCP	ALLOW VNET INBOUND	65000	Allow	VIRTUAL_NETWORK	*		VIRTUAL_NETWORK	*
*	ALLOW AZURE LOAD	65001	Allow	AZURE_LOADBALAN	*		*	*
*	BALANCER INBOUND			CER				
*	DENY ALL INBOUND	65500	Deny	*	*	*	*	*

Type: Outbound

Destination	Protocol	Name	Priority	Action	Source Address	Source Port	Destination	
					Prefix	Range	Address Prefix	Port
Range	---	---	---	---	-----	-----	-----	-----
	-----	ALLOW VNET OUTBOUND	65000	Allow	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*
*	ALLOW INTERNET	65001	Allow	*	*		INTERNET	*
*	OUTBOUND							
*	DENY ALL OUTBOUND	65500	Deny	*	*	*	*	*

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'
  
```

Expected output:

```

Name      : NSG-FrontEnd
Location  : Central US
Label     : Front end subnet NSG
Rules     :

```

Type: Inbound

Destination	Protocol	Name	Priority	Action	Source Address	Source Port	Destination	
					Prefix	Range	Address Prefix	Port
Range	----	----	-----	-----	-----	-----	-----	-----
		rdp-rule	100	Allow	INTERNET	*	*	3389
TCP		web-rule	200	Allow	INTERNET	*	*	80
TCP		ALLOW VNET INBOUND	65000	Allow	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*
*		ALLOW AZURE LOAD	65001	Allow	AZURE_LOADBALAN	*	*	*
*		BALANCER INBOUND			CER			
*		DENY ALL INBOUND	65500	Deny	*	*	*	*

Type: Outbound

Destination	Protocol	Name	Priority	Action	Source Address	Source Port	Destination	
					Prefix	Range	Address Prefix	Port
Range	----	----	-----	-----	-----	-----	-----	-----
		ALLOW VNET OUTBOUND	65000	Allow	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*
*		ALLOW INTERNET	65001	Allow	*	*	INTERNET	*
*		OUTBOUND						
*		DENY ALL OUTBOUND	65500	Deny	*	*	*	*

How to create the NSG for the back end subnet

- Create a network security group named **NSG-BackEnd**.

```

New-AzureNetworkSecurityGroup -Name "NSG-BackEnd" -Location uswest ` 
    -Label "Back end subnet NSG"

```

Expected output:

Name	Location	Label
NSG-BackEnd	West US	Back end subnet NSG

- 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'

```

Expected output:

Name	:	NSG-BackEnd						
Location	:	Central US						
Label	:	Back end subnet NSG						
Rules	:							
Type: Inbound								
Destination	Protocol	Name	Priority	Action	Source Address	Source Port	Destination	
Range					Prefix	Range	Address Prefix	Port
-----	-----	----	-----	-----	-----	-----	-----	-----
TCP	*	fe-rule	100	Allow	192.168.1.0/24	*	*	1433
	*	ALLOW VNET INBOUND	65000	Allow	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*
	*	ALLOW AZURE LOAD	65001	Allow	AZURE_LOADBALAN	*	*	*
	*	BALANCER INBOUND			CER			
	*	DENY ALL INBOUND	65500	Deny	*	*	*	*
Type: Outbound								
Destination	Protocol	Name	Priority	Action	Source Address	Source Port	Destination	
Range					Prefix	Range	Address Prefix	Port
-----	-----	----	-----	-----	-----	-----	-----	-----
*	*	ALLOW VNET OUTBOUND	65000	Allow	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*
*	*	ALLOW INTERNET	65001	Allow	*	*	INTERNET	*
*	*	OUTBOUND						
*	*	DENY ALL OUTBOUND	65500	Deny	*	*	*	*

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 '*'

```

Expected output:

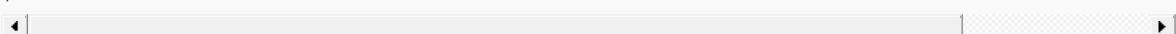
Name : NSG-BackEnd
 Location : Central US
 Label : Back end subnet NSG
 Rules :

Type: Inbound

Destination	Protocol	Name	Priority	Action	Source Address	Source Port	Destination	
					Prefix	Range	Address Prefix	Port
Range		----	-----	-----	-----	-----	-----	-----
		fe-rule	100	Allow	192.168.1.0/24	*	*	1433
TCP		ALLOW VNET INBOUND	65000	Allow	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*
*		ALLOW AZURE LOAD	65001	Allow	AZURE_LOADBALAN	*	*	*
*		BALANCER INBOUND			CER			
*		DENY ALL INBOUND	65500	Deny	*	*	*	*

Type: Outbound

Destination	Protocol	Name	Priority	Action	Source Address	Source Port	Destination	
					Prefix	Range	Address Prefix	Port
Range		----	-----	-----	-----	-----	-----	-----
		block-internet	200	Deny	*	*	INTERNET	*
*		ALLOW VNET OUTBOUND	65000	Allow	VIRTUAL_NETWORK	*	VIRTUAL_NETWORK	*
*		ALLOW INTERNET	65001	Allow	*	*	INTERNET	*
*		OUTBOUND						
*		DENY ALL OUTBOUND	65500	Deny	*	*	*	*



How to create NSGs (classic) in the Azure CLI

1/17/2017 • 8 min to read • [Edit on GitHub](#)

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](#).

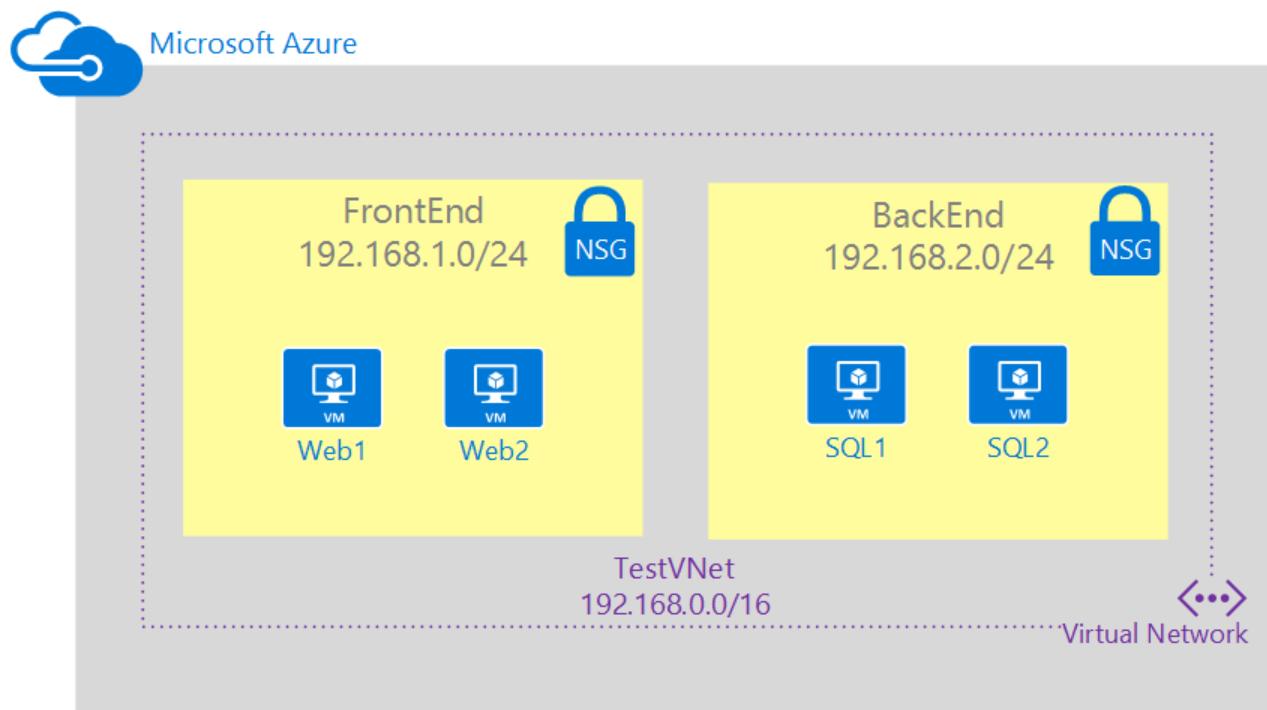
IMPORTANT

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 will use the scenario below.



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

- **NSG-FrontEnd.** The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule.** This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule.** This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd.** The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule.** This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule.** This 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 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, first build the test environment by [creating a VNet](#).

How to create the NSG for the front end subnet

To create an NSG named named **NSG-FrontEnd** based on the scenario above, follow the steps below.

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 config mode` command to switch to classic mode, as shown below.

```
azure config mode asm
```

Expected output:

```
info: New mode is asm
```

3. Run the `azure network nsg create` command to create an NSG.

```
azure network nsg create -l uswest -n NSG-FrontEnd
```

Expected output:

```
info: Executing command network nsg create
info: Creating a network security group "NSG-FrontEnd"
info: Looking up the network security group "NSG-FrontEnd"
data: Name : NSG-FrontEnd
data: Location : West US
data: Security group rules:
data:   Name           Source IP      Source Port Destination IP
Destination Port Protocol Type Action Prior
ity Default
data:   -----
----- -----
data:   ALLOW VNET OUTBOUND      VIRTUAL_NETWORK *      VIRTUAL_NETWORK *
*   Outbound Allow 65000
true
data:   ALLOW VNET INBOUND      VIRTUAL_NETWORK *      VIRTUAL_NETWORK *
*   Inbound Allow 65000
true
data:   ALLOW AZURE LOAD BALANCER INBOUND AZURE_LOADBALANCER *      *
*   Inbound Allow 65001
true
data:   ALLOW INTERNET OUTBOUND      *      *      INTERNET *
*   Outbound Allow 65001
true
data:   DENY ALL OUTBOUND      *      *      *
*   Outbound Deny 65500
true
data:   DENY ALL INBOUND      *      *      *
*   Inbound Deny 65500
true
info: network nsg 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*.

4. Run the `azure network nsg rule create` command to create a 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
```

Expected output:

```
info: Executing command network nsg rule create  
info: Looking up the network security group "NSG-FrontEnd"  
info: Creating a network security rule "rdp-rule"  
info: Looking up the network security group "NSG-FrontEnd"  
data: Name : rdp-rule  
data: Source address prefix : INTERNET  
data: Source Port : *  
data: Destination address prefix : *  
data: Destination Port : 3389  
data: Protocol : TCP  
data: Type : Inbound  
data: Action : Allow  
data: Priority : 100  
info: network nsg rule create command OK
```

Parameters:

- **-a (or --nsg-name)**. Name of the NSG in which the rule will be created. For our scenario, *NSG-FrontEnd*.
- **-n (or --name)**. Name for the new rule. For our scenario, *rdp-rule*.
- **-c (or --action)**. Access level for the rule (Deny or Allow).
- **-p (or --protocol)**. Protocol (Tcp, Udp, or *) for the rule.
- **-r (or --type)**. Direction of connection (Inbound or Outbound).
- **-y (or --priority)**. Priority for the rule.
- **-f (or --source-address-prefix)**. Source address prefix in CIDR or using default tags.
- **-o (or --source-port-range)**. Source port, or port range.
- **-e (or --destination-address-prefix)**. Destination address prefix in CIDR or using default tags.
- **-u (or --destination-port-range)**. Destination port, or port range.

5. Run the `azure network nsg rule create` command to 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
```

Expected output:

```
info: Executing command network nsg rule create
info: Looking up the network security group "NSG-FrontEnd"
info: Creating a network security rule "web-rule"
info: Looking up the network security group "NSG-FrontEnd"
data: Name : web-rule
data: Source address prefix : INTERNET
data: Source Port : *
data: Destination address prefix : *
data: Destination Port : 80
data: Protocol : TCP
data: Type : Inbound
data: Action : Allow
data: Priority : 200
info: network nsg rule create command OK
```

- Run the `azure network nsg subnet add` command to link the NSG to the front end subnet.

```
azure network nsg subnet add -a NSG-FrontEnd --vnet-name TestVNet --subnet-name FrontEnd
```

Expected output:

```
info: Executing command network nsg subnet add
info: Looking up the network security group "NSG-FrontEnd"
info: Looking up the subnet "FrontEnd"
info: Looking up network configuration
info: Creating a network security group "NSG-FrontEnd"
info: network nsg subnet add command OK
```

How to create the NSG for the back end subnet

To create an NSG named named *NSG-BackEnd* based on the scenario above, follow the steps below.

- Run the `azure network nsg create` command to create an NSG.

```
azure network nsg create -l uswest -n NSG-BackEnd
```

Expected output:

```

info: Executing command network nsg create
info: Creating a network security group "NSG-BackEnd"
info: Looking up the network security group "NSG-BackEnd"
data: Name : NSG-BackEnd
data: Location : West US
data: Security group rules:
data:          Name           Source IP      Source Port Destination IP
Destination Port Protocol Type     Action Prior
ity Default
data: -----
----- -----
data: ALLOW VNET OUTBOUND          VIRTUAL_NETWORK *          VIRTUAL_NETWORK *
*   Outbound Allow 65000
true
data: ALLOW VNET INBOUND           VIRTUAL_NETWORK *          VIRTUAL_NETWORK *
*   Inbound Allow 65000
true
data: ALLOW AZURE LOAD BALANCER INBOUND AZURE_LOADBALANCER *          *          *
*   Inbound Allow 65001
true
data: ALLOW INTERNET OUTBOUND      *          *          INTERNET *
*   Outbound Allow 65001
true
data: DENY ALL OUTBOUND            *          *          *
*   Outbound Deny 65500
true
data: DENY ALL INBOUND             *          *          *
*   Inbound Deny 65500
true
info: network nsg 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*.

2. Run the `azure network nsg rule create` command to 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

```

Expected output:

```

info: Executing command network nsg rule create
info: Looking up the network security group "NSG-BackEnd"
info: Creating a network security rule "sql-rule"
info: Looking up the network security group "NSG-BackEnd"
data: Name : sql-rule
data: Source address prefix : 192.168.1.0/24
data: Source Port : *
data: Destination address prefix : *
data: Destination Port : 1433
data: Protocol : TCP
data: Type : Inbound
data: Action : Allow
data: Priority : 100
info: network nsg rule create command OK

```

3. Run the `azure network nsg rule create` command to 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
```

Expected output:

```
info: Executing command network nsg rule create
info: Looking up the network security group "NSG-BackEnd"
info: Creating a network security rule "web-rule"
info: Looking up the network security group "NSG-BackEnd"
data: Name : web-rule
data: Source address prefix : *
data: Source Port : *
data: Destination address prefix : INTERNET
data: Destination Port : 80
data: Protocol : TCP
data: Type : Outbound
data: Action : Deny
data: Priority : 200
info: network nsg rule create command OK
```

- Run the `azure network nsg subnet add` command to link the NSG to the back end subnet.

```
azure network nsg subnet add -a NSG-BackEnd --vnet-name TestVNet --subnet-name BackEnd
```

Expected output:

```
info: Executing command network nsg subnet add
info: Looking up the network security group "NSG-BackEndX"
info: Looking up the subnet "BackEnd"
info: Looking up network configuration
info: Creating a network security group "NSG-BackEndX"
info: network nsg subnet add command OK
```

Create User-Defined Routes (UDR) using PowerShell

1/17/2017 • 5 min to read • [Edit on GitHub](#)

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.

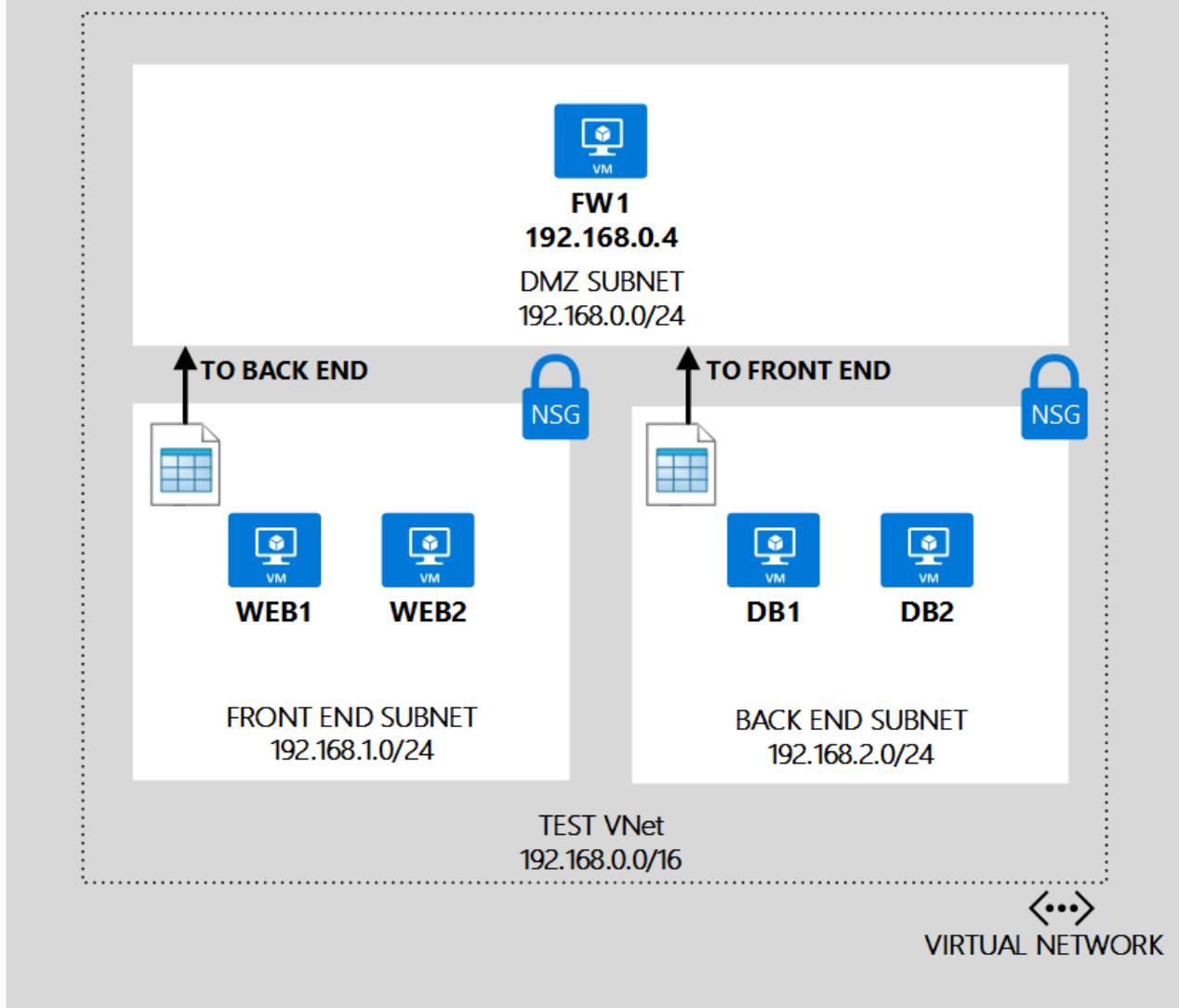
This article covers the Resource Manager deployment model. You can also [create UDRs in the classic deployment model](#).

Scenario

To better illustrate how to create UDRs, this document will use the scenario below.



AZURE REGION



In this scenario you will create one UDR for the *Front end subnet* and another UDR for the *Back end subnet*, as described below:

- **UDR-FrontEnd.** The front end UDR will be applied to the *FrontEnd* subnet, and contain one route:
 - **RouteToBackend.** This route will send all traffic to the back end subnet to the **FW1** virtual machine.
- **UDR-BackEnd.** The back end UDR will be applied to the *BackEnd* subnet, and contain one route:
 - **RouteToFrontend.** This route will send all traffic to the front end subnet to the **FW1** virtual machine.

The combination of these routes will ensure that all traffic destined from one subnet to another will be routed to the **FW1** virtual machine, which is being used as a virtual appliance. You also need to turn on IP forwarding for that VM, to ensure it can receive traffic destined to other VMs.

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 deploying [this template](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal.

Prerequisite: Install the Azure PowerShell Module

To perform the steps in this article, you'll need to [to install and configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.

NOTE

If you don't have an Azure account, you'll need one. Go sign up for a [free trial here](#).

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, complete the following steps:

1. Create a route used to send all traffic destined to the back-end subnet (192.168.2.0/24) to be routed to the **FW1** virtual appliance (192.168.0.4).

```
$route = New-AzureRmRouteConfig -Name RouteToBackEnd `  
-AddressPrefix 192.168.2.0/24 -NextHopType VirtualAppliance `  
-NextHopIpAddress 192.168.0.4
```

2. Create a route table named **UDR-FrontEnd** in the **westus** region that contains the route.

```
$routeTable = New-AzureRmRouteTable -ResourceGroupName TestRG -Location westus `  
-Name UDR-FrontEnd -Route $route
```

3. Create a variable that contains the VNet where the subnet is. In our scenario, the VNet is named **TestVNet**.

```
$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName TestRG -Name TestVNet
```

4. Associate the route table created above to the **FrontEnd** subnet.

```
Set-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet -Name FrontEnd `  
-AddressPrefix 192.168.1.0/24 -RouteTable $routeTable
```

WARNING

The output for the command above shows the content for the virtual network configuration object, which only exists on the computer where you are running PowerShell. You need to run the **Set-AzureVirtualNetwork** cmdlet to save these settings to Azure.

5. Save the new subnet configuration in Azure.

```
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

Expected output:

```

Name : TestVNet
ResourceGroupName : TestRG
Location : westus
Id : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet
Etag : W/"[Id]"
ProvisioningState : Succeeded
Tags :
      Name Value
      =====
      displayName VNet

AddressSpace : {
    "AddressPrefixes": [
        "192.168.0.0/16"
    ]
}
DhcpOptions : {
    "DnsServers": null
}
NetworkInterfaces : null
Subnets : [
    ...,
    {
        "Name": "FrontEnd",
        "Etag": "W/"[Id]\\"",
        "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICWEB2/ipConfigurations/ipconfig
1"
        "AddressPrefix": "192.168.1.0/24",
        "IpConfigurations": [
            {
                "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICWEB1/ipConfigurations/ipconfig
1"
            },
            {
                "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd"
            },
            "RouteTable": {
                "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/routeTables/UDR-FrontEnd"
            },
            "ProvisioningState": "Succeeded"
        ],
        "NetworkSecurityGroup": {
            "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-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 above, follow the steps below.

1. Create a route used to send all traffic destined to the front-end subnet (192.168.1.0/24) to be routed to the **FW1** virtual appliance (192.168.0.4).

```
$route = New-AzureRmRouteConfig -Name RouteToFrontEnd `  
-AddressPrefix 192.168.1.0/24 -NextHopType VirtualAppliance `  
-NextHopIpAddress 192.168.0.4
```

2. Create a route table named **UDR-BackEnd** in the **uswest** region that contains the route created above.

```
$routeTable = New-AzureRmRouteTable -ResourceGroupName TestRG -Location westus `  
-Name UDR-BackEnd -Route $route
```

3. Associate the route table created above to the **BackEnd** subnet.

```
Set-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet -Name BackEnd `  
-AddressPrefix 192.168.2.0/24 -RouteTable $routeTable
```

4. Save the new subnet configuration in Azure.

```
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

Expected output:

```

Name : TestVNet
ResourceGroupName : TestRG
Location : westus
Id : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet
Etag : W/"[Id]"
ProvisioningState : Succeeded
Tags :
      Name Value
      =====
      displayName VNet

AddressSpace : {
    "AddressPrefixes": [
        "192.168.0.0/16"
    ]
}
DhcpOptions : {
    "DnsServers": null
}
NetworkInterfaces : null
Subnets : [
    ...,
    {
        "Name": "BackEnd",
        "Etag": "W/"[Id]\\"",
        "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/BackEnd",
        "AddressPrefix": "192.168.2.0/24",
        "IpConfigurations": [
            {
                "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICSQL2/ipConfigurations/ipconfig
1"
            },
            {
                "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICSQL1/ipConfigurations/ipconfig
1"
            }
        ],
        "NetworkSecurityGroup": {
            "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-BacEnd"
        },
        "RouteTable": {
            "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/routeTables/UDR-BackEnd"
        },
        "ProvisioningState": "Succeeded"
    }
]

```

Enable IP forwarding on FW1

To enable IP forwarding in the NIC used by **FW1**, follow the steps below.

1. Create a variable that contains the settings for the NIC used by FW1. In our scenario, the NIC is named **NICFW1**.

```
$nicfw1 = Get-AzureRmNetworkInterface -ResourceGroupName TestRG -Name NICFW1
```

2. Enable IP forwarding, and save the NIC settings.

```
$nicfw1.EnableIPForwarding = 1
Set-AzureRmNetworkInterface -NetworkInterface $nicfw1
```

Expected output:

```
Name : NICFW1
ResourceGroupName : TestRG
Location : westus
Id : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICFW1
Etag : W/"[Id]"
ProvisioningState : Succeeded
Tags :
    Name      Value
    ======  =====
    displayName NetworkInterfaces - DMZ

VirtualMachine : {
    "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/fw1"
}
IpConfigurations : [
    {
        "Name": "ipconfig1",
        "Etag": "W/"[Id]\\"",
        "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICFW1/ipConfigurations/ipconfig1
",
        "PrivateIpAddress": "192.168.0.4",
        "PrivateIpAllocationMethod": "Static",
        "Subnet": {
            "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/DMZ"
        },
        "PublicIpAddress": {
            "Id": "/subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/PIPFW1"
        },
        "LoadBalancerBackendAddressPools": [],
        "LoadBalancerInboundNatRules": [],
        "ProvisioningState": "Succeeded"
    }
]
DnsSettings : {
    "DnsServers": [],
    "AppliedDnsServers": [],
    "InternalDnsNameLabel": null,
    "InternalFqdn": null
}
EnableIPForwarding : True
NetworkSecurityGroup : null
Primary : True
```

Create User-Defined Routes (UDR) using the Azure CLI

1/17/2017 • 6 min to read • [Edit on GitHub](#)

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.

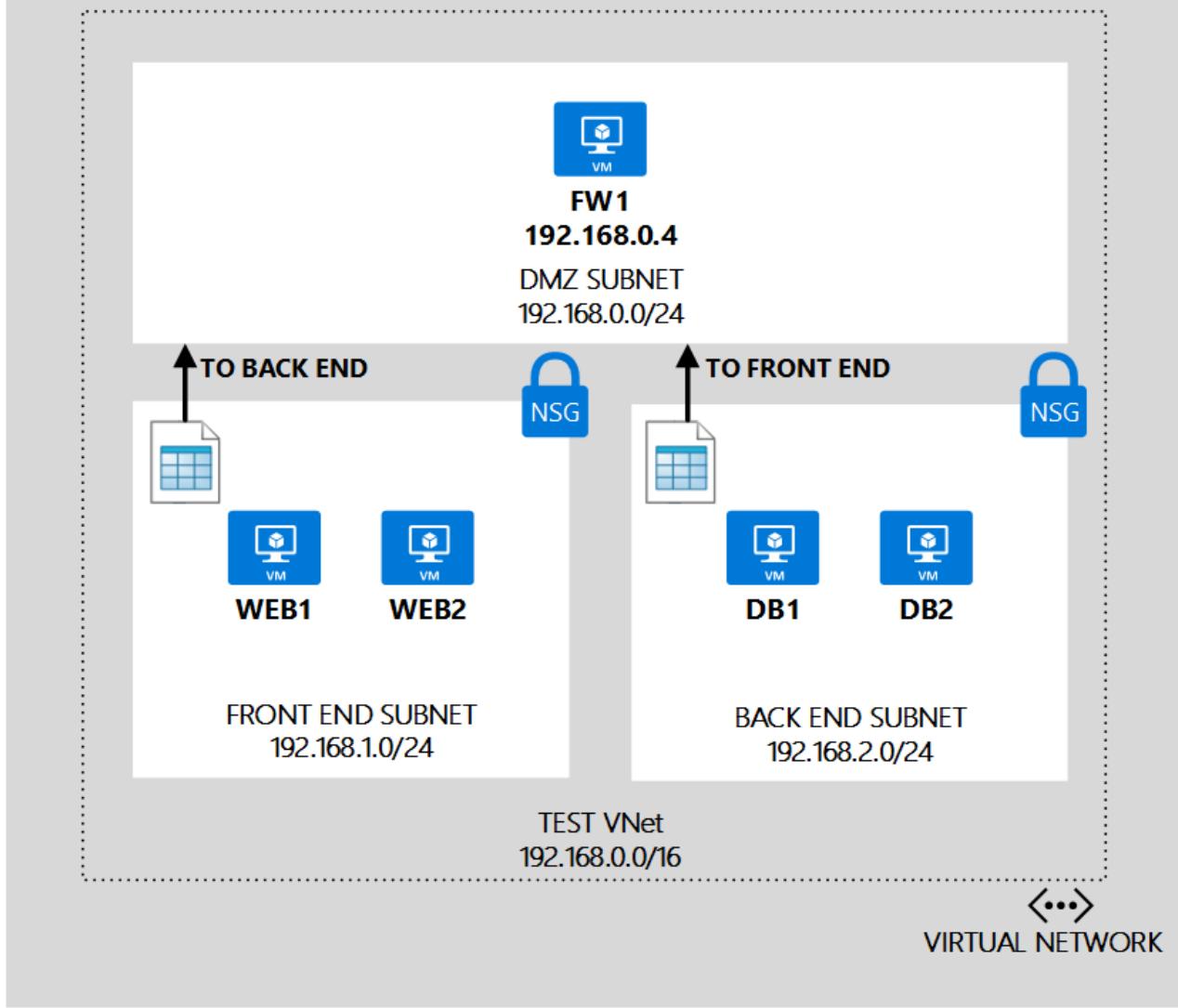
This article covers the Resource Manager deployment model. You can also create UDRs in the [classic deployment model](#).

Scenario

To better illustrate how to create UDRs, this document will use the scenario below.



AZURE REGION



In this scenario you will create one UDR for the *Front end subnet* and another UDR for the *Back end subnet*, as described below:

- **UDR-FrontEnd.** The front end UDR will be applied to the *FrontEnd* subnet, and contain one route:
 - **RouteToBackend.** This route will send all traffic to the back end subnet to the **FW1** virtual machine.
- **UDR-BackEnd.** The back end UDR will be applied to the *BackEnd* subnet, and contain one route:
 - **RouteToFrontend.** This route will send all traffic to the front end subnet to the **FW1** virtual machine.

The combination of these routes will ensure that all traffic destined from one subnet to another will be routed to the **FW1** virtual machine, which is being used as a virtual appliance. You also need to turn on IP forwarding for that 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, first build the test environment by deploying [this template](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal.

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 create a route table for the front-end subnet:

```
azure network route-table create -g TestRG -n UDR-FrontEnd -l uswest
```

Output:

```
info: Executing command network route-table create
info: Looking up route table "UDR-FrontEnd"
info: Creating route table "UDR-FrontEnd"
info: Looking up route table "UDR-FrontEnd"
data:  Id                      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
routeTables/UDR-FrontEnd
data:  Name                     : UDR-FrontEnd
data:  Type                      : Microsoft.Network/routeTables
data:  Location                  : westus
data:  Provisioning state       : Succeeded
info:  network route-table create command OK
```

Parameters:

- **-g (or --resource-group)**. Name of the resource group where the UDR will be created. For our scenario, *TestRG*.
- **-l (or --location)**. Azure region where the new UDR will be created. For our scenario, *westus*.
- **-n (or --name)**. Name for the new UDR. For our scenario, *UDR-FrontEnd*.

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):

```
azure network route-table route create -g TestRG -r UDR-FrontEnd -n RouteToBackEnd -a 192.168.2.0/24 -y
VirtualAppliance -p 192.168.0.4
```

Output:

```

info: Executing command network route-table route create
info: Looking up route "RouteToBackEnd" in route table "UDR-FrontEnd"
info: Creating route "RouteToBackEnd" in a route table "UDR-FrontEnd"
info: Looking up route "RouteToBackEnd" in route table "UDR-FrontEnd"
data: Id : /subscriptions/[Subscription
Id]/TestRG/providers/Microsoft.Network/
routeTables/UDR-FrontEnd/routes/RouteToBackEnd
data: Name : RouteToBackEnd
data: Provisioning state : Succeeded
data: Next hop type : VirtualAppliance
data: Next hop IP address : 192.168.0.4
data: Address prefix : 192.168.2.0/24
info: network route-table route create 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*.
- **-y (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*.

3. Run the following command to associate the route table created above with the **FrontEnd** subnet:

```
azure network vnet subnet set -g TestRG -e TestVNet -n FrontEnd -r UDR-FrontEnd
```

Output:

```

info: Executing command network vnet subnet set
info: Looking up the subnet "FrontEnd"
info: Looking up route table "UDR-FrontEnd"
info: Setting subnet "FrontEnd"
info: Looking up the subnet "FrontEnd"
data: Id : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
virtualNetworks/TestVNet/subnets/FrontEnd
data: Type : Microsoft.Network/virtualNetworks/subnets
data: ProvisioningState : Succeeded
data: Name : FrontEnd
data: Address prefix : 192.168.1.0/24
data: Network security group : [object Object]
data: Route Table : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
routeTables/UDR-FrontEnd
data: IP configurations:
data: /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICWEB1/ipConfigurations/ipconfig1
data: /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICWEB2/ipConfigurations/ipconfig1
data:
info: network vnet subnet set command OK

```

Parameters:

- **-e (or --vnet-name)**. Name of the VNet where the subnet is located. For our scenario, *TestVNet*.

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 above, complete the following steps:

1. Run the following command to create a route table for the back-end subnet:

```
azure network route-table create -g TestRG -n UDR-BackEnd -l westus
```

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 create -g TestRG -r UDR-BackEnd -n RouteToFrontEnd -a 192.168.1.0/24 -y  
VirtualAppliance -p 192.168.0.4
```

3. Run the following command to associate the route table with the **BackEnd** subnet:

```
azure network vnet subnet set -g TestRG -e TestVNet -n BackEnd -r UDR-BackEnd
```

Enable IP forwarding on FW1

To enable IP forwarding in the NIC used by **FW1**, complete the following steps:

1. Run the command that follows and notice the value for **Enable IP forwarding**. It should be set to *false*.

```
azure network nic show -g TestRG -n NICFW1
```

Output:

```
info: Executing command network nic show
info: Looking up the network interface "NICFW1"
data: Id : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
networkInterfaces/NICFW1
data: Name : NICFW1
data: Type : Microsoft.Network/networkInterfaces
data: Location : westus
data: Provisioning state : Succeeded
data: MAC address : 00-0D-3A-30-95-B3
data: Enable IP forwarding : false
data: Tags : displayName=NetworkInterfaces - DMZ
data: Virtual machine : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/
virtualMachines/FW1
data: IP configurations:
data: Name : ipconfig1
data: Provisioning state : Succeeded
data: Public IP address : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
publicIPAddresses/PIPFW1
data: Private IP address : 192.168.0.4
data: Private IP Allocation Method : Static
data: Subnet : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
virtualNetworks/TestVNet/subnets/DMZ
data:
info: network nic show command OK
```

2. Run the following command to enable IP forwarding:

```
azure network nic set -g TestRG -n NICFW1 -f true
```

Output:

```
info: Executing command network nic set
info: Looking up the network interface "NICFW1"
info: Updating network interface "NICFW1"
info: Looking up the network interface "NICFW1"
data: Id : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
networkInterfaces/NICFW1
data: Name : NICFW1
data: Type : Microsoft.Network/networkInterfaces
data: Location : westus
data: Provisioning state : Succeeded
data: MAC address : 00-0D-3A-30-95-B3
data: Enable IP forwarding : true
data: Tags : displayName=NetworkInterfaces - DMZ
data: Virtual machine : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/
virtualMachines/FW1
data: IP configurations:
data: Name : ipconfig1
data: Provisioning state : Succeeded
data: Public IP address : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
publicIPAddresses/PIPFW1
data: Private IP address : 192.168.0.4
data: Private IP Allocation Method : Static
data: Subnet : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/
virtualNetworks/TestVNet/subnets/DMZ
data:
info: network nic set command OK
```

Parameters:

- **-f (or --enable-ip-forwarding)**. *true or false.*

Create User-Defined Routes (UDR) using a template

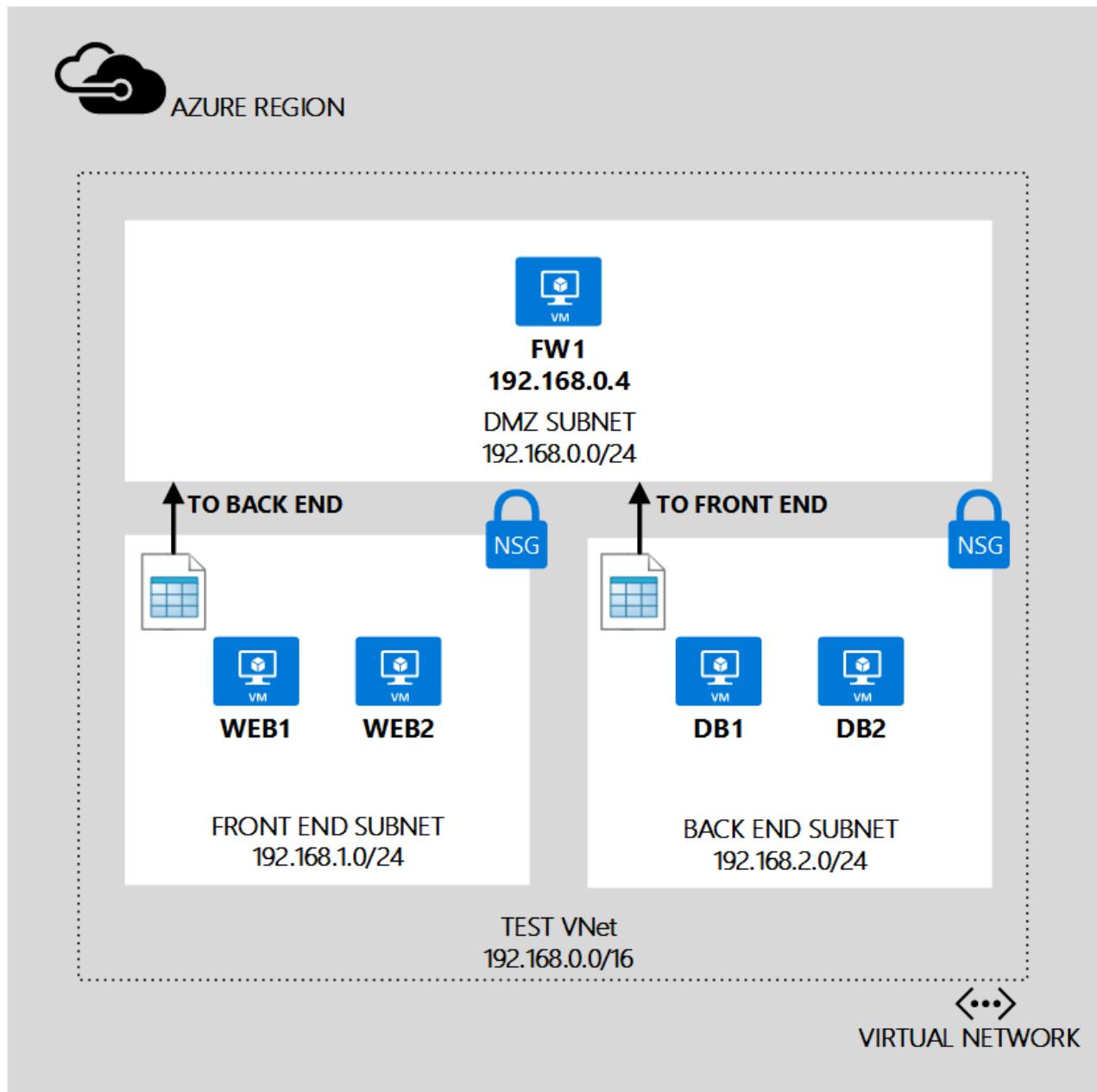
1/17/2017 • 7 min to read • [Edit on GitHub](#)

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.

Scenario

To better illustrate how to create UDRs, this document will use the scenario below.



In this scenario you will create one UDR for the *Front end subnet* and another UDR for the *Back end subnet*, as described below:

- **UDR-FrontEnd**. The front end UDR will be applied to the *FrontEnd* subnet, and contain one route:
 - **RouteToBackend**. This route will send all traffic to the back end subnet to the **FW1** virtual machine.
- **UDR-BackEnd**. The back end UDR will be applied to the *BackEnd* subnet, and contain one route:
 - **RouteToFrontend**. This route will send all traffic to the front end subnet to the **FW1** virtual machine.

The combination of these routes will ensure that all traffic destined from one subnet to another will be routed to the **FW1** virtual machine, which is being used as a virtual appliance. You also need to turn on IP forwarding for that VM, to ensure it can receive traffic destined to other VMs.

UDR resources in a template file

You can view and download the [sample template](#).

The following section shows the definition of the front-end UDR in the **azureddeploy-vnet-nsg-udr.json** file for the scenario:

```
"apiVersion": "2015-06-15",
"type": "Microsoft.Network/routeTables",
"name": "[parameters('frontEndRouteTableName')]",
"location": "[resourceGroup().location]",
"tags": {
  "displayName": "UDR - FrontEnd"
},
"properties": {
  "routes": [
    {
      "name": "RouteToBackEnd",
      "properties": {
        "addressPrefix": "[parameters('backEndSubnetPrefix')]",
        "nextHopType": "VirtualAppliance",
        "nextHopIpAddress": "[parameters('vmaIpAddress')]"
      }
    }
  ]
}
```

To associate the UDR to the front-end subnet, you have to change the subnet definition in the template, and use the reference id for the UDR.

```
"subnets": [
  "name": "[parameters('frontEndSubnetName')]",
  "properties": {
    "addressPrefix": "[parameters('frontEndSubnetPrefix')]",
    "networkSecurityGroup": {
      "id": "[resourceId('Microsoft.Network/networkSecurityGroups', parameters('frontEndNSGName'))]"
    },
    "routeTable": {
      "id": "[resourceId('Microsoft.Network/routeTables', parameters('frontEndRouteTableName'))]"
    }
  }
],
```

Notice the same being done for the back-end NSG and the back-end subnet in the template.

You also need to ensure that the **FW1** VM has the IP forwarding property enabled on the NIC that will be used to receive and forward packets. The section below shows the definition of the NIC for FW1 in the **azureddeploy-nsg-udr.json** file, based on the scenario above.

```

"apiVersion": "2015-06-15",
"type": "Microsoft.Network/networkInterfaces",
"location": "[variables('location')]",
"tags": {
  "displayName": "NetworkInterfaces - DMZ"
},
"name": "[concat(variables('fwVMSettings').nicName, copyindex(1))]",
"dependsOn": [
  "[concat('Microsoft.Network/publicIPAddresses/', variables('fwVMSettings').pipName, copyindex(1))]",
  "[concat('Microsoft.Resources/deployments/', 'vnetTemplate')]"
],
"properties": {
  "ipConfigurations": [
    {
      "name": "ipconfig1",
      "properties": {
        "enableIPForwarding": true,
        "privateIPAllocationMethod": "Static",
        "privateIPAddress": "[concat('192.168.0.', copyindex(4))]",
        "publicIPAddress": {
          "id": "[resourceId('Microsoft.Network/publicIPAddresses', concat(variables('fwVMSettings').pipName, copyindex(1)))]"
        },
        "subnet": {
          "id": "[variables('dmzSubnetRef')]"
        }
      }
    }
  ],
  "copy": {
    "name": "fwniccount",
    "count": "[parameters('fwCount')]"
  }
}

```

Deploy the template by using click to deploy

The sample template available in the public repository uses a parameter file containing the default values used to generate the scenario described above. To deploy this template using click to deploy, follow [this link](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal.

1. If you have never used Azure PowerShell, see [How to Install and Configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.
2. Run the following command to create a resource group:

```
New-AzureRmResourceGroup -Name TestRG -Location westus
```

3. Run the following command to deploy the template:

```

New-AzureRmResourceGroupDeployment -Name DeployUDR -ResourceGroupName TestRG ` 
  -TemplateUri https://raw.githubusercontent.com/telmosampaio/azure-templates/master/IaaS-NSG-` 
  UDR/azuredeploy.json ` 
  -TemplateParameterUri https://raw.githubusercontent.com/telmosampaio/azure-templates/master/IaaS-` 
  NSG-UDR/azuredeploy.parameters.json

```

Expected output:

```

ResourceGroupName : TestRG
Location         : westus
ProvisioningState : Succeeded
Tags             :
Permissions      :
    Actions  NotActions
    =====  ========
    *

Resources       :
    Name          Type           Location
    =====        ======        =====
    ASFW          Microsoft.Compute/availabilitySets   westus
    ASSQL         Microsoft.Compute/availabilitySets   westus
    ASWEB         Microsoft.Compute/availabilitySets   westus
    FW1           Microsoft.Compute/virtualMachines   westus
    SQL1          Microsoft.Compute/virtualMachines   westus
    SQL2          Microsoft.Compute/virtualMachines   westus
    WEB1          Microsoft.Compute/virtualMachines   westus
    WEB2          Microsoft.Compute/virtualMachines   westus
    NICFW1        Microsoft.Network/networkInterfaces westus
    NICSQ1L       Microsoft.Network/networkInterfaces westus
    NICSQ2L       Microsoft.Network/networkInterfaces westus
    NICWEB1       Microsoft.Network/networkInterfaces westus
    NICWEB2       Microsoft.Network/networkInterfaces westus
    NSG-BackEnd   Microsoft.Network/networkSecurityGroups westus
    NSG-FrontEnd  Microsoft.Network/networkSecurityGroups westus
    PIPFW1        Microsoft.Network/publicIPAddresses westus
    PIPSQL1       Microsoft.Network/publicIPAddresses westus
    PIPSQL2       Microsoft.Network/publicIPAddresses westus
    PIPWEB1       Microsoft.Network/publicIPAddresses westus
    PIPWEB2       Microsoft.Network/publicIPAddresses westus
    UDR-BackEnd   Microsoft.Network/routeTables     westus
    UDR-FrontEnd  Microsoft.Network/routeTables     westus
    TestVNet      Microsoft.Network/virtualNetworks westus
    testvnetstorageprm Microsoft.Storage/storageAccounts westus
    testvnetstoragestd Microsoft.Storage/storageAccounts westus

ResourceId      : /subscriptions/[Subscription Id]/resourceGroups/TestRG

```

Deploy the template by using the Azure CLI

To deploy the ARM template by using the Azure CLI, complete the following 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 following command to switch to Resource Manager mode:

```
azure config mode arm
```

Here is the expected output for the command above:

```
info:  New mode is arm
```

3. From your browser, navigate to <https://raw.githubusercontent.com/telmosampaio/azure-templates/master/IaaS-NSG-UDR/azuredetect.parameters.json>, copy the contents of the json file, and paste into a new file in your computer. For this scenario, you would be copying the values below to a file named **c:\udr\azuredetect.parameters.json**.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "fwCount": {
      "value": 1
    },
    "webCount": {
      "value": 2
    },
    "sqlCount": {
      "value": 2
    }
  }
}
```

- Run the following command to deploy the new VNet by using the template and parameter files you downloaded and modified above:

```
azure group create -n TestRG -l westus --template-uri
'https://raw.githubusercontent.com/telmosampaio/azure-templates/master/IaaS-NSG-UDR/azuredeploy.json' -e
'c:\udr\azuredeploy.parameters.json'
```

Expected output:

```
info: Executing command group create
info: Getting resource group TestRG
info: Updating resource group TestRG
info: Updated resource group TestRG
info: Initializing template configurations and parameters
info: Creating a deployment
info: Created template deployment "azuredeploy"
data: Id: /subscriptions/[Subscription Id]/resourceGroups/TestRG
data: Name: TestRG
data: Location: westus
data: Provisioning State: Succeeded
data: Tags: null
data:
info: group create command OK
```

- Run the following command to view the resources created in the new resource group:

```
azure group show TestRG
```

Expected result:

```
info: Executing command group show
info: Listing resource groups
info: Listing resources for the group
data: Id: /subscriptions/[Subscription Id]/resourceGroups/TestRG
data: Name: TestRG
data: Location: westus
data: Provisioning State: Succeeded
data: Tags: null
data: Resources:
data:
data:   Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/availabilitySets/ASFW
data:     Name   : ASFW
data:     Type   : availabilitySets
data:     Location: westus
data:     Tags    : displayName=AvailabilitySet - DMZ
data:
```

```
data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/availabilitySets/ASSQL
    data:    Name    : ASSQL
    data:    Type    : availabilitySets
    data:    Location: westus
    data:    Tags    : displayName=AvailabilitySet - SQL
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/availabilitySets/ASWEB
    data:    Name    : ASWEB
    data:    Type    : availabilitySets
    data:    Location: westus
    data:    Tags    : displayName=AvailabilitySet - Web
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/FW1
    data:    Name    : FW1
    data:    Type    : virtualMachines
    data:    Location: westus
    data:    Tags    : displayName=VMs - DMZ
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/SQL1
    data:    Name    : SQL1
    data:    Type    : virtualMachines
    data:    Location: westus
    data:    Tags    : displayName=VMs - SQL
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/SQL2
    data:    Name    : SQL2
    data:    Type    : virtualMachines
    data:    Location: westus
    data:    Tags    : displayName=VMs - SQL
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/WEB1
    data:    Name    : WEB1
    data:    Type    : virtualMachines
    data:    Location: westus
    data:    Tags    : displayName=VMs - Web
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/WEB2
    data:    Name    : WEB2
    data:    Type    : virtualMachines
    data:    Location: westus
    data:    Tags    : displayName=VMs - Web
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICFW1
    data:    Name    : NICFW1
    data:    Type    : networkInterfaces
    data:    Location: westus
    data:    Tags    : displayName=NetworkInterfaces - DMZ
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICSQL1
    data:    Name    : NICSQL1
    data:    Type    : networkInterfaces
    data:    Location: westus
    data:    Tags    : displayName=NetworkInterfaces - SQL
    data:
    data:    Id      : /subscriptions/[Subscription]
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICSQL2
    data:    Name    : NICSQL2
    data:    Type    : networkInterfaces
    data:    Location: westus
    data:    Tags    : displayName=NetworkInterfaces - SQL
```

```
    data:    tags : displayName=Network Interfaces - SQL
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICWEB1
    data:    Name   : NICWEB1
    data:    Type   : networkInterfaces
    data:    Location: westus
    data:    Tags   : displayName=NetworkInterfaces - Web
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICWEB2
    data:    Name   : NICWEB2
    data:    Type   : networkInterfaces
    data:    Location: westus
    data:    Tags   : displayName=NetworkInterfaces - Web
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-BackEnd
    data:    Name   : NSG-BackEnd
    data:    Type   : networkSecurityGroups
    data:    Location: westus
    data:    Tags   : displayName=NSG - Front End
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd
    data:    Name   : NSG-FrontEnd
    data:    Type   : networkSecurityGroups
    data:    Location: westus
    data:    Tags   : displayName=NSG - Remote Access
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/PIPFW1
    data:    Name   : PIPFW1
    data:    Type   : publicIPAddresses
    data:    Location: westus
    data:    Tags   : displayName=PublicIPAddresses - DMZ
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/PIPSQL1
    data:    Name   : PIPSQL1
      data:    Type   : publicIPAddresses
    data:    Location: westus
    data:    Tags   : displayName=PublicIPAddresses - SQL
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/PIPSQL2
    data:    Name   : PIPSQL2
    data:    Type   : publicIPAddresses
    data:    Location: westus
    data:    Tags   : displayName=PublicIPAddresses - SQL
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/PIPWEB1
    data:    Name   : PIPWEB1
    data:    Type   : publicIPAddresses
    data:    Location: westus
    data:    Tags   : displayName=PublicIPAddresses - Web
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/PIPWEB2
    data:    Name   : PIPWEB2
    data:    Type   : publicIPAddresses
    data:    Location: westus
    data:    Tags   : displayName=PublicIPAddresses - Web
  data:
  data:    Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/routeTables/UDR-BackEnd
    data:    Name   : UDR-BackEnd
    data:    Type   : routeTables
```

```
data:      Location: westus
data:      Tags     : displayName=Route Table - Back End
data:
data:      Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/routeTables/UDR-FrontEnd
data:      Name    : UDR-FrontEnd
data:      Type    : routeTables
data:      Location: westus
data:      Tags    : displayName=UDR - FrontEnd
data:
data:      Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet
data:      Name    : TestVNet
data:      Type    : virtualNetworks
data:      Location: westus
data:      Tags    : displayName=VNet
data:
data:      Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Storage/storageAccounts/testvnetstorageprm
data:      Name    : testvnetstorageprm
data:      Type    : storageAccounts
data:      Location: westus
data:      Tags    : displayName=Storage Account - Premium
data:
data:      Id      : /subscriptions/[Subscription
Id]/resourceGroups/TestRG/providers/Microsoft.Storage/storageAccounts/testvnetstoragestd
data:      Name    : testvnetstoragestd
data:      Type    : storageAccounts
data:      Location: westus
data:      Tags    : displayName=Storage Account - Simple
data:
data:      Permissions:
data:      Actions: *
data:      NotActions:
data:
info:   group show command OK
```

TIP

If you do not see all the resources, run the `azure group deployment show` command to ensure the provisioning state of the deployment is *Succeeded*.

Control routing and use virtual appliances (classic) using PowerShell

1/17/2017 • 4 min to read • [Edit on GitHub](#)

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

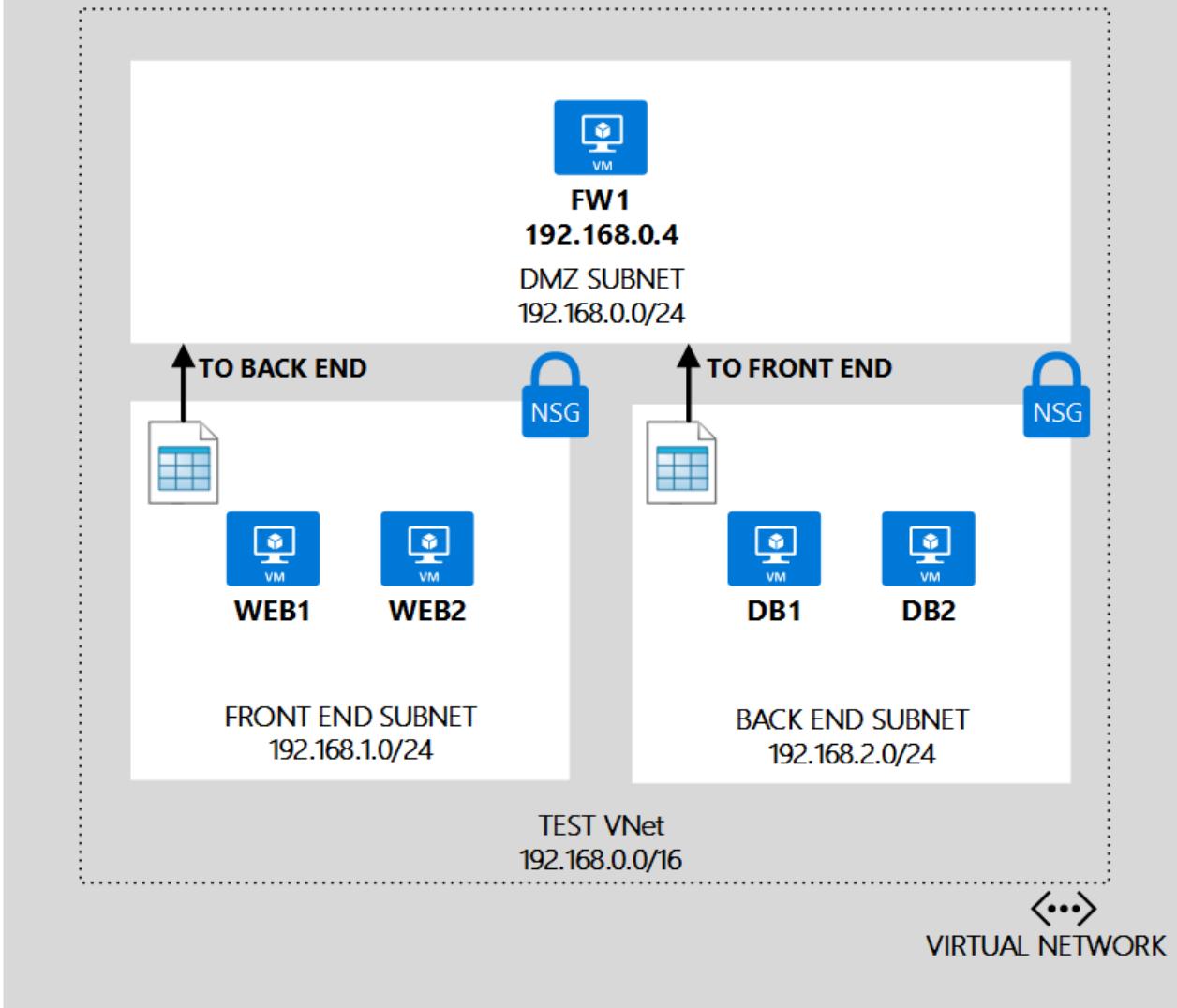
Before you work with Azure resources, it's important to understand that Azure currently has two deployment models: Azure Resource Manager and classic. Make sure you understand [deployment models and tools](#) before you work with any Azure resource. You can view the documentation for different tools 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 will use the scenario below.



AZURE REGION



In this scenario you will create one UDR for the *Front end subnet* and another UDR for the *Back end subnet* , as described below:

- **UDR-FrontEnd.** The front end UDR will be applied to the *FrontEnd* subnet, and contain one route:
 - **RouteToBackend.** This route will send all traffic to the back end subnet to the **FW1** virtual machine.
- **UDR-BackEnd.** The back end UDR will be applied to the *BackEnd* subnet, and contain one route:
 - **RouteToFrontend.** This route will send all traffic to the front end subnet to the **FW1** virtual machine.

The combination of these routes will ensure that all traffic destined from one subnet to another will be routed to the **FW1** virtual machine, which is being used as a virtual appliance. You also need to turn on IP forwarding for that 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'll need to [to install and configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.

NOTE

If you don't have an Azure account, you'll need one. Go sign up for a [free trial here](#).

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.

- 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"
```

Output:

Name	Location	Label
---	-----	-----
UDR-FrontEnd	West US	Route table for front end subnet

- 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
```

Output:

Name : UDR-FrontEnd			
Location : West US			
Label : Route table for frontend subnet			
Routes :			
Name Address Prefix Next hop type Next hop IP address			
----	-----	-----	-----
RouteToBackEnd 192.168.2.0/24 VirtualAppliance 192.168.0.4			

- 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:

- 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"
```

- 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
```

Output:

```
Disabled
```

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

1/17/2017 • 5 min to read • [Edit on GitHub](#)

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.

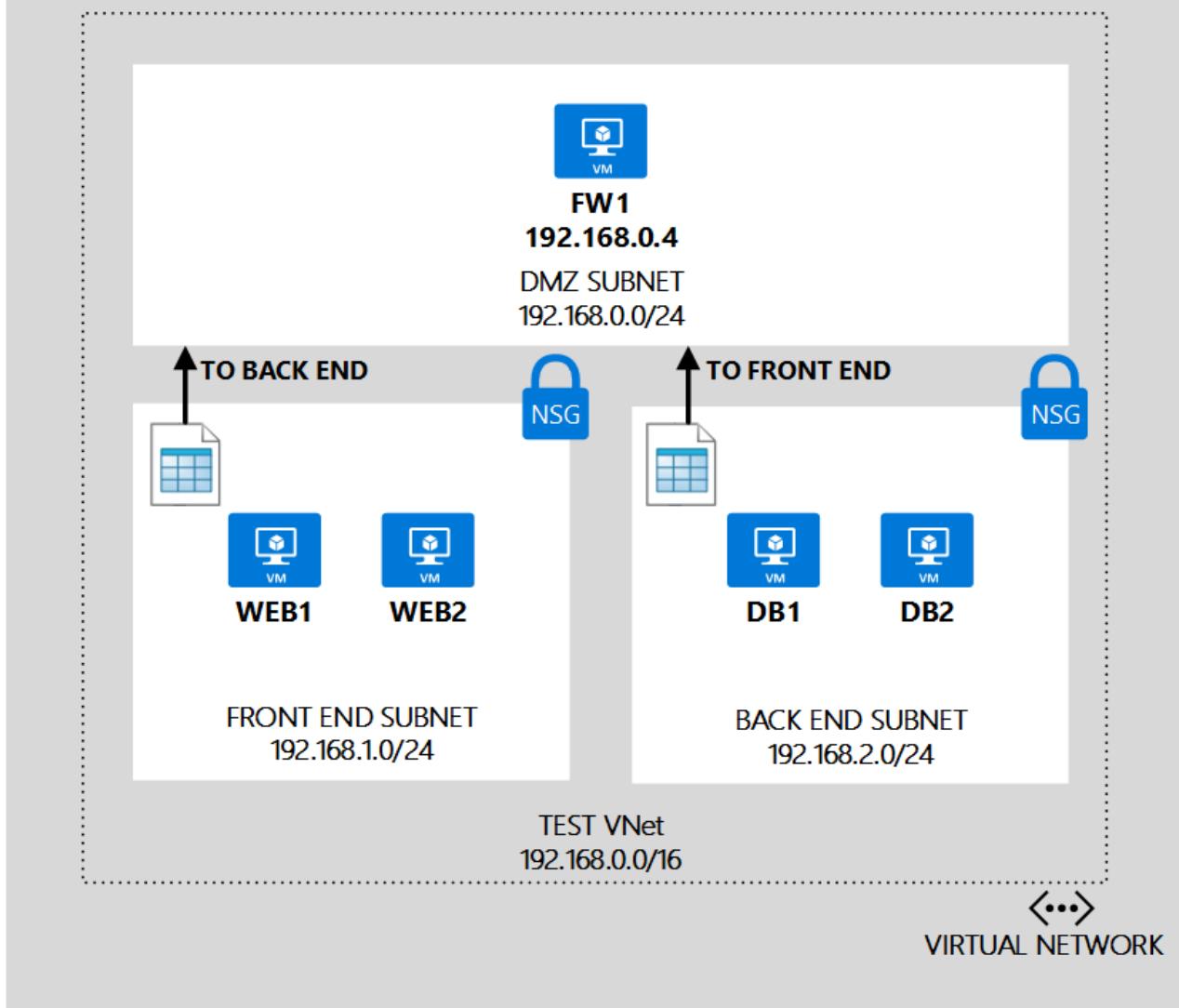
This article covers the classic deployment model. You can also [control routing and use virtual appliances in the Resource Manager deployment model](#).

Scenario

To better illustrate how to create UDRs, this document will use the scenario below.



AZURE REGION



In this scenario you will create one UDR for the *Front end subnet* and another UDR for the *Back end subnet*, as described below:

- **UDR-FrontEnd.** The front end UDR will be applied to the *FrontEnd* subnet, and contain one route:
 - **RouteToBackend.** This route will send all traffic to the back end subnet to the **FW1** virtual machine.
- **UDR-BackEnd.** The back end UDR will be applied to the *BackEnd* subnet, and contain one route:
 - **RouteToFrontend.** This route will send all traffic to the front end subnet to the **FW1** virtual machine.

The combination of these routes will ensure that all traffic destined from one subnet to another will be routed to the **FW1** virtual machine, which is being used as a virtual appliance. You also need to turn on IP forwarding for that 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*.
- 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:

- Run the following command to create a route table for the back-end subnet:

```
azure network route-table create -n UDR-BackEnd -l uswest
```

- 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
```

- 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
```

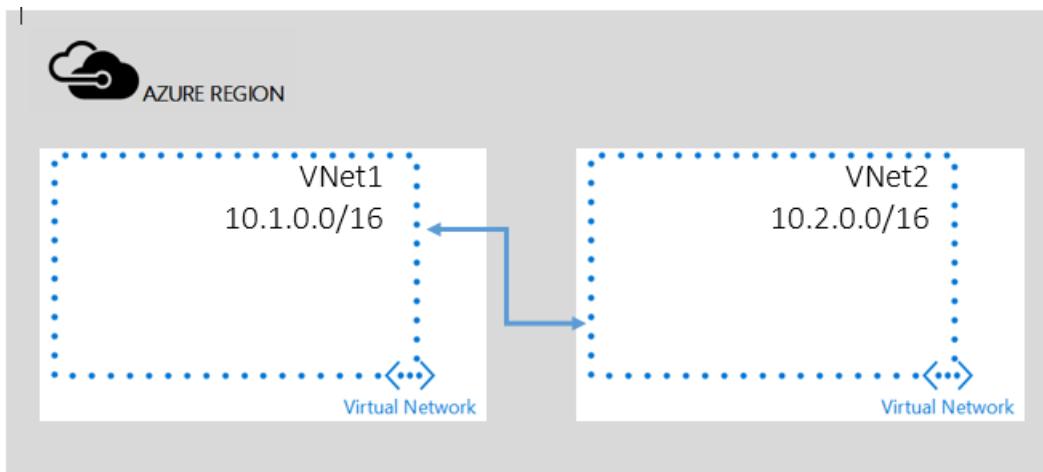
Create a virtual network peering using the Azure portal

1/17/2017 • 5 min to read • [Edit on GitHub](#)

VNet Peering is a mechanism to connect two Virtual Networks in the same region through the Azure backbone network. Once peered, the two Virtual Networks will appear like a single Virtual Network for all connectivity purposes. Read the [VNet Peering overview](#) if you are not familiar with VNet Peering.

Peering VNets in the same subscription

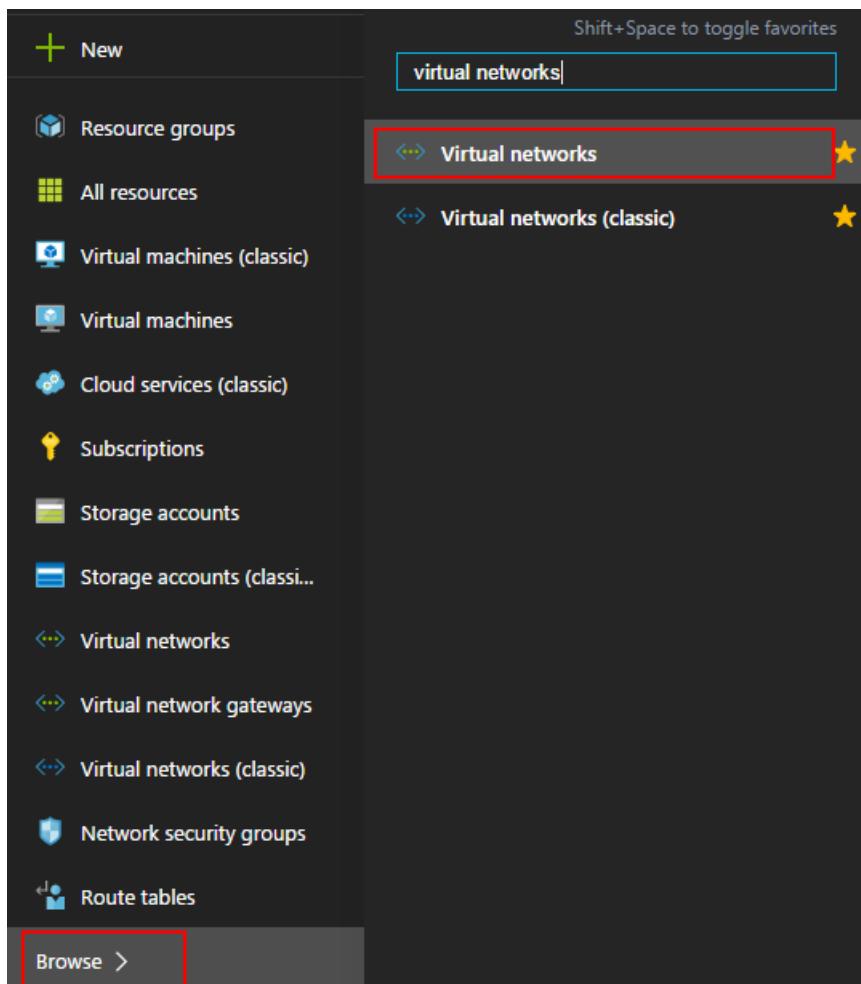
In this scenario you will create a peering between two VNets named **VNet1** and **VNet2** belonging to the same subscription.



VNet peering will allow full connectivity between the entire address space of peered virtual networks.

To create a VNet peering based on the scenario above by using the Azure portal, follow the steps below.

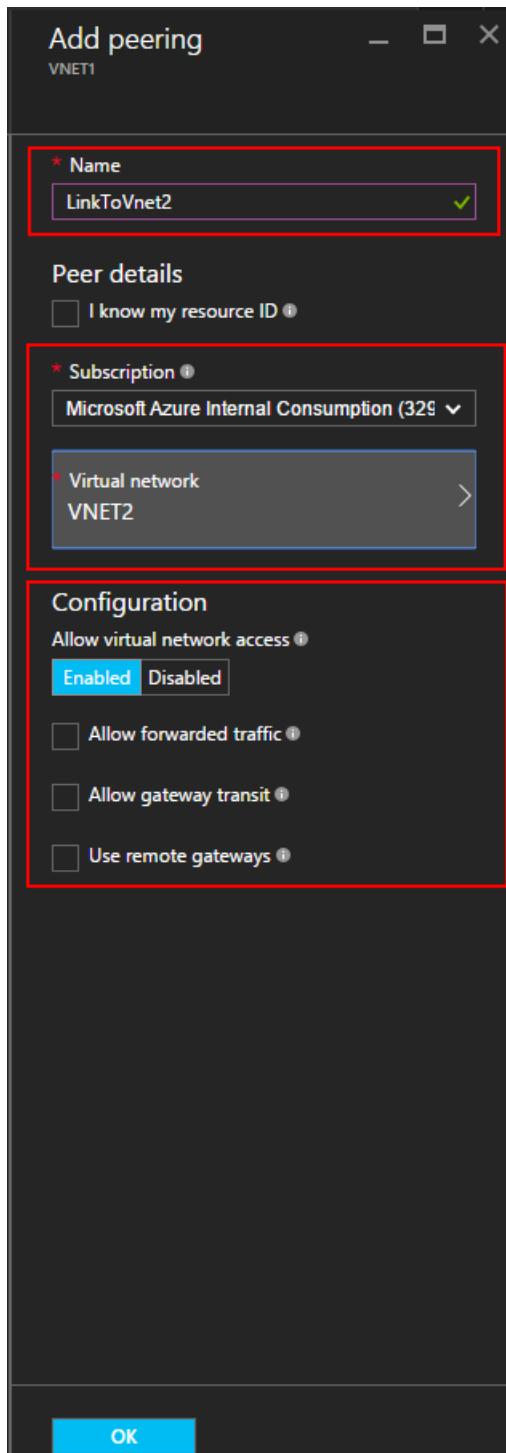
1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. To establish VNET peering, you need to create two links, one for each direction, between two VNets. You can create VNET peering link for VNET1 to VNET2 first. On the portal, Click **Browse > choose Virtual Networks**



3. In Virtual Networks blade, choose VNET1, click Peerings, then click Add

The screenshot shows the "VNET1 - Peerings" blade. On the left, there is a sidebar with the following options: "Overview", "Activity logs", "Access control (IAM)", "Tags", "Address space", "Connected devices", "Subnets", "DNS servers", and "Peerings". The "Peerings" option is highlighted with a red box. On the right, there is a main area with a search bar labeled "Search (Ctrl+ /)" and a "NAME" field. Below the search bar, it says "No results.". In the top right corner, there is a large "Add" button, which is also highlighted with a red box.

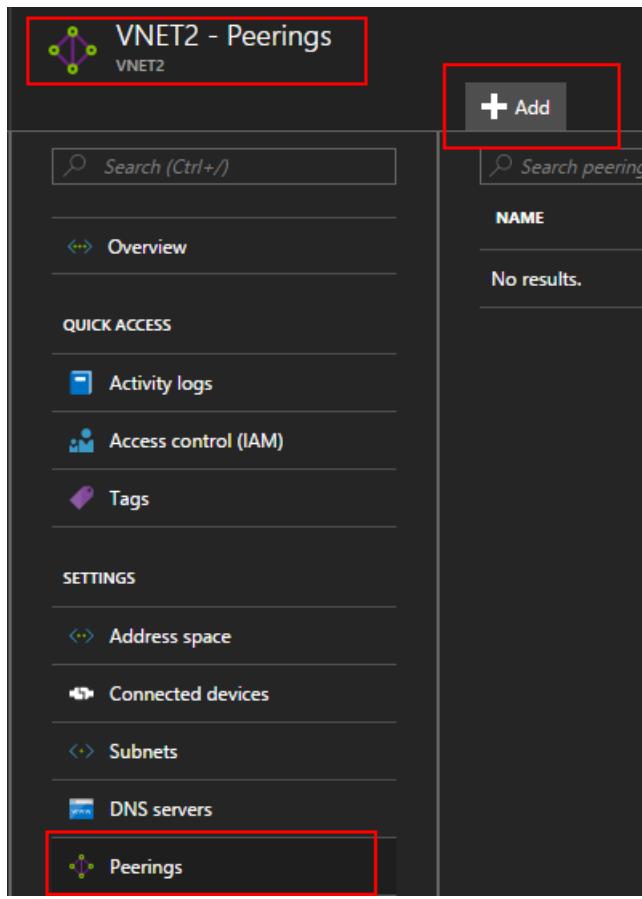
4. In the Add Peering blade, give a peering link name LinkToVnet2, choose the subscription and the peer Virtual Network VNET2, click OK.



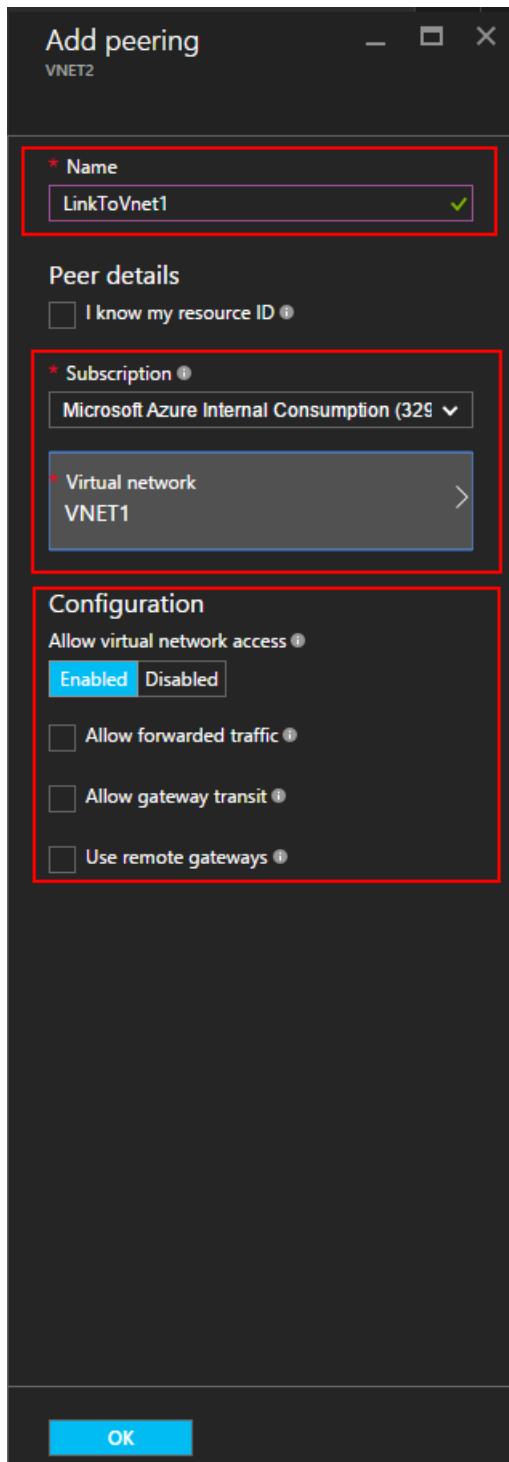
- Once this VNET peering link is created. You can see the link state as following:

Search peerings				
NAME	STATUS	PEER	GATEWAY TRANSIT	...
LinkToVnet2	Initiated	VNET2	Disabled	...

- Next create the VNET peering link for VNET2 to VNET1. In Virtual Networks blade, choose VNET2, click Peerings, then click Add



7. In the Add Peering blade, give a peering link name LinkToVnet1, choose the subscription and the peer Virtual Network, Click OK.



8. Once this VNET peering link is created. You can see the link state as following:

+ Add				
Search peerings				
NAME	STATUS	PEER	GATEWAY TRANSIT	...
LinkToVNet1	Connected	VNET1	Disabled	...

9. Check the state for LinkToVnet2 and it now changes to Connected as well.

Add				
<input type="text"/> Search peerings				
NAME	STATUS	PEER	GATEWAY TRANSIT	...
LinkToVnet2	Connected	VNET2	Disabled	...

NOTE

VNET peering is only established if both links are connected.

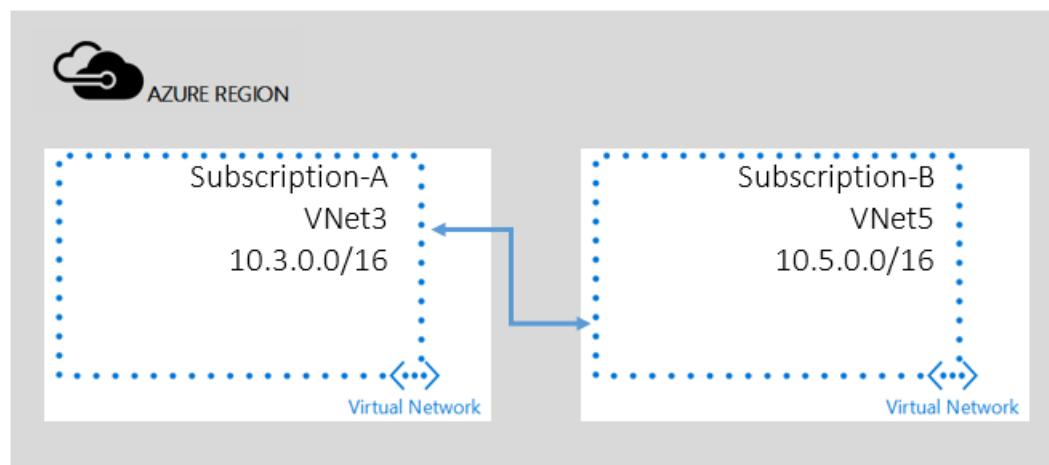
There are a few configurable properties for each link:

OPTION	DESCRIPTION	DEFAULT
AllowVirtualNetworkAccess	Whether address space of Peer VNet to be included as part of the Virtual_network Tag	Yes
AllowForwardedTraffic	Whether traffic not originating from a peered VNet is accepted or dropped	No
AllowGatewayTransit	Allows the peer VNet to use your VNet gateway	No
UseRemoteGateways	Use your peer's VNet gateway. The peer VNet must have a gateway configured and AllowGatewayTransit is selected. You cannot use this option if you have a gateway configured	No

Each link in VNet peering has a set of above properties. From portal, you can click the VNet Peering Link and change any available options, click Save to make the change effect.

Peering across subscriptions

In this scenario you will create a peering between two VNets belonging to different subscriptions.



VNet peering relies on role-based access control (RBAC) for authorization. For cross-subscriptions scenario, you first need to grant sufficient permission to users who will create the peering link:

NOTE

If the same user has the privilege over both subscriptions, then you can skip step1-4 below.

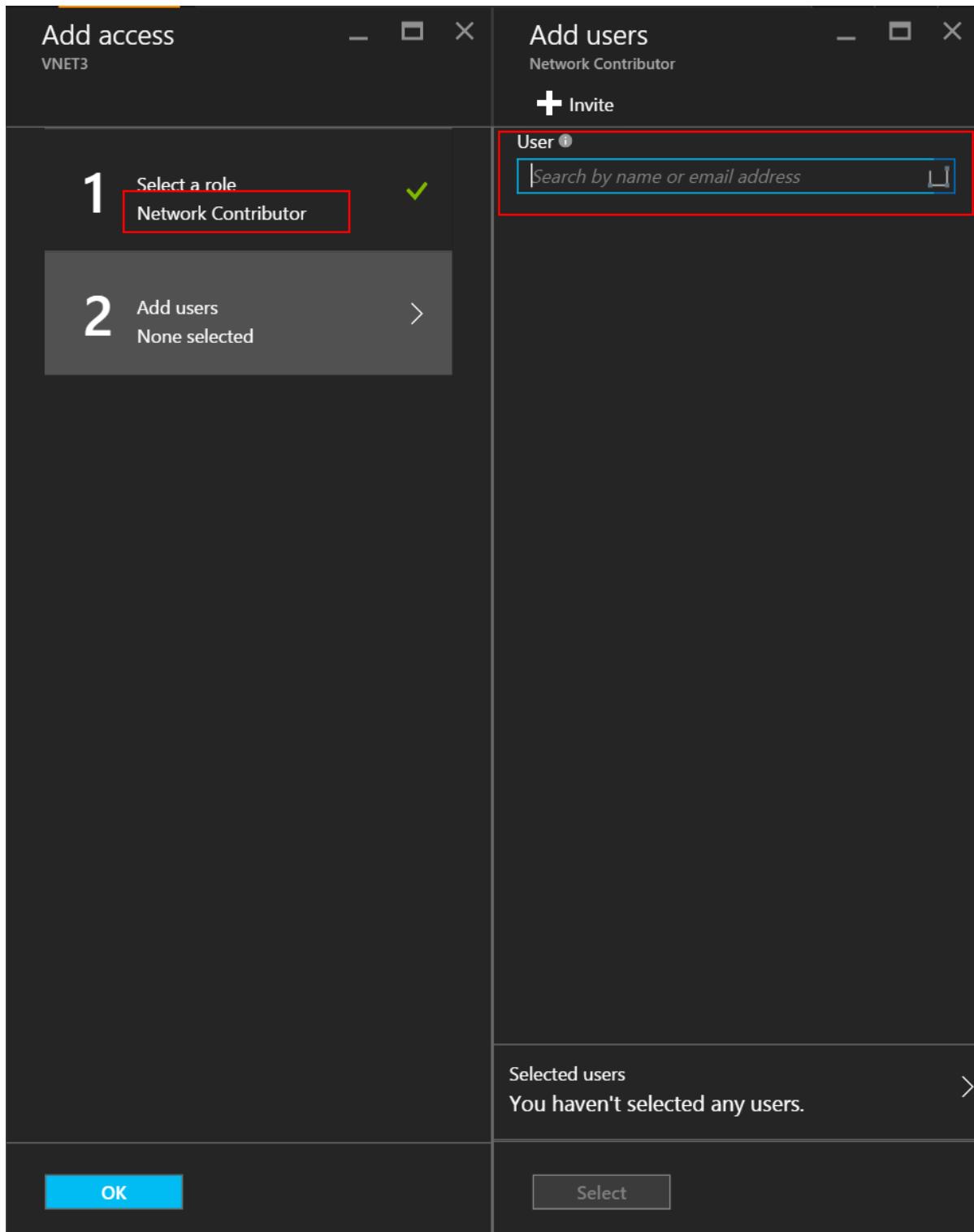
1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. In this example we will use two subscriptions A and B and two users UserA and UserB with privileges in the subscriptions respectively
3. On the portal, Click Browse, choose Virtual Networks. Click the VNET and click Add.

The screenshot shows the 'VNET3 - Users' blade in the Azure portal. At the top, there's a 'Search (Ctrl+ /)' bar. Below it, a 'QUICK ACCESS' sidebar with 'Overview', 'Activity logs', and 'Access control (IAM)' (which is highlighted with a red box). The main area has a 'Roles' section with a 'User' table:

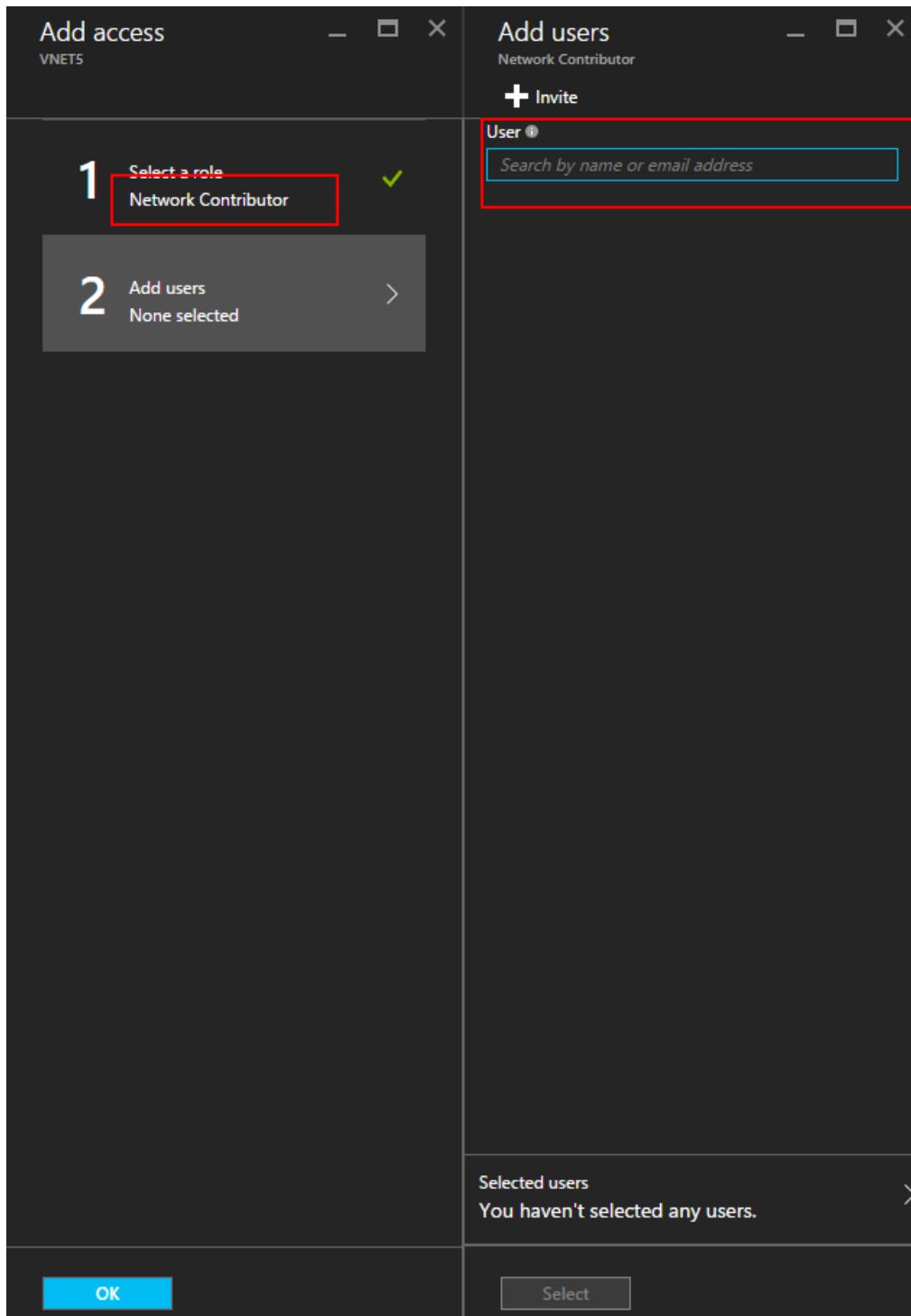
USER	ROLE	ACCESS
Subscription admins	Owner	Inherited

At the top right of the main area, there are 'Add' and 'Roles' buttons, both highlighted with red boxes.

4. On the Add access blade, click select a role and choose Network Contributor, click Add Users, type the UserB sign in name, and click OK.



5. Then login to Azure portal with UserB who is the privilege user for SubscriptionB. Follow above steps to add UserA as Network Contributor.

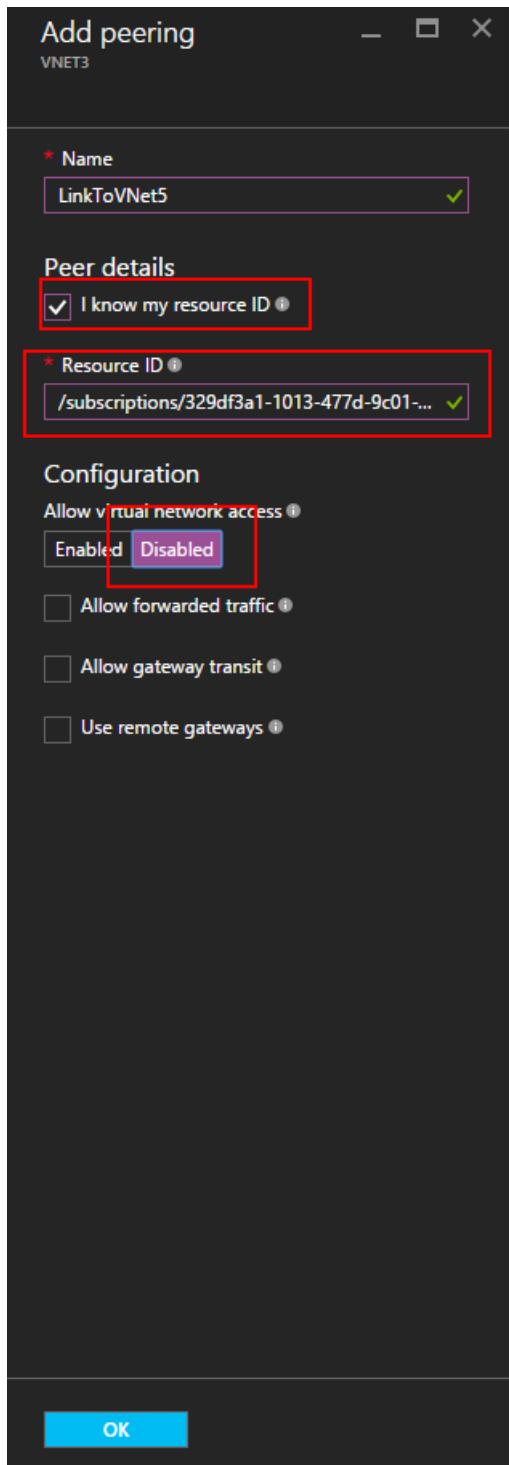


NOTE

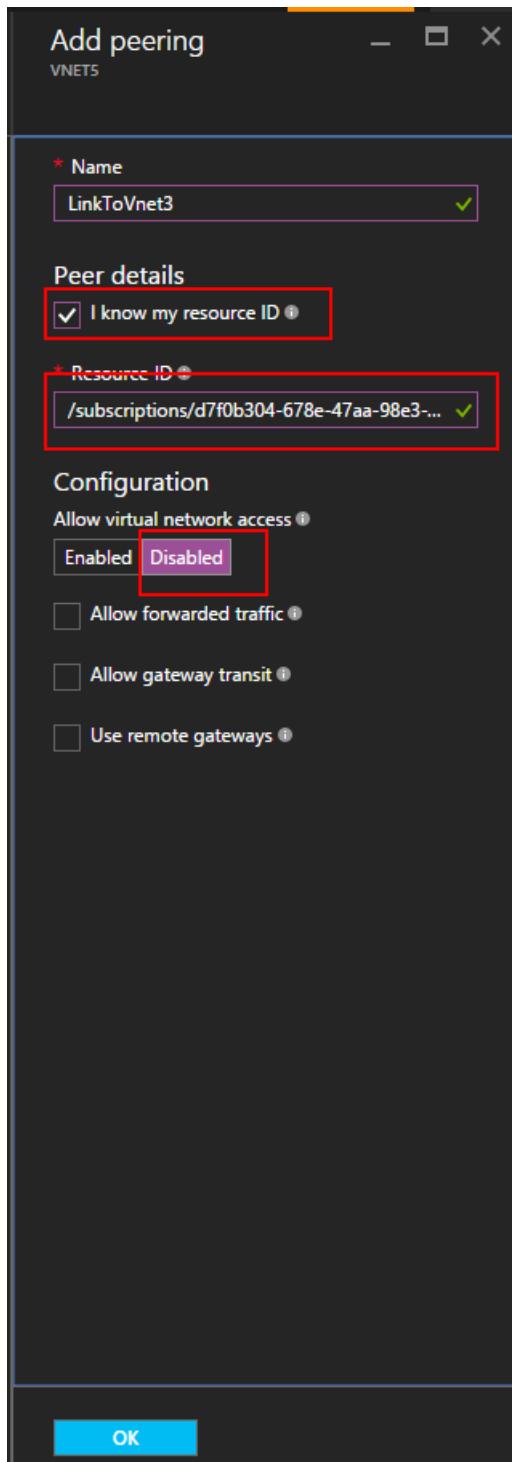
You can log off and log on both user sessions in browser to ensure the authorization is enabled successfully.

6. Login to the portal as UserA, navigate to the VNET3 blade, click Peering, check 'I Know my resource ID' checkbox and type the resource ID for VNET5 in below format.

/subscriptions/{SubscriptionID}/resourceGroups/{ResourceGroupName}/providers/Microsoft.Network/Virtu
alNetwork/{VNETname}



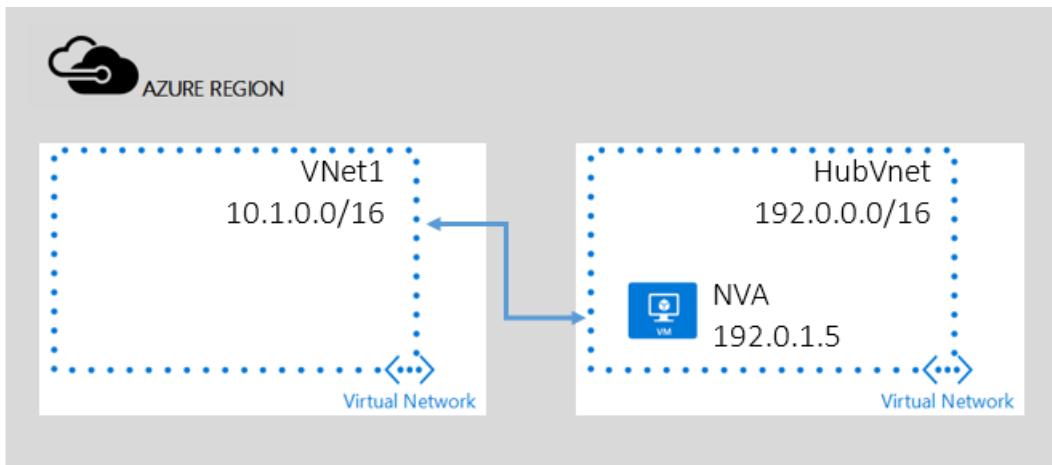
7. Login to the portal as UserB and follow above step to create peering link from VNET5 to VNet3.



8. Peering will be established and any Virtual machine in VNet3 should be able to communicate with any virtual machine in VNet5

Service Chaining - Transit through peered VNet

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. In this scenario, there are two VNets in a subscription, HubVNet and VNet1 as described in the diagram below. You deploy Network Virtual Appliance(NVA) in VNet HubVNet. After establishing VNet peering between HubVNet and VNet1, you can set up User Defined Routes and specify the next hop to NVA in the HubVNet.

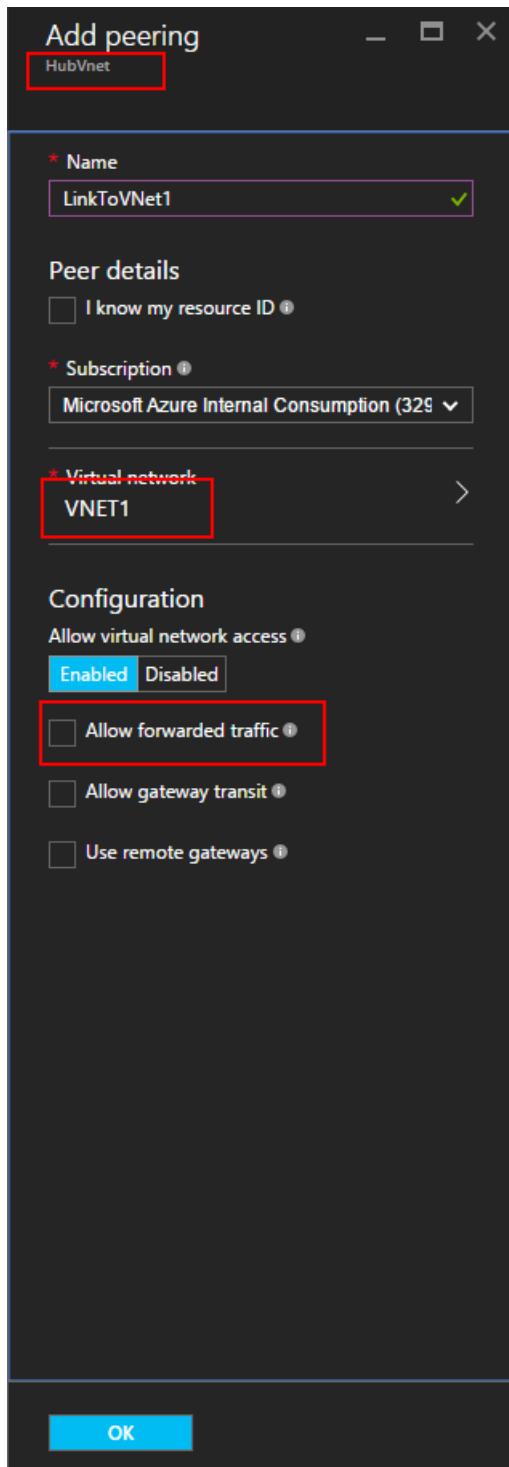


NOTE

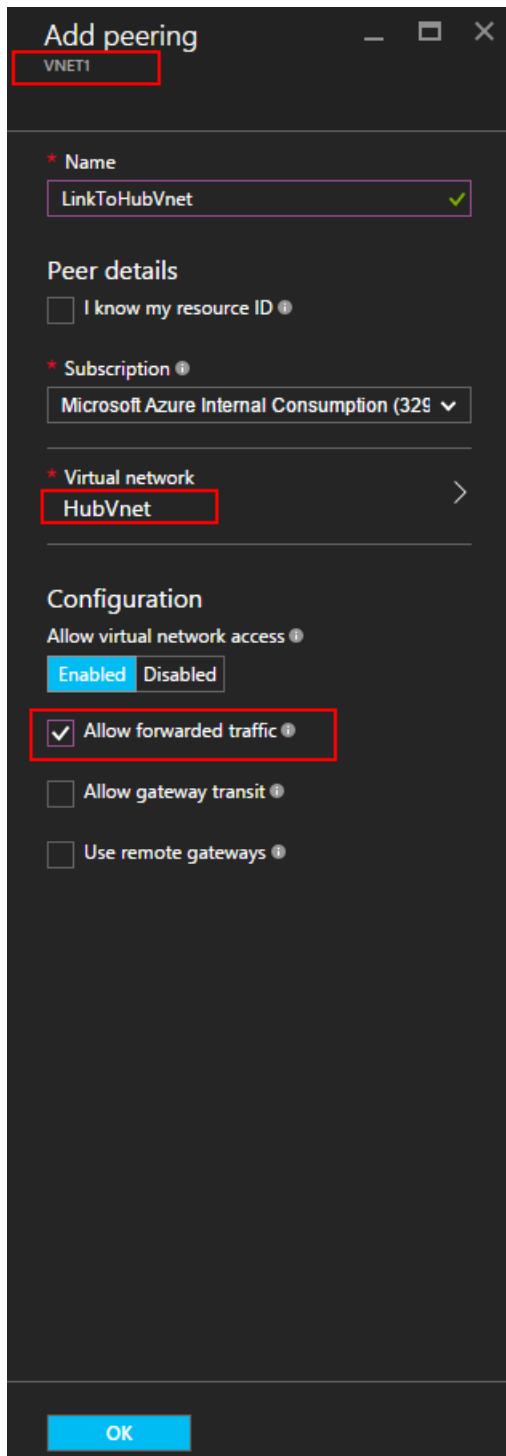
For the simplicity, assume all VNets here are in the same subscription. But it also works for cross-subscription scenario.

The key property to enable Transit routing is the "Allow Forwarded Traffic" parameter. This allows accepting and sending traffic from/to the NVA in the peered VNet.

1. As a first step, VNET peering links from HubVnet to VNET1. Note that Allow Forwarded Traffic option is not selected for the link.



2. As a next step, peering links from VNET1 to HubVnet can be created. Note that Allow forwarded traffic option is selected.



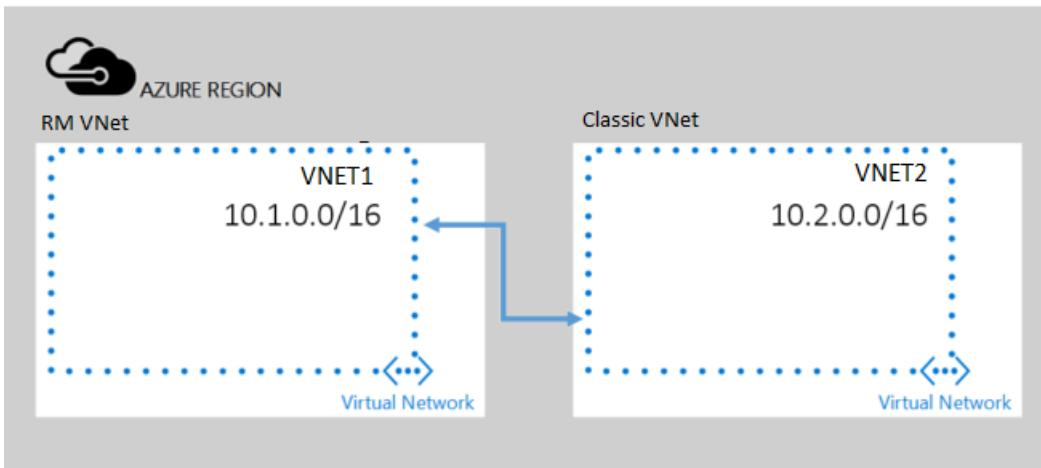
3. After peering is established, you can refer to this [article](#) and define User Defined Route(UDR) to redirect VNet1 traffic through a virtual appliance to use its capabilities. When you specify the Next Hop address in route, you can set it to the IP address of virtual appliance in peer VNet HubVNet

Peering virtual networks from RM to classic

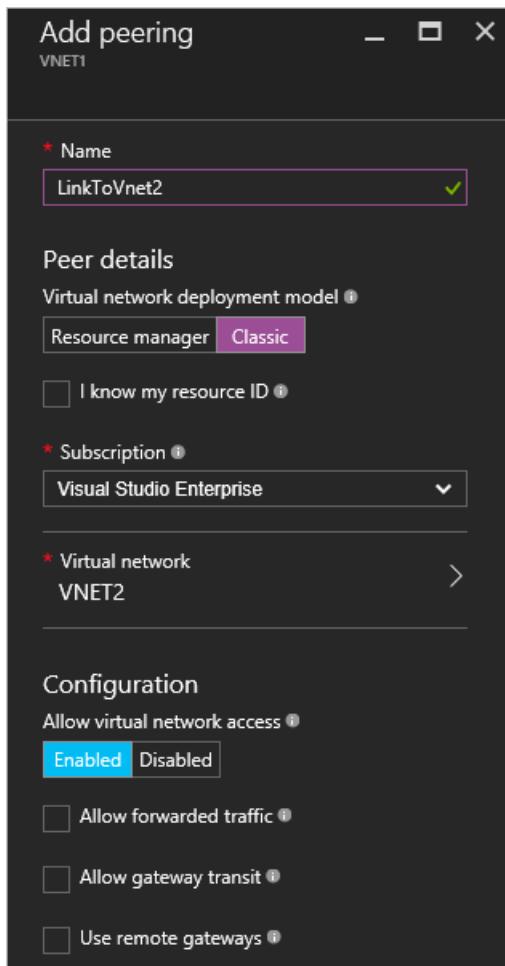
In this scenario, you will create a peering between two VNets, namely **VNET1** and **VNET2** belonging to Azure Resource Manager deployment model and classic deployment model respectively.

NOTE

The virtual networks must be in the same subscription.



1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. To establish VNET peering in this scenario, you need to create only one link, from the virtual network in Azure resource manager to the one in classic. That is, from **VNET1** to **VNET2**. On the portal, Click **Browse** > choose **Virtual Networks**
3. In the Virtual networks blade, choose **VNET1**. Click **Peerings**, then click **Add**.
4. In the Add Peering blade, name your link. Here it is called **LinkToVNet2**. Under Peer details, select **Classic**.
5. Then choose the subscription and the peer Virtual Network **VNET2**. Then click OK.

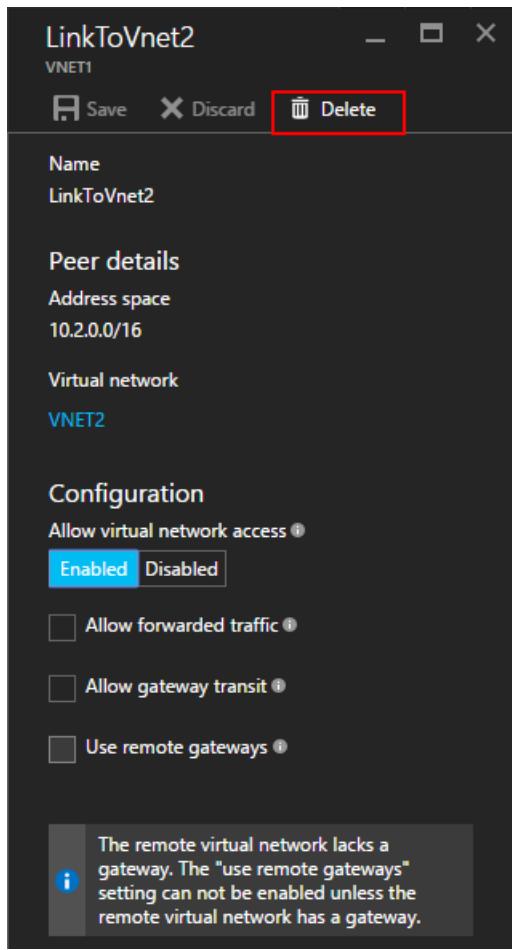


6. Once this VNet peering link is created, the two virtual networks are peered and you will be able to see the following:

Name	Status	Peer	Gateway Transit
LinkToVnet2	Connected	VNET2	Disabled

Remove VNet Peering

1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Go to virtual network blade, click Peerings, click the Link you want to remove, click button Delete.



- Once you remove one link in VNET peering, the peer link state will go to disconnected.

Search peerings				
NAME	STATUS	PEER	GATEWAY TRANSIT	...
LinkToVNet1	Disconnected	VNET1	Disabled	...

- In this state, you cannot re-create the link until the peer link state changes to Initiated. We recommend you remove the both links before you re-create the VNET peering.

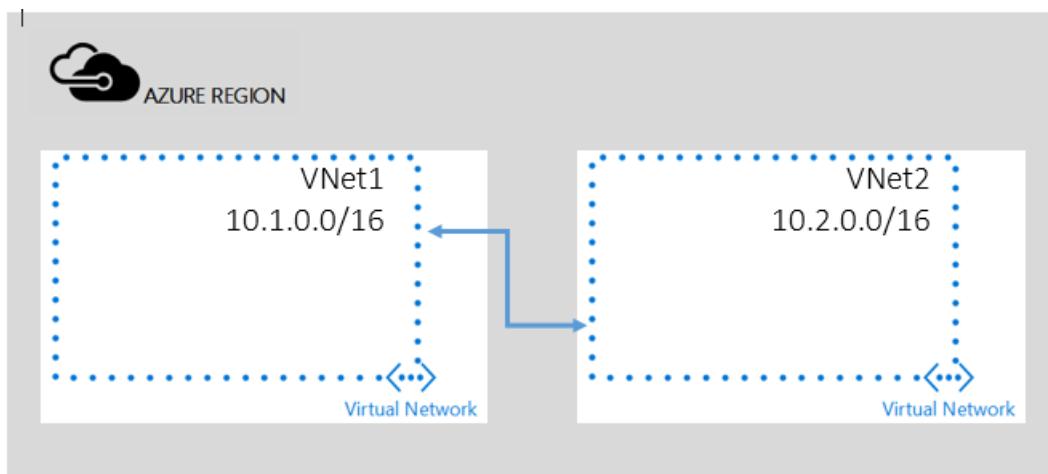
Create VNet Peering using Powershell cmdlets

1/17/2017 • 7 min to read • [Edit on GitHub](#)

VNet Peering is a mechanism to connect two Virtual Networks in the same region through the Azure backbone network. Once peered, the two Virtual Networks will appear like a single Virtual Network for all connectivity purposes. Read the [VNet Peering overview](#) if you are not familiar with VNet Peering.

Peering VNets in the same subscription

In this scenario you will create a peering between two VNets named **VNet1** and **VNet2** belonging to the same subscription.



VNet peering will allow full connectivity between the entire address space of peered virtual networks.

To create a VNet peering by using PowerShell, please follow the steps below:

1. If you have never used Azure PowerShell, see [How to Install and Configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.

NOTE

PowerShell cmdlet for managing VNet peering is shipped with [Azure PowerShell 1.6](#).

2. Read virtual network objects:

```
$vnet1 = Get-AzureRmVirtualNetwork -ResourceGroupName vnet101 -Name vnet1  
$vnet2 = Get-AzureRmVirtualNetwork -ResourceGroupName vnet101 -Name vnet2
```

3. To establish VNet peering, you need to create two links, one for each direction. The following step will create a VNet peering link for VNet1 to VNet2 first:

```
Add-AzureRmVirtualNetworkPeering -Name LinkToVNet2 -VirtualNetwork $vnet1 -RemoteVirtualNetworkId  
$vnet2.Id
```

Output shows:

```

Name          : LinkToVNet2
Id: /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/vnet101/providers/Microsoft.Network/virtualNetworks/vnet1/virtualNetworkPeerin
gs/LinkToVNet2
Etag          : W/"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
ResourceGroupName   : vnet101
VirtualNetworkName    : vnet1
PeeringState        : Initiated
ProvisioningState    : Succeeded
RemoteVirtualNetwork  : {
                        "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/vnet101/providers/Microsoft.Network/virtualNetworks/vnet2"
}
AllowVirtualNetworkAccess  : True
AllowForwardedTraffic    : False
AllowGatewayTransit      : False
UseRemoteGateways        : False
RemoteGateways          : null
RemoteVirtualNetworkAddressSpace : null

```

4. This step will create a VNet peering link for VNet2 to VNet1:

```
Add-AzureRmVirtualNetworkPeering -Name LinkToVNet1 -VirtualNetwork $vnet2 -RemoteVirtualNetworkId
$vnet1.Id
```

Output shows:

```

Name          : LinkToVNet1
Id: /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/vnet101/providers/Microsoft.Network/virtualNetworks/vnet2/virtualNetworkPeerin
gs/LinkToVNet1
Etag          : W/"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
ResourceGroupName   : vnet101
VirtualNetworkName    : vnet2
PeeringState        : Connected
ProvisioningState    : Succeeded
RemoteVirtualNetwork  : {
                        "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/vnet101/providers/Microsoft.Network/virtualNetworks/vnet1"
}
AllowVirtualNetworkAccess  : True
AllowForwardedTraffic    : False
AllowGatewayTransit      : False
UseRemoteGateways        : False
RemoteGateways          : null
RemoteVirtualNetworkAddressSpace : null

```

5. Once the VNet peering link is created, you can see the link state below:

```
Get-AzureRmVirtualNetworkPeering -VirtualNetworkName vnet1 -ResourceGroupName vnet101 -Name linktovnet2
```

Output shows:

```

Name          : LinkToVNet2
Id           : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/vnet101/providers/Microsoft.Network/virtualNetworks/vnet1/virtualNetworkPeerin
gs/LinkToVNet2
Etag         : W/"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
ResourceGroupName   : vnet101
VirtualNetworkName    : vnet1
PeeringState        : Connected
ProvisioningState    : Succeeded
RemoteVirtualNetwork  : {
                        "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/vnet101/providers/Microsoft.Network/virtualNetworks/vnet2"
}
AllowVirtualNetworkAccess  : True
AllowForwardedTraffic     : False
AllowGatewayTransit       : False
UseRemoteGateways        : False
RemoteGateways           : null
RemoteVirtualNetworkAddressSpace : null

```

There are a few configurable properties for VNet peering:

OPTION	DESCRIPTION	DEFAULT
AllowVirtualNetworkAccess	Whether address space of Peer VNet to be included as part of the Virtual_network Tag	Yes
AllowForwardedTraffic	Whether traffic not originating from a peered VNet is accepted or dropped	No
AllowGatewayTransit	Allows the peer VNet to use your VNet gateway	No
UseRemoteGateways	Use your peer's VNet gateway. The peer VNet must have a gateway configured and AllowGatewayTransit selected. You cannot use this option if you have a gateway configured	No

Each link in VNet peering has the set of properties above. For example, you can set

AllowVirtualNetworkAccess to True for VNet peering link VNet1 to VNet2 and set it to False for the VNet peering link in the other direction.

```

$LinktoVNet2 = Get-AzureRmVirtualNetworkPeering -VirtualNetworkName vnet1 -ResourceGroupName vnet101 -
Name LinkToVNet2
$LinktoVNet2.AllowForwardedTraffic = $true
Set-AzureRmVirtualNetworkPeering -VirtualNetworkPeering $LinktoVNet2

```

You can run Get-AzureRmVirtualNetworkPeering to double check the property value after the change. From the output, you can see AllowForwardedTraffic changes set to True after running the above cmdlets.

```

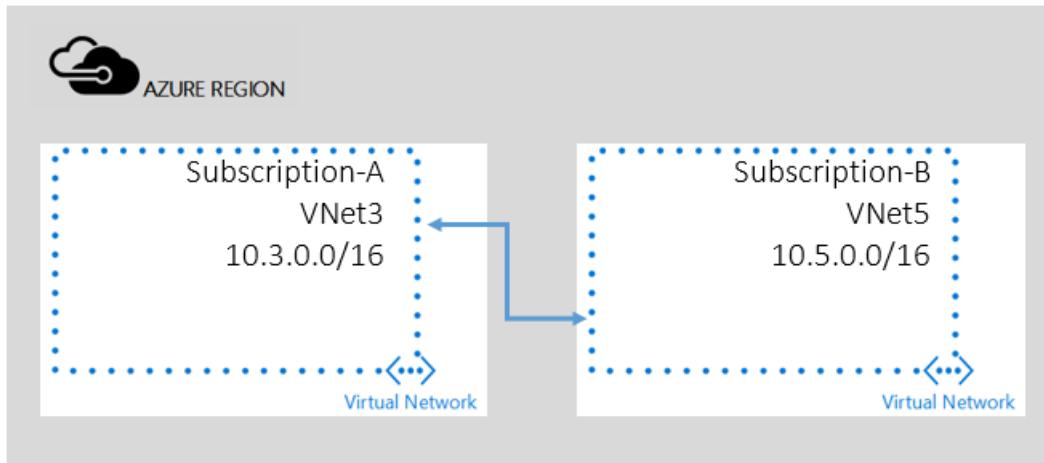
Name : LinkToVNet2
Id : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/vnet101/providers/Microsoft.Network/virtualNetworks/vnet1/virtualNetworkPeerings/LinkToVNet2
Etag : W/"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
ResourceGroupName : vnet101
VirtualNetworkName : vnet1
PeeringState : Connected
ProvisioningState : Succeeded
RemoteVirtualNetwork : {
    "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/vnet101/providers/Microsoft.Network/virtualNetworks/vnet2"
}
AllowVirtualNetworkAccess : True
AllowForwardedTraffic : True
AllowGatewayTransit : False
UseRemoteGateways : False
RemoteGateways : null
RemoteVirtualNetworkAddressSpace : null

```

After peering is established in this scenario, you should be able to initiate the connections from any virtual machine to any virtual machine of both VNets. By default, AllowVirtualNetworkAccess is True and VNet peering will provision the proper ACLs to allow the communication between VNets. You can still apply network security group (NSG) rules to block connectivity between specific subnets or virtual machines to gain fine grain control of access between two virtual networks. For more information about creating NSG rules, please refer to this [article](#).

Peering across subscriptions

In this scenario you will create a peering between two VNets belonging to different subscriptions.



VNet peering relies on role-based access control (RBAC) for authorization. For cross-subscriptions scenario, you first need to grant sufficient permission to users who will create the peering link:

NOTE

If the same user has the privilege over both subscriptions, then you can skip step1-4 below.

To create VNet peering across subscriptions using PowerShell, please follow the steps below:

1. Sign in to Azure with privileged User-A's account for Subscription-A and run the following cmdlet:

```
New-AzureRmRoleAssignment -SignInName <UserB ID> -RoleDefinitionName "Network Contributor" -Scope /subscriptions/<Subscription-A-ID>/resourceGroups/<ResourceGroupName>/providers/Microsoft.Network/VirtualNetworks/VNet5
```

This is not a requirement, peering can be established even if users individually raise peering requests for their respective VNets as long as the requests match. Adding a privileged user of the other VNet as a user in the local VNet makes it easier to do the setup.

2. Sign in to Azure with privileged User-B's account for Subscription-B and run the following cmdlet:

```
New-AzureRmRoleAssignment -SignInName <UserA ID> -RoleDefinitionName "Network Contributor" -Scope /subscriptions/<Subscription-B-ID>/resourceGroups/<ResourceGroupName>/providers/Microsoft.Network/VirtualNetworks/VNet3
```

3. In User-A's login session, run the cmdlet below:

```
$vnet3 = Get-AzureRmVirtualNetwork -ResourceGroupName hr-vnets -Name vnet3  
  
Add-AzureRmVirtualNetworkPeering -Name LinkToVNet5 -VirtualNetwork $vnet3 -RemoteVirtualNetworkId "/subscriptions/<Subscription-B-ID>/resourceGroups/<ResourceGroupName>/providers/Microsoft.Network/virtualNetworks/VNet5" -  
BlockVirtualNetworkAccess
```

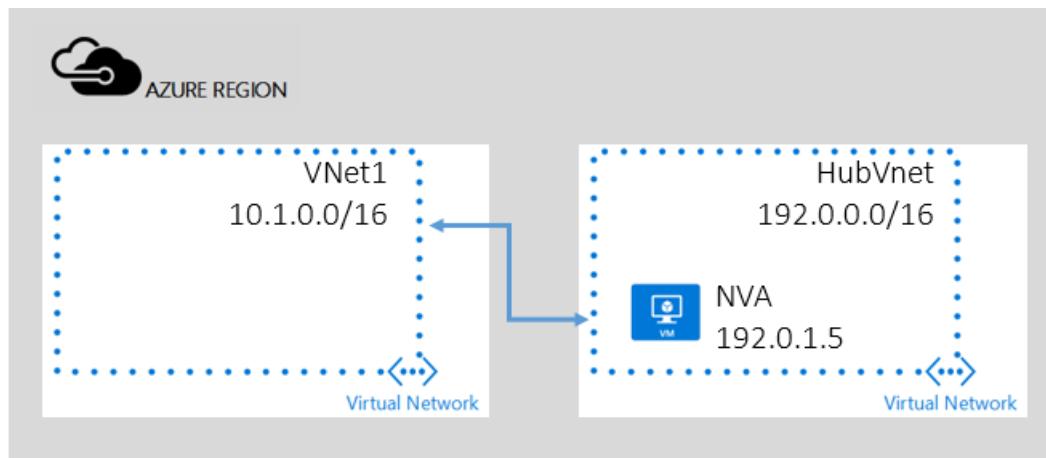
4. In User-B's login session, run the cmdlet below:

```
$vnet5 = Get-AzureRmVirtualNetwork -ResourceGroupName vendor-vnets -Name vnet5  
  
Add-AzureRmVirtualNetworkPeering -Name LinkToVNet3 -VirtualNetwork $vnet5 -RemoteVirtualNetworkId "/subscriptions/<Subscription-A-ID>/resourceGroups/<ResourceGroupName>/providers/Microsoft.Network/virtualNetworks/VNet3" -  
BlockVirtualNetworkAccess
```

5. After peering is established, any virtual machine in VNet3 should be able to communicate with any virtual machine in VNet5.

Service Chaining - Transit through peered VNet

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. In this scenario, there are two VNets in a subscription, HubVNet and VNet1 as described in the diagram below. You deploy Network Virtual Appliance(NVA) in VNet HubVNet. After establishing VNet peering between HubVNet and VNet1, you can set up User Defined Routes and specify the next hop to NVA in the HubVNet.



NOTE

For the simplicity, assume all VNets here are in the same subscription. But it also works for cross-subscription scenario.

The key property to enable Transit routing is the "Allow Forwarded Traffic" parameter. This allows accepting and sending traffic from/to the NVA in the peered VNet.

1. In this scenario, you can run the PowerShell cmdlets below to establish the VNet peering. You need to set the AllowForwardedTraffic property to True and link VNET1 to HubVNet, which allows the inbound traffic from outside of the peering VNet address space.

```
$hubVNet = Get-AzureRmVirtualNetwork -ResourceGroupName vnet101 -Name HubVNet
$vnet1 = Get-AzureRmVirtualNetwork -ResourceGroupName vnet101 -Name vnet1

Add-AzureRmVirtualNetworkPeering -Name LinkToHub -VirtualNetwork $vnet1 -RemoteVirtualNetworkId
$HubVNet.Id -AllowForwardedTraffic

Add-AzureRmVirtualNetworkPeering -Name LinkToVNet1 -VirtualNetwork $HubVNet -RemoteVirtualNetworkId
$vnet1.Id
```

2. After peering is established, you can refer to this [article](#) and define a user-defined route (UDR) to redirect VNet1 traffic through a virtual appliance to use its capabilities. When you specify the next hop address in the route, you can set it to the IP address of the virtual appliance in the peer VNet HubVNet. Below is a sample:

```
$route = New-AzureRmRouteConfig -Name TestNVA -AddressPrefix 10.3.0.0/16 -NextHopType VirtualAppliance -
NextHopIpAddress 192.0.1.5

$routeTable = New-AzureRmRouteTable -ResourceGroupName VNet101 -Location brazilsouth -Name TestRT -Route
$route

$vnet1 = Get-AzureRmVirtualNetwork -ResourceGroupName VNet101 -Name VNet1

Set-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet1 -Name subnet-1 -AddressPrefix 10.1.1.0/24 -
RouteTable $routeTable

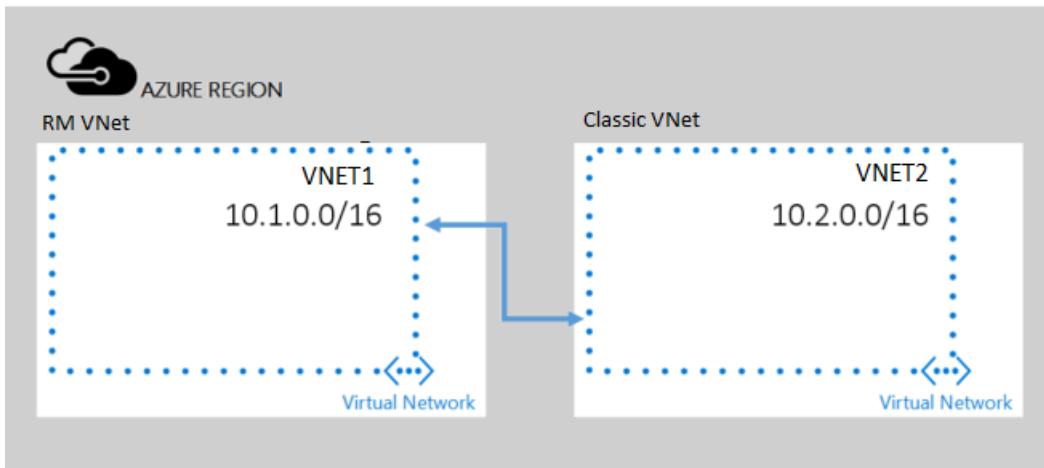
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet1
```

Peering virtual networks from RM to classic

In this scenario, you will create a peering between two VNets, namely **VNET1** and **VNET2** belonging to Azure Resource Manager deployment model and classic deployment model respectively.

NOTE

The virtual networks must be in the same subscription.



To create a VNet peering between a classic virtual network and an Azure Resource Manager virtual network in PowerShell, follow the steps below:

1. Read virtual network object for **VNET1**, the Azure Resource Manager virtual network as follows:

```
$vnet1 = Get-AzureRmVirtualNetwork -ResourceGroupName vnet101 -Name vnet1
```

2. To establish VNet peering in this scenario, only one link is needed, specifically a link from **VNET1** to **VNET2**.

This step requires knowing your classic VNet's resource ID. The resource group ID format looks like:

```
/subscriptions/{SubscriptionID}/resourceGroups/{ResourceGroupName}/providers/Microsoft.ClassicNetwork/virtualNetworks/{VirtualNetworkName}
```

Be sure to replace SubscriptionID, ResourceGroupName, and VirtualNetworkName with the appropriate names.

This can be accomplished by the following:

```
Add-AzureRmVirtualNetworkPeering -Name LinkToVNet2 -VirtualNetwork $vnet1 -RemoteVirtualNetworkId /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Network/virtualNetworks/VNET2
```

3. Once the VNet peering link is created, you can see the link state as shown in the output below:

```
Name : LinkToVNet2
Id   : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.Network/virtualNetworks/VNET1/virtualNetworkPeerings/LinkToVNet2
Etag : W/"acecbd0f-766c-46be-aa7e-d03e41c46b16"
ResourceGroupName : MyResourceGroup
VirtualNetworkName : VNET1
PeeringState      : Connected
ProvisioningState : Succeeded
RemoteVirtualNetwork :
                    {
                      "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx/resourceGroups/MyResourceGroup/providers/Microsoft.ClassicNetwork/virtualNetworks/VNET2"
                    }
AllowVirtualNetworkAccess : True
AllowForwardedTraffic    : False
AllowGatewayTransit      : False
UseRemoteGateways        : False
RemoteGateways          : null
RemoteVirtualNetworkAddressSpace : null
```

Remove VNet Peering

1. In order to remove the VNet peering, you need to run the following cmdlet:

```
Remove-AzureRmVirtualNetworkPeering
```

Remove both links, using the following commands:

```
Remove-AzureRmVirtualNetworkPeering -ResourceGroupName vnet101 -VirtualNetworkName vnet1 -  
Name linktovnet2 Remove-AzureRmVirtualNetworkPeering -ResourceGroupName vnet101 -  
VirtualNetworkName vnet1 -Name linktovnet2
```

2. Once you remove one link in a VNET peering, the peer link state will go to disconnected. In this state, you cannot re-create the link until the peer link state changes to Initiated. We recommend you remove both links before you re-create the VNet peering.

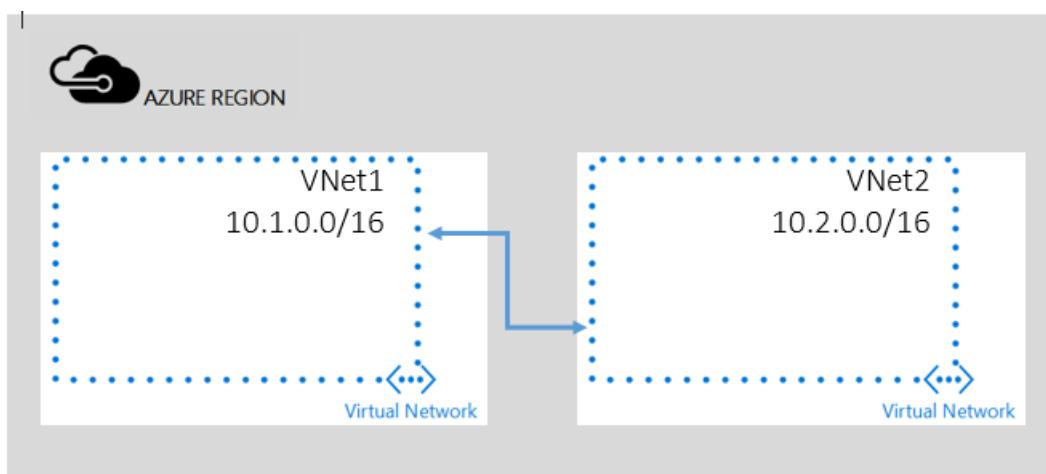
Create VNet Peering using Resource Manager templates

1/17/2017 • 8 min to read • [Edit on GitHub](#)

VNet Peering is a mechanism to connect two Virtual Networks in the same region through the Azure backbone network. Once peered, the two Virtual Networks will appear like a single Virtual Network for all connectivity purposes. Read the [VNet Peering overview](#) if you are not familiar with VNet Peering.

Peering VNets in the same subscription

In this scenario you will create a peering between two VNets named **VNet1** and **VNet2** belonging to the same subscription.



VNet peering will allow full connectivity between the entire address space of peered virtual networks.

To create a VNet peering by using Resource Manager templates, please follow the steps below:

1. If you have never used Azure PowerShell, see [How to Install and Configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.

NOTE

The PowerShell cmdlet for managing VNet peering is shipped with [Azure PowerShell 1.6](#).

2. The text below shows the definition of a VNet peering link for VNet1 to VNet2, based on the scenario above.
Copy the content below and save it to a file named VNetPeeringVNet1.json.

```
{
"$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": "VNet1/LinkToVNet2",
    "location": "[resourceGroup().location]",
    "properties": {
        "allowVirtualNetworkAccess": true,
        "allowForwardedTraffic": false,
        "allowGatewayTransit": false,
        "useRemoteGateways": false,
        "remoteVirtualNetwork": {
            "id": "[resourceId('Microsoft.Network/virtualNetworks', 'vnet2')]"
        }
    }
}
]
}
```

3. The section below shows the definition of a VNet peering link for VNet2 to VNet1, based on the scenario above. Copy the content below and save it to a file named VNetPeeringVNet2.json.

```
{
"$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": "VNet2/LinkToVNet1",
    "location": "[resourceGroup().location]",
    "properties": {
        "allowVirtualNetworkAccess": true,
        "allowForwardedTraffic": false,
        "allowGatewayTransit": false,
        "useRemoteGateways": false,
        "remoteVirtualNetwork": {
            "id": "[resourceId('Microsoft.Network/virtualNetworks', 'vnet1')]"
        }
    }
}
]
```

As seen in the template above, there are a few configurable properties for VNet peering:

OPTION	DESCRIPTION	DEFAULT
AllowVirtualNetworkAccess	Whether or not the address space of a peer VNet is included as part of the virtual_network tag.	Yes

OPTION	DESCRIPTION	DEFAULT
AllowForwardedTraffic	Whether traffic not originating from a peered VNet is accepted or dropped.	No
AllowGatewayTransit	Allows the peer VNet to use your VNet gateway.	No
UseRemoteGateways	Use your peer's VNet gateway. The peer VNet must have a gateway configured and AllowGatewayTransit selected. You cannot use this option if you have a gateway configured.	No

Each link in VNet peering has the set of properties above. For example, you can set AllowVirtualNetworkAccess to True for VNet peering link VNet1 to VNet2 and set it to False for the VNet peering link in the other direction.

4. To deploy the template file, you can run the New-AzureRmResourceGroupDeployment cmdlet to create or update the deployment. For more information about using Resource Manager templates, please refer to this [article](#).

```
New-AzureRmResourceGroupDeployment -ResourceGroupName <resource group name> -TemplateFile <template file path> -DeploymentLogLevel all
```

NOTE

Please replace the resource group name and template file as appropriate.

Below is an example based on the scenario above:

```
New-AzureRmResourceGroupDeployment -ResourceGroupName VNet101 -TemplateFile .\VNetPeeringVNet1.json -DeploymentLogLevel all
```

Output shows:

```
DeploymentName      : VNetPeeringVNet1
ResourceGroupName  : VNet101
ProvisioningState   : Succeeded
Timestamp          : 7/26/2016 9:05:03 AM
Mode               : Incremental
TemplateLink       :
Parameters         :
Outputs            :
DeploymentLogLevel : RequestContent, ResponseContent
```

```
New-AzureRmResourceGroupDeployment -ResourceGroupName VNet101 -TemplateFile .\VNetPeeringVNet2.json -DeploymentLogLevel all
```

Output shows:

```

DeploymentName      : VNetPeeringVNet2
ResourceGroupName   : VNet101
ProvisioningState   : Succeeded
Timestamp          : 7/26/2016 9:07:22 AM
Mode               : Incremental
TemplateLink       :
Parameters         :
Outputs            :
DeploymentLogLevel : RequestContent, ResponseContent

```

- After the deployment is finished, you can run the cmdlet below to view the peering state:

```
Get-AzureRmVirtualNetworkPeering -VirtualNetworkName VNet1 -ResourceGroupName VNet101 -Name linktoVNet2
```

Output shows:

```

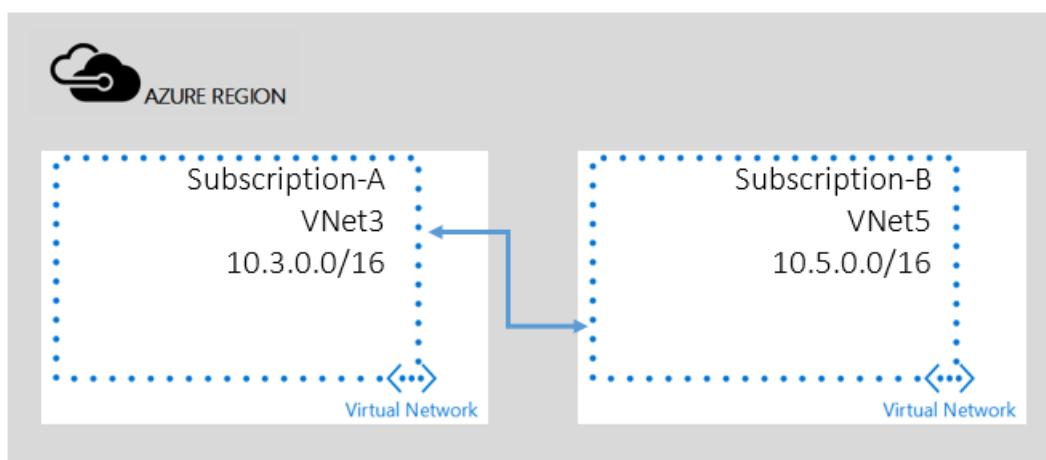
Name      : LinkToVNet2
Id        : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/VNet101/providers/Microsoft.Network/virtualNetworks/VNet1/virtualNetworkPeerings/LinkToVNet2
Etag      : W/"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
ResourceGroupName : VNet101
VirtualNetworkName  : VNet1
ProvisioningState   : Succeeded
RemoteVirtualNetwork : {
    "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/VNet101/providers/Microsoft.Network/virtualNetworks/VNet2"
}
AllowVirtualNetworkAccess : True
AllowForwardedTraffic   : False
AllowGatewayTransit     : False
UseRemoteGateways       : False
RemoteGateways          : null
RemoteVirtualNetworkAddressSpace : null

```

After peering is established in this scenario, you should be able to initiate the connections from any virtual machine to any virtual machine in both VNets. By default, AllowVirtualNetworkAccess is True and VNet peering will provision the proper ACLs to allow the communication between VNets. You can still apply network security group (NSG) rules to block connectivity between specific subnets or virtual machines to gain fine-grain control of access between two virtual networks. For more information of creating NSG rules, please refer to this [article](#).

Peering across subscriptions

In this scenario you will create a peering between two VNets belonging to different subscriptions.



VNet peering relies on role-based access control (RBAC) for authorization. For cross-subscriptions scenario, you first need to grant sufficient permission to users who will create the peering link:

NOTE

If the same user has the privilege over both subscriptions, then you can skip step1-4 below.

To create a VNet peering across subscriptions, please follow the steps below:

1. Sign in to Azure with privileged User-A's account in Subscription-A and run the following cmdlet:

```
New-AzureRmRoleAssignment -SignInName <UserB ID> -RoleDefinitionName "Network Contributor" -Scope /subscriptions/<Subscription-A-ID>/resourceGroups/<ResourceGroupName>/providers/Microsoft.Network/VirtualNetwork/VNet5
```

This is not a requirement, peering can be established even if users individually raise peering requests for their respective Vnets as long as the requests match. Adding a privileged user of the other VNet as users in the local VNet makes it easier to do the setup.

2. Sign in to Azure with privileged User-B's account for Subscription-B and run the following cmdlet:

```
New-AzureRmRoleAssignment -SignInName <UserA ID> -RoleDefinitionName "Network Contributor" -Scope /subscriptions/<Subscription-B-ID>/resourceGroups/<ResourceGroupName>/providers/Microsoft.Network/VirtualNetwork/VNet3
```

3. In User-A's login session, run this cmdlet:

```
New-AzureRmResourceGroupDeployment -ResourceGroupName VNet101 -TemplateFile .\VNetPeeringVNet3.json -DeploymentLogLevel all
```

Here is how the JSON file is defined.

```
{
  "$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": "VNet3/LinkToVNet5",
      "location": "[resourceGroup().location]",
      "properties": {
        "allowVirtualNetworkAccess": true,
        "allowForwardedTraffic": false,
        "allowGatewayTransit": false,
        "useRemoteGateways": false,
        "remoteVirtualNetwork": {
          "id": "/subscriptions/<Subscription-B-ID>/resourceGroups/<resource group name>/providers/Microsoft.Network/virtualNetworks/VNet5"
        }
      }
    }
  ]
}
```

4. In User-B's login session, run the following cmdlet:

```
New-AzureRmResourceGroupDeployment -ResourceGroupName VNet101 -TemplateFile .\VNetPeeringVNet5.json -DeploymentLogLevel all
```

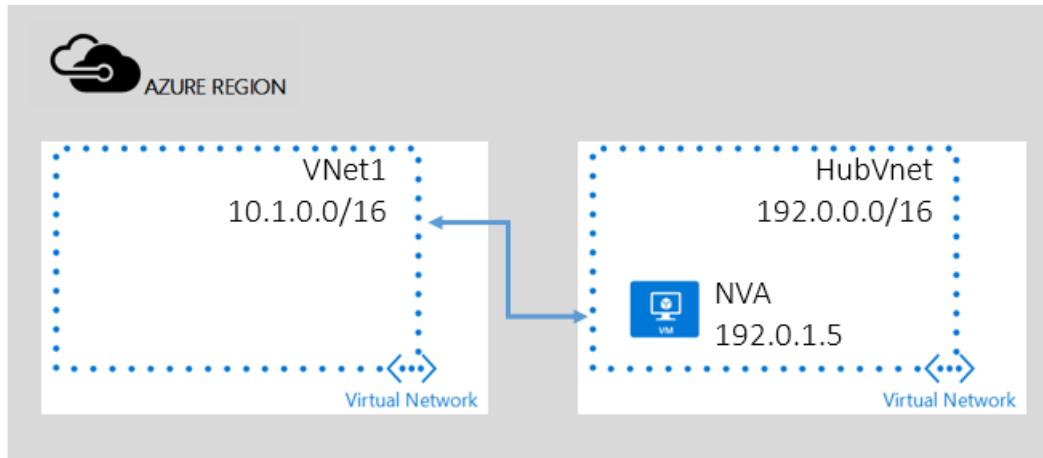
Here is how the JSON file is defined:

```
{
  "$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": "VNet5/LinkToVNet3",
      "location": "[resourceGroup().location]",
      "properties": {
        "allowVirtualNetworkAccess": true,
        "allowForwardedTraffic": false,
        "allowGatewayTransit": false,
        "useRemoteGateways": false,
        "remoteVirtualNetwork": {
          "id": "/subscriptions/Subscription-A-ID /resourceGroups/<resource group name>/providers/Microsoft.Network/virtualNetworks/VNet3"
        }
      }
    }
  ]
}
```

After peering is established in this scenario, you should be able to initiate the connections from any virtual machine to any virtual machine of both VNets across different subscriptions.

Service Chaining - Transit through peered VNet

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. In this scenario, there are two VNets in a subscription, HubVNet and VNet1 as described in the diagram below. You deploy Network Virtual Appliance(NVA) in VNet HubVNet. After establishing VNet peering between HubVNet and VNet1, you can set up User Defined Routes and specify the next hop to NVA in the HubVNet.



NOTE

For the simplicity, assume all VNets here are in the same subscription. But it also works for cross-subscription scenario.

The key property to enable Transit routing is the "Allow Forwarded Traffic" parameter. This allows accepting and sending traffic from/to the NVA in the peered VNet.

1. In this scenario, you can deploy the sample template below to establish the VNet peering. You'll need to set the AllowForwardedTraffic property to True, which allows the network virtual appliance in the peered VNet to send and receive traffic.

Here is the template for creating a VNet peering from HubVNet to VNet1. Note that AllowForwardedTraffic is set to false.

```
{  
    "$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": "HubVNet/LinkToVNet1",  
            "location": "[resourceGroup().location]",  
            "properties": {  
                "allowVirtualNetworkAccess": true,  
                "allowForwardedTraffic": false,  
                "allowGatewayTransit": false,  
                "useRemoteGateways": false,  
                "remoteVirtualNetwork": {  
                    "id": "[resourceId('Microsoft.Network/virtualNetworks', 'vnet1')]"  
                }  
            }  
        }  
    ]  
}
```

2. Here is the template for creating a VNet peering from VNet1 to HubVnet. Note that AllowForwardedTraffic is set to true.

```
{
"$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": "VNet1/LinkToHubVNet",
"location": "[resourceGroup().location]",
"properties": {
"allowVirtualNetworkAccess": true,
"allowForwardedTraffic": true,
"allowGatewayTransit": false,
"useRemoteGateways": false,
"remoteVirtualNetwork": {
"id": "[resourceId('Microsoft.Network/virtualNetworks', 'HubVnet')]"
}
}
}
]
}
```

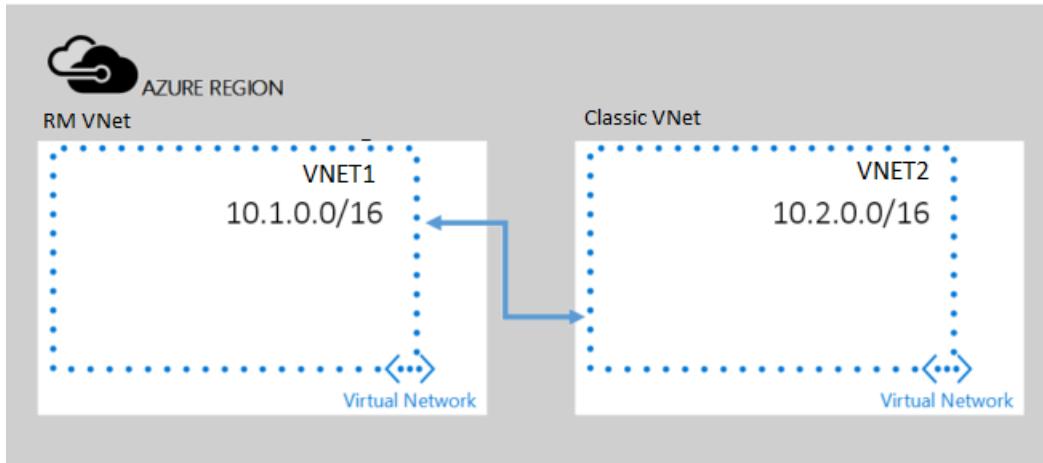
- After peering is established, you can refer to this [article](#) to define user-defined routes(UDR) to redirect VNet1 traffic through a virtual appliance to use its capabilities. When you specify the next hop address in route, you can set it to the IP address of the virtual appliance in the peer VNet HubVNet.

Peering virtual networks from RM to classic

In this scenario, you will create a peering between two VNets, namely **VNET1** and **VNET2** belonging to Azure Resource Manager deployment model and classic deployment model respectively.

NOTE

The virtual networks must be in the same subscription.



To create a peering between virtual networks from different deployment models, follow the steps below:

- The text below shows the definition of a VNet peering link for VNET1 to VNET2 in this scenario. Only one link is required to peer a classic virtual network to a Azure resource manager virtual network.

Be sure to put in your subscription ID for where the classic virtual network or VNET2 is located and change MyResourceGroup to the appropriate resource group name.

```
{
  "$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": "VNET1/LinkToVNET2",
      "location": "[resourceGroup().location]",
      "properties": {
        "allowVirtualNetworkAccess": true,
        "allowForwardedTraffic": false,
        "allowGatewayTransit": false,
        "useRemoteGateways": false,
        "remoteVirtualNetwork": {
          "id": "[resourceId('Microsoft.ClassicNetwork/virtualNetworks', 'VNET2')]"
        }
      }
    }
  ]
}
```

2. To deploy the template file, run the following cmdlet to create or update the deployment.

```
New-AzureRmResourceGroupDeployment -ResourceGroupName MyResourceGroup -TemplateFile .\VnetPeering.json -DeploymentDebugLogLevel all

Output shows:

DeploymentName      : VnetPeering
ResourceGroupName   : MyResourceGroup
ProvisioningState   : Succeeded
Timestamp           : XX/XX/YYYY 5:42:33 PM
Mode                : Incremental
TemplateLink        :
Parameters          :
Outputs             :
DeploymentDebugLogLevel : RequestContent, ResponseContent
```

3. After the deployment succeeds, you can run the following cmdlet to view the peering state:

```
Get-AzureRmVirtualNetworkPeering -VirtualNetworkName VNET1 -ResourceGroupName MyResourceGroup -Name LinkToVNET2
```

Output shows:

```
Name : LinkToVNET2
Id   : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxx/resourceGroups/MyResource
Group/providers/Microsoft.Network/virtualNetworks/VNET1/virtualNetworkPeering
s/LinkToVNET2
Etag   : W/"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
ResourceGroupName : MyResourceGroup
VirtualNetworkName : VNET1
PeeringState       : Connected
ProvisioningState  : Succeeded
RemoteVirtualNetwork : {
    "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/M
yResourceGroup/providers/Microsoft.ClassicNetwork/virtualNetworks/VNET2"
}
AllowVirtualNetworkAccess : True
AllowForwardedTraffic   : False
AllowGatewayTransit     : False
UseRemoteGateways       : False
RemoteGateways          : null
RemoteVirtualNetworkAddressSpace : null
```

After peering is established between a classic VNet and a resource manager VNet, you should be able to initiate connections from any virtual machine in VNET1 to any virtual machine in VNET2 and vice versa.

Create a VM with a static public IP using the Azure portal

1/17/2017 • 2 min to read • [Edit on GitHub](#)

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. 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.

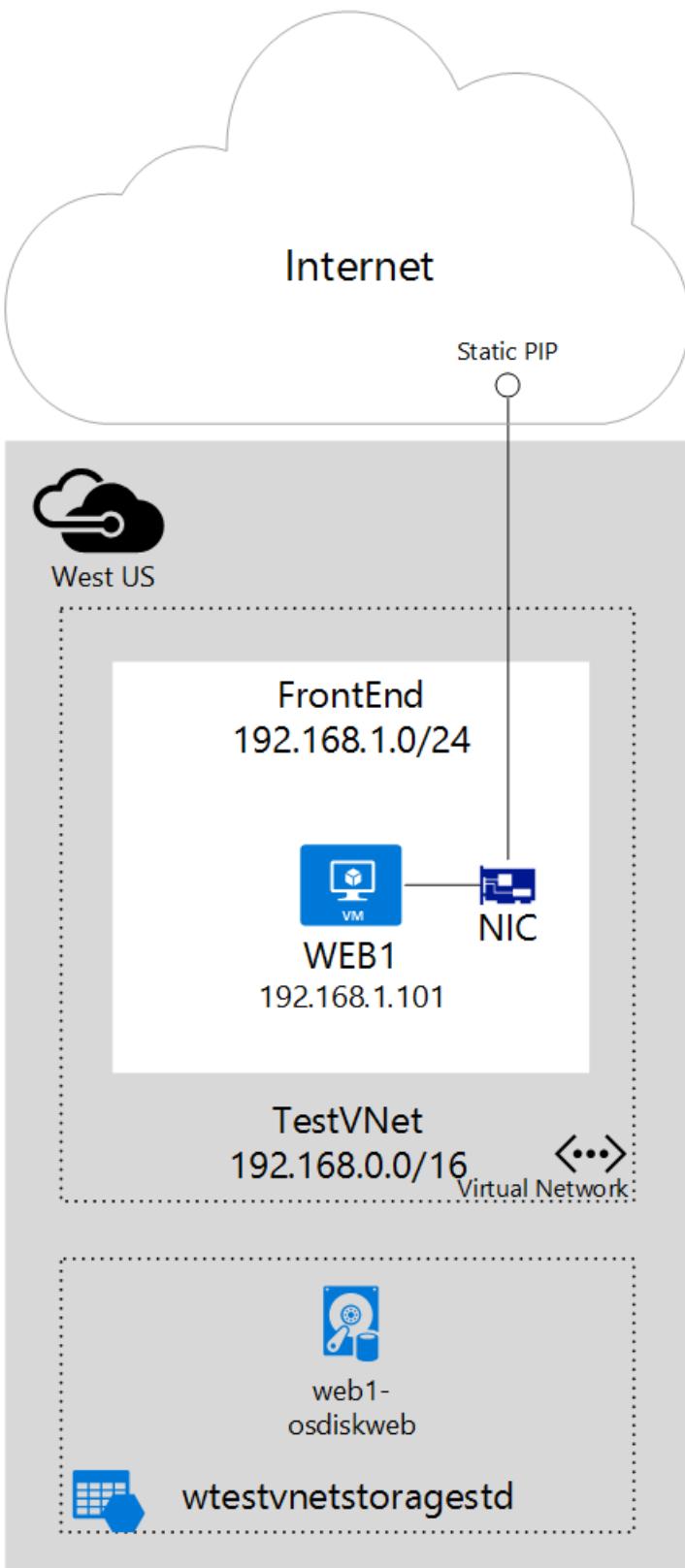
NOTE

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 **New** > **Compute** > **Windows Server 2012 R2 Datacenter**.
3. In the **Select a deployment model** list, select **Resource Manager** and click **Create**.
4. In the **Basics** blade, enter the VM information as shown below, and then click **OK**.

* Name
WEB1 ✓

* User name
adminUser ✓

* Password
***** ✓

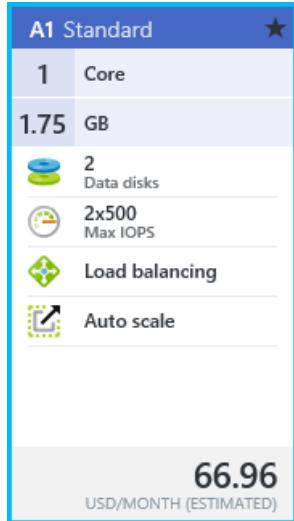
* Subscription
Microsoft Azure Internal Consumption

* Resource group
PublicIPTest ✗ ✓

Select existing

* Location
East US 2

5. In the **Choose a size** blade, click **A1 Standard** as shown below, and then click **Select**.



6. In the **Settings** blade, click **Public IP address**, then in the **Create public IP address** blade, under **Assignment**, click **Static** as shown below. And then click **OK**.

* Name
WEB1 ✓

Assignment
Dynamic Static

7. In the **Settings** blade, click **OK**.
8. Review the **Summary** blade, as shown below, 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** blade will be displayed as shown below

WEB1
Virtual machine

Settings Connect Start Restart Stop Delete

Essentials

Resource group: PublicIPTest

Status: Running

Location: East US 2

Subscription name: Microsoft Azure Internal Consumption

Subscription ID: 628dad04-b5d1-4f10-b3a4-dc61d88cf97c

Computer name: WEB1

Size: Standard A1 (1 core, 1.75 GB memory)

Operating system: Windows

Public IP address/DNS name label: 104.208.232.131/<none>

Virtual network/subnet: PublicIPTest/default

All settings →

Monitoring

CPU percentage

100%

80%

60%

40%

20%

0%

11 AM 11:15 AM 11:30 AM 11:45 AM

CPU UTILIZATION
0.08 %

Add tiles +

Add a group +

Settings

WEB1

INVESTIGATE

- Audit logs
- Boot diagnostics
- Reset password
- Troubleshoot
- New support request

MANAGE

- Properties
- Disks
- Network interfaces
- Availability set
- Extensions
- Size

MONITOR

- Alert rules
- Diagnostics

RESOURCE MANAGEMENT

- Users
- Tags

Create a VM with a static public IP using PowerShell

1/17/2017 • 4 min to read • [Edit on GitHub](#)

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. 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.

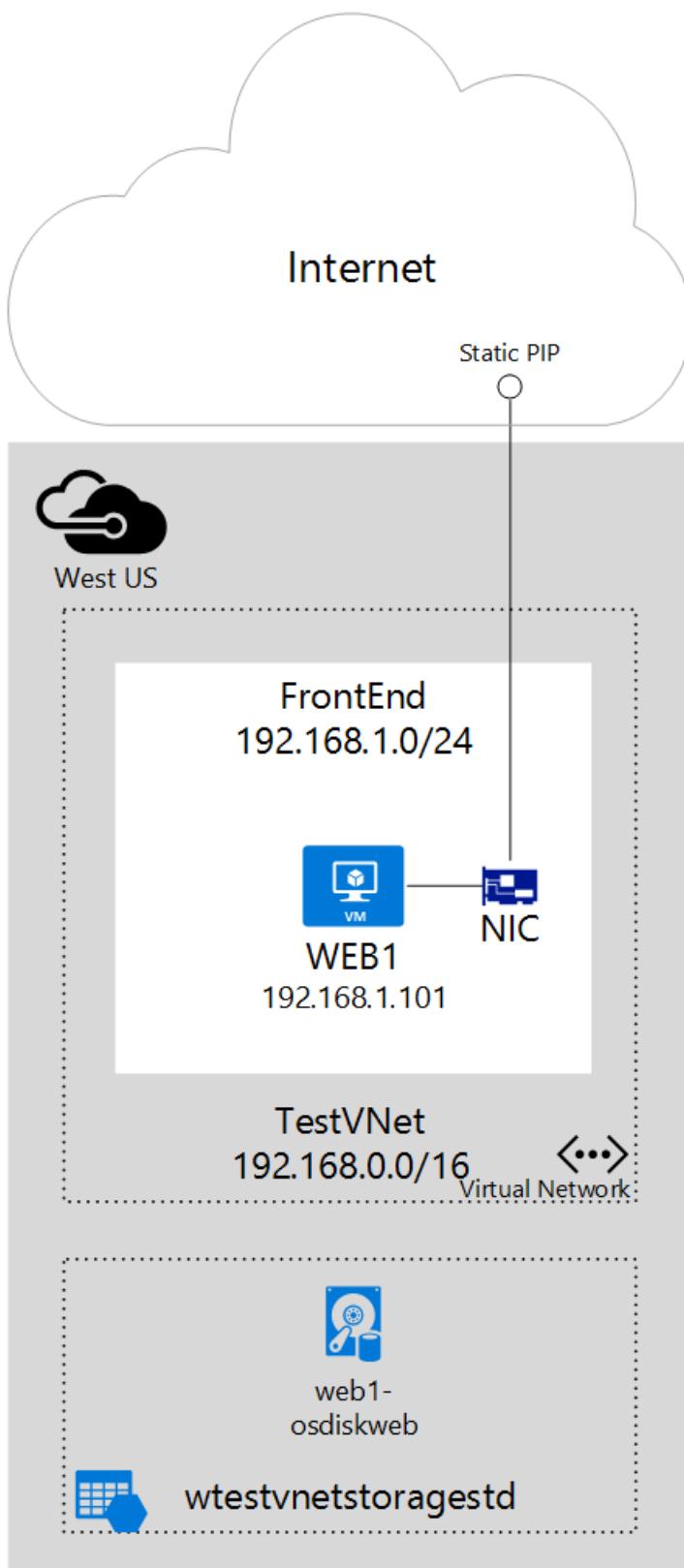
NOTE

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'll need to [to install and configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.

NOTE

If you don't have an Azure account, you'll need one. Go sign up for a [free trial here](#).

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.

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"
```

Step 2 - 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
```

Step 3 - 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 + ".vhd"  
$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.

Step 4 - Run the script

After making any necessary changes, and understanding the script show above, run the script.

1. From a PowerShell console, or PowerShell ISE, run the script above.

2. The following output should be displayed after a few minutes:

```
ResourceGroupName : IaaSStory
Location         : westus
ProvisioningState : Succeeded
Tags             :
ResourceId       : /subscriptions/[Subscription ID]/resourceGroups/IaaSStory

AddressSpace      : Microsoft.Azure.Commands.Network.Models.PSAddressSpace
DhcpOptions      : Microsoft.Azure.Commands.Network.Models.PSDhcpOptions
Subnets          : {FrontEnd}
ProvisioningState : Succeeded
AddressSpaceText  : {
    "AddressPrefixes": [
        "192.168.0.0/16"
    ]
}
DhcpOptionsText  : {}
SubnetsText      : [
    {
        "Name": "FrontEnd",
        "AddressPrefix": "192.168.1.0/24"
    }
]
ResourceGroupName : IaaSStory
Location         : westus
ResourceGuid     : [Id]
Tag              :
TagsTable        :
Name             : WTestVNet
Etag             : W/"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
Id               : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/WTestVNet

AddressSpace      : Microsoft.Azure.Commands.Network.Models.PSAddressSpace
DhcpOptions      : Microsoft.Azure.Commands.Network.Models.PSDhcpOptions
Subnets          : {FrontEnd}
ProvisioningState : Succeeded
AddressSpaceText  : {
    "AddressPrefixes": [
        "192.168.0.0/16"
    ]
}
DhcpOptionsText  : {
    "DnsServers": []
}
SubnetsText      : [
    {
        "Name": "FrontEnd",
        "Etag": [Id],
        "Id": "/subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/WTestVNet/subnets/FrontEnd",
        "AddressPrefix": "192.168.1.0/24",
        "IpConfigurations": [],
        "ProvisioningState": "Succeeded"
    }
]
ResourceGroupName : IaaSStory
Location         : westus
ResourceGuid     : [Id]
Tag              :
TagsTable        :
Name             : WTestVNet
Etag             : [Id]
Id               : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/WTestVNet

TrackingOperationId : [Id]
```

RequestId	:	[Id]
Status	:	Succeeded
StatusCode	:	OK
Output	:	
StartTime	:	[Subscription Id]
EndTime	:	[Subscription Id]
Error	:	
ErrorText	:	

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

1/17/2017 • 5 min to read • [Edit on GitHub](#)

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. 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.

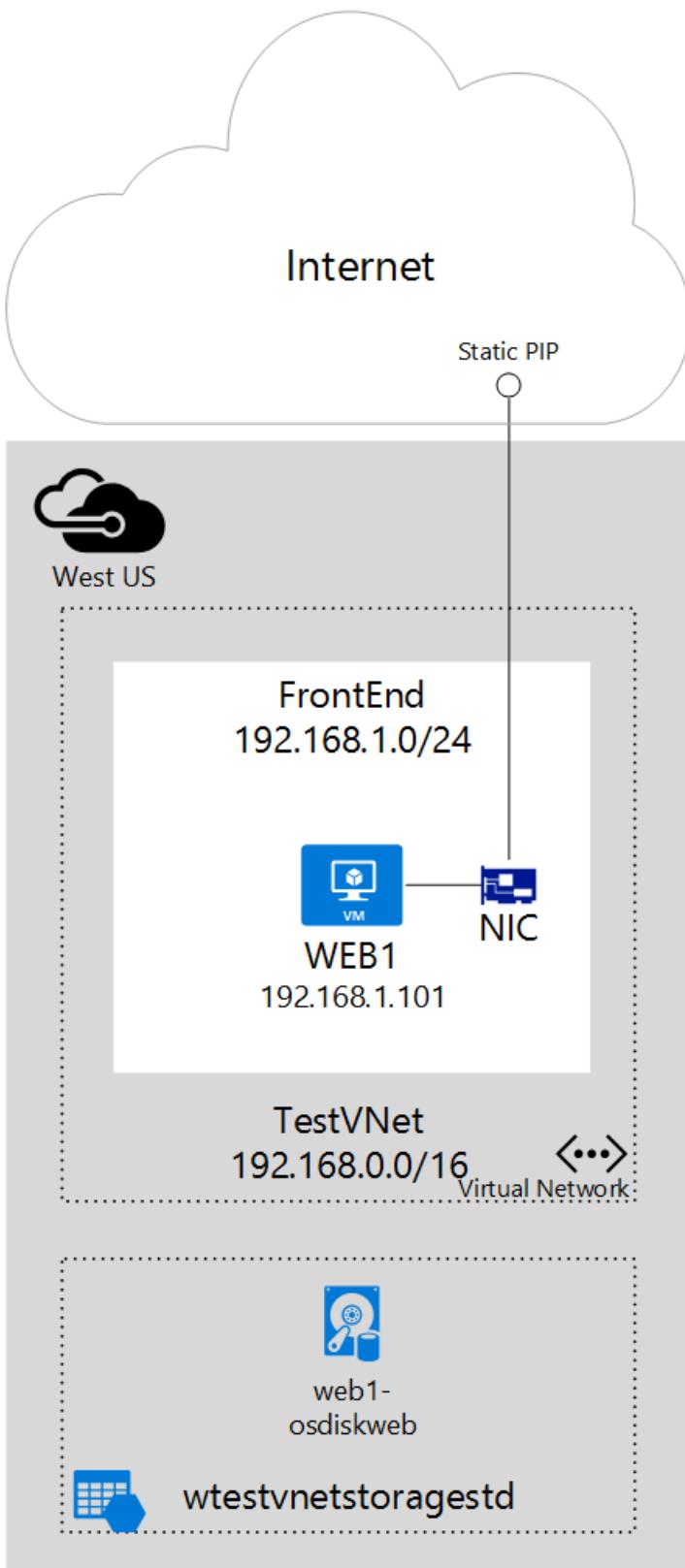
NOTE

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 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.

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:

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 for the new resource group
rgName="IaaSStory"
location="westus"

# Set variables for VNet
vnetName="TestVNet"
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="Canonical"
offer="UbuntuServer"
sku="14.04.2-LTS"
version="latest"
vmName="WEB1"
osDiskName="osdisk"
nicName="NICWEB1"
privateIPAddress="192.168.1.101"
username='adminuser'
password='adminP@ssw0rd'
pipName="PIPWEB1"
dnsName="iaasstoryws1"
```

Step 2 - 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.

```
azure group create $rgName $location
```

2. Create the VNet and subnet.

```
azure network vnet create --resource-group $rgName \
    --name $vnetName \
    --address-prefixes $vnetPrefix \
    --location $location
azure network vnet subnet create --resource-group $rgName \
    --vnet-name $vnetName \
    --name $subnetName \
    --address-prefix $subnetPrefix
```

3. Create the public IP resource.

```
azure network public-ip create --resource-group $rgName \
    --name $pipName \
    --location $location \
    --allocation-method Static \
    --domain-name-label $dnsName
```

4. Create the network interface (NIC) for the VM in the subnet created above, with the public IP. Notice the first set of commands are used to retrieve the **Id** of the subnet created above.

```
subnetId=$(azure network vnet subnet show --resource-group $rgName \
    --vnet-name $vnetName \
    --name $subnetName|grep Id)"
subnetId=${subnetId#*/}

azure network nic create --name $nicName \
    --resource-group $rgName \
    --location $location \
    --private-ip-address $privateIPAddress \
    --subnet-id $subnetId \
    --public-ip-name $pipName
```

TIP

The first command above uses [grep](#) and [string manipulation](#) (more specifically, substring removal).

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

```
azure storage account create $stdStorageAccountName \
    --resource-group $rgName \
    --location $location --type LRS
```

Step 3 - Create the VM

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

1. Create the VM.

```
azure vm create --resource-group $rgName \
  --name $vmName \
  --location $location \
  --vm-size $vmSize \
  --subnet-id $subnetId \
  --nic-names $nicName \
  --os-type linux \
  --image-urn $publisher:$offer:$sku:$version \
  --storage-account-name $stdStorageAccountName \
  --storage-account-container-name vhds \
  --os-disk-vhd $osDiskName.vhd \
  --admin-username $username \
  --admin-password $password
```

2. Save the script file.

Step 4 - Run the script

After making any necessary changes, and understanding the script show above, run the script.

1. From a bash console, run the script above.

```
sh myscript.sh
```

2. The output below should be displayed after a few minutes.

```
info:  Executing command group create
info:  Getting resource group IaaSStory
info:  Creating resource group IaaSStory
info:  Created resource group IaaSStory
data:  Id:          /subscriptions/[Subscription ID]/resourceGroups/IaaSStory
data:  Name:        IaaSStory
data:  Location:   westus
data:  Provisioning State: Succeeded
data:  Tags: null
data:
info:  group create command OK
info:  Executing command network vnet create
info:  Looking up virtual network "TestVNet"
info:  Creating virtual network "TestVNet"
info:  Loading virtual network state
data:  Id           : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/TestVNet
data:  Name         : TestVNet
data:  Type         : Microsoft.Network/virtualNetworks
data:  Location     : westus
data:  ProvisioningState : Succeeded
data:  Address prefixes:
data:    192.168.0.0/16
info:  network vnet create command OK
info:  Executing command network vnet subnet create
info:  Looking up the subnet "FrontEnd"
info:  Creating subnet "FrontEnd"
info:  Looking up the subnet "FrontEnd"
data:  Id           : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data:  Type         : Microsoft.Network/virtualNetworks/subnets
data:  ProvisioningState : Succeeded
data:  Name         : FrontEnd
data:  Address prefix       : 192.168.1.0/24
data:
info:  network vnet subnet create command OK
info:  Executing command network public-ip create
info:  Looking up the public ip "PIPWEB1"
info:  Creating public ip address "PIPWEB1"
```

```
info: Creating public ip address PIPWEB1
info: Looking up the public ip "PIPWEB1"
data: Id : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/publicIPAddresses/PIPWEB1
data: Name : PIPWEB1
data: Type : Microsoft.Network/publicIPAddresses
data: Location : westus
data: Provisioning state : Succeeded
data: Allocation method : Static
data: Idle timeout : 4
data: IP Address : 40.78.63.253
data: Domain name label : iaasstoryws1
data: FQDN : iaasstoryws1.westus.cloudapp.azure.com
info: network public-ip create command OK
info: Executing command network nic create
info: Looking up the network interface "NICWEB1"
info: Looking up the public ip "PIPWEB1"
info: Creating network interface "NICWEB1"
info: Looking up the network interface "NICWEB1"
data: Id : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/networkInterfaces/NICWEB1
data: Name : NICWEB1
data: Type : Microsoft.Network/networkInterfaces
data: Location : westus
data: Provisioning state : Succeeded
data: Enable IP forwarding : false
data: IP configurations:
data: Name : NIC-config
data: Provisioning state : Succeeded
data: Public IP address : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/publicIPAddresses/PIPWEB1
data: Private IP address : 192.168.1.101
data: Private IP Allocation Method : Static
data: Subnet : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory2/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data:
info: network nic 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 the VM "WEB1"
info: Using the VM Size "Standard_A1"
info: The [OS, Data] Disk or image configuration requires storage account
info: Looking up the storage account iaasstorystorage
info: Looking up the NIC "NICWEB1"
info: Creating VM "WEB1"
info: vm create command OK
```

Create a VM with a static public IP using a template

1/17/2017 • 4 min to read • [Edit on GitHub](#)

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. 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.

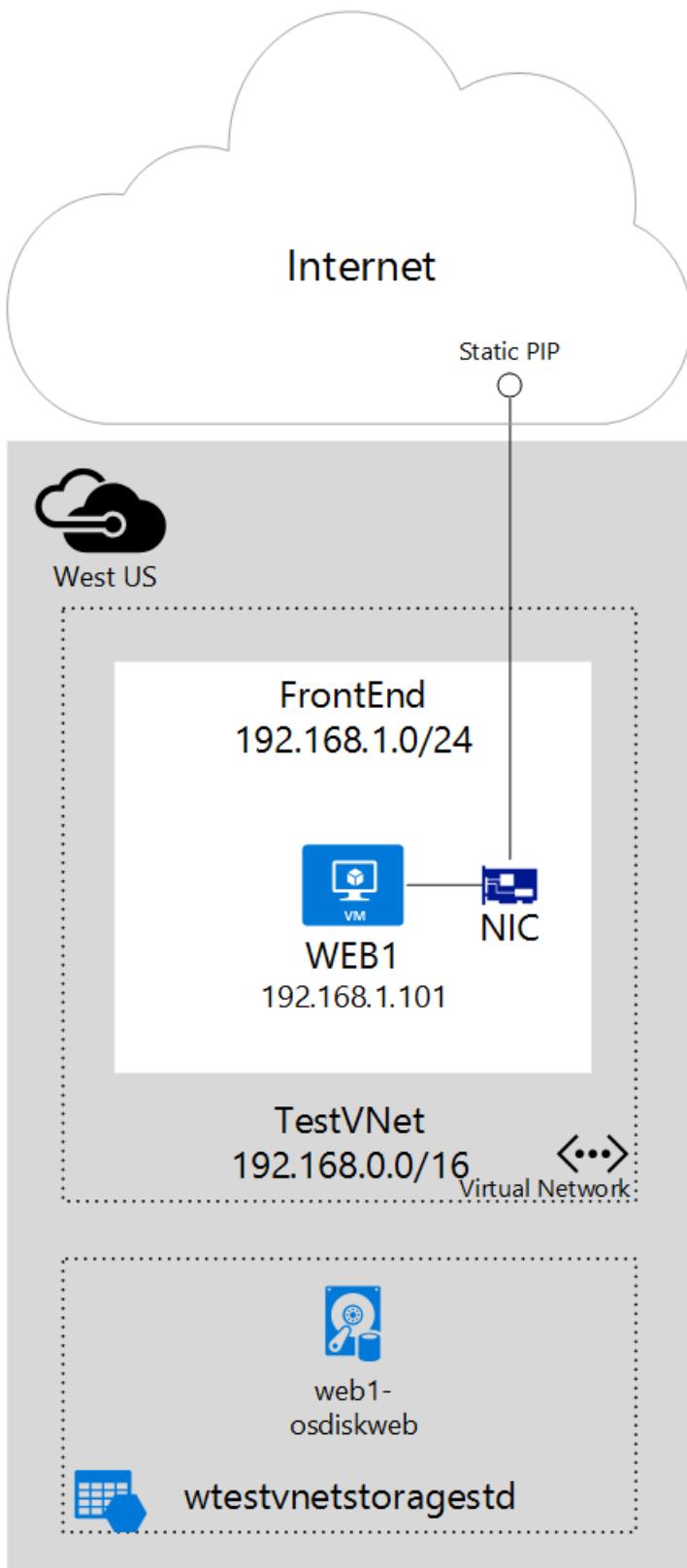
NOTE

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.

Public IP address resources in a template file

You can view and download the [sample template](#).

The following section shows the definition of the public IP resource, based on the scenario above:

```
{
  "apiVersion": "2015-06-15",
  "type": "Microsoft.Network/publicIPAddresses",
  "name": "[variables('webVMSetting').pipName]",
  "location": "[variables('location')]",
  "properties": {
    "publicIPAllocationMethod": "Static"
  },
  "tags": {
    "displayName": "PublicIPAddress - Web"
  }
},
```

Notice the **publicIPAllocationMethod** property, which is set to *Static*. This property can be either *Dynamic* (default value) or *Static*. Setting it to static guarantees that the public IP address assigned will never change.

The following section shows the association of the public IP address with a network interface:

```
{
  "apiVersion": "2015-06-15",
  "type": "Microsoft.Network/networkInterfaces",
  "name": "[variables('webVMSetting').nicName]",
  "location": "[variables('location')]",
  "tags": {
    "displayName": "NetworkInterface - Web"
  },
  "dependsOn": [
    "[concat('Microsoft.Network/publicIPAddresses/', variables('webVMSetting').pipName)]",
    "[concat('Microsoft.Network/virtualNetworks/', parameters('vnetName'))]"
  ],
  "properties": {
    "ipConfigurations": [
      {
        "name": "ipconfig1",
        "properties": {
          "privateIPAllocationMethod": "Static",
          "privateIPAddress": "[variables('webVMSetting').ipAddress]",
          "publicIPAddress": {
            "id": "[resourceId('Microsoft.Network/publicIPAddresses', variables('webVMSetting').pipName)]"
          },
          "subnet": {
            "id": "[variables('frontEndSubnetRef')]"
          }
        }
      }
    ]
  }
},
```

Notice the **publicIPAddress** property pointing to the **Id** of a resource named **variables('webVMSetting').pipName**. That is the name of the public IP resource shown above.

Finally, the network interface above is listed in the **networkProfile** property of the VM being created.

```
"networkProfile": {
  "networkInterfaces": [
    {
      "id": "[resourceId('Microsoft.Network/networkInterfaces', variables('webVMSetting').nicName)]"
    }
  ]
}
```

Deploy the template by using click to deploy

The sample template available in the public repository uses a parameter file containing the default values used to generate the scenario described above. To deploy this template using click to deploy, click **Deploy to Azure** in the Readme.md file for the [VM with static PIP](#) template. Replace the default parameter values if desired and enter values for the blank parameters. Follow the instructions in the portal to create a virtual machine with a static public IP address.

Deploy the template by using PowerShell

To deploy the template you downloaded by using PowerShell, follow the steps below.

1. If you have never used Azure PowerShell, complete the steps in the [How to Install and Configure Azure PowerShell](#) article.
2. In a PowerShell console, run the `New-AzureRmResourceGroup` cmdlet to create a new resource group, if necessary. If you already have a resource group created, go to step 3.

```
New-AzureRmResourceGroup -Name PIPTEST -Location westus
```

Expected output:

```
ResourceGroupName : PIPTEST
Location         : westus
ProvisioningState : Succeeded
Tags             :
ResourceId       : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/StaticPublicIP
```

3. In a PowerShell console, run the `New-AzureRmResourceGroupDeployment` cmdlet to deploy the template.

```
New-AzureRmResourceGroupDeployment -Name DeployVM -ResourceGroupName PIPTEST ` 
    -TemplateUri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/IaaS-` 
        -Story/03-Static-public-IP/azuredeploy.json ` 
            -TemplateParameterUri https://raw.githubusercontent.com/Azure/azure-quickstart-` 
                -templates/master/IaaS-Story/03-Static-public-IP/azuredeploy.parameters.json
```

Expected output:

```

DeploymentName      : DeployVM
ResourceGroupName  : PIPTEST
ProvisioningState   : Succeeded
Timestamp          : [Date and time]
Mode                : Incremental
TemplateLink       :
    Uri           : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/IaaS-Story/03-Static-public-IP/azuredeploy.json
ContentVersion     : 1.0.0.0

Parameters         :
    Name          Type          Value
    =====        =====        =====
    vnetName      String        WTestVNet
    vnetPrefix    String        192.168.0.0/16
    frontEndSubnetName String        FrontEnd
    frontEndSubnetPrefix String        192.168.1.0/24
    storageAccountNamePrefix String        iaasestd
    stdStorageType String        Standard_LRS
    osType         String        Windows
    adminUsername String        adminUser
    adminPassword SecureString
Outputs            :

```

Deploy the template by using the Azure CLI

To deploy the template by using the Azure CLI, complete the following steps:

1. If you have never used Azure CLI, follow the steps in the [Install and Configure the Azure CLI](#) article to install and configure it.
2. Run the `azure config mode` command to switch to Resource Manager mode, as shown below.

```
azure config mode arm
```

The expected output for the command above:

```
info: New mode is arm
```

3. Open the [parameter file](#), select its content, and save it to a file in your computer. For this example, the parameters are saved to a file named *parameters.json*. Change the parameter values within the file if desired, but at a minimum, it's recommended that you change the value for the *adminPassword* parameter to a unique, complex password.
4. Run the `azure group deployment create` cmd to deploy the new VNet by using the template and parameter files you downloaded and modified above. In the command below, replace with the path you saved the file to.

```
azure group create -n PIPTEST2 -l westus --template-uri https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/IaaS-Story/03-Static-public-IP/azuredeploy.json -e <path>\parameters.json
```

Expected output (lists parameter values used):

```
info: Executing command group create
+ Getting resource group PIPTEST2
+ Creating resource group PIPTEST2
info: Created resource group PIPTEST2
+ Initializing template configurations and parameters
+ Creating a deployment
info: Created template deployment "azuredeploy"
data: Id: /subscriptions/[Subscription ID]/resourceGroups/PIPTEST2
data: Name: PIPTEST2
data: Location: westus
data: Provisioning State: Succeeded
data: Tags: null
data:
info: group create command OK
```

Reserved IP addresses (Classic)

1/17/2017 • 5 min to read • [Edit on GitHub](#)

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 shutdown or 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 will be the same even as resources are shutdown or 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 will not be 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 will not change even when VMs in the cloud service are stop or 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 will know the source IP address and won't have to update your firewall rules due to an IP change.

FAQ

1. Can I use a reserved IP for all Azure services?
 - 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?
 - See the [Azure limits](#) article.
3. Is there a charge for reserved IPs?
 - See [Reserved IP Address Pricing Details](#) for pricing information.
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 a particular region. This reserved IP address is associated to your subscription.
5. Can I use this with affinity group based VNets?
 - Reserved IPs are only supported in regional VNets. It is not supported for VNets that are associated with affinity groups. For more information about associating a VNet with a region or an affinity group, see [About Regional VNets and Affinity Groups](#).

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
```

Once an IP is reserved, it remains associated to your subscription until you delete it. To delete the reserved IP shown above, 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 script below 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'll still need to 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 a ILPIP) that is assigned directly to your VM. It cannot be reserved. See [Instance-level Public IP \(ILPIP\)](#) for more information.

Remove a reserved IP from a running deployment

To remove the reserved IP added to the new service created in the script above, 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 new cloud service named *TestService2* with a new VM named *TestVM2*, and then associates the existing reserved IP named *MyReservedIP* 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 sample xml below 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 private IP address in the Azure portal

1/17/2017 • 4 min to read • [Edit on GitHub](#)

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.

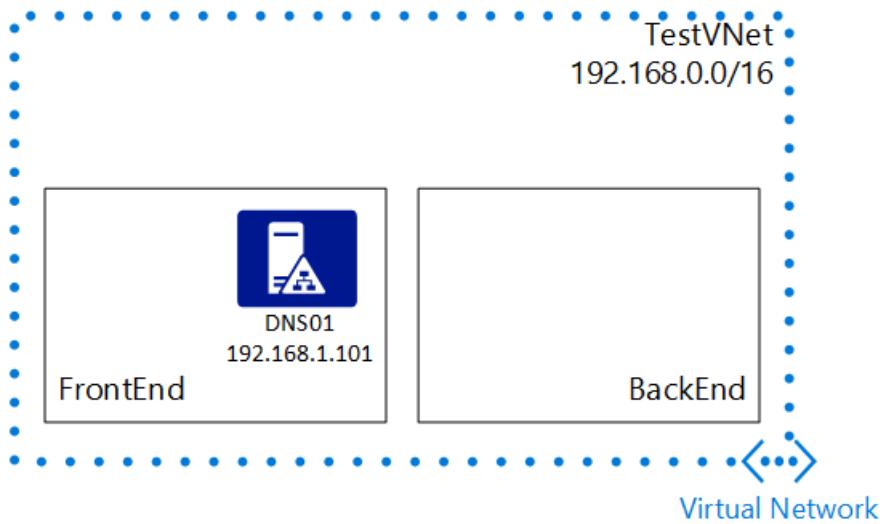
IMPORTANT

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**.

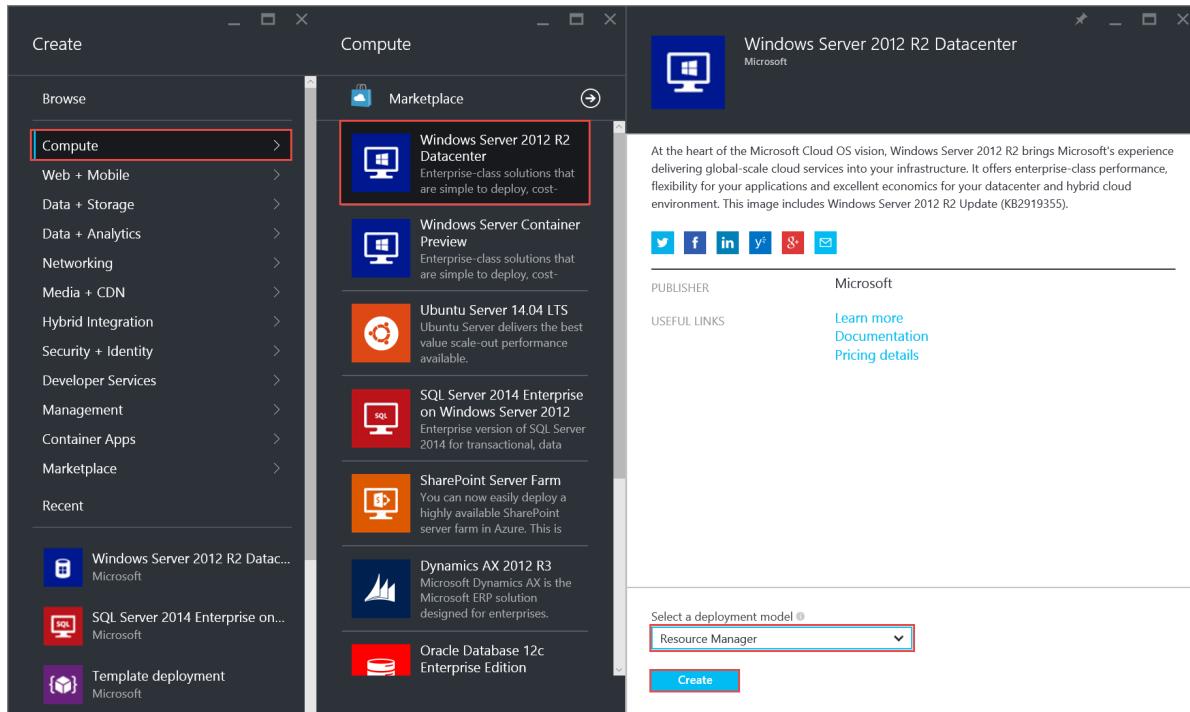
The sample steps below expect a simple environment already created. If you want to run the steps as they are displayed in this document, first build the test environment described in [create a vnet](#).

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 the steps below.

1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Click **NEW > 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 figure below.

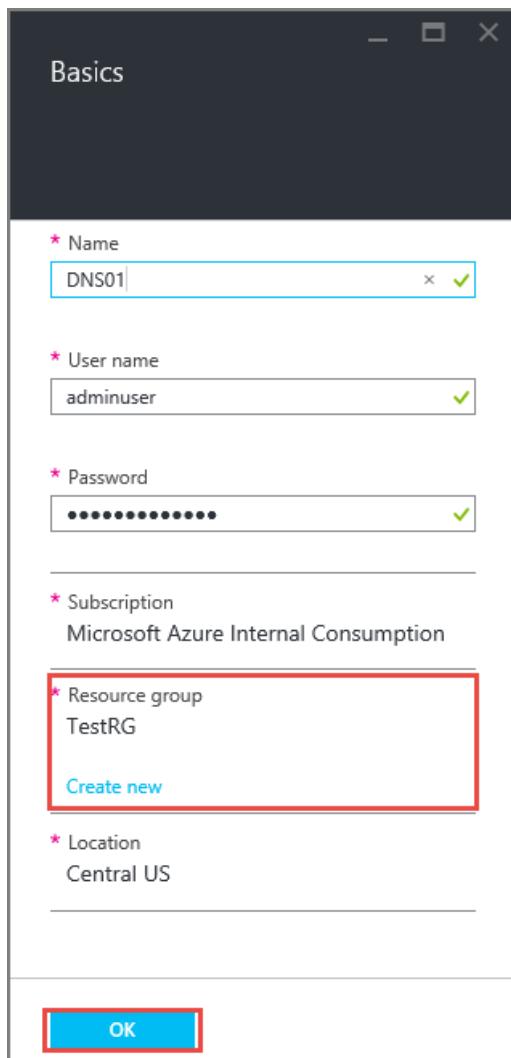


3. In the **Basics** blade, enter the name of the VM to be created (*DNS01* in our scenario), the local administrator account, and password, as seen in the figure below.

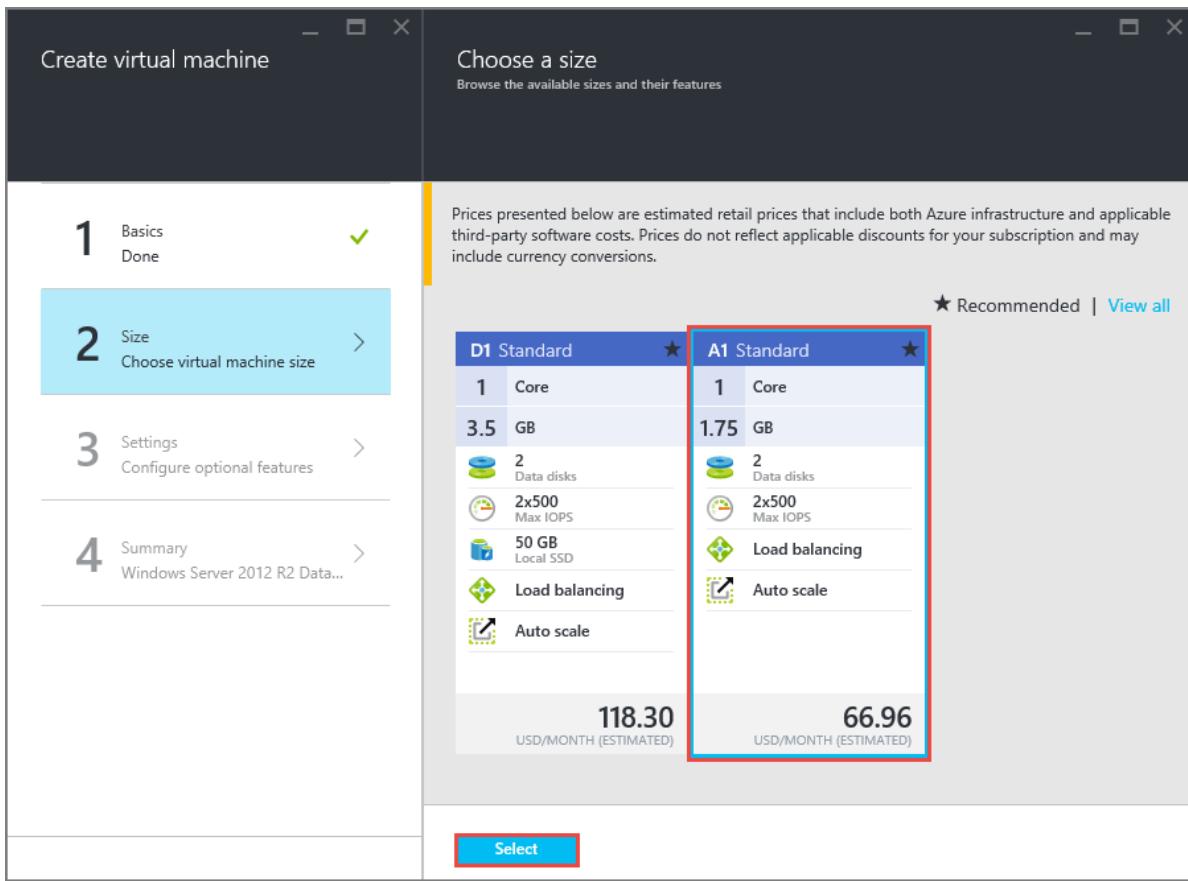
The screenshot shows the 'Basics' blade for creating a VM. It includes fields for:

- Name:** DNS01
- User name:** adminuser
- Password:** (Redacted)
- Subscription:** Microsoft Azure Internal Consumption
- Resource group:** (Empty field with 'Select existing' link)
- Location:** Central US

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** blade, select **A1 Standard**, and then click **Select**.



6. In the **Settings** blade, make sure the following properties are set with the values below, and then click **OK**.

-**Storage account:** *vnetstorage*

- **Network:** *TestVNet*
- **Subnet:** *FrontEnd*

The screenshot shows two windows side-by-side. The left window is titled 'Create virtual machine' and displays a step-by-step process:

- Step 1: Basics** (Done) - Status: ✓
- Step 2: Size** (Done) - Status: ✓
- Step 3: Settings** (Configure optional features) - Status: > (highlighted in blue)
- Step 4: Summary** (Windows Server 2012 R2 Data...) - Status: >

The right window is titled 'Settings' and shows configuration options:

- Storage**: Disk type: Standard (selected), Premium (SSD) (disabled)
- Network**: Virtual network: TestVNet, Subnet: FrontEnd (192.168.1.0/24) (both highlighted with red boxes)
- Monitoring**: Diagnostics: Enabled (selected), Disabled (disabled)
- Availability**: Availability set: None

A large blue 'OK' button is at the bottom of the settings blade.

7. In the **Summary** blade, click **OK**. Notice the tile below displayed in your dashboard.



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 steps below.

- 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

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

dns01995
Network interface

Settings Delete Change security...

Essentials

- Resource group: TestRG
- Location: Central US
- Subscription name: Microsoft Azure Internal Consumption
- Subscription ID: 628dad04-b5d1-4f10-b3a4-dc61d88cf97c

[All settings →](#)

Operations

Events
DNS01995

Settings

dns01995

- Properties
- IP addresses** (highlighted)
- DNS servers
- Users
- Tags

IP addresses

dns01995

Save Discard

Public IP address settings

Public IP address: **Enabled**

IP address: DNS01 (40.122.203.66)

Private IP address settings

Virtual network: TestNet

Subnet: FrontEnd (192.168.1.0/24)

Assignment

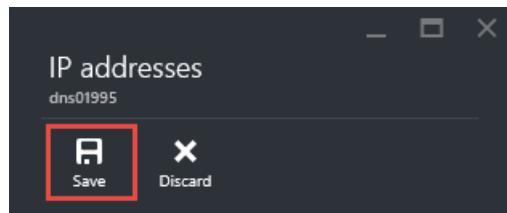
Dynamic (highlighted) **Static**

* IP address: 192.168.1.6

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 the steps below:

- From the **IP addresses** blade shown above, click **Static** under **Assignment**.
- 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 that the IP address you typed is already in use. Try a different IP address.

How to remove a static private IP address from a VM

To remove the static private IP address from the VM created above, follow the step below.

1. From the **IP addresses** blade shown above, click **Dynamic** under **Assignment**, and then click **Save**.

Next steps

- Learn about [reserved public IP](#) addresses.
- Learn about [instance-level public IP \(ILPIP\)](#) addresses.
- Consult the [Reserved IP REST APIs](#).

Set and manage a static private IP address using PowerShell

1/17/2017 • 4 min to read • [Edit on GitHub](#)

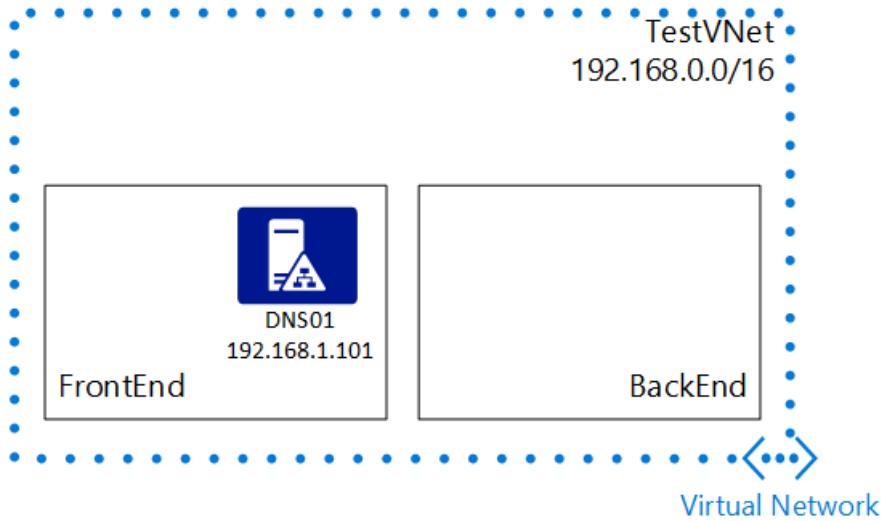
Your IaaS virtual machines (VMs) and PaaS role instances in a virtual network automatically receive a private IP address from a range that you specify, based on the subnet they are connected to. That address is retained by the VMs and role instances, until they are decommissioned. You decommission a 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 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*, 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 using the NIC created above.

```
$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
```

Expected output:

```
EndTime          : [Date and time]
Error           :
Output          :
StartTime        : [Date and time]
Status          : Succeeded
TrackingOperationId : [Id]
RequestId       : [Id]
StatusCode      : OK
```

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 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 VM

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 an existing VM

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

```

Next steps

- Learn about [reserved public IP](#) addresses.
- Learn about [instance-level public IP \(ILPIP\)](#) addresses.
- Consult the [Reserved IP REST APIs](#).

How to set a static private IP address in Azure CLI

1/17/2017 • 8 min to read • [Edit on GitHub](#)

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.

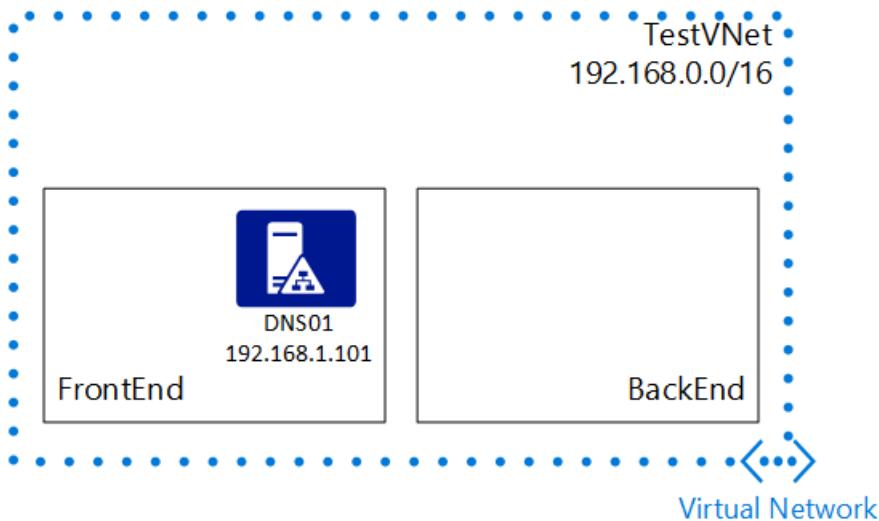
IMPORTANT

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 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 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. 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 config mode** command to switch to Resource Manager mode, as shown below.

```
azure config mode arm
```

Expected output:

```
info: New mode is arm
```

3. Run the **azure network public-ip create** to create a public IP for the VM. The list shown after the output explains the parameters used.

```
azure network public-ip create -g TestRG -n TestPIP -l centralus
```

Expected output:

```
info: Executing command network public-ip create
+ Looking up the public ip "TestPIP"
+ Creating public ip address "TestPIP"
+ Looking up the public ip "TestPIP"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/TestPIP
data: Name : TestPIP
data: Type : Microsoft.Network/publicIPAddresses
data: Location : centralus
data: Provisioning state : Succeeded
data: Allocation method : Dynamic
data: Idle timeout : 4
info: network public-ip create command OK
```

- **-g (or --resource-group)**. Name of the resource group the public IP will be created in.
 - **-n (or --name)**. Name of the public IP.
 - **-l (or --location)**. Azure region where the public IP will be created. For our scenario, *centralus*.
4. Run the **azure network nic create** command to create a NIC with a static private IP. The list shown after the output explains the parameters used.

```
azure network nic create -g TestRG -n TestNIC -l centralus -a 192.168.1.101 -m TestVNet -k FrontEnd
```

Expected output:

```

+ Looking up the network interface "TestNIC"
+ Looking up the subnet "FrontEnd"
+ Creating network interface "TestNIC"
+ Looking up the network interface "TestNIC"
data:   Id                      : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC
data:   Name                     : TestNIC
data:   Type                     : Microsoft.Network/networkInterfaces
data:   Location                 : centralus
data:   Provisioning state      : Succeeded
data:   Enable IP forwarding    : false
data:   IP configurations:
data:     Name                   : NIC-config
data:     Provisioning state    : Succeeded
data:     Private IP address    : 192.168.1.101
data:     Private IP Allocation Method : Static
data:     Subnet                 : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data:
info:   network nic create command OK

```

- **-a (or --private-ip-address)**. Static private IP address for the NIC.
- **-m (or --subnet-vnet-name)**. Name of the VNet where the NIC will be created.
- **-k (or --subnet-name)**. Name of the subnet where the NIC will be created.

5. Run the **azure vm create** command to create the VM using the public IP and NIC created above. The list shown after the output explains the parameters used.

```

azure vm create -g TestRG -n DNS01 -l centralus -y Windows -f TestNIC -i TestPIP -F TestVNet -j FrontEnd
-o vnetstorage -q bd507d3a70934695bc2128e3e5a255ba__RightImage-Windows-2012R2-x64-v14.2 -u adminuser -p
AdminP@ssw0rd

```

Expected output:

```

info:   Executing command vm create
+ Looking up the VM "DNS01"
info:   Using the VM Size "Standard_A1"
warn:   The image "bd507d3a70934695bc2128e3e5a255ba__RightImage-Windows-2012R2-x64-v14.2" will be used
for VM
info:   The [OS, Data] Disk or image configuration requires storage account
+ Looking up the storage account vnetstorage
+ Looking up the NIC "TestNIC"
info:   Found an existing NIC "TestNIC"
info:   Found an IP configuration with virtual network subnet id "/subscriptions/628dad04-b5d1-4f10-
b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd"
in the NIC "TestNIC"
info:   Found public ip parameters, trying to setup PublicIP profile
+ Looking up the public ip "TestPIP"
info:   Found an existing PublicIP "TestPIP"
info:   Configuring identified NIC IP configuration with PublicIP "TestPIP"
+ Updating NIC "TestNIC"
+ Looking up the NIC "TestNIC"
+ Creating VM "DNS01"
info:   vm create command OK

```

- **-y (or --os-type)**. Type of operating system for the VM, either *Windows* or *Linux*.
- **-f (or --nic-name)**. Name of the NIC the VM will use.
- **-i (or --public-ip-name)**. Name of the public IP the VM will use.
- **-F (or --vnet-name)**. Name of the VNet where the VM will be created.
- **-j (or --vnet-subnet-name)**. Name of the subnet where the VM will be created.

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 values for *Private IP alloc-method* and *Private IP address*:

```
azure vm show -g TestRG -n DNS01
```

Expected output:

```
info: Executing command vm show
+ Looking up the VM "DNS01"
+ Looking up the NIC "TestNIC"
+ Looking up the public ip "TestPIP"
data: Id :/subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/DNS01
data: ProvisioningState :Succeeded
data: Name :DNS01
data: Location :centralus
data: Type :Microsoft.Compute/virtualMachines
data:
data: Hardware Profile:
data: Size :Standard_A1
data:
data: Storage Profile:
data: Source image:
data: Id :/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/services/images/bd507d3a70934695bc2128e3e5a255ba__RightImage-Windows-2012R2-x64-v14.2
data:
data: OS Disk:
data: OSType :Windows
data: Name :cli08d7bd987a0112a8-os-1441774961355
data: Caching :ReadWrite
data: CreateOption :FromImage
data: Vhd:
data: Uri :https://vnetstorage2.blob.core.windows.net/vhds/cli08d7bd987a0112a8-
os-1441774961355vhd
data:
data: OS Profile:
data: Computer Name :DNS01
data: User Name :adminuser
data: Windows Configuration:
data: Provision VM Agent :true
data: Enable automatic updates :true
data:
data: Network Profile:
data: Network Interfaces:
data: Network Interface #1:
data: Id :/subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC
data: Primary :true
data: MAC Address :00-0D-3A-90-1A-A8
data: Provisioning State :Succeeded
data: Name :TestNIC
data: Location :centralus
data: Private IP alloc-method :Static
data: Private IP address :192.168.1.101
data: Public IP address :40.122.213.159
info: vm show command OK
```

How to remove a static private IP address from a VM

You cannot remove a static private IP address from a NIC in Azure CLI for Resource Manager. You must create a new NIC that uses a dynamic IP, remove the previous NIC from the VM, and then add the new NIC to the VM. To

change the NIC for the VM used int eh commands above, follow the steps below.

1. Run the **azure network nic create** command to create a new NIC using dynamic IP allocation. Notice how you do not need to specify the IP address this time.

```
azure network nic create -g TestRG -n TestNIC2 -l centralus -m TestVNet -k FrontEnd
```

Expected output:

```
info: Executing command network nic create
+ Looking up the network interface "TestNIC2"
+ Looking up the subnet "FrontEnd"
+ Creating network interface "TestNIC2"
+ Looking up the network interface "TestNIC2"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC2
data: Name : TestNIC2
data: Type : Microsoft.Network/networkInterfaces
data: Location : centralus
data: Provisioning state : Succeeded
data: Enable IP forwarding : false
data: IP configurations:
data: Name : NIC-config
data: Provisioning state : Succeeded
data: Private IP address : 192.168.1.6
data: Private IP Allocation Method : Dynamic
data: Subnet : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data:
info: network nic create command OK
```

2. 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:

```
info: Executing command vm set
+ Looking up the VM "DNS01"
+ Looking up the NIC "TestNIC2"
+ Updating VM "DNS01"
info: vm set command OK
```

3. If wanted, run the **azure network nic delete** command to delete the old NIC.

```
azure network nic delete -g TestRG -n TestNIC --quiet
```

Expected output:

```
info: Executing command network nic delete
+ Looking up the network interface "TestNIC"
+ Deleting network interface "TestNIC"
info: network nic delete command OK
```

How to add a static private IP address to an existing VM

To add a static private IP address to the NIC used by teh VM created using the script above, run the following command:

```
azure network nic set -g TestRG -n TestNIC2 -a 192.168.1.101
```

Expected output:

```
info: Executing command network nic set
+ Looking up the network interface "TestNIC2"
+ Updating network interface "TestNIC2"
+ Looking up the network interface "TestNIC2"
data: Id : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/TestNIC2
data: Name : TestNIC2
data: Type : Microsoft.Network/networkInterfaces
data: Location : centralus
data: Provisioning state : Succeeded
data: MAC address : 00-0D-3A-90-29-25
data: Enable IP forwarding : false
data: Virtual machine : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Compute/virtualMachines/DNS01
data: IP configurations:
data: Name : NIC-config
data: Provisioning state : Succeeded
data: Private IP address : 192.168.1.101
data: Private IP Allocation Method : Static
data: Subnet : /subscriptions/628dad04-b5d1-4f10-b3a4-
dc61d88cf97c/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data:
info: network nic set command OK
```

Next steps

- Learn about [reserved public IP addresses](#).
- Learn about [instance-level public IP \(ILPIP\) addresses](#).
- Consult the [Reserved IP REST APIs](#).

How to set a static private IP address (classic) in the Azure portal

1/17/2017 • 3 min to read • [Edit on GitHub](#)

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.

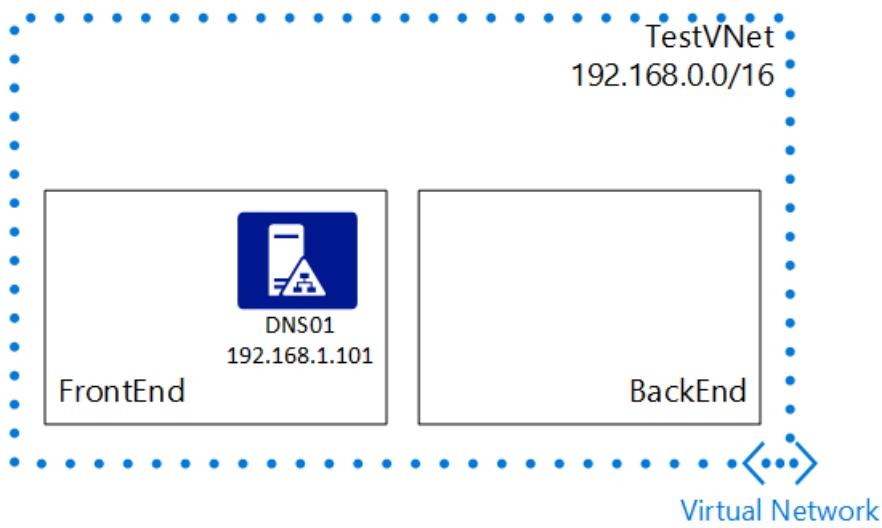
IMPORTANT

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](#).

Scenario

To better illustrate how to configure a static IP address for a VM, this document will use the scenario below.



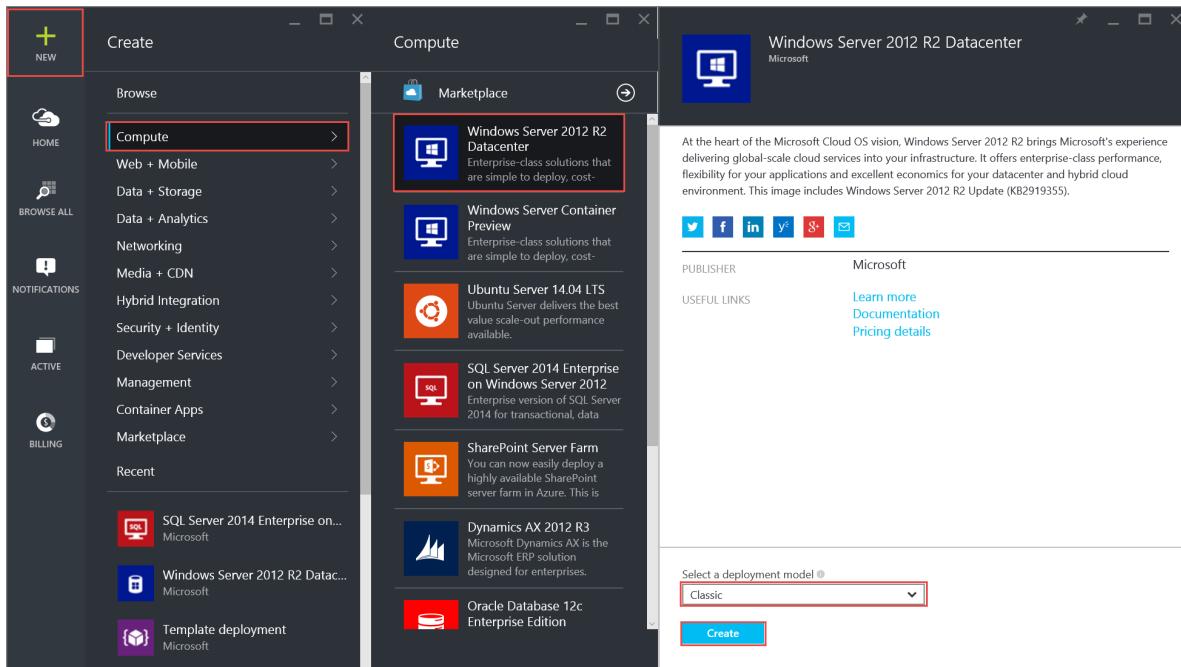
In this scenario you will create a VM named **DNS01** in the **FrontEnd** subnet, and set it to use a static IP address of **192.168.1.101**.

The sample steps below 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 vnet](#).

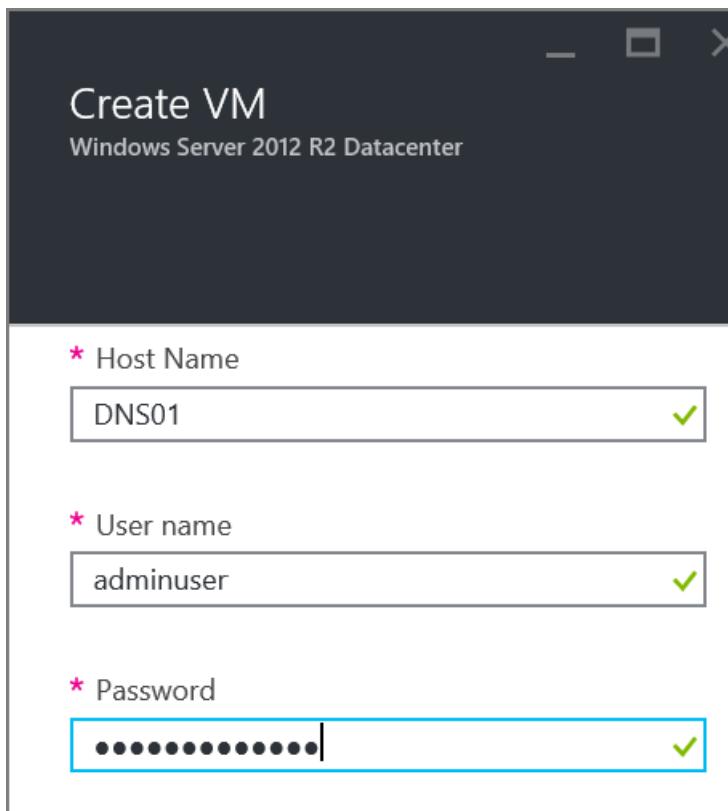
How to specify a static private IP address when creating a VM

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*, follow the steps below:

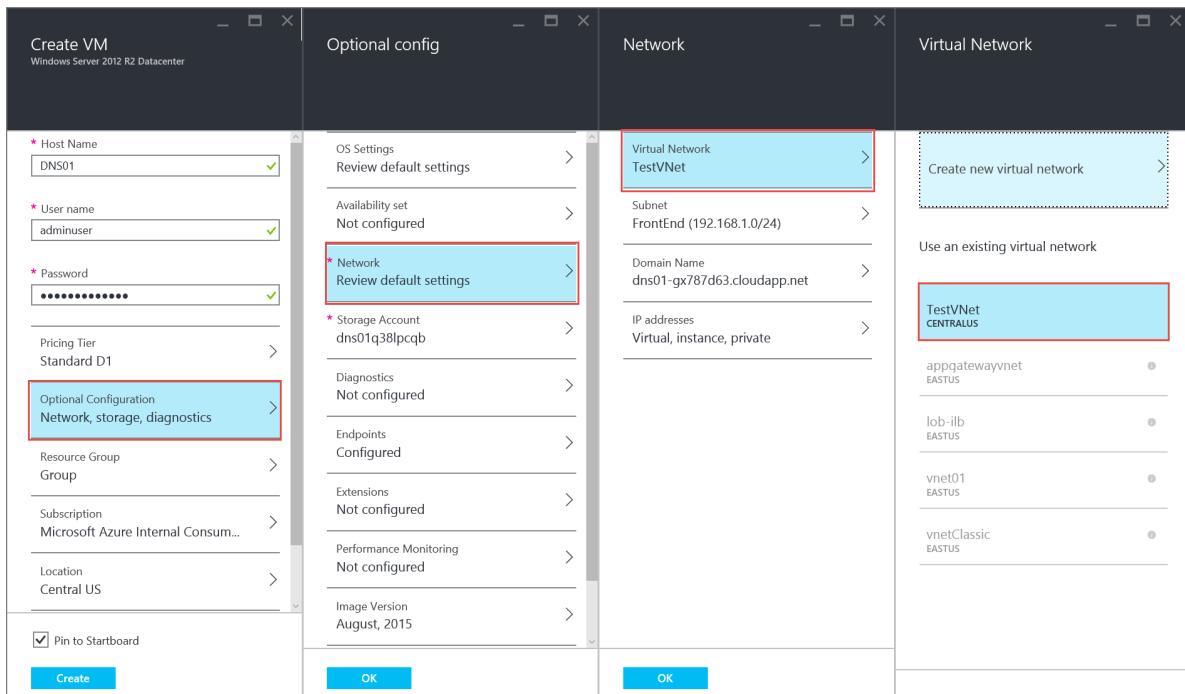
1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Click **NEW > Compute > Windows Server 2012 R2 Datacenter**, notice that the **Select a deployment model** list already shows **Classic**, and then click **Create**.



3. In the **Create VM** blade, enter the name of the VM to be created (*DNS01* in our scenario), the local administrator account, and password.



4. Click **Optional Configuration > Network > Virtual Network**, and then click **TestVNet**. If **TestVNet** is not available, make sure you are using the *Central US* location and have created the test environment described at the beginning of this article.



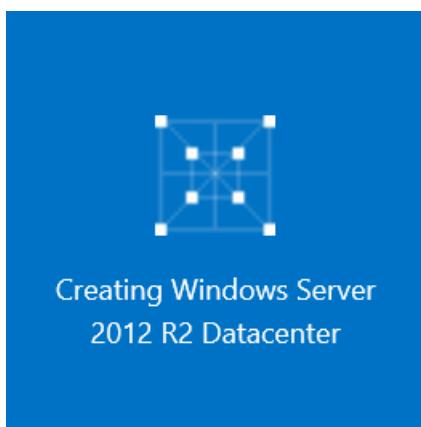
5. In the **Network** blade, make sure the subnet currently selected is *FrontEnd*, then click **IP addresses**, under **IP address assignment** click **Static**, and then enter **192.168.1.101** for **IP Address** as seen below.

The screenshot shows two overlapping windows from the Azure portal:

- Network** window (left):
 - Virtual Network: TestVNet
 - Subnet: FrontEnd (192.168.1.0/24)
 - Domain Name: dns01-gx787d63.cloudapp.net
 - IP addresses**: Virtual, instance, private (highlighted with a blue box)
- IP addresses** window (right):
 - Virtual IP address**:
 - Virtual IP address assignment: Dynamic (selected)
 - Instance IP address**:
 - Instance IP address: Off (selected)
 - Idle timeout (in minutes)**: A slider set to 4.
 - Private IP address**:
 - IP address assignment: Static (selected)
 - IP Address*: 192.168.1.101 (highlighted with a red box)

Both windows have an **OK** button at the bottom right.

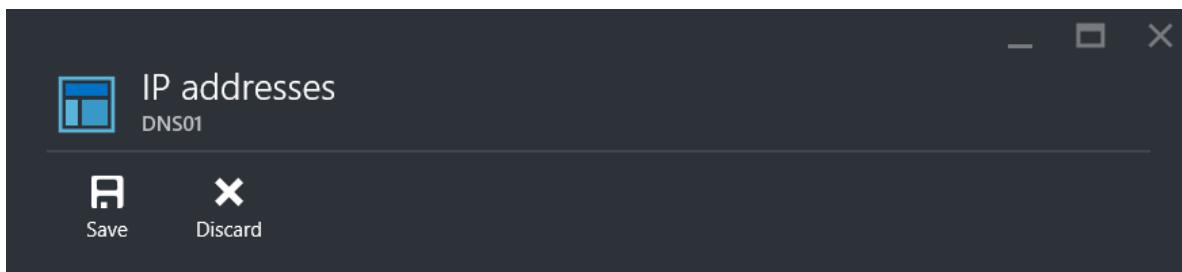
- Click **OK** in the **IP addresses** blade, then click **OK** in the **Network** blade, and click **OK** in the **Optional config** blade.
- In the **Create VM** blade, click **Create**. Notice the tile below displayed in your dashboard.



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 steps below.

1. From the Azure portal, click **BROWSE ALL** > **Virtual machines (classic)** > **DNS01** > **All settings** > **IP addresses** and notice the IP address assignment and IP address as seen below.



Virtual IP address

IP address assignment ⓘ Dynamic

IP address 40.122.208.41

Instance IP address

Instance IP address ⓘ

Private IP address

IP address assignment

Subnet 192.168.1.0/24

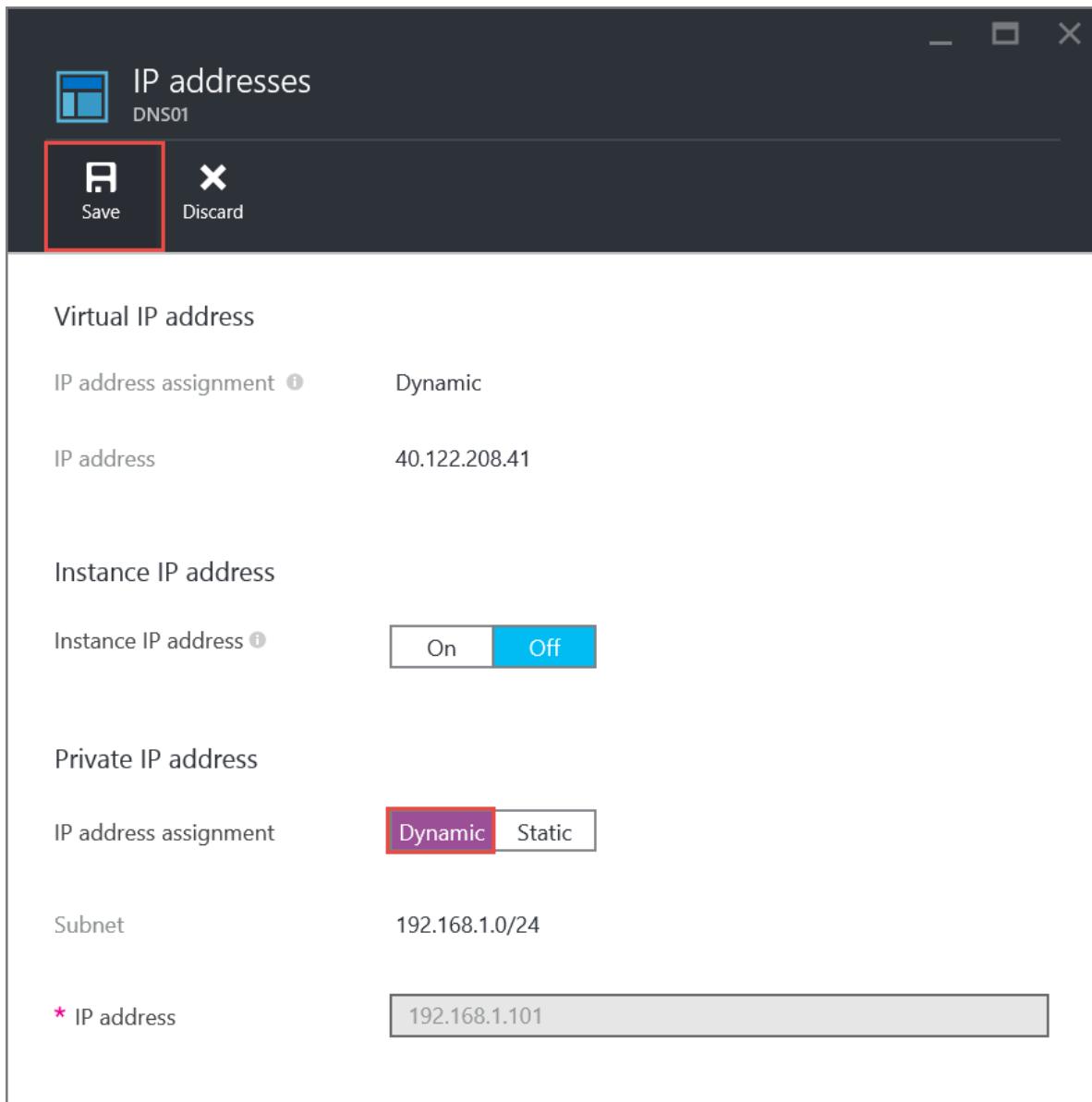
* IP address 192.168.1.101



How to remove a static private IP address from a VM

To remove the static private IP address from the VM created above, follow the steps below.

1. From the **IP addresses** blade shown above, click **Dynamic** to the right of **IP address assignment**, then click **Save**, and then click **Yes**.



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 the steps below:

1. From the **IP addresses** blade shown above, click **Static** to the right of **IP address assignment**.
2. Type **192.168.1.101** for **IP address**, then click **Save**, and then click **Yes**.

Next steps

- Learn about [reserved public IP](#) addresses.
- Learn about [instance-level public IP \(ILPIP\)](#) addresses.
- Consult the [Reserved IP REST APIs](#).

How to set a static private IP address (classic) in PowerShell

1/17/2017 • 3 min to read • [Edit on GitHub](#)

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.

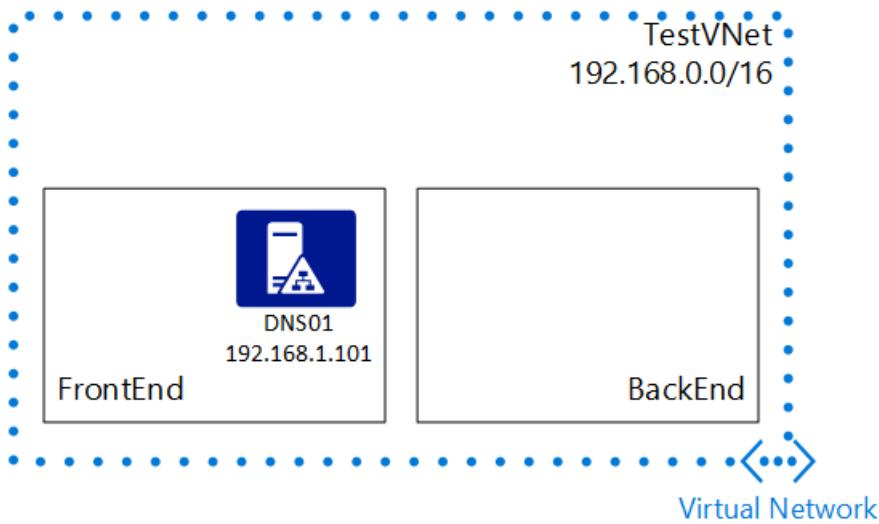
IMPORTANT

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](#).

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. 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 verify if a specific IP address is available

To verify if the IP address `192.168.1.101` is available in a VNet named `TestVNet`, run the following PowerShell command and verify the value for `IsAvailable`:

```
Test-AzureStaticVNetIP -VNetName TestVNet -IPAddress 192.168.1.101
```

Expected output:

```
IsAvailable      : True
AvailableAddresses : {}
OperationDescription : Test-AzureStaticVNetIP
OperationId       : fd3097e1-5f4b-9cac-8afa-bba1e3492609
OperationStatus    : Succeeded
```

How to specify a static private IP address when creating a VM

The PowerShell script below creates a new cloud service named `TestService`, then retrieves an image from Azure, creates a VM named `DNS01` in the new cloud service using the retrieved image, sets the VM to be in a subnet named `FrontEnd`, and sets `192.168.1.7` as a static private IP address for the VM:

```
New-AzureService -ServiceName TestService -Location "Central US"
$image = Get-AzureVMImage | where {$_.ImageName -like "*RightImage-Windows-2012R2-x64*"}
New-AzureVMConfig -Name DNS01 -InstanceSize Small -ImageName $image.ImageName |
  Add-AzureProvisioningConfig -Windows -AdminUsername adminuser -Password MyP@ssw0rd!! |
  Set-AzureSubnet -SubnetNames FrontEnd |
  Set-AzureStaticVNetIP -IPAddress 192.168.1.7 |
New-AzureVM -ServiceName TestService -VNetName TestVNet
```

Expected output:

```
WARNING: No deployment found in service: 'TestService'.
OperationDescription OperationId                               OperationStatus
----- ----- -----
New-AzureService      fcf705f1-d902-011c-95c7-b690735e7412 Succeeded
New-AzureVM           3b99a86d-84f8-04e5-888e-b6fc3c73c4b9 Succeeded
```

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 PowerShell command and observe the values for `IpAddress`:

```
Get-AzureVM -Name DNS01 -ServiceName TestService
```

Expected output:

```

DeploymentName      : TestService
Name               : DNS01
Label              :
VM                : Microsoft.WindowsAzure.Commands.ServiceManagement.Model.PersistentVM
InstanceStateStatus : Provisioning
IpAddress          : 192.168.1.7
InstanceStateDetails: Windows is preparing your computer for first use...
PowerState          : Started
InstanceErrorCode   :
InstanceFaultDomain: 0
InstanceName        : DNS01
InstanceUpgradeDomain: 0
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 private IP address from a VM

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

```

Get-AzureVM -ServiceName TestService -Name DNS01 |
  Remove-AzureStaticVNetIP |
  Update-AzureVM

```

Expected output:

OperationDescription	OperationId	OperationStatus
Update-AzureVM	052fa6f6-1483-0ede-a7bf-14f91f805483	Succeeded

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 script above, run the following command:

```

Get-AzureVM -ServiceName TestService -Name DNS01 |
  Set-AzureStaticVNetIP -IPAddress 192.168.1.7 |
  Update-AzureVM

```

Expected output:

OperationDescription	OperationId	OperationStatus
Update-AzureVM	77d8cae2-87e6-0ead-9738-7c7dae9810cb	Succeeded

Next steps

- Learn about [reserved public IP](#) addresses.
- Learn about [instance-level public IP \(ILPIP\)](#) addresses.
- Consult the [Reserved IP REST APIs](#).

How to set a static private IP address (classic) in Azure CLI

1/17/2017 • 4 min to read • [Edit on GitHub](#)

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
```

Next steps

- Learn about [reserved public IP](#) addresses.
- Learn about [instance-level public IP \(ILPIP\)](#) addresses.
- Consult the [Reserved IP REST APIs](#).

Create a VM with multiple NICs using PowerShell

1/17/2017 • 7 min to read • [Edit on GitHub](#)

You can create virtual machines (VMs) in Azure and attach multiple network interfaces (NICs) to each of your VMs. Multi NIC is a requirement for many network virtual appliances, such as application delivery and WAN optimization solutions. Multi NIC also provides more network traffic management functionality, including isolation of traffic between a front end NIC and back end NIC(s), or separation of data plane traffic from management plane traffic.

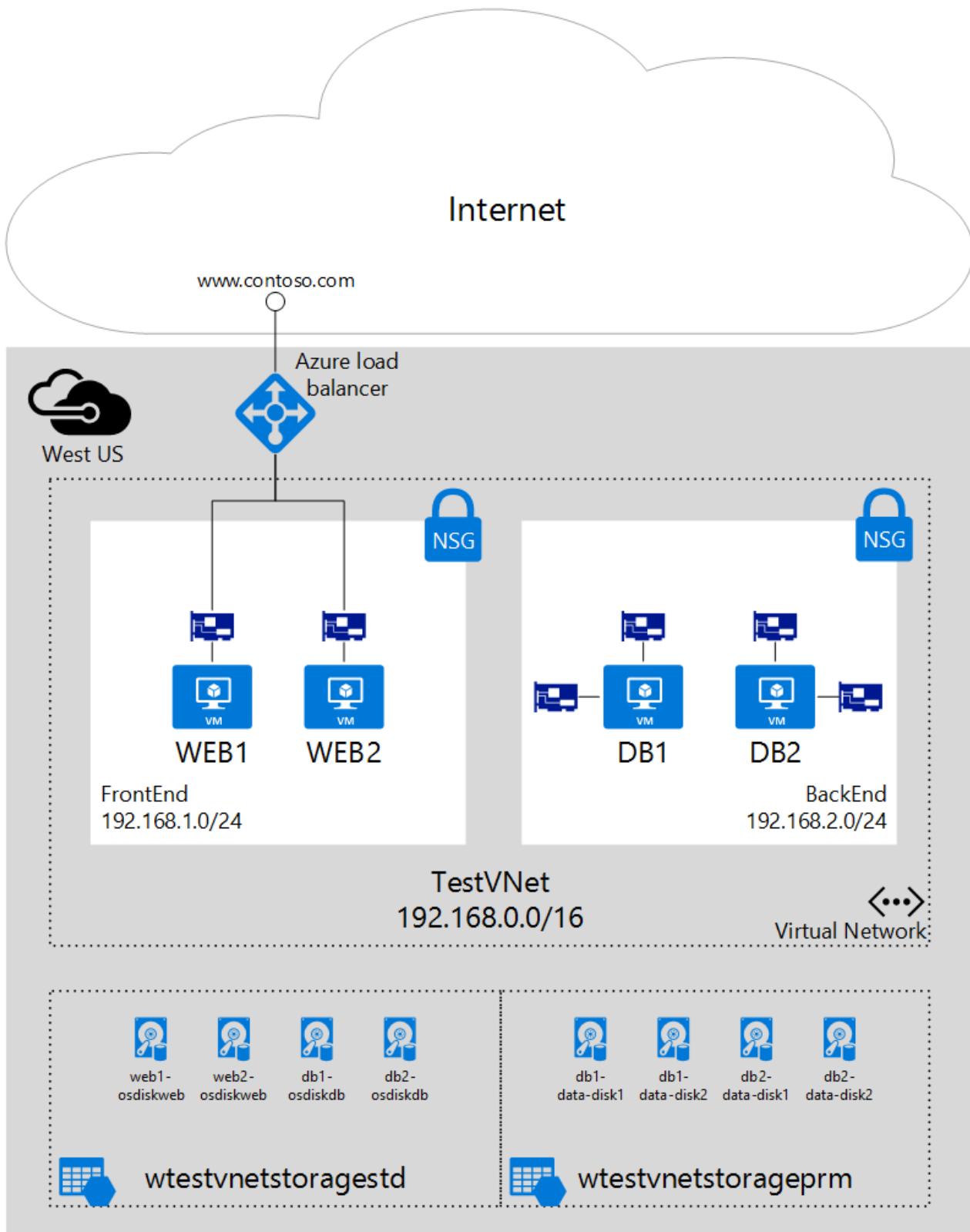
Before you can implement multi NICs in VMs, it is necessary to understand when you can use multi NICs, and how they are used. Read the [multi NIC overview](#) to learn more about VMs with multiple NICs.

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 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. These database servers will be 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 figure below 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 following steps:

1. Navigate to [the template page](#).
2. In the template page, to the right of **Parent resource group**, click **Deploy to Azure**.
3. If needed, change the parameter values, then follow the steps in the Azure preview portal to deploy the

resource group.

IMPORTANT

Make sure your storage account names are unique. You cannot have duplicate storage account names in Azure.

Prerequisite: Install the Azure PowerShell Module

To perform the steps in this article, you'll need to [to install and configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.

NOTE

If you don't have an Azure account, you'll need one. Go sign up for a [free trial here](#).

Create 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 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](#).

```
$existingRGName      = "IaaSStory"
$location            = "West US"
$vnetName            = "WTestVNet"
$backendSubnetName   = "BackEnd"
$remoteAccessNSGName = "NSG-RemoteAccess"
$stdStorageAccountName = "wtestvnetstoragestd"
```

2. Change the values of the variables below based on the values you want to use for your backend deployment.

```

$backendRGName      = "IaaSStory-Backend"
$prmStorageAccountName = "wtestvnetstorageprm"
$avSetName          = "ASDB"
$vmSize              = "Standard_DS3"
$publisher            = "MicrosoftSQLServer"
$offer                = "SQL2014SP1-WS2012R2"
$sku                  = "Standard"
$version              = "latest"
$vmNamePrefix        = "DB"
$osDiskPrefix         = "osdiskdb"
$dataDiskPrefix       = "datadisk"
$diskSize             = "120"
$nicNamePrefix        = "NICDB"
$ipAddressPrefix      = "192.168.2."
$numberOfVMs          = 2

```

3. Retrieve the existing resources needed for your deployment.

```

$vnet           = Get-AzureRmVirtualNetwork -Name $vnetName -ResourceGroupName $existingRGName
$backendSubnet = $vnet.Subnets | ?{$_ . Name -eq $backendSubnetName}
$remoteAccessNSG = Get-AzureRmNetworkSecurityGroup -Name $remoteAccessNSGName -ResourceGroupName $existingRGName
$stdStorageAccount = Get-AzureRmStorageAccount -Name $stdStorageAccountName -ResourceGroupName $existingRGName

```

Step 2 - Create necessary resources for your VMs

You need to create a new resource group, a storage account for the data disks, and an availability set for all VMs. You also need the local administrator account credentials for each VM. To create these resources, execute the following steps.

1. Create a new resource group.

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

2. Create a new premium storage account in the resource group created above.

```
$prmStorageAccount = New-AzureRmStorageAccount -Name $prmStorageAccountName ` 
-ResourceGroupName $backendRGName -Type Premium_LRS -Location $location
```

3. Create a new availability set.

```
$avSet = New-AzureRmAvailabilitySet -Name $avSetName -ResourceGroupName $backendRGName -Location $location
```

4. Get the local administrator account credentials to be used for each VM.

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

Step 3 - Create the NICs and back-end 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.

1. 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++){
```

2. Create the NIC used for database access.

```
$nic1Name = $nicNamePrefix + $suffixNumber + "-DA"
$ipAddress1 = $ipAddressPrefix + ($suffixNumber + 3)
$nic1 = New-AzureRmNetworkInterface -Name $nic1Name -ResourceGroupName $backendRGName ` 
-Location $location -SubnetId $backendSubnet.Id -PrivateIpAddress $ipAddress1
```

3. Create the NIC used for remote access. Notice how this NIC has an NSG associated to it.

```
$nic2Name = $nicNamePrefix + $suffixNumber + "-RA"
$ipAddress2 = $ipAddressPrefix + (53 + $suffixNumber)
$nic2 = New-AzureRmNetworkInterface -Name $nic2Name -ResourceGroupName $backendRGName ` 
-Location $location -SubnetId $backendSubnet.Id -PrivateIpAddress $ipAddress2 ` 
-NetworkSecurityGroupId $remoteAccessNSG.Id
```

4. Create `vmConfig` object.

```
$vmName = $vmNamePrefix + $suffixNumber
$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmSize -AvailabilitySetId $avSet.Id
```

5. Create two data disks per VM. Notice that the data disks are in the premium storage account created earlier.

```
$dataDisk1Name = $vmName + "—" + $osDiskPrefix + "-1"
$data1VhdUri = $prmStorageAccount.PrimaryEndpoints.Blob.ToString() + "vhds/" + $dataDisk1Name + ".vhd"
Add-AzureRmVMDataDisk -VM $vmConfig -Name $dataDisk1Name -DiskSizeInGB $diskSize ` 
-VhdUri $data1VhdUri -CreateOption empty -Lun 0

$dataDisk2Name = $vmName + "—" + $osDiskPrefix + "-2"
$data2VhdUri = $prmStorageAccount.PrimaryEndpoints.Blob.ToString() + "vhds/" + $dataDisk2Name + ".vhd"
Add-AzureRmVMDataDisk -VM $vmConfig -Name $dataDisk2Name -DiskSizeInGB $diskSize ` 
-VhdUri $data2VhdUri -CreateOption empty -Lun 1
```

6. Configure the operating system, and image to be used for the VM.

```
$vmConfig = Set-AzureRmVMOperatingSystem -VM $vmConfig -Windows -ComputerName $vmName -Credential $cred
-ProvisionVMAgent -EnableAutoUpdate
$vmConfig = Set-AzureRmVMSourceImage -VM $vmConfig -PublisherName $publisher -Offer $offer -Skus $sku - 
Version $version
```

7. Add the two NICs created above to the `vmConfig` object.

```
$vmConfig = Add-AzureRmVMNetworkInterface -VM $vmConfig -Id $nic1.Id -Primary
$vmConfig = Add-AzureRmVMNetworkInterface -VM $vmConfig -Id $nic2.Id
```

8. Create the OS disk and create the VM. Notice the `)` ending the `for` loop.

```
$osDiskName = $vmName + "—" + $osDiskSuffix
$osVhdUri = $stdStorageAccount.PrimaryEndpoints.Blob.ToString() + "vhds/" + $osDiskName + ".vhd"
$vmConfig = Set-AzureRmVMOSDisk -VM $vmConfig -Name $osDiskName -VhdUri $osVhdUri -CreateOption
fromImage
New-AzureRmVM -VM $vmConfig -ResourceGroupName $backendRGName -Location $location
}
```

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.

- Save your script and run it from the **PowerShell** command prompt, or **PowerShell ISE**. You will see the initial output, as follows:

```

ResourceGroupName : IaaSStory-Backend
Location         : westus
ProvisioningState : Succeeded
Tags             :
Permissions      :
  Actions  NotActions
  =====  ========
  *          *

ResourceId       : /subscriptions/[Subscription ID]/resourceGroups/IaaSStory-Backend

```

- After a few minutes, fill out the credentials prompt and click **OK**. The output below represents a single VM. Notice the entire process took 8 minutes to complete.

```

ResourceGroupName      :
Id                   :
Name                 : DB2
Type                 :
Location             :
Tags                :
TagsText             : null
AvailabilitySetReference : Microsoft.Azure.Management.Compute.Models.AvailabilitySetReference
AvailabilitySetReferenceText : {
    "ReferenceUri": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/IaaSStory-Backend/providers/Microsoft.Compute/availabilitySets/ASDB"
}
Extensions           :
ExtensionsText       : null
HardwareProfile      : Microsoft.Azure.Management.Compute.Models.HardwareProfile
HardwareProfileText : {
    "VirtualMachineSize": "Standard_DS3"
}
InstanceView          :
InstanceViewText     : null
NetworkProfile        :
NetworkProfileText   : null
OSProfile            :
OSProfileText        : null
Plan                 :
PlanText             : null
ProvisioningState    :
StorageProfile       : Microsoft.Azure.Management.Compute.Models.StorageProfile
StorageProfileText  : {
    "DataDisks": [
        {
            "Lun": 0,
            "Caching": null,
            "CreateOption": "empty",
            "DiskSizeGB": 127,
            "Name": "DB2-disk-1",
            "SourceImage": null,
            "VirtualHardDisk": {
                "Uri":
                "https://wtestvnetstorageprm.blob.core.windows.net/vhds/DB2-disk-1.vhd"
            }
        }
    ],
    "ImageReference": null,
    "OSDisk": null
}
DataDiskNames         : {DB2-disk-1}
NetworkInterfaceIDs :
RequestId            :

```

```

StatusCode : 0

ResourceGroupName : 
Id : 
Name : DB2
Type : 
Location : 
Tags : 
TagsText : null
AvailabilitySetReference : Microsoft.Azure.Management.Compute.Models.AvailabilitySetReference
AvailabilitySetReferenceText : {
    "ReferenceUri": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/IaaSStory-Backend/providers/
        Microsoft.Compute/availabilitySets/ASDB"
}
Extensions : 
ExtensionsText : null
HardwareProfile : Microsoft.Azure.Management.Compute.Models.HardwareProfile
HardwareProfileText : {
    "VirtualMachineSize": "Standard_DS3"
}
InstanceView : 
InstanceViewText : null
NetworkProfile : 
NetworkProfileText : null
OSProfile : 
OSProfileText : null
Plan : 
PlanText : null
ProvisioningState : 
StorageProfile : Microsoft.Azure.Management.Compute.Models.StorageProfile
StorageProfileText : {
    "DataDisks": [
        {
            "Lun": 0,
            "Caching": null,
            "CreateOption": "empty",
            "DiskSizeGB": 127,
            "Name": "DB2-disk-1",
            "SourceImage": null,
            "VirtualHardDisk": {
                "Uri":
"https://wtestvnetstorageprm.blob.core.windows.net/vhds/DB2-disk-1.vhd"
            }
        },
        {
            "Lun": 1,
            "Caching": null,
            "CreateOption": "empty",
            "DiskSizeGB": 127,
            "Name": "DB2-disk-2",
            "SourceImage": null,
            "VirtualHardDisk": {
                "Uri":
"https://wtestvnetstorageprm.blob.core.windows.net/vhds/DB2-disk-2.vhd"
            }
        }
    ],
    "ImageReference": null,
    "OSDisk": null
}
DataDiskNames : {DB2-disk-1, DB2-disk-2}
NetworkInterfaceIDs : 
RequestId : 
StatusCode : 0
EndTime : [Date] [Time]
Error : 
Output : 
StartTime : [Date] [Time]

```

Status	: Succeeded
TrackingOperationId	: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
RequestId	: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
StatusCode	: OK

Create a VM with multiple NICs using the Azure CLI

1/17/2017 • 9 min to read • [Edit on GitHub](#)

You can create virtual machines (VMs) in Azure and attach multiple network interfaces (NICs) to each of your VMs. Multi NIC is a requirement for many network virtual appliances, such as application delivery and WAN optimization solutions. Multi NIC also provides more network traffic management functionality, including isolation of traffic between a front end NIC and back end NIC(s), or separation of data plane traffic from management plane traffic.

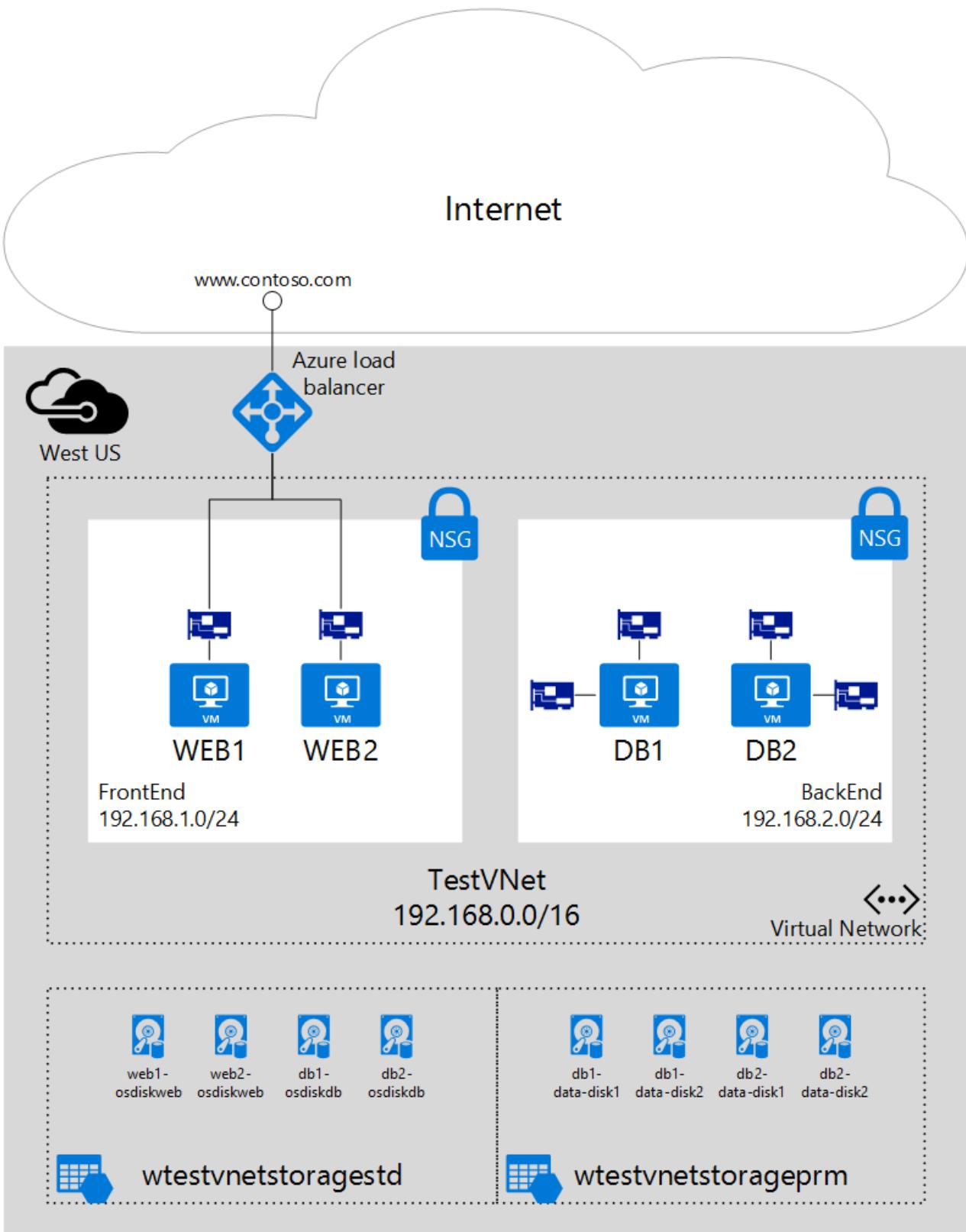
Before you can implement multi NICs in VMs, it is necessary to understand when you can use multi NICs, and how they are used. Read the [multi NIC overview](#) to learn more about VMs with multiple NICs.

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 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. These database servers will be 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 figure below 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 following steps:

1. Navigate to [the template page](#).
2. In the template page, to the right of **Parent resource group**, click **Deploy to Azure**.
3. If needed, change the parameter values, then follow the steps in the Azure preview portal to deploy the

resource group.

IMPORTANT

Make sure your storage account names are unique. You cannot have duplicate storage account names in Azure.

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 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](#). 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](#).

```
existingRGName="IaaSStory"
location="westus"
vnetName="WTestVNet"
backendSubnetName="BackEnd"
remoteAccessNSGName="NSG-RemoteAccess"
```

2. Change the values of the variables below based on the values you want to use for your backend deployment.

```
backendRGName="IaaSStory-Backend"
prmStorageAccountName="wtestvnetstorageprm"
avSetName="ASDB"
vmSize="Standard_DS3"
diskSize=127
publisher="Canonical"
offer="UbuntuServer"
sku="14.04.2-LTS"
version="latest"
vmNamePrefix="DB"
osDiskName="osdiskdb"
dataDiskName="datadisk"
nicNamePrefix="NICDB"
ipAddressPrefix="192.168.2."
username='adminuser'
password='adminP@ssw0rd'
number0fVMs=2
```

3. Retrieve the ID for the `BackEnd` subnet where the VMs will be created. You need to do this since the NICs to be associated to this subnet are in a different resource group.

```
subnetId=$(azure network vnet subnet show --resource-group $existingRGName \
    --vnet-name $vnetName \
    --name $backendSubnetName|grep Id)"
subnetId=${subnetId#*/}
```

TIP

The first command above uses `grep` and `string manipulation` (more specifically, substring removal).

4. Retrieve the ID for the `NSG-RemoteAccess` NSG. You need to do this since the NICs to be associated to this NSG are in a different resource group.

```
nsgId=$(azure network nsg show --resource-group $existingRGName \
    --name $remoteAccessNSGName|grep Id)"
nsgId=${nsgId#*/}
```

Step 2 - Create necessary resources for your VMs

1. Create a new resource group for all backend resources. Notice the use of the `$backendRGName` variable for the resource group name, and `$location` for the Azure region.

```
azure group create $backendRGName $location
```

2. Create a premium storage account for the OS and data disks to be used by yours VMs.

```
azure storage account create $prmStorageAccountName \
    --resource-group $backendRGName \
    --location $location \
    --type PLRS
```

3. Create an availability set for the VMs.

```
azure availset create --resource-group $backendRGName \
    --location $location \
    --name $avSetName
```

Step 3 - Create the NICs and back-end VMs

1. Start a loop to create multiple VMs, based on the `numberOfVMs` variables.

```
for ((suffixNumber=1;suffixNumber<=numberOfVMs;suffixNumber++));  
do
```

2. For each VM, create a NIC for database access.

```
nic1Name=$nicNamePrefix$suffixNumber-DA  
x=$((suffixNumber+3))  
ipAddress1=$ipAddressPrefix$x  
azure network nic create --name $nic1Name \  
--resource-group $backendRGName \  
--location $location \  
--private-ip-address $ipAddress1 \  
--subnet-id $subnetId
```

3. For each VM, create a NIC for remote access. Notice the `--network-security-group` parameter, used to associate the NIC to an NSG.

```
nic2Name=$nicNamePrefix$suffixNumber-RA  
x=$((suffixNumber+53))  
ipAddress2=$ipAddressPrefix$x  
azure network nic create --name $nic2Name \  
--resource-group $backendRGName \  
--location $location \  
--private-ip-address $ipAddress2 \  
--subnet-id $subnetId $vnetName \  
--network-security-group-id $nsgId
```

4. Create the VM.

```
azure vm create --resource-group $backendRGName \  
--name $vmNamePrefix$suffixNumber \  
--location $location \  
--vm-size $vmSize \  
--subnet-id $subnetId \  
--availset-name $avSetName \  
--nic-names $nic1Name,$nic2Name \  
--os-type linux \  
--image-urn $publisher:$offer:$sku:$version \  
--storage-account-name $prmStorageAccountName \  
--storage-account-container-name vhds \  
--os-disk-vhd $osDiskName$suffixNumber.vhd \  
--admin-username $username \  
--admin-password $password
```

5. For each VM, create two data disks, and end the loop with the `done` command.

```

azure vm disk attach-new --resource-group $backendRGName \
--vm-name $vmNamePrefix$suffixNumber \
--storage-account-name $prmStorageAccountName \
--storage-account-container-name vhds \
--vhd-name $dataDiskName$suffixNumber-1.vhd \
--size-in-gb $diskSize \
--lun 0

azure vm disk attach-new --resource-group $backendRGName \
--vm-name $vmNamePrefix$suffixNumber \
--storage-account-name $prmStorageAccountName \
--storage-account-container-name vhds \
--vhd-name $dataDiskName$suffixNumber-2.vhd \
--size-in-gb $diskSize \
--lun 1
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 group create
info: Getting resource group IaaSStory-Backend
info: Creating resource group IaaSStory-Backend
info: Created resource group IaaSStory-Backend
data: Id: /subscriptions/[Subscription ID]/resourceGroups/IaaSStory-Backend
data: Name: IaaSStory-Backend
data: Location: westus
data: Provisioning State: Succeeded
data: Tags: null
data:
info: group create command OK
info: Executing command storage account create
info: Creating storage account
info: storage account create command OK
info: Executing command availset create
info: Looking up the availability set "ASDB"
info: Creating availability set "ASDB"
info: availset create command OK
info: Executing command network nic create
info: Looking up the network interface "NICDB1-DA"
info: Creating network interface "NICDB1-DA"
info: Looking up the network interface "NICDB1-DA"
data: Id : /subscriptions/[Subscription ID]/resourceGroups/IaaSStory-
Backend/providers/Microsoft.Network/networkInterfaces/NICDB1-DA
data: Name : NICDB1-DA
data: Type : Microsoft.Network/networkInterfaces
data: Location : westus
data: Provisioning state : Succeeded
data: Enable IP forwarding : false
data: IP configurations:
data: Name : NIC-config
data: Provisioning state : Succeeded
data: Private IP address : 192.168.2.4
data: Private IP Allocation Method : Static
data: Subnet : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/WTestVNet/subnets/BackEnd
data:
info: network nic create command OK
info: Executing command network nic create
info: Looking up the network interface "NICDB1-RA"
info: Creating network interface "NICDB1-RA"
info: Looking up the network interface "NICDB1-RA"
data: Id : /subscriptions/[Subscription ID]/resourceGroups/IaaSStory-

```

```

Backend/providers/Microsoft.Network/networkInterfaces/NICDB1-RA
data:  Name          : NICDB1-RA
data:  Type          : Microsoft.Network/networkInterfaces
data:  Location       : westus
data:  Provisioning state : Succeeded
data:  Enable IP forwarding : false
data:  Network security group : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/networkSecurityGroups/NSG-RemoteAccess
data:  IP configurations:
data:    Name          : NIC-config
data:    Provisioning state : Succeeded
data:    Private IP address : 192.168.2.54
data:    Private IP Allocation Method : Static
data:    Subnet         : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/WTestVNet/subnets/BackEnd
data:
info:  network nic create command OK
info:  Executing command vm create
info:  Looking up the VM "DB1"
info:  Using the VM Size "Standard_DS3"
info:  The [OS, Data] Disk or image configuration requires storage account
info:  Looking up the storage account wtestvnetstorageprm
info:  Looking up the availability set "ASDB"
info:  Found an Availability set "ASDB"
info:  Looking up the NIC "NICDB1-DA"
info:  Looking up the NIC "NICDB1-RA"
info:  Creating VM "DB1"

```

- After a few minutes, the execution will end and you will see the rest of the output as shown below.

```

info:  vm create command OK
info:  Executing command vm disk attach-new
info:  Looking up the VM "DB1"
info:  Looking up the storage account wtestvnetstorageprm
info:  New data disk location: https://wtestvnetstorageprm.blob.core.windows.net/vhds/datadisk1-1.vhd
info:  Updating VM "DB1"
info:  vm disk attach-new command OK
info:  Executing command vm disk attach-new
info:  Looking up the VM "DB1"
info:  Looking up the storage account wtestvnetstorageprm
info:  New data disk location: https://wtestvnetstorageprm.blob.core.windows.net/vhds/datadisk1-2.vhd
info:  Updating VM "DB1"
info:  vm disk attach-new command OK
info:  Executing command network nic create
info:  Looking up the network interface "NICDB2-DA"
info:  Creating network interface "NICDB2-DA"
info:  Looking up the network interface "NICDB2-DA"
data:  Id          : /subscriptions/[Subscription ID]/resourceGroups/IaaSStory-
Backend/providers/Microsoft.Network/networkInterfaces/NICDB2-DA
data:  Name          : NICDB2-DA
data:  Type          : Microsoft.Network/networkInterfaces
data:  Location       : westus
data:  Provisioning state : Succeeded
data:  Enable IP forwarding : false
data:  IP configurations:
data:    Name          : NIC-config
data:    Provisioning state : Succeeded
data:    Private IP address : 192.168.2.5
data:    Private IP Allocation Method : Static
data:    Subnet         : /subscriptions/[Subscription
ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/WTestVNet/subnets/BackEnd
data:
info:  network nic create command OK
info:  Executing command network nic create
info:  Looking up the network interface "NICDB2-RA"
info:  Creating network interface "NICDB2-RA"
info:  Looking up the network interface "NICDB2-RA"

```

```
data:   Id                      : /subscriptions/[Subscription ID]/resourceGroups/IaaSStory-Backend/providers/Microsoft.Network/networkInterfaces/NICDB2-RA
data:   Name                     : NICDB2-RA
data:   Type                     : Microsoft.Network/networkInterfaces
data:   Location                  : westus
data:   Provisioning state       : Succeeded
data:   Enable IP forwarding     : false
data:   Network security group   : /subscriptions/[Subscription ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/networkSecurityGroups/NSG-RemoteAccess
data:   IP configurations:
data:     Name                   : NIC-config
data:     Provisioning state    : Succeeded
data:     Private IP address    : 192.168.2.55
data:     Private IP Allocation Method : Static
data:     Subnet                 : /subscriptions/[Subscription ID]/resourceGroups/IaaSStory/providers/Microsoft.Network/virtualNetworks/WTestVNet/subnets/BackEnd
data:
info:  network nic create command OK
info:  Executing command vm create
info:  Looking up the VM "DB2"
info:  Using the VM Size "Standard_DS3"
info:  The [OS, Data] Disk or image configuration requires storage account
info:  Looking up the storage account wtestvnetstorageprm
info:  Looking up the availability set "ASDB"
info:  Found an Availability set "ASDB"
info:  Looking up the NIC "NICDB2-DA"
info:  Looking up the NIC "NICDB2-RA"
info:  Creating VM "DB2"
info:  vm create command OK
info:  Executing command vm disk attach-new
info:  Looking up the VM "DB2"
info:  Looking up the storage account wtestvnetstorageprm
info:  New data disk location: https://wtestvnetstorageprm.blob.core.windows.net/vhds/datadisk2-1.vhd
info:  Updating VM "DB2"
info:  vm disk attach-new command OK
info:  Executing command vm disk attach-new
info:  Looking up the VM "DB2"
info:  Looking up the storage account wtestvnetstorageprm
info:  New data disk location: https://wtestvnetstorageprm.blob.core.windows.net/vhds/datadisk2-2.vhd
info:  Updating VM "DB2"
info:  vm disk attach-new command OK
```

Create a VM with multiple NICs using a template

1/17/2017 • 8 min to read • [Edit on GitHub](#)

You can create virtual machines (VMs) in Azure and attach multiple network interfaces (NICs) to each of your VMs. Multi NIC is a requirement for many network virtual appliances, such as application delivery and WAN optimization solutions. Multi NIC also provides more network traffic management functionality, including isolation of traffic between a front end NIC and back end NIC(s), or separation of data plane traffic from management plane traffic.

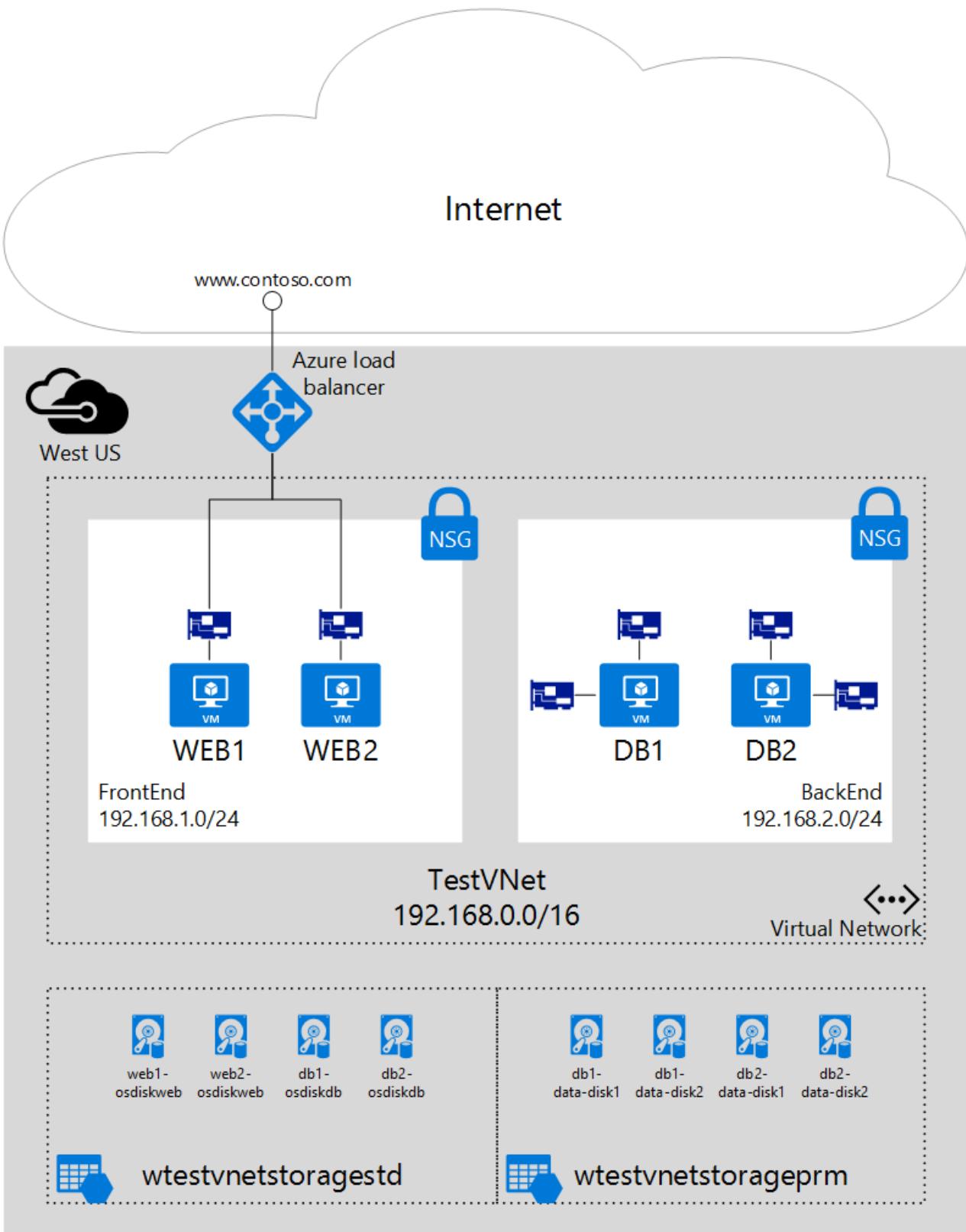
Before you can implement multi NICs in VMs, it is necessary to understand when you can use multi NICs, and how they are used. Read the [multi NIC overview](#) to learn more about VMs with multiple NICs.

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 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. These database servers will be 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 figure below 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 following steps:

1. Navigate to [the template page](#).
2. In the template page, to the right of **Parent resource group**, click **Deploy to Azure**.
3. If needed, change the parameter values, then follow the steps in the Azure preview portal to deploy the

resource group.

IMPORTANT

Make sure your storage account names are unique. You cannot have duplicate storage account names in Azure.

Understand the deployment template

Before you deploy the template provided with this documentation, make sure you understand what it does. The following steps provide a good overview of the template:

1. Navigate to [the template page](#).
2. Click **azuredeploy.json** to open the template file.
3. Notice the **osType** parameter listed below. This parameter is used to select what VM image to use for the database server, along with multiple operating system related settings.

```
"osType": {  
    "type": "string",  
    "defaultValue": "Windows",  
    "allowedValues": [  
        "Windows",  
        "Ubuntu"  
    ],  
    "metadata": {  
        "description": "Type of OS to use for VMs: Windows or Ubuntu."  
    }  
},
```

4. Scroll down to the list of variables, and check the definition for the **dbVMSetting** variables, listed below. It receives one of the array elements contained in the **dbVMSettings** variable. If you are familiar with software development terminology, you can view the **dbVMSettings** variable as a hash table, or a dictionary.

```
"dbVMSetting": "[variables('dbVMSettings')[parameters('osType')]]"
```

5. Suppose you decide to deploy Windows VMs running SQL in the back-end. Then the value for **osType** would be *Windows*, and the **dbVMSetting** variable would contain the element listed below, which represents the first value in the **dbVMSettings** variable.

```
"Windows": {  
    "vmSize": "Standard_DS3",  
    "publisher": "MicrosoftSQLServer",  
    "offer": "SQL2014SP1-WS2012R2",  
    "sku": "Standard",  
    "version": "latest",  
    "vmName": "DB",  
    "osdisk": "osdiskdb",  
    "datadisk": "datadiskdb",  
    "nicName": "NICDB",  
    "ipAddress": "192.168.2.",  
    "extensionDeployment": "",  
    "avsetName": "ASDB",  
    "remotePort": 3389,  
    "dbPort": 1433  
},
```

6. Notice the **vmSize** contains the value *Standard_DS3*. Only certain VM sizes allow for the use of multiple NICs. You can verify which VM sizes support multiple NICs by reading the [Windows VM sizes](#) and [Linux VM](#)

sizes articles.

7. Scroll down to **resources** and notice the first element. It describes a storage account. This storage account will be used to maintain the data disks used by each database VM. In this scenario, each database VM has an OS disk stored in regular storage, and two data disks stored in SSD (premium) storage.

```
{  
  "apiVersion": "2015-05-01-preview",  
  "type": "Microsoft.Storage/storageAccounts",  
  "name": "[parameters('prmStorageName')]",  
  "location": "[variables('location')]",  
  "tags": {  
    "displayName": "Storage Account - Premium"  
  },  
  "properties": {  
    "accountType": "[parameters('prmStorageType')]"  
  }  
},
```

8. Scroll down to the next resource, as listed below. This resource represents the NIC used for database access in each database VM. Notice the use of the **copy** function in this resource. The template allows you to deploy as many VMs as you want, based on the **dbCount** parameter. Therefore you need to create the same amount of NICs for database access, one for each VM.

```
{  
  "apiVersion": "2015-06-15",  
  "type": "Microsoft.Network/networkInterfaces",  
  "name": "[concat(variables('dbVMSetting').nicName,'-DA-', copyindex(1))]",  
  "location": "[variables('location')]",  
  "tags": {  
    "displayName": "NetworkInterfaces - DB DA"  
  },  
  "copy": {  
    "name": "dbniccount",  
    "count": "[parameters('dbCount')]"  
  },  
  "properties": {  
    "ipConfigurations": [  
      {  
        "name": "ipconfig1",  
        "properties": {  
          "privateIPAllocationMethod": "Static",  
          "privateIPAddress": "[concat(variables('dbVMSetting').ipAddress,copyindex(4))]",  
          "subnet": {  
            "id": "[variables('backEndSubnetRef')]"  
          }  
        }  
      }  
    ]  
  },  
},
```

9. Scroll down to the next resource, as listed below. This resource represents the NIC used for management in each database VM. Once again, you need one of these NICs for each database VM. Notice the **networkSecurityGroup** element, linking an NSG that allows access to RDP/SSH to this NIC only.

```
{
  "apiVersion": "2015-06-15",
  "type": "Microsoft.Network/networkInterfaces",
  "name": "[concat(variables('dbVMSetting').nicName, '-RA-', copyindex(1))]",
  "location": "[variables('location')]",
  "tags": {
    "displayName": "NetworkInterfaces - DB RA"
  },
  "copy": {
    "name": "dbniccount",
    "count": "[parameters('dbCount')]"
  },
  "properties": {
    "ipConfigurations": [
      {
        "name": "ipconfig1",
        "properties": {
          "networkSecurityGroup": {
            "id": "[resourceId('Microsoft.Network/networkSecurityGroups',
parameters('remoteAccessNSGName'))]"
          },
          "privateIPAllocationMethod": "Static",
          "privateIPAddress": "[concat(variables('dbVMSetting').ipAddress,copyindex(54))]",
          "subnet": {
            "id": "[variables('backEndSubnetRef')]"
          }
        }
      }
    ]
  }
},
```

10. Scroll down to the next resource, as listed below. This resource represents an availability set to be shared by all database VMs. That way, you guarantee that there will always be one VM in the set running during maintenance.

```
{
  "apiVersion": "2015-06-15",
  "type": "Microsoft.Compute/availabilitySets",
  "name": "[variables('dbVMSetting').avsetName]",
  "location": "[variables('location')]",
  "tags": {
    "displayName": "AvailabilitySet - DB"
  }
},
```

11. Scroll down to the next resource. This resource represents the database VMs, as seen in the first few lines listed below. Notice the use of the **copy** function again, ensuring that multiple VMs are created based on the **dbCount** parameter. Also notice the **dependsOn** collection. It lists two NICs being necessary to be created before the VM is deployed, along with the availability set, and the storage account.

```

"apiVersion": "2015-06-15",
"type": "Microsoft.Compute/virtualMachines",
"name": "[concat(variables('dbVMSetting').vmName,copyindex(1))]",
"location": "[variables('location')]",
"dependsOn": [
    "[concat('Microsoft.Network/networkInterfaces/', variables('dbVMSetting').nicName,'-DA-',copyindex(1))]",
    "[concat('Microsoft.Network/networkInterfaces/', variables('dbVMSetting').nicName,'-RA-',copyindex(1))]",
    "[concat('Microsoft.Compute/availabilitySets/', variables('dbVMSetting').avsetName)]",
    "[concat('Microsoft.Storage/storageAccounts/', parameters('prmStorageName'))]"
],
"tags": {
    "displayName": "VMs - DB"
},
"copy": {
    "name": "dbvmcount",
    "count": "[parameters('dbCount')]"
},

```

12. Scroll down in the VM resource to the **networkProfile** element, as listed below. Notice that there are two NICs being reference for each VM. When you create multiple NICs for a VM, you must set the **primary** property of one of the NICs to *true*, and the rest to *false*.

```

"networkProfile": {
    "networkInterfaces": [
        {
            "id": "[resourceId('Microsoft.Network/networkInterfaces',
concat(variables('dbVMSetting').nicName,'-DA-',copyindex(1)))]",
            "properties": { "primary": true }
        },
        {
            "id": "[resourceId('Microsoft.Network/networkInterfaces',
concat(variables('dbVMSetting').nicName,'-RA-',copyindex(1)))]",
            "properties": { "primary": false }
        }
    ]
}

```

Deploy the ARM template by using click to deploy

IMPORTANT

Make sure you follow the [pre-requisites](#) steps before following the instructions below.

The sample template available in the public repository uses a parameter file containing the default values used to generate the scenario described above. To deploy this template using click to deploy, follow [this link](#), to the right of **Backend resource group (see documentation)** click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal.

The figure below shows the contents of the new resource group, after deployment.

The screenshot shows the Azure portal interface for a resource group named "IaaSStory-Backend". At the top, there are three buttons: "Settings", "Add", and "Delete". Below this, the "Essentials" section displays the following details:

Subscription name	Subscription ID
<subscription name>	<subscription ID> 7c
Last deployment	Location
10/28/2015 (Succeeded)	West US

A "All settings" link is located at the bottom right of this section. The main area is titled "Summary" and contains a "Resources" list. The listed resources are:

- ASDB
- DB1
- DB2
- NICDB-DA-1
- NICDB-DA-2
- NICDB-RA-1
- NICDB-RA-2
- wtestvnetstorageprm

Deploy the template by using PowerShell

To deploy the template you downloaded by using PowerShell, install and configure PowerShell by completing the steps in the [Install and configure PowerShell](#) article and then complete the following steps:

Run the `New-AzureRmResourceGroup` cmdlet to create a resource group using the template.

```
>New-AzureRmResourceGroup -Name IaaSStory-Backend -Location uswest `  
TemplateFile 'https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/IaaS-Story/11-`  
MultiNIC/azuredeploy.json' `  
-TemplateParameterFile 'https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/IaaS-Story/11-`  
MultiNIC/azuredeploy.parameters.json'
```

Expected output:

```

ResourceGroupName : IaaSStory-Backend
Location         : westus
ProvisioningState : Succeeded
Tags             :
Permissions      :
    Actions  NotActions
    =====  ========
    *
Resources       :
    Name          Type           Location
    ======  =====  =====
    ASDB          Microsoft.Compute/availabilitySets westus
    DB1           Microsoft.Compute/virtualMachines westus
    DB2           Microsoft.Compute/virtualMachines westus
    NICDB-DA-1   Microsoft.Network/networkInterfaces westus
    NICDB-DA-2   Microsoft.Network/networkInterfaces westus
    NICDB-RA-1   Microsoft.Network/networkInterfaces westus
    NICDB-RA-2   Microsoft.Network/networkInterfaces westus
    wtestvnetstorageprm Microsoft.Storage/storageAccounts westus
ResourceId      : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/IaaSStory-Backend

```

Deploy the template by using the Azure CLI

To deploy the template by using the Azure CLI, follow the steps below.

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 config mode` command to switch to Resource Manager mode, as shown below.

```
azure config mode arm
```

The expected output follows:

```
info: New mode is arm
```

3. Open the [parameter file](#), select its contents, and save it to a file in your computer. For this example, we saved the parameters file to *parameters.json*.
4. Run the `azure group deployment create` cmdlet to deploy the new VNet by using the template and parameter files you downloaded and modified above. The list shown after the output explains the parameters used.

```
azure group create -n IaaSStory-Backend -l westus --template-uri
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/IaaS-Story/11-
MultiNIC/azuredeploy.json -e parameters.json
```

Expected output:

```
info: Executing command group create
+ Getting resource group IaaSStory-Backend
+ Creating resource group IaaSStory-Backend
info: Created resource group IaaSStory-Backend
+ Initializing template configurations and parameters
+ Creating a deployment
info: Created template deployment "azuredeploy"
data: Id: /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/IaaSStory-Backend
data: Name: IaaSStory-Backend
data: Location: westus
data: Provisioning State: Succeeded
data: Tags: null
data:
info: group create command OK
```

Create a VM (Classic) with multiple NICs using PowerShell

1/17/2017 • 5 min to read • [Edit on GitHub](#)

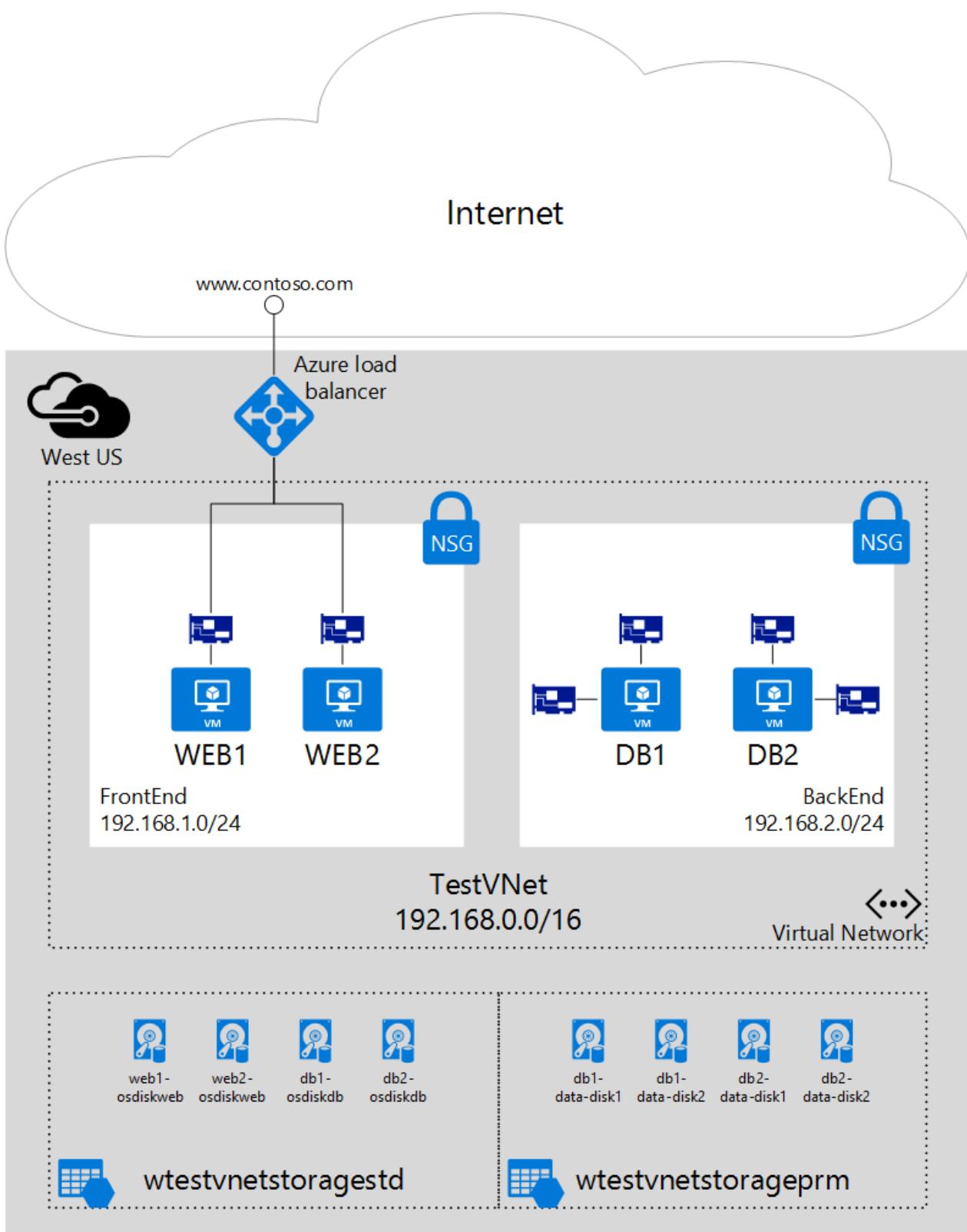
You can create virtual machines (VMs) in Azure and attach multiple network interfaces (NICs) to each of your VMs. Multiple NICs enable separation of traffic types across NICs. For example, one NIC might communicate with the Internet, while another communicates only with internal resources not connected 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 will walk 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. These database servers will be 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 figure below 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'll need to [to install and configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.

NOTE

If you don't have an Azure account, you'll need one. Go sign up for a [free trial here](#).

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
```

3. 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
```

4. 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
```

5. 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.

1. 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++){
```

2. 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
```

3. Provision the VM as a Windows VM.

```
Add-AzureProvisioningConfig -VM $vmConfig -Windows `|  
-AdminUsername $cred.UserName `|  
-Password $cred.Password
```

4. 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
```

5. 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 output below will be displayed.

New-AzureVM	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx	Succeeded
New-AzureVM	xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx	Succeeded

Create a VM (Classic) with multiple NICs using the Azure CLI

1/17/2017 • 5 min to read • [Edit on GitHub](#)

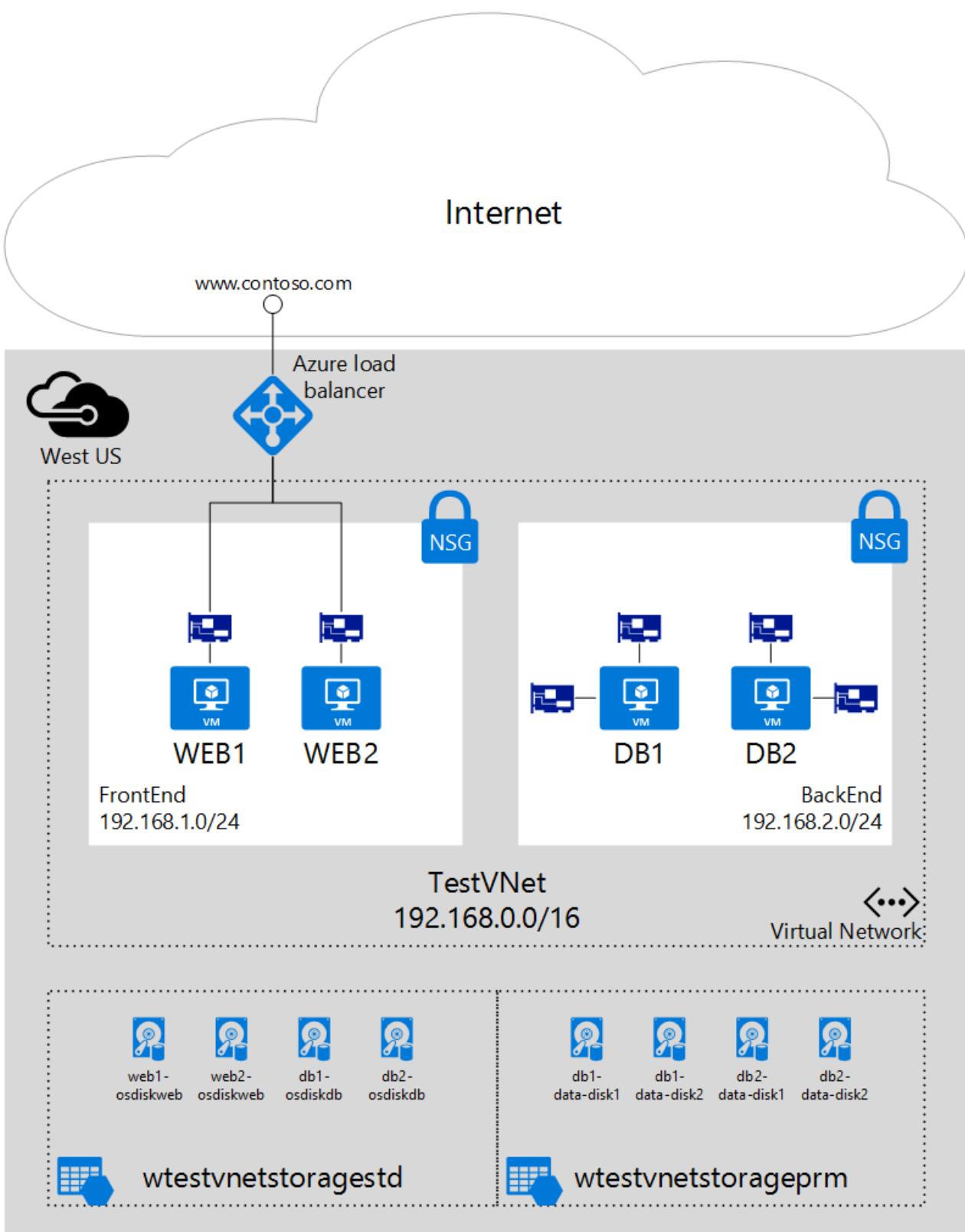
You can create virtual machines (VMs) in Azure and attach multiple network interfaces (NICs) to each of your VMs. Multiple NICs enable separation of traffic types across NICs. For example, one NIC might communicate with the Internet, while another communicates only with internal resources not connected 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 will walk 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. These database servers will be 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 figure below 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
```

Assign multiple IP addresses to virtual machines using the Azure portal

1/17/2017 • 11 min to read • [Edit on GitHub](#)

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.

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.

WARNING

This feature is currently in preview release and may not have the same level of availability and reliability as features that are in general availability release. The feature is not supported, may have constrained capabilities, and may not be available in all Azure locations. For the most up-to-date notifications on availability and status of this feature, check the [Azure Virtual Network updates](#) page.

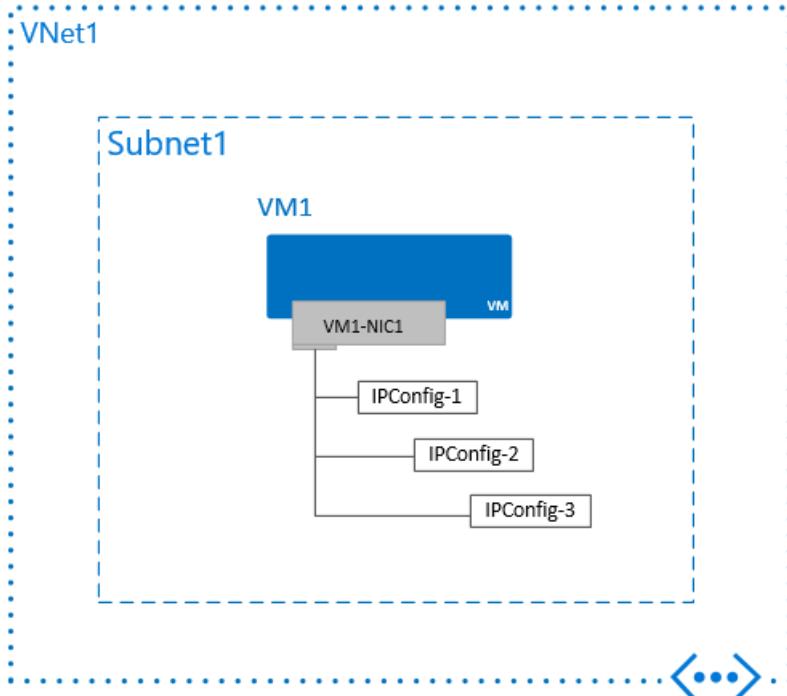
Scenario

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 *dynamic* private IP address (default) and a *static* public IP address.
- **IPConfig-2:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-3:** Assigns a *dynamic* private IP address and no public IP address.



Microsoft
Azure



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, you must create it using PowerShell or the Azure CLI. Click the PowerShell or CLI options at the top of this article to learn how. You can create a VM with a single static private IP address and (optionally) a single public IP address using 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 types 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 a NIC 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.

Core steps

1. Register for the preview by sending an email to [Multiple IPs](#) with your subscription ID and intended use. Do not attempt to complete the remaining steps:
 - Until you receive an e-mail notifying you that you've been accepted into the preview
 - Without following the instructions in the email you receive
2. Browse to the Azure portal at <https://portal.azure.com> and sign into it, if necessary.

3. In the portal, click **More services** > type *virtual machines* in the filter box, and then click **Virtual machines**.
4. In the **Virtual machines** blade, click the VM you want to add IP addresses to. Click **Network interfaces** in the virtual machine blade 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:

NAME	PUBLIC IP ADDRESS	PRIVATE IP ADDRESS	SECURITY GROUP
myNIC	52.161.29.217	10.0.0.4	-

5. In the blade that appears for the NIC you selected, click **IP configurations**, as shown in the following picture:

NAME	TYPE	PRIVATE IP ADDRESS	PUBLIC IP ADDRESS
IPConfig-1	Primary	10.0.0.4 (Dynamic)	52.161.29.217 (myPu...)
IPConfig-2	Secondary	10.0.0.5 (Static)	52.161.16.38 (myPub...)
IPConfig-3	Secondary	10.0.0.6 (Dynamic)	-

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** blade that appears, create an IP configuration named *IPConfig-4* with *10.0.0.7* as a *Static* private IP address then click **OK**, as shown in the following picture:

The screenshot shows the 'Add IP configuration' blade. On the left, there's a summary section with 'IP forwarding settings' (disabled), 'Virtual network' (myVnet), and a table of 'IP configurations' showing three existing entries: 'IPConfig-1' (Primary, Dynamic, 10.0.0.4, 52.161.29.217), 'IPConfig-2' (Secondary, Static, 10.0.0.5, 52.161.16.38), and 'IPConfig-3' (Secondary, Dynamic, 10.0.0.6, -). A search bar for 'Search IP configurations' is also present. On the right, the configuration details for 'IPConfig-4' are being entered. The 'Name' is 'IPConfig-4', 'Type' is 'Secondary', 'Allocation' is 'Static' (selected), 'IP address' is '10.0.0.7' (highlighted with a red box), and 'Public IP address' is 'Disabled'. The 'OK' button at the bottom is highlighted.

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 will show an X for the IP address and you'll need to select a different one.

If you prefer that the private IP address **Allocation method** be *Dynamic*, select it instead and you won't need to specify an IP address.

- Once you click OK, the blade will close and you'll see the new IP configuration listed, as shown in the following picture:

NAME	TYPE	PRIVATE IP ADDRESS	PUBLIC IP ADDRESS
IPConfig-1	Primary	10.0.0.4 (Dynamic)	52.161.29.217 (myPu... ...)
IPConfig-2	Secondary	10.0.0.5 (Static)	52.161.16.38 (myPub... ...)
IPConfig-3	Secondary	10.0.0.6 (Dynamic)	- ...
IPConfig-4	Secondary	10.0.0.7 (Static)	- ...

Click **OK** to close the **Add IP configuration** blade.

- You can click **Add** to add additional IP configurations, or close all open blades to finish adding IP addresses.
- 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 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.

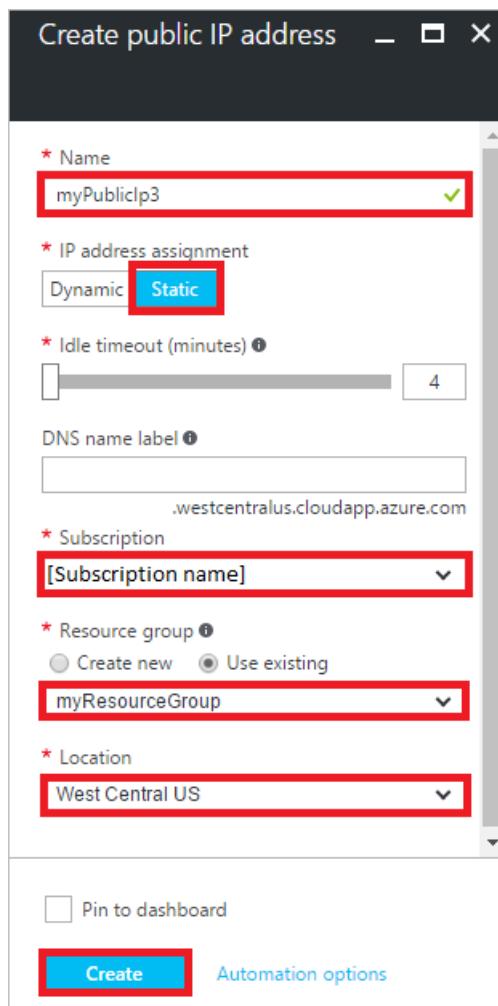
NOTE

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 **New > Networking > Public IP address**.
3. In the **Create public IP address** blade 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** blade 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** blade that appears, as shown in the following picture:

The screenshot shows two blades side-by-side. The left blade is titled 'Add IP configuration' and has a 'myNIC' tab selected. It contains fields for 'Name' (IPConfig-4), 'Type' (Primary/Secondary), and 'Allocation' (Dynamic/Static). Under 'Public IP address', 'Enabled' is selected. A red box highlights the 'IP address' section, which contains a link to 'Configure required settings'. The right blade is titled 'Choose public IP address' and lists three public IP resources: 'myPublicIp3' (selected and highlighted with a red box), 'myPublicIp1', and 'myPublicIp2'. Each resource shows its name, resource group, and IP address.

Once you've selected the public IP address resource, click **OK** and the blade will 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.

- Review the new IP configuration, as shown in the following picture:

NAME	TYPE	PRIVATE IP ADDRESS	PUBLIC IP ADDRESS
IPConfig-1	Primary	10.0.0.4 (Dynamic)	52.161.29.217 (myPu... ...)
IPConfig-2	Secondary	10.0.0.5 (Static)	52.161.16.38 (myPub... ...)
IPConfig-3	Secondary	10.0.0.6 (Dynamic)	- ...
IPConfig-4	Secondary	10.0.0.7 (Dynamic)	52.161.8.58 (myPubli... ...)

NOTE

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.

- You can click **Add** to add additional IP configurations, or close all open blades to finish adding 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. Do not add the public IP address to the operating system.

Associate the public IP address resource to an existing IP configuration

- Complete the steps in the [Core steps](#) section of this article.
- Select the IP configuration you want to add the public IP address resource to, enable the Public IP address, and select an existing, available public IP address resource. In the example shown in the following picture, the *myPublicIp3* public IP address resource is associated to *IPConfig-3*.

Once you've selected the public IP address resource, click **Save** and the blades will 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.

- Review the new IP configuration, as shown in the following picture:

NAME	TYPE	PRIVATE IP ADDRESS	PUBLIC IP ADDRESS
IPConfig-1	Primary	10.0.0.4 (Dynamic)	52.161.29.217 (myPu... ...)
IPConfig-2	Secondary	10.0.0.5 (Static)	52.161.16.38 (myPub... ...)
IPConfig-3	Secondary	10.0.0.6 (Dynamic)	52.161.8.58 (myPubli... ...)
IPConfig-4	Secondary	10.0.0.7 (Dynamic)	- ...

- 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 following steps 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 **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.
- Click **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.
- Click the **Advanced** button and add additional IP addresses. Add each of the secondary private IP addresses listed in step 8 to the NIC with the same subnet specified for the primary IP address.
- 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.

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: `vi filename`.

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>
```

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. Copy the *ifcfg-eth0* file and name it *ifcfg-eth0:0* with the following command:

```
cp ifcfg-eth0 ifcfg-eth0:0
```

6. Edit the *ifcfg-eth0:0* file with the following command:

```
vi ifcfg-eth1
```

7. Change the device to the appropriate name in the file; *eth0:0* in this case, with the following command:

```
DEVICE=eth0:0
```

8. Change the *IPADDR = YourPrivateIPAddress* line to reflect the IP address.

9. Save the file with the following command:

```
:wq
```

10. Restart the network services and make sure the changes are successful by running the following commands:

```
/etc/init.d/network restart  
Ipconfig
```

You should see the IP address you added, *eth0:0*, in the list returned.

Assign multiple IP addresses to virtual machines using PowerShell

1/17/2017 • 14 min to read • [Edit on GitHub](#)

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.

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.

WARNING

This feature is currently in preview release and may not have the same level of availability and reliability as features that are in general availability release. The feature is not supported, may have constrained capabilities, and may not be available in all [Azure locations](#). For the most up-to-date notifications on availability and status of this feature, check the [Azure Virtual Network updates](#) page.

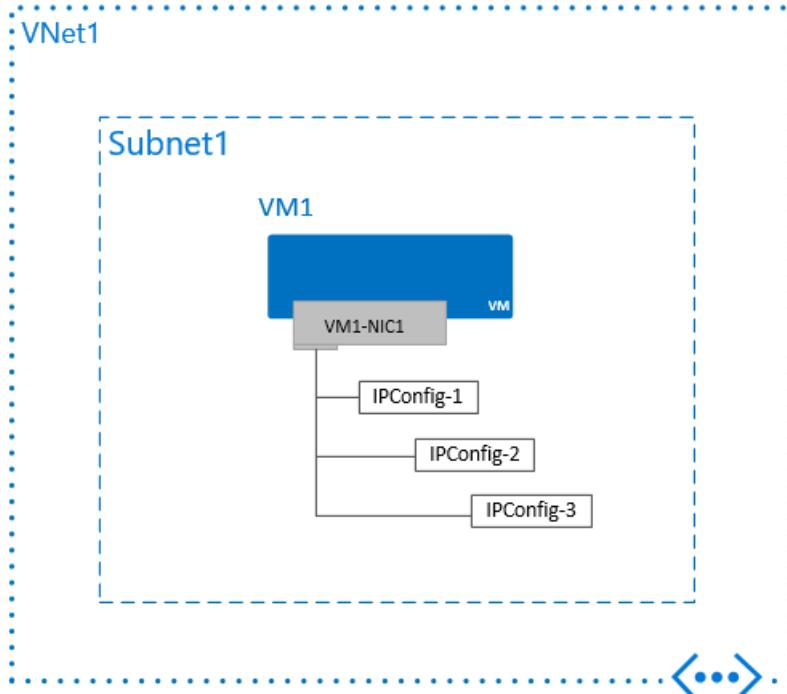
Scenario

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 *dynamic* private IP address (default) and a *static* public IP address.
- **IPConfig-2:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-3:** Assigns a *dynamic* private IP address and no public IP address.



Microsoft
Azure



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 names and IP address types 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. Register for the preview by sending an email to [Multiple IPs](#) with your subscription ID and intended use. Do not attempt to complete the remaining steps:
 - Until you receive an e-mail notifying you that you've been accepted into the preview
 - Without following the instructions in the email you receive
3. Complete steps 1-4 of the [Create a Windows VM](#) article. Do not complete step 5 (creation of public IP resource and network interface). If you change the names of any variables used in that article, change the names of the variables in the remaining steps too. To create a Linux VM, select a Linux operating system instead of Windows.
4. Create a variable to store the subnet object created in Step 4 (Create a VNet) of the Create a Windows VM article by typing the following command:

```
$SubnetName = $mySubnet.Name  
$Subnet = $myVnet.Subnets | Where-Object { $_.Name -eq $SubnetName }
```

5. Define the IP configurations you want to assign to the NIC. You can add, remove, or change the configurations as necessary. The following configurations are described in the scenario:

IPConfig-1

Enter the commands that follow to create:

- A public IP address resource with a static public IP address
- An IP configuration with the public IP address resource and a dynamic private IP address

```
$myPublicIp1      = New-AzureRmPublicIpAddress -Name "myPublicIp1" -ResourceGroupName  
$myResourceGroup -Location $location -AllocationMethod Static  
$IpConfigName1   = "IPConfig-1"  
$IpConfig1       = New-AzureRmNetworkInterfaceIpConfig -Name $IpConfigName1 -Subnet $Subnet -  
PublicIpAddress $myPublicIp1 -Primary
```

Note the `-Primary` switch in the previous command. 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.

IPConfig-2

Change the value of the **\$IPAddress** variable that follows to an available, valid address on the subnet you created. To check whether the address 10.0.0.5 is available on the subnet, enter the command

```
Test-AzureRmPrivateIPAddressAvailability -IPAddress 10.0.0.5 -VirtualNetwork $myVnet
```

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. Enter the following commands to create a new public IP address resource and a new IP configuration with a static public IP address and a static private IP address:

```
$IpConfigName2 = "IPConfig-2"  
$IPAddress    = 10.0.0.5  
$myPublicIp2  = New-AzureRmPublicIpAddress -Name "myPublicIp2" -ResourceGroupName $myResourceGroup  
`  
-Location $location -AllocationMethod Static  
$IpConfig2     = New-AzureRmNetworkInterfaceIpConfig -Name $IpConfigName2 `  
-Subnet $Subnet -PrivateIpAddress $IPAddress -PublicIpAddress $myPublicIp2
```

IPConfig-3

Enter the following commands to create an IP configuration with a dynamic private IP address and no public IP address:

```
$IpConfigName3 = "IpConfig-3"  
$IpConfig3    = New-AzureRmNetworkInterfaceIpConfig -Name $IPConfigName3 -Subnet $Subnet
```

6. Create the NIC using the IP configurations defined in the previous step by entering the following command:

```
$myNIC = New-AzureRmNetworkInterface -Name myNIC -ResourceGroupName $myResourceGroup  
-Location $location -IpConfiguration $IpConfig1,$IpConfig2,$IpConfig3
```

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.

7. Complete step 6 of the [Create a VM](#) article.

WARNING

Step 6 in the Create a VM article fails if:

- You changed the variable named \$myNIC to something else in step 6 of this article.
- You haven't completed the previous steps of this article and the [Create a VM](#) article.

8. Enter the following command to view the private IP addresses and public IP address resources assigned to the NIC:

```
$myNIC.IpConfigurations | Format-Table Name, PrivateIPAddress, PublicIPAddress, Primary
```

9. 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 a NIC 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. Register for the preview by sending an email to [Multiple IPs](#) with your subscription ID and intended use. Do not attempt to complete the remaining steps:
 - Until you receive an e-mail notifying you that you've been accepted into the preview
 - Without following the instructions in the email you receive
3. 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"  
$myResourceGroup = "myResourceGroup"  
$location      = "westcentralus"
```

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
```

4. Create a variable and set it to the existing NIC by typing the following command:

```
$myNIC = Get-AzureRmNetworkInterface -Name $NICname -ResourceGroupName $myResourceGroup
```

5. 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 $myResourceGroup  
$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
```

Look for text similar to the following text in the returned output:

```
Subnet : {  
    "Id":  
        "/subscriptions/[Id]/resourceGroups/myResourceGroup/providers/Microsoft.Network/virtualNetworks/myVnet/su  
        bnets/mySubnet"
```

In this output, *myVnet* is the VNet and *mySubnet* is the subnet the NIC is connected to.

6. 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. If you want to add a dynamic private IP address, remove `-PrivateIpAddress 10.0.0.7` before entering the command. 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 $myResourceGroup ` -Location $location -AllocationMethod Static
```

To create a new IP configuration with a dynamic private IP address and the associated *myPublicIp3* public IP address resource, enter the following command:

```
Add-AzureRmNetworkInterfaceIpConfig -Name IPConfig-4 -NetworkInterface `$myNIC -Subnet $Subnet -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
```

Look for a line similar to the one that follows in the returned output:

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 $myResourceGroup ` -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
```

7. Set the NIC with the new IP configuration by entering the following command:

```
Set-AzureRmNetworkInterface -NetworkInterface $myNIC
```

8. 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
```

9. 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 following steps 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 **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.
 - Click **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.
6. From a command prompt, type `ipconfig /all`. All IP addresses you added are shown and DHCP is turned off.

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: `vi filename`.

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>
```

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. Copy the *ifcfg-eth0* file and name it *ifcfg-eth0:0* with the following command:

```
cp ifcfg-eth0 ifcfg-eth0:0
```

6. Edit the *ifcfg-eth0:0* file with the following command:

```
vi ifcfg-eth1
```

7. Change the device to the appropriate name in the file; *eth0:0* in this case, with the following command:

```
DEVICE=eth0:0
```

8. Change the *IPADDR = YourPrivateIPAddress* line to reflect the IP address.

9. Save the file with the following command:

```
:wq
```

10. Restart the network services and make sure the changes are successful by running the following commands:

```
/etc/init.d/network restart  
Ipconfig
```

You should see the IP address you added, *eth0:0*, in the list returned.

Assign multiple IP addresses to virtual machines using Azure CLI

1/17/2017 • 12 min to read • [Edit on GitHub](#)

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.

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.

WARNING

This feature is currently in preview release and may not have the same level of availability and reliability as features that are in general availability release. The feature is not supported, may have constrained capabilities, and may not be available in all Azure locations. For the most up-to-date notifications on availability and status of this feature, check the [Azure Virtual Network updates](#) page.

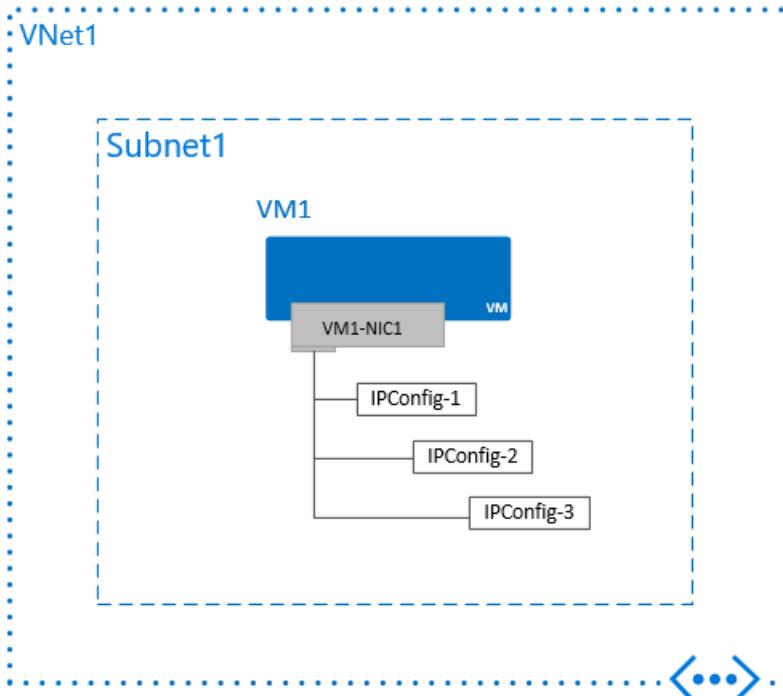
Scenario

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 *dynamic* private IP address (default) and a *static* public IP address.
- **IPConfig-2:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-3:** Assigns a *dynamic* private IP address and no public IP address.



Microsoft
Azure



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 names and IP address types as required for your implementation.

1. Install and configure the Azure CLI by following the steps in the [Install and Configure the Azure CLI](#) article and log into your Azure account.
2. Register for the preview by sending an email to [Multiple IPs](#) with your subscription ID and intended use. Do not attempt to complete the remaining steps:
 - Until you receive an e-mail notifying you that you've been accepted into the preview
 - Without following the instructions in the email you receive
3. [Create a resource group](#) followed by a [virtual network and subnet](#). Change the `--address-prefixes` and `--address-prefix` fields to the following to follow the exact scenario outlined in this article:

```
--address-prefixes 10.0.0.0/16
--address-prefix 10.0.0.0/24
```

NOTE

The referenced article above uses West Europe as the location to create resources, but this article uses West Central US. Make location changes appropriately.

4. [Create a storage account](#) for your VM.
5. Create the NIC and the IP configurations you want to assign to the NIC. You can add, remove, or change the configurations as necessary. The following configurations are described in the scenario:

IPConfig-1

Enter the commands that follow to create:

- A public IP address resource with a static public IP address
- An IP configuration with the public IP address resource and a dynamic private IP address

```
azure network public-ip create --resource-group myResourceGroup --location westcentralus --name myPublicIP --domain-name-label mypublicdns --allocation-method Static
```

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.

```
azure network nic create --resource-group myResourceGroup --location westcentralus --subnet-vnet-name myVnet --subnet-name mySubnet --name myNic1 --public-ip-name myPublicIP
```

IPConfig-2

Enter the following commands to create a new public IP address resource and a new IP configuration with a static public IP address and a static private IP address:

```
azure network public-ip create --resource-group myResourceGroup --location westcentralus --name myPublicIP2 --domain-name-label mypublicdns2 --allocation-method Static

azure network nic ip-config create --resource-group myResourceGroup --nic-name myNic1 --name IPConfig-2 --private-ip-address 10.0.0.5 --public-ip-name myPublicIP2
```

IPConfig-3

Enter the following commands to create an IP configuration with a dynamic private IP address and no public IP address:

```
azure network nic ip-config create --resource-group myResourceGroup --nic-name myNic1 --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.

6. Create a Linux VM article. Be sure to remove the `--availset-name myAvailabilitySet` property as it is not required for this scenario. Use the appropriate location based on your scenario.

WARNING

Step 6 in the Create a VM article fails if the VM size is not supported in the location you selected. Run the following command to get a full list of VMs in US West Central, for example: `azure vm sizes --location westcentralus`. This location name can be changed based on your scenario.

To change the VM size to Standard DS2 v2, for example, simply add the following property

`--vm-size Standard_DS3_v2` to the `azure vm create` command in step 6.

7. Enter the following command to view the NIC and the associated IP configurations:

```
azure network nic show --resource-group myResourceGroup --name myNic1
```

8. 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 NIC by completing the steps that follow. The examples build upon the [scenario](#) described in this article.

1. Open Azure CLI and complete the remaining steps in this section within a single CLI session. If you don't already have Azure CLI installed and configured, complete the steps in the [Install and Configure the Azure CLI](#) article and log into your Azure account.
2. Register for the preview by sending an email to [Multiple IPs](#) with your subscription ID and intended use. Do not attempt to complete the remaining steps:
 - Until you receive an e-mail notifying you that you've been accepted into the preview
 - Without following the instructions in the email you receive
3. 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 below. If you want to add a dynamic private IP address, remove `--private-ip-address 10.0.0.7` before entering the command. When specifying a static IP address, it must be an unused address for the subnet.

```
azure 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.

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 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:

```
azure network public-ip create --resource-group myResourceGroup --location westcentralus --name myPublicIP3 --domain-name-label mypublicdns3
```

To create a new IP configuration with a dynamic private IP address and the associated *myPublicIP3* public IP address resource, enter the following command:

```
azure network nic ip-config create --resource-group myResourceGroup --nic-name myNic --name IPConfig-4 - -public-ip-name 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:

```
azure network nic ip-config list --resource-group myResourceGroup --nic-name myNic1
```

Look for a line similar to the one that follows in the returned output:

Name	Provisioning state	Primary	Private IP allocation	Private IP version	Private IP
address	Subnet	Public IP			
default-ip-config	Succeeded	true	Dynamic	IPv4	10.0.0.4
mySubnet	myPublicIP				
IPConfig-2	Succeeded	false	Static	IPv4	10.0.0.5
mySubnet	myPublicIP2				
IPConfig-3	Succeeded	false	Dynamic	IPv4	10.0.0.6
mySubnet					

Since the **Public IP** 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:

```
azure network public-ip create --resource-group myResourceGroup --location westcentralus --name myPublicIP3 --domain-name-label mypublicdns3 --allocation-method Static
```

Enter the following command to associate the public IP address resource to the existing IP configuration named *IPConfig-3*:

```
azure network nic ip-config set --resource-group myResourceGroup --nic-name myNic1 --name IPConfig-3 --public-ip-name myPublicIP3
```

4. View the private IP addresses and the public IP address resources assigned to the NIC by entering the following command:

```
azure network nic ip-config list --resource-group myResourceGroup --nic-name myNic1
```

You should see output similar to the following:

Name	Provisioning state	Primary address	Private IP allocation	Private IP version	Private IP
address	Subnet	Public IP			
default-ip-config	Succeeded	mySubnet	true	Dynamic	IPv4
IPConfig-2	Succeeded	mySubnet	false	Static	IPv4
IPConfig-3	Succeeded	mySubnet	false	Dynamic	IPv4
		myPublicIP			10.0.0.4
		myPublicIP2			10.0.0.5
		myPublicIP3			10.0.0.6

5. 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 following steps 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 **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.
 - Click **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.
6. From a command prompt, type `ipconfig /all`. All IP addresses you added are shown and DHCP is turned off.

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: vi *filename*.

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>
```

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. Copy the *ifcfg-eth0* file and name it *ifcfg-eth0:0* with the following command:

```
cp ifcfg-eth0 ifcfg-eth0:0
```

6. Edit the *ifcfg-eth0:0* file with the following command:

```
vi ifcfg-eth1
```

7. Change the device to the appropriate name in the file; *eth0:0* in this case, with the following command:

```
DEVICE=eth0:0
```

8. Change the *IPADDR = YourPrivateIPAddress* line to reflect the IP address.

9. Save the file with the following command:

```
:wq
```

10. Restart the network services and make sure the changes are successful by running the following commands:

```
/etc/init.d/network restart  
Ipconfig
```

You should see the IP address you added, *eth0:0*, in the list returned.

Assign multiple IP addresses to virtual machines using an Azure Resource Manager template

1/17/2017 • 12 min to read • [Edit on GitHub](#)

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.

This article explains how to create a virtual machine (VM) through the Azure Resource Manager deployment model using a Resource Manager template. Multiple public and private IP addresses cannot be assigned to the same NIC when deploying a VM through the classic deployment model. To learn more about Azure deployment models, read the [Understand deployment models](#) article.

WARNING

This feature is currently in preview release and may not have the same level of availability and reliability as features that are in general availability release. The feature is not supported, may have constrained capabilities, and may not be available in all [Azure locations](#). For the most up-to-date notifications on availability and status of this feature, check the [Azure Virtual Network updates page](#).

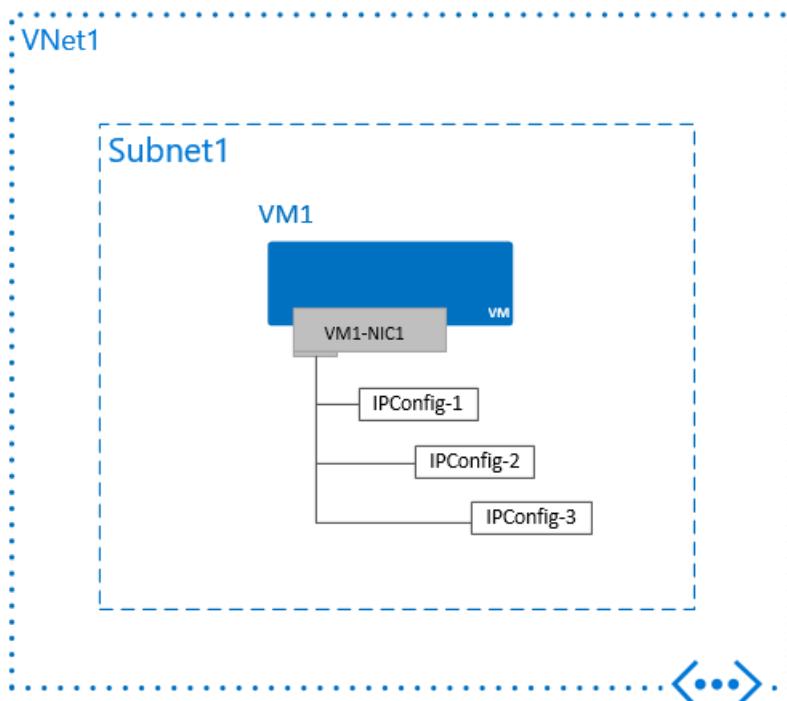
Scenario

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 *dynamic* private IP address (default) and a *static* public IP address.
- **IPConfig-2:** Assigns a *static* private IP address and a *static* public IP address.
- **IPConfig-3:** Assigns a *dynamic* private IP address and no public IP address.



Microsoft
Azure



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.

Template description

Deploying a template enables you to quickly and consistently create Azure resources with different configuration values. Read the [Resource Manager template walkthrough](#) article if you're not familiar with Azure Resource Manager templates. The [Deploy a VM with multiple IP addresses](#) template is utilized in this article.

Deploying the template creates the following resources:

RESOURCE	NAME	DESCRIPTION
Network interface	<i>myNic1</i>	The three IP configurations described in the scenario section of this article are created and assigned to this NIC.
Public IP address resource	2 are created: <i>myPublicIP</i> and <i>myPublicIP2</i>	These resources are assigned static public IP addresses and are assigned to the <i>IPConfig-1</i> and <i>IPConfig-2</i> IP configurations described in the scenario.
VM	<i>myVM1</i>	A Standard DS3 VM.

RESOURCE	NAME	DESCRIPTION
Virtual network	<i>myVNet1</i>	A virtual network with one subnet named <i>mySubnet</i> .
Storage account	Unique to the deployment	A storage account.

When deploying the template, you must specify values for the following parameters:

NAME	DESCRIPTION
adminUsername	Admin username. The username must comply with Azure username requirements .
adminPassword	Admin password The password must comply with Azure password requirements .
dnsLabelPrefix	DNS name for PublicIPAddressName1. The DNS name will resolve to one of the public IP addresses assigned to the VM. The name must be unique within the Azure region (location) you create the VM in.
dnsLabelPrefix1	DNS name for PublicIPAddressName2. The DNS name will resolve to one of the public IP addresses assigned to the VM. The name must be unique within the Azure region (location) you create the VM in.
OSVersion	The Windows/Linux version for the VM. The operating system is a fully patched image of the given Windows/Linux version selected.
imagePublisher	The Windows/Linux image publisher for the selected VM.
imageOffer	The Windows/Linux image for the selected VM.

Each of the resources deployed by the template is configured with several default settings. You can view these settings through either of the following methods:

- **View the template on GitHub:** If you're familiar with templates, you can view the settings within the [template](#).
- **View the settings after deploying:** If you're not familiar with templates, you can deploy the template using steps in one of the following sections and then view the settings after deployment.

You can use the Azure portal, PowerShell, or the Azure command-line interface (CLI) to deploy the template. All methods produce the same result. To deploy the template, complete the steps in one of the following sections :

Deploy using the Azure portal

To deploy the template using the Azure portal, complete the following steps:

1. Register for the preview by sending an email to [Multiple IPs](#) with your subscription ID and intended use. Do not attempt to complete the remaining steps:
 - Until you receive an e-mail notifying you that you've been accepted into the preview
 - Without following the instructions in the email you receive
2. Modify the template, if desired. The template deploys the resources and settings listed in the [resources](#) section

of this article. To learn more about templates and how to author them, read the [Authoring Azure Resource Manager templates](#) article.

3. Deploy the template with one of the following methods:

- **Select the template in the portal:** Complete the steps in the [Deploy resources from custom template](#) article. Choose the pre-existing template named *101-vm-multiple-ipconfig*.
- **Directly:** Click the following button to open the template directly in the portal: 

Regardless of the method you choose, you'll need to supply values for the [parameters](#) listed previously in this article. After the VM is deployed, connect to the VM and add the private IP addresses to the operating system you deployed by completing the steps 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.

Deploy using PowerShell

To deploy the template using PowerShell, complete the following steps:

1. Register for the preview by sending an email to [Multiple IPs](#) with your subscription ID and intended use. Do not attempt to complete the remaining steps:
 - Until you receive an e-mail notifying you that you've been accepted into the preview
 - Without following the instructions in the email you receive
2. Deploy the template by completing the steps in the [Deploy a template with PowerShell](#) article. The article describes multiple options for deploying a template. If you choose to deploy using the `-TemplateUri` parameter, the URI for this template is <https://raw.githubusercontent.com/azure/azure-quickstart-templates/master/101-vm-multiple-ipconfig/azureddeploy.json>. If you choose to deploy using the `-TemplateFile` parameter, copy the contents of the [template file](#) from GitHub into a new file on your machine. Modify the template contents, if desired. The template deploys the resources and settings listed in the [resources](#) section of this article. To learn more about templates and how to author them, read the [Authoring Azure Resource Manager templates](#) article.

Regardless of the option you choose to deploy the template with, you must supply values for the parameter values listed in the [parameters](#) section of this article. If you choose to supply parameters using a parameters file, copy the contents of the [parameters file](#) from GitHub into a new file on your computer. Modify the values in the file. Use the file you created as the value for the `-TemplateParameterFile` parameter.

To determine valid values for the OSVersion, ImagePublisher, and imageOffer parameters, complete the steps in the [Navigate and select Windows VM images article](#).

TIP

If you're not sure whether a dnslabelprefix is available, enter the

```
Test-AzureRmDnsAvailability -DomainNameLabel <name-you-want-to-use> -Location <location>
```

 command to find out. If it is available, the command will return `True`.

3. After the VM is deployed, connect to the VM and add the private IP addresses to the operating system you deployed by completing the steps 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.

Deploy using the Azure CLI

To deploy the template using the Azure CLI 1.0, complete the following steps:

1. Register for the preview by sending an email to [Multiple IPs](#) with your subscription ID and intended use. Do not attempt to complete the remaining steps:

- Until you receive an e-mail notifying you that you've been accepted into the preview
 - Without following the instructions in the email you receive
2. Deploy the template by completing the steps in the [Deploy a template with the Azure CLI](#) article. The article describes multiple options for deploying the template. If you choose to deploy using the `--template-uri` (-f), the URI for this template is <https://raw.githubusercontent.com/azure/azure-quickstart-templates/master/101-vm-multiple-ipconfig/azuredeploy.json>. If you choose to deploy using the `--template-file` (-f) parameter, copy the contents of the [template file](#) from GitHub into a new file on your machine. Modify the template contents, if desired. The template deploys the resources and settings listed in the [resources](#) section of this article. To learn more about templates and how to author them, read the [Authoring Azure Resource Manager templates](#) article.

Regardless of the option you choose to deploy the template with, you must supply values for the parameter values listed in the [parameters](#) section of this article. If you choose to supply parameters using a parameters file, copy the contents of the [parameters file](#) from GitHub into a new file on your computer. Modify the values in the file. Use the file you created as the value for the `--parameters-file` (-e) parameter.

To determine valid values for the OSVersion, ImagePublisher, and imageOffer parameters, complete the steps in the [Navigate and select Windows VM images article](#) article.

3. After the VM is deployed, connect to the VM and add the private IP addresses to the operating system you deployed by completing the steps 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 following steps 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 **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.
 - Click **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.
6. From a command prompt, type `ipconfig /all`. All IP addresses you added are shown and DHCP is turned off.

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: vi *filename*.

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>
```

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. Copy the *ifcfg-eth0* file and name it *ifcfg-eth0:0* with the following command:

```
cp ifcfg-eth0 ifcfg-eth0:0
```

6. Edit the *ifcfg-eth0:0* file with the following command:

```
vi ifcfg-eth1
```

7. Change the device to the appropriate name in the file; *eth0:0* in this case, with the following command:

```
DEVICE=eth0:0
```

8. Change the *IPADDR = YourPrivateIPAddress* line to reflect the IP address.

9. Save the file with the following command:

```
:wq
```

10. Restart the network services and make sure the changes are successful by running the following commands:

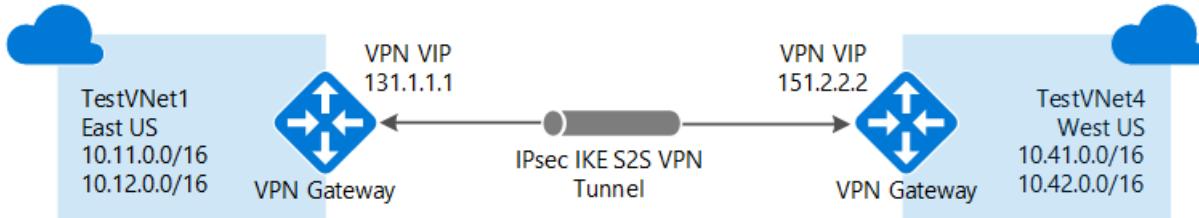
```
/etc/init.d/network restart  
Ipconfig
```

You should see the IP address you added, *eth0:0*, in the list returned.

Configure a VNet-to-VNet connection for Resource Manager using PowerShell

1/17/2017 • 15 min to read • [Edit on GitHub](#)

This article walks you through the steps to create a connection between VNets in the Resource Manager deployment model by using VPN Gateway. The virtual networks can be in the same or different regions, and from the same or different subscriptions.



Deployment models and methods for VNet-to-VNet connections

It's important to understand that Azure currently works with two deployment models: Resource Manager and classic. Before you begin your configuration, verify that you are using the instructions for the deployment model that you want to work in. The two models are not completely compatible with each other.

For example, if you are working with a virtual network that was created using the classic deployment model and wanted to add a connection to the VNet, you would use the deployment methods that correspond to the classic deployment model, not Resource Manager. If you are working with a virtual network that was created using the Resource Manager deployment model, you would use the deployment methods that correspond with Resource Manager, not classic.

For information about the deployment models, see [Understanding Resource Manager deployment and classic deployment](#).

The following table shows the currently available deployment models and methods for VNet-to-VNet configurations. When an article with configuration steps is available, we link directly to it from this table.

DEPLOYMENT MODEL/METHOD	AZURE PORTAL	CLASSIC PORTAL	POWERSHELL
Classic	Not Supported	Article*	Supported
Resource Manager	Article+	Not Supported	Article
Connections between different deployment models	Article*	Article*	Article

(+) denotes this deployment method is available only for VNets in the same subscription.

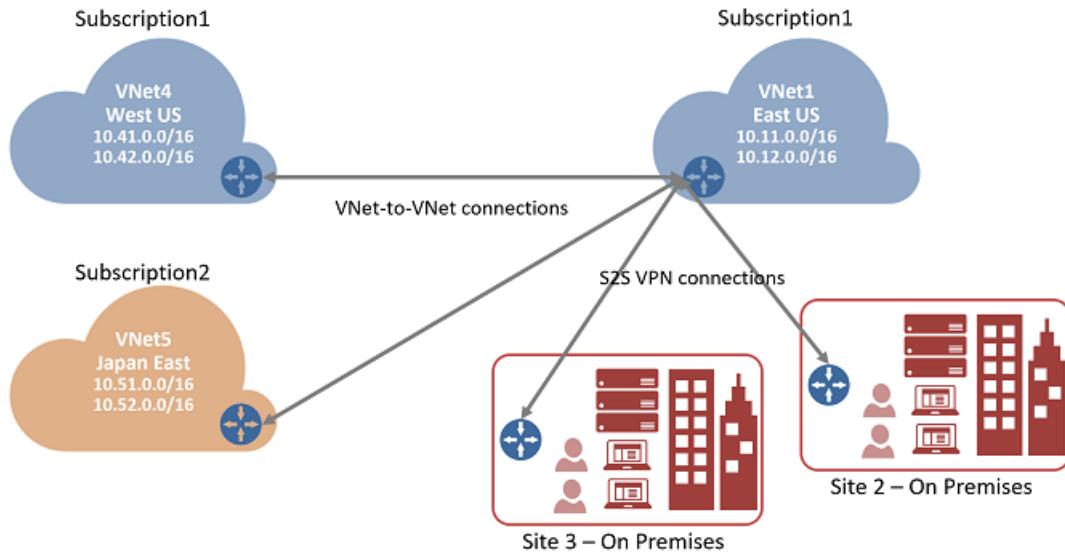
(*) denotes that this deployment method also requires PowerShell.

VNet peering

It's also possible to connect VNets without using a VPN gateway. If your VNets are in the same region, you may want to consider connecting them by using VNet peering. For more information, see the [VNet peering](#) article.

About VNet-to-VNet connections

Connecting a virtual network to another virtual network (VNet-to-VNet) is similar to connecting a VNet to an on-premises site location. Both connectivity types use an Azure VPN gateway to provide a secure tunnel using IPsec/IKE. The VNets you connect can be in different regions. Or in different subscriptions. You can even combine VNet-to-VNet communication with multi-site configurations. This lets you establish network topologies that combine cross-premises connectivity with inter-virtual network connectivity, as shown in the following diagram:



Why connect virtual networks?

You may want to connect virtual networks 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 FAQ

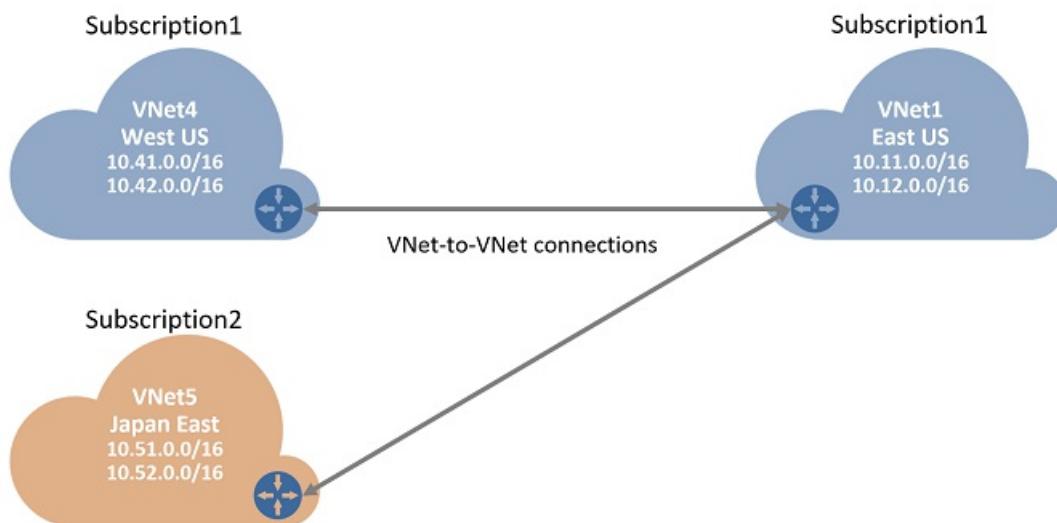
- The virtual networks can be in the same or different Azure regions (locations).
- A cloud service or a load balancing endpoint CANNOT span across virtual networks, even if they are connected together.
- Connecting multiple Azure virtual networks together doesn't require any on-premises VPN gateways unless cross-premises connectivity is required.
- VNet-to-VNet supports connecting virtual networks. It does not support connecting virtual machines or cloud services NOT in a virtual network.
- VNet-to-VNet requires Azure VPN gateways with RouteBased (previously called Dynamic Routing) VPN types.
- Virtual network connectivity can be used simultaneously with multi-site VPNs. There is a maximum of 10 (Default/Standard Gateways) or 30 (HighPerformance Gateways) VPN tunnels for a virtual network VPN gateway connecting to either other virtual networks, or on-premises sites.
- The address spaces of the virtual networks and on-premises local network sites must not overlap. Overlapping address spaces will cause the creation of VNet-to-VNet connections to fail.

- Redundant tunnels between a pair of virtual networks are supported when one virtual network gateway is configured as active-active.
- 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.
- VNet-to-VNet traffic travels across the Microsoft Network, not the Internet.
- VNet-to-VNet traffic within the same region is free for both directions. Cross region VNet-to-VNet egress traffic is charged with the outbound inter-VNet data transfer rates based on the source regions. Please refer to the [pricing page](#) for details.

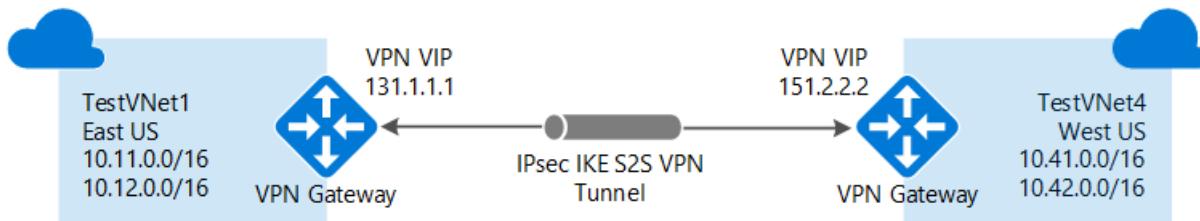
Which set of steps should I use?

In this article, you see two different sets of steps. One set of steps for [VNets that reside in the same subscription](#), and another for [VNets that reside in different subscriptions](#). The key difference between the sets is whether you can create and configure all virtual network and gateway resources within the same PowerShell session.

The steps in this article use variables that are declared at the beginning of each section. If you already are working with existing VNets, modify the variables to reflect the settings in your own environment.



How to connect VNets that are in the same subscription



Before you begin

Before beginning, you need to install the Azure Resource Manager PowerShell cmdlets. See [How to install and configure Azure PowerShell](#) for more information about installing the PowerShell cmdlets.

Step 1 - Plan your IP address ranges

In the following steps, we create two virtual networks along with their respective gateway subnets and configurations. We 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.

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
- DNS Server: 8.8.8.8
- GatewayName: VNet1GW
- Public IP: VNet1GWIP
- VPNTYPE: RouteBased
- Connection(1to4): VNet1toVNet4
- Connection(1to5): VNet1toVNet5
- 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
- DNS Server: 8.8.8.8
- GatewayName: VNet4GW
- Public IP: VNet4GWIP
- VPNTYPE: RouteBased
- Connection: VNet4toVNet1
- ConnectionType: VNet2VNet

Step 2 - Create and configure TestVNet1

1. Declare your variables

Start by declaring 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"
$DNS1 = "8.8.8.8"
$GWName1 = "VNet1GW"
$GWIPName1 = "VNet1GWIP"
$GWIPconfName1 = "gwipconf1"
$Connection14 = "VNet1toVNet4"
$Connection15 = "VNet1toVNet5"
```

2. Connect to your subscription

Switch to PowerShell mode to use the Resource Manager cmdlets. Open your PowerShell console and connect to your account. Use the following example to help you connect:

```
Login-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 will fail.

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 $GWSubName1 -AddressPrefix $GWSubPrefix1
```

5. Create TestVNet1

```
New-AzureRmVirtualNetwork -Name $VNetName1 -ResourceGroupName $RG1  
-Location $Location1 -AddressPrefix $VNetPrefix11,$VNetPrefix12 -Subnet $fesub1,$besub1,$gwsu
```

6. Request a public IP address

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 take a while (45 minutes or more to complete).

```
New-AzureRmVirtualNetworkGateway -Name $GWName1 -ResourceGroupName $RG1  
-Location $Location1 -IpConfigurations $gwipconf1 -GatewayType Vpn  
-VpnType RouteBased -GatewaySku Standard
```

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"  
$DNS4 = "8.8.8.8"  
$GWName4 = "VNet4GW"  
$GWIPName4 = "VNet4GWIP"  
$GWIPconfName4 = "gwipconf4"  
$Connection41 = "VNet4toVNet1"
```

Before you continue, make sure you are still connected to Subscription 1.

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  
$gwsb4 = New-AzureRmVirtualNetworkSubnetConfig -Name $GWSubName4 -AddressPrefix $GWSubPrefix4
```

4. Create TestVNet4

```
New-AzureRmVirtualNetwork -Name $VnetName4 -ResourceGroupName $RG4 `  
-Location $Location4 -AddressPrefix $VnetPrefix41,$VnetPrefix42 -Subnet $fesub4,$besub4,$gwsb4
```

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 $gwpip4
```

7. Create the TestVNet4 gateway

```
New-AzureRmVirtualNetworkGateway -Name $GWName4 -ResourceGroupName $RG4 `  
-Location $Location4 -IpConfigurations $gwipconf4 -GatewayType Vpn `  
-VpnType RouteBased -GatewaySku Standard
```

Step 4 - Connect the gateways

1. Get both virtual network gateways

In this example, because both gateways are in the same subscription, this step can be completed 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

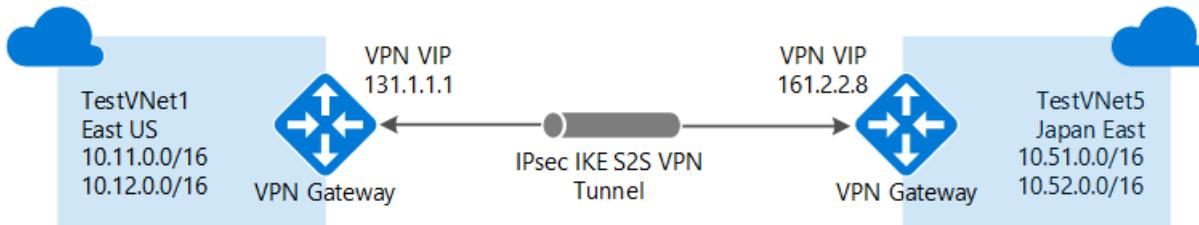
This step is similar to the one above, except you are creating the connection from TestVNet4 to TestVNet1. Make sure the shared keys match.

```
New-AzureRmVirtualNetworkGatewayConnection -Name $Connection41 -ResourceGroupName $RG4 ` 
-VirtualNetworkGateway1 $vnet4gw -VirtualNetworkGateway2 $vnet1gw -Location $Location4 ` 
-ConnectionType Vnet2Vnet -SharedKey 'AzureA1b2C3'
```

The connection should be established after a few minutes.

4. Verify your connection. See the section [How to verify your connection](#).

How to connect VNets that are in different subscriptions



In this scenario, we connect TestVNet1 and TestVNet5. TestVNet1 and TestVNet5 reside in a different subscription. The steps for this configuration add an additional VNet-to-VNet connection in order to connect TestVNet1 to TestVNet5.

The difference here 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.

The instructions continue from the previous steps listed above. You must complete [Step 1](#) and [Step 2](#) to create and configure TestVNet1, and the VPN Gateway for TestVNet1. Once you complete Step 1 and Step 2, continue with Step 5 to create TestVNet5.

Step 5 - Verify the additional 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: 10.52.255.0/27
- DNS Server: 8.8.8.8
- GatewayName: VNet5GW
- Public IP: VNet5GWIP
- VPNTYPE: RouteBased
- Connection: VNet5toVNet1
- ConnectionType: VNet2VNet

Additional Values for TestVNet1:

- Connection: VNet1toVNet5

Step 6 - 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"
$DNS5 = "8.8.8.8"
$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:

```
Login-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 TestVNet4

```
$fesub5 = New-AzureRmVirtualNetworkSubnetConfig -Name $FESubName5 -AddressPrefix $FESubPrefix5
$besub5 = New-AzureRmVirtualNetworkSubnetConfig -Name $BESubName5 -AddressPrefix $BESubPrefix5
$gwsub5 = New-AzureRmVirtualNetworkSubnetConfig -Name $GWSubName5 -AddressPrefix $GWSubPrefix5
```

5. Create TestVNet5

```
New-AzureRmVirtualNetwork -Name $VnetName5 -ResourceGroupName $RG5 -Location $Location5 ` 
-AddressPrefix $VnetPrefix51,$VnetPrefix52 -Subnet $fesub5,$besub5,$gwsub5
```

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 Standard
```

Step 7 - Connecting the gateways

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

Make sure you log in and connect to Subscription 1.

```
$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/resourceGroupsTestRG1/providers/Microsoft.Network/virtualNetworkGateways/VNet1GW
```

2. [Subscription 5] Get the virtual network gateway for Subscription 5

Make sure you log in and connect to Subscription 5.

```
$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.

Make sure you connect to Subscription 1.

```
$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.

Make sure you connect to Subscription 5.

```
$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 "  
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 created and want to test.

```
Get-AzureRmVirtualNetworkGatewayConnection -Name MyGWConnection -ResourceGroupName MyRG
```

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.

```
Body:  
{  
    "name": "MyGWConnection",  
    "id":  
    "/subscriptions/086cfcaa0-0d1d-4b1c-94544-  
f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/connections/MyGWConnection",  
    "properties": {  
        "provisioningState": "Succeeded",  
        "resourceGuid": "1c484f82-23ec-47e2-8cd8-231107450446b",  
        "virtualNetworkGateway1": {  
            "id":  
            "/subscriptions/086cfcaa0-0d1d-4b1c-94544-  
f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/virtualNetworkGa  
teways/vnetgw1"  
        },  
        "localNetworkGateway2": {  
            "id":  
            "/subscriptions/086cfcaa0-0d1d-4b1c-94544-  
f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/localNetworkGate  
ways/LocalSite"  
        },  
        "connectionType": "IPsec",  
        "routingWeight": 10,  
        "sharedKey": "abc123",  
        "connectionStatus": "Connected",  
        "ingressBytesTransferred": 33509044,  
        "egressBytesTransferred": 4142431  
    }  
}
```

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 in the portal

1/17/2017 • 15 min to read • [Edit on GitHub](#)

Azure currently has two management models: classic and Resource Manager (RM). If you have been using Azure for some time, you probably have Azure VMs and instance roles running in a classic VNet. Your newer VMs and role instances may be running in a VNet created in Resource Manager. This article walks you through connecting classic VNets to Resource Manager VNets to allow the resources located in the separate deployment models to communicate with each other over a gateway connection.

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.

Deployment models and methods for VNet-to-VNet connections

It's important to know that Azure currently works with two deployment models: Resource Manager and classic. Before you begin your configuration, make sure that you understand the deployment models and tools. You'll need to know which model that you want to work in. Not all networking features are supported yet for both models. For information about the deployment models, see [Understanding Resource Manager deployment and classic deployment](#).

We update the following table as new articles and additional tools become available for this configuration. When an article is available, we link directly to it from the table.

VNet-to-VNet

DEPLOYMENT MODEL/METHOD	AZURE PORTAL	CLASSIC PORTAL	POWERSHELL
Classic	Not Supported	Article*	Supported
Resource Manager	Article+	Not Supported	Article
Connections between different deployment models	Article*	Article*	Article

(+) denotes this deployment method is available only for VNets in the same subscription.

(*) denotes that this deployment method also requires PowerShell.

VNet peering

It's also possible to connect VNets without using a VPN gateway. If your VNets are in the same region, you may want to consider connecting them by using VNet peering. For more information, see the [VNet peering](#) article.

Before beginning

The following steps walk you through the settings necessary to configure a dynamic or route-based gateway for each VNet and create a VPN connection between the gateways. This configuration does not support static or

policy-based gateways.

In this article, we use the classic portal, the Azure portal, and PowerShell. Currently, it's not possible to create this configuration using only the Azure portal.

Prerequisites

- Both VNets have already been created.
- 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.
- You have installed the latest PowerShell cmdlets (1.0.2 or later). See [How to install and configure Azure PowerShell](#) for more information. Make sure you install both the Service Management (SM) and the Resource Manager (RM) cmdlets.

Example settings

You can use the example settings as reference.

Classic VNet settings

VNet Name = ClassicVNet

Location = West US

Virtual Network Address Spaces = 10.0.0.0/24

Subnet-1 = 10.0.0.0/27

GatewaySubnet = 10.0.0.32/29

Local Network Name = RMVNetLocal

Resource Manager VNet settings

VNet Name = RMVNet

Resource Group = RG1

Virtual Network IP Address Spaces = 192.168.0.0/16

Subnet-1 = 192.168.1.0/24

GatewaySubnet = 192.168.0.0/26

Location = East US

Virtual network gateway name = RMGateway

Gateway public IP name = gwpip

Gateway type = VPN

VPN type = Route-based

Local network gateway = ClassicVNetLocal

Section 1: Configure classic VNet settings

In this section, we create the local network and the gateway for your classic VNet. The instructions in this section use the classic portal. Currently, the Azure portal does not offer all the settings that pertain to a classic VNet.

Part 1 - Create a new local network

Open the [classic portal](#) and sign in with your Azure account.

1. On the bottom left corner of the screen, click **NEW > Network Services > Virtual Network > Add local network**.
2. In the **Specify your local network details** window, type a name for the RM VNet you want to connect to. In the **VPN device IP address (optional)** box, type any valid public IP address. This is just a temporary placeholder. You change this IP address later. On the bottom right corner of the window, click the arrow button.
3. On the **Specify the address space** page, in the **Starting IP** text box, type the network prefix and CIDR block for the Resource Manager VNet you want to connect to. This setting is used to specify the address space to route to the RM VNet.

Part 2 - Associate the local network to your VNet

1. Click **Virtual Networks** at the top of the page to switch to the Virtual Networks screen, then click to select your classic VNet. On the page for your VNet, click **Configure** to navigate to the configuration page.
2. Under the **site-to-site connectivity** connection section, select the **Connect to the local network** checkbox. Then select the local network that you created. If you have multiple local networks that you created, be sure to select the one that you created to represent your Resource Manager VNet from the dropdown.
3. Click **Save** at the bottom of the page.

Part 3 - Create the gateway

1. After saving the settings, click **Dashboard** at the top of the page to change to the Dashboard page. On the bottom of the Dashboard page, click **Create Gateway**, then click **Dynamic Routing**. Click **Yes** to begin creating your gateway. A Dynamic Routing gateway is required for this configuration.
2. Wait for the gateway to be created. This can sometimes take 45 minutes or more to complete.

Part 4 - View the gateway public IP address

After the gateway has been created, you can view the gateway IP address on the **Dashboard** page. This is the public IP address of your gateway. Write down or copy the public IP address. You use it in later steps when you create the local network for your Resource Manager VNet configuration.

Section 2: Configure Resource Manager VNet settings

In this section, we create the virtual network gateway and the local network for your Resource Manager VNet. Don't start the following steps until after you have retrieved the public IP address for the classic VNet's gateway.

The screenshots are provided as examples. Be sure to replace the values with your own. If you are creating this configuration as an exercise, refer to these [values](#).

Part 1 - Create a gateway subnet

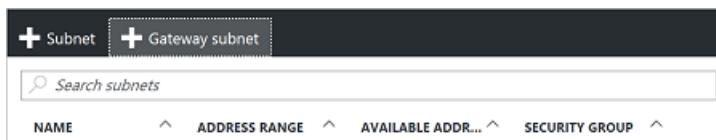
Before connecting your virtual network to a gateway, you first need to create the gateway subnet for the virtual network to which you want to connect. Create a gateway subnet with CIDR count of /28 or larger (/27, /26, etc.)

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?](#)

From a browser, navigate to the [Azure portal](#) and sign in with your Azure account.

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 blade, click **Subnets** to expand the Subnets blade.
3. On the **Subnets** blade, click **+Gateway subnet** at the top. This will open the **Add subnet** blade.



4. The **Name** for your subnet will automatically be 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.

Add subnet
RMVNet1

* Name
GatewaySubnet

* Address range (CIDR block) 192.168.0.0/24
192.168.0.0 - 192.168.0.255 (256 addresses)

5. Click **OK** at the bottom of the blade to create the subnet.

Part 2 - Create a virtual network gateway

- 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** blade, click **Create** at the bottom of the blade. This opens the **Create virtual network gateway** blade.
- On the **Create virtual network gateway** blade, fill in the values for your virtual network gateway.

Create virtual network g... — □ ×

* Name

Gateway type VPN ExpressRoute

VPN type Route-based Policy-based

* SKU Standard

* Virtual network Choose a virtual network

* Public IP address Choose a public IP address

* Subscription Windows Azure Internal Consumption

Resource group —

* Location East US

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.

6. **SKU:** Select the gateway SKU from the dropdown. The SKUs listed in the dropdown depend on the VPN type you select.
7. **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, the virtual network will not appear in the 'Choose a virtual network' dropdown.
8. Choose the virtual network to which you want to add this gateway. Click **Virtual network** to open the **Choose a virtual network** blade. 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.
9. Choose a public IP address. Click **Public IP address** to open the **Choose public IP address** blade. Click **+Create New** to open the **Create public IP address blade**. Input a name for your public IP address. This blade creates a public IP address object to which a public IP address will be dynamically assigned. Click **OK** to save your changes to this blade.
10. **Subscription:** Verify that the correct subscription is selected.
11. **Resource group:** This setting is determined by the Virtual Network that you select.
12. Don't adjust the **Location** after you've specified the previous settings.
13. Verify the settings. You can select **Pin to dashboard** at the bottom of the blade if you want your gateway to appear on the dashboard.
14. Click **Create** to begin creating the gateway. The settings will be 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.



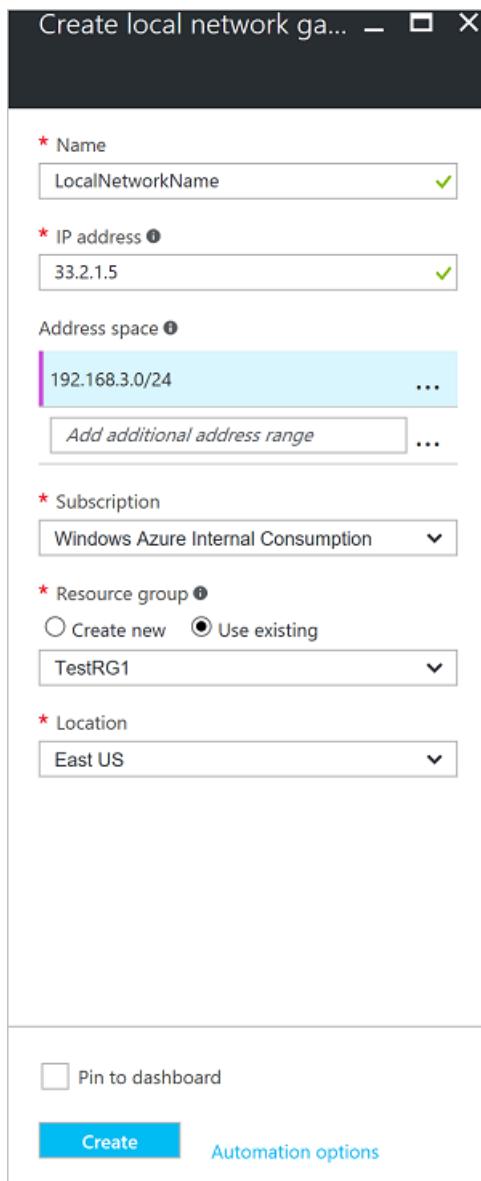
15. After the gateway is created, you can view the IP address that has been assigned to it by looking at the virtual network in the portal. The gateway will appear as a connected device. You can click the connected device (your virtual network gateway) to view more information.

Part 3 - Create a local network gateway

The 'local network gateway' typically refers to your on-premises location. It tells Azure which IP address ranges to route to the location and the public IP address of the device for that location. However, in this case, it refers to the address range and public IP address associated with your classic VNet and virtual network gateway.

Give the local network gateway a name by which Azure can refer to it. You can create your local network gateway while your virtual network gateway is being created. For this configuration, you use the public IP address that was assigned to your classic VNet gateway in the [previous section](#).

1. In the portal, from **All resources**, click **+Add**. In the **Everything** blade search box, type **Local network gateway**, then click to search. This will return a list. Click **Local network gateway** to open the blade, then click **Create** to open the **Create local network gateway** blade.



2. On the **Create local network gateway blade**, specify a **Name** for your local network gateway object.
3. 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, this is 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, you will specify the public IP address that was assigned to the virtual network gateway for that VNet.
4. **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.
5. For **Subscription**, verify that the correct subscription is showing.
6. For **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.
7. For **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.
8. Click **Create** to create the local network gateway.

Part 4 - Copy the public IP address

Once the virtual network gateway has finished creating, copy the public IP address that is associated with the gateway. You use it when you configure the local network settings for your classic VNet.

Section 3: Modify the local network for the classic VNet

Open the [classic portal](#).

1. In the classic portal, scroll down on the left side and click **Networks**. On the **networks** page, click **Local Networks** at the top of the page.
2. Click to select the local network that you configured in Part 1. At the bottom of the page, click **Edit**.
3. On the **Specify your local network details** page, replace the placeholder IP address with the public IP address for the Resource Manager gateway that you created in the previous section. Click the arrow to move to the next section. Verify that the **Address Space** is correct, and then click the checkmark to accept the changes.

Section 4: Create the connection

In this section, we create the connection between the VNets. The steps for this require PowerShell. You cannot create this connection in either of the portals. Make sure you have downloaded and installed both the classic (SM) and Resource Manager (RM) PowerShell cmdlets.

1. Log in to your Azure account in the PowerShell console. The following cmdlet prompts you for the login credentials for your Azure Account. After logging in, your account settings are downloaded so that they are available to Azure PowerShell.

```
Login-AzureRmAccount
```

Get a list of your Azure subscriptions if you have more than one subscription.

```
Get-AzureRmSubscription
```

Specify the subscription that you want to use.

```
Select-AzureRmSubscription -SubscriptionName "Name of subscription"
```

2. Add your Azure Account to use the classic PowerShell cmdlets. To do so, you can use the following command:

```
Add-AzureAccount
```

3. Set your shared key by running the following sample. In this sample, `-VNetName` is the name of the classic VNet and `-LocalNetworkSiteName` is the name you specified for the local network when you configured it in the classic portal. The `-SharedKey` is a value that you can generate and specify. The value you specify here must be the same value that you specify in the next step when you create your connection.

```
Set-AzureVNetGatewayKey -VNetName ClassicVNet `  
-LocalNetworkSiteName RMVNetLocal -SharedKey abc123
```

4. Create the VPN connection by running the following commands:

Set the variables

```
$vnet01gateway = Get-AzureRMLocalNetworkGateway -Name ClassicVNetLocal -ResourceGroupName RG1  
$vnet02gateway = Get-AzureRmVirtualNetworkGateway -Name RMGateway -ResourceGroupName RG1
```

Create the connection

Note that the `-ConnectionType` is 'IPsec', not 'Vnet2Vnet'. In this sample, `-Name` is the name that you want to call your connection. The following sample creates a connection named '*rm-to-classic-connection*'.

```
New-AzureRmVirtualNetworkGatewayConnection -Name rm-to-classic-connection -ResourceGroupName RG1 `  
-Location "East US" -VirtualNetworkGateway1 `  
$vnet02gateway -LocalNetworkGateway2 `  
$vnet01gateway -ConnectionType IPsec -RoutingWeight 10 -SharedKey 'abc123'
```

Verify your connection

You can verify your connection by using the classic portal, the Azure portal, or PowerShell. You can use the following steps to verify your connection. Replace the values with your own.

To verify your connection by using PowerShell

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 created and want to test.

```
Get-AzureRmVirtualNetworkGatewayConnection -Name MyGWConnection -ResourceGroupName MyRG
```

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.

```
Body:  
{  
    "name": "MyGWConnection",  
    "id":  
    "/subscriptions/086cfcaa0-0d1d-4b1c-94544-  
f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/connections/MyGWConnection",  
    "properties": {  
        "provisioningState": "Succeeded",  
        "resourceGuid": "1c484f82-23ec-47e2-8cd8-231107450446b",  
        "virtualNetworkGateway1": {  
            "id":  
            "/subscriptions/086cfcaa0-0d1d-4b1c-94544-  
f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/virtualNetworkGa  
teways/vnetgw1"  
        },  
        "localNetworkGateway2": {  
            "id":  
            "/subscriptions/086cfcaa0-0d1d-4b1c-94544-  
f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/localNetworkGate  
ways/LocalSite"  
        },  
        "connectionType": "IPsec",  
        "routingWeight": 10,  
        "sharedKey": "abc123",  
        "connectionStatus": "Connected",  
        "ingressBytesTransferred": 33509044,  
        "egressBytesTransferred": 4142431  
    }  
}
```

To verify your connection by using the Azure portal

In the Azure portal, you can view the connection status by navigating to the connection. There are multiple ways to do this. 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 RG1	Data in 2.35 KB
Status Connected	Data out 3.14 KB
Location East US	Virtual network RMVNet
Subscription name Windows Azure Internal Consumption	Virtual network gateway RMGateway (40.114.5.29)
Subscription ID	Local network gateway Site2 (40.76.7.127)

VNet-to-VNet FAQ

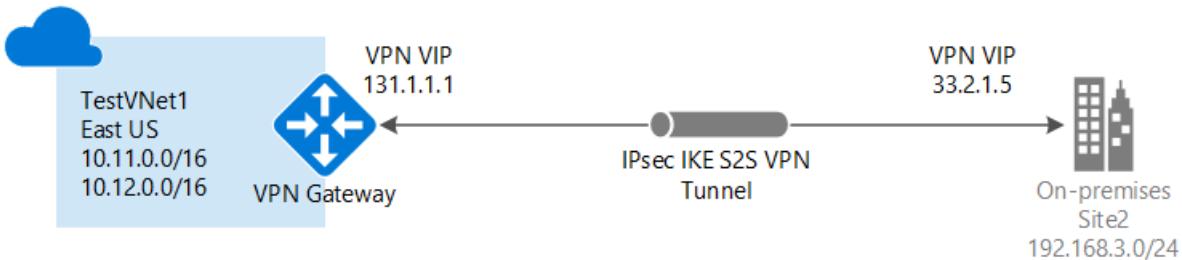
View the FAQ details for additional information about VNet-to-VNet connections.

- The virtual networks can be in the same or different Azure regions (locations).
- A cloud service or a load balancing endpoint CANNOT span across virtual networks, even if they are connected together.
- Connecting multiple Azure virtual networks together doesn't require any on-premises VPN gateways unless cross-premises connectivity is required.
- VNet-to-VNet supports connecting virtual networks. It does not support connecting virtual machines or cloud services NOT in a virtual network.
- VNet-to-VNet requires Azure VPN gateways with RouteBased (previously called Dynamic Routing) VPN types.
- Virtual network connectivity can be used simultaneously with multi-site VPNs. There is a maximum of 10 (Default/Standard Gateways) or 30 (HighPerformance Gateways) VPN tunnels for a virtual network VPN gateway connecting to either other virtual networks, or on-premises sites.
- The address spaces of the virtual networks and on-premises local network sites must not overlap. Overlapping address spaces will cause the creation of VNet-to-VNet connections to fail.
- Redundant tunnels between a pair of virtual networks are supported when one virtual network gateway is configured as active-active.
- 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.
- VNet-to-VNet traffic travels across the Microsoft Network, not the Internet.
- VNet-to-VNet traffic within the same region is free for both directions. Cross region VNet-to-VNet egress traffic is charged with the outbound inter-VNet data transfer rates based on the source regions. Please refer to the [pricing page](#) for details.

Create a VNet with a Site-to-Site connection using the Azure portal

1/17/2017 • 14 min to read • [Edit on GitHub](#)

This article walks you through creating a virtual network and a Site-to-Site VPN gateway connection to your on-premises network using the Azure Resource Manager deployment model and the Azure portal. Site-to-Site connections can be used for cross-premises and hybrid configurations.



Deployment models and methods for Site-to-Site connections

It's important to understand that Azure currently works with two deployment models: Resource Manager and classic. Before you begin your configuration, verify that you are using the instructions for the deployment model that you want to work in. The two models are not completely compatible with each other.

For example, if you are working with a virtual network that was created using the classic deployment model and wanted to add a connection to the VNet, you would use the deployment methods that correspond to the classic deployment model, not Resource Manager. If you are working with a virtual network that was created using the Resource Manager deployment model, you would use the deployment methods that correspond with Resource Manager, not classic.

For information about the deployment models, see [Understanding Resource Manager deployment and classic deployment](#).

The following table shows the currently available deployment models and methods for Site-to-Site configurations. When an article with configuration steps is available, we link directly to it from this table.

DEPLOYMENT MODEL/METHOD	AZURE PORTAL	CLASSIC PORTAL	POWERSHELL
Resource Manager	Article	Not Supported	Article
Classic	Supported**	Article*	Article+

(*) denotes that the classic portal can only support creating one S2S VPN connection.

(**) denotes that an end-to-end scenario is not yet available for the Azure portal.

(+) denotes that this article is written for multi-site connections.

Additional configurations

If you want to connect VNets together, but are not creating a connection to an on-premises location, see [Configure a VNet-to-VNet connection](#). If you want to add a Site-to-Site connection to a VNet that already has a connection, see [Add a S2S connection to a VNet with an existing VPN gateway connection](#).

Before you begin

Verify that you have the following items before beginning your configuration:

- A compatible VPN device and someone who is able to configure it. See [About VPN Devices](#). If you aren't familiar with configuring your VPN device, or 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.
- An externally facing public IP address for your VPN device. This IP address cannot be located behind a NAT.
- An Azure subscription. If you don't already have an Azure subscription, you can activate your [MSDN subscriber benefits](#) or sign up for a [free account](#).

Sample configuration values for this exercise

When using these steps as an exercise, you can use the sample configuration values:

- **VNet Name:** TestVNet1
- **Address Space:** 10.11.0.0/16 and 10.12.0.0/16
- **Subnets:**
 - FrontEnd: 10.11.0.0/24
 - BackEnd: 10.12.0.0/24
 - GatewaySubnet: 10.12.255.0/27
- **Resource Group:** TestRG1
- **Location:** East US
- **DNS Server:** 8.8.8.8
- **Gateway Name:** VNet1GW
- **Public IP:** VNet1GWIP
- **VPN Type:** Route-based
- **Connection Type:** Site-to-site (IPsec)
- **Gateway Type:** VPN
- **Local Network Gateway Name:** Site2
- **Connection Name:** VNet1toSite2

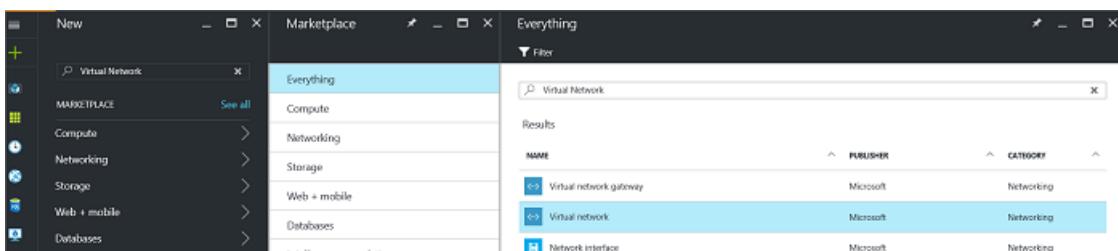
1. Create a virtual network

If you already have a VNet, verify that the settings are compatible with your VPN gateway design. Pay particular attention to any subnets that may overlap with other networks. If you have overlapping subnets, your connection won't work properly. If your VNet is configured with the correct settings, you can begin the steps in the [Specify a DNS server](#) section.

To create a virtual network

To create a VNet in the Resource Manager deployment model by using the Azure portal, follow the steps below. The screenshots are provided as examples. Be sure to replace the values with your own. For more information about working with virtual networks, see the [Virtual Network Overview](#).

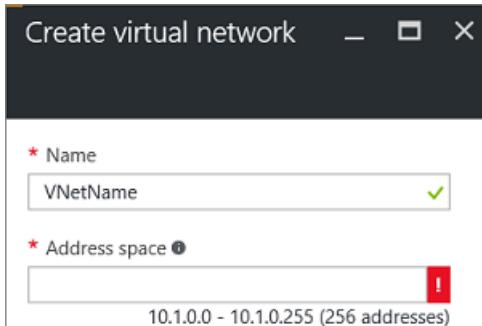
1. From a browser, navigate to the [Azure portal](#) and, if necessary, sign in with your Azure account.
2. Click **New**. In the **Search the marketplace** field, type "Virtual Network". Locate **Virtual Network** from the returned list and click to open the **Virtual Network** blade.



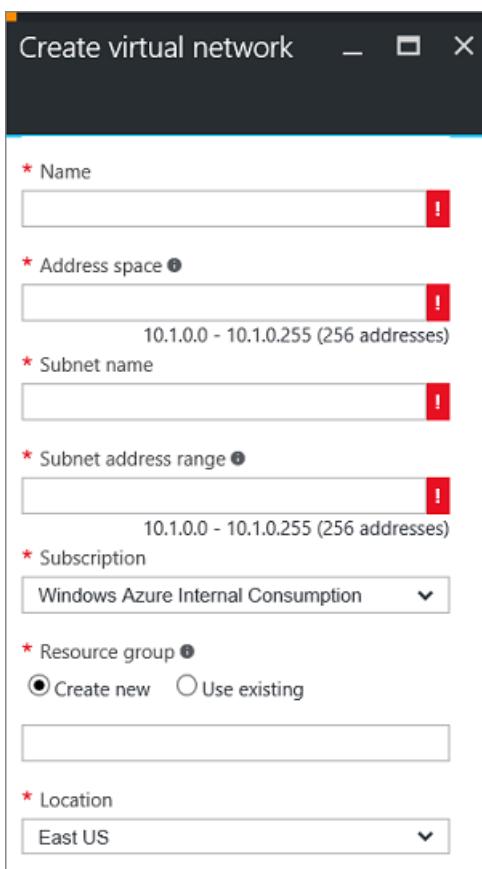
3. Near the bottom of the Virtual Network blade, from the **Select a deployment model** list, select **Resource Manager**, and then click **Create**.



4. On the **Create virtual network** blade, configure the VNet settings. When you fill in the fields, the red exclamation mark will become a green check mark when the characters entered in the field are valid.



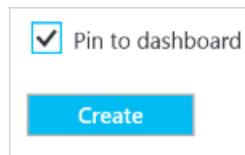
5. The **Create virtual network** blade looks similar to the following example. There may be values that are auto-filled. If so, replace the values with your own.



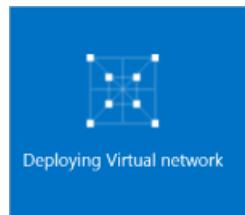
6. **Name:** Enter the name for your Virtual Network.
7. **Address space:** Enter the address space. If you have multiple address spaces to add, add your first address space. You can add additional address spaces later, after creating the VNet.
8. **Subnet name:** Add the subnet name and subnet address range. You can add additional subnets later, after creating the VNet.
9. **Subscription:** Verify that the Subscription listed is the correct one. You can change subscriptions by using the drop-down.
10. **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](#).

11. **Location:** Select the location for your VNet. The location determines where the resources that you deploy to this VNet will reside.
12. Select **Pin to dashboard** if you want to be able to find your VNet easily on the dashboard, and then click **Create**.



13. 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. Add additional address space and subnets

You can add additional address space and subnets to your VNet once it has been created.

To add address space

1. To add additional address space, under the **Settings** section for your virtual network blade, click **Address space** to open the Address space blade.
2. Add the additional address space, and then click **Save** at the top of the blade.

A screenshot of the Azure Virtual Network blade. The left sidebar shows navigation links: Overview, Activity log, Access control (IAM), Tags, and SETTINGS (Address space, Connected devices, Subnets, DNS). The "Address space" link is highlighted with a blue background. The main content area shows two address ranges: 10.41.0.0/16 and 10.42.0.0/16, each with a three-dot ellipsis menu. Below them is a text input field labeled "Add additional address range" with a three-dot ellipsis menu. At the top of the blade, there are "Save" and "Discard" buttons, with "Save" being highlighted with a red box.

To create subnets

1. To create subnets, in the **Settings** section of your virtual network blade, click **Subnets** to open the **Subnets** blade.
2. In the Subnets blade, click **+Subnet** to open the **Add subnet** blade. Name your new subnet and specify the address range.

The screenshot shows the Azure portal interface for creating a new subnet. On the left, there's a list of existing subnets: 'FrontEnd' with address range 10.41.0.0/24. On the right, a new subnet is being created with the following details:

- Name:** BackEnd
- Address range (CIDR block):** 10.42.0.0/24 (10.42.0.0 - 10.42.0.255 (256 addresses))

- Click **OK** at the bottom of the blade to save your changes.

OK

3. Specify a DNS server

To specify a DNS server

This setting allows you to specify the DNS server that you want to use for name resolution for this virtual network. It does not create a DNS server.

- On the **Settings** page for your virtual network, navigate to **DNS Servers** and click to open the DNS servers blade.
- On the **DNS Servers** page, under **DNS servers**, select **Custom**.
- In the **DNS Server** field, in the **Add DNS server** box, enter the IP address of the DNS server that you want to use for name resolution.
- When you are done adding DNS servers, click **Save** at the top of the blade to save your configuration.

The screenshot shows the 'DNS servers' blade in the Azure portal. The left sidebar has a 'SETTINGS' section with options like 'Address space', 'Connected devices', 'Subnets', and 'DNS servers' (which is highlighted). The main area shows the 'DNS servers' configuration:

- DNS servers:** Default (Azure-provided) is unselected, and Custom is selected.
- DNS SERVER:** There is a button labeled 'Add DNS server'.

4. Create a gateway subnet

Before connecting your virtual network to a gateway, you first need to create the gateway subnet for the virtual network to which you want to connect. If possible, it's best to create a gateway subnet using a CIDR block of /28 or /27 in order to provide enough IP addresses to accommodate additional future configuration requirements.

If you are creating this configuration as an exercise, refer to these [values](#) when creating your gateway subnet.

To create a gateway subnet

- 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 blade, click **Subnets** to expand the Subnets blade.
3. On the **Subnets** blade, click **+Gateway subnet** at the top. This will open the **Add subnet** blade.

The screenshot shows the 'Subnets' blade in the Azure portal. At the top, there are two buttons: '+ Subnet' and '+ Gateway subnet'. The '+ Gateway subnet' button is highlighted with a red box. Below the buttons is a search bar labeled 'Search subnets'. Underneath the search bar are four filter buttons: 'NAME', 'ADDRESS RANGE', 'AVAILABLE ADDR...', and 'SECURITY GROUP'. There is also a small 'X' icon in the top right corner of the blade.

4. The **Name** for your subnet will automatically be 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.

The screenshot shows the 'Add subnet' blade. At the top, it says 'RMVNet1'. The main area has two fields: 'Name' (containing 'GatewaySubnet') and 'Address range (CIDR block)' (containing '192.168.0.0/24'). A red box highlights the 'Address range (CIDR block)' field. Below the address range is the text '192.168.0.0 - 192.168.0.255 (256 addresses)'. The blade has standard window controls (minimize, maximize, close) in the top right corner.

5. Click **OK** at the bottom of the blade to create the subnet.

5. Create a virtual network gateway

If you are creating this configuration as an exercise, you can refer to the [sample configuration values](#).

To create a virtual network gateway

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** blade, click **Create** at the bottom of the blade. This opens the **Create virtual network gateway** blade.
2. On the **Create virtual network gateway** blade, fill in the values for your virtual network gateway.

Create virtual network g... — □ X

* Name

Gateway type ⓘ
VPN ExpressRoute

VPN type ⓘ
Route-based Policy-based

* SKU ⓘ
Standard

* Virtual network ⓘ >
Choose a virtual network

* Public IP address ⓘ >
Choose a public IP address

* Subscription
Windows Azure Internal Consumption

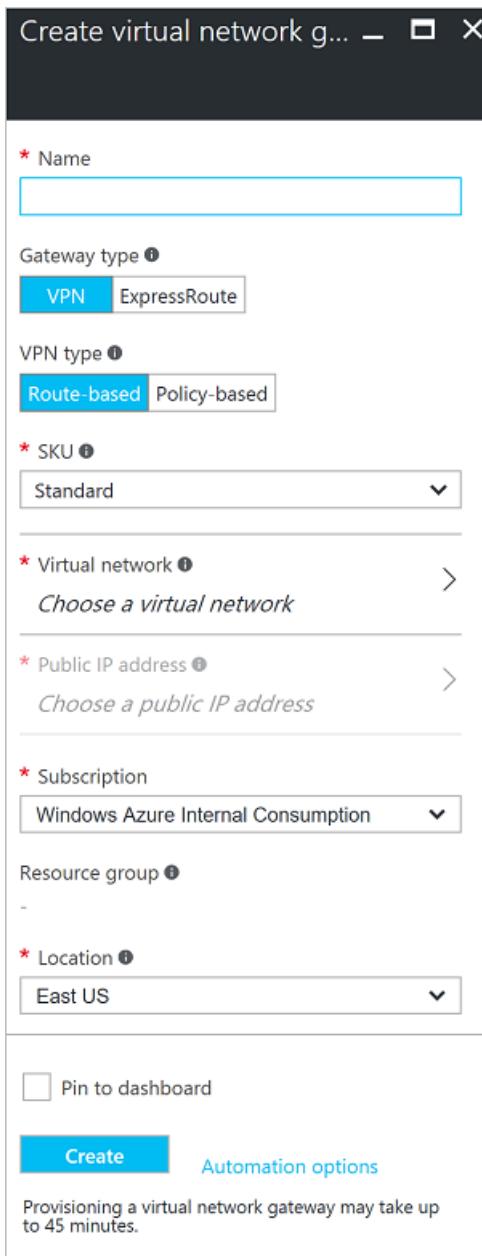
Resource group ⓘ
-

* Location ⓘ
East US

Pin to dashboard

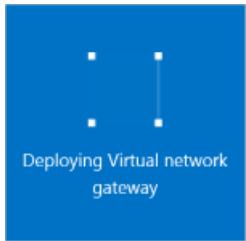
Create [Automation options](#)

Provisioning a virtual network gateway may take up to 45 minutes.



3. **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.
4. **Gateway type:** Select **VPN**. VPN gateways use the virtual network gateway type **VPN**.
5. **VPN type:** Select the VPN type that is specified for your configuration. Most configurations require a Route-based VPN type.
6. **SKU:** Select the gateway SKU from the dropdown. The SKUs listed in the dropdown depend on the VPN type you select.
7. **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, the virtual network will not appear in the 'Choose a virtual network' dropdown.
8. Choose the virtual network to which you want to add this gateway. Click **Virtual network** to open the **Choose a virtual network** blade. 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.
9. Choose a public IP address. Click **Public IP address** to open the **Choose public IP address** blade. Click **+Create New** to open the **Create public IP address blade**. Input a name for your public IP address. This blade creates a public IP address object to which a public IP address will be dynamically assigned. Click **OK** to save your changes to this blade.
10. **Subscription:** Verify that the correct subscription is selected.
11. **Resource group:** This setting is determined by the Virtual Network that you select.

12. Don't adjust the **Location** after you've specified the previous settings.
13. Verify the settings. You can select **Pin to dashboard** at the bottom of the blade if you want your gateway to appear on the dashboard.
14. Click **Create** to begin creating the gateway. The settings will be 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.



15. After the gateway is created, you can view the IP address that has been assigned to it by looking at the virtual network in the portal. The gateway will appear as a connected device. You can click the connected device (your virtual network gateway) to view more information.

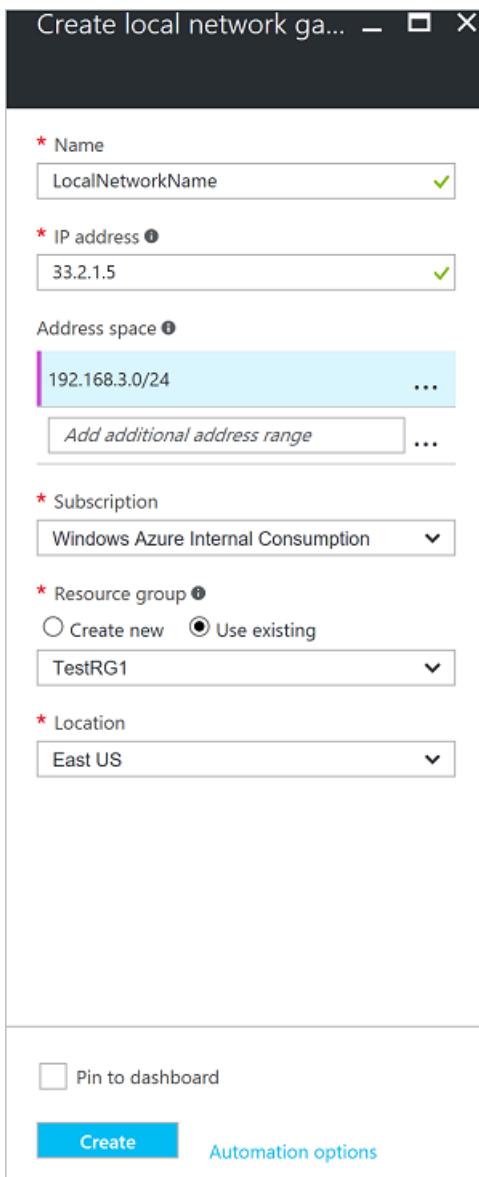
6. Create a local network gateway

The 'local network gateway' refers to your on-premises location. Give the local network gateway a name by which Azure can refer to it.

If you are creating this configuration as an exercise, you can refer to the [sample configuration values](#).

To create a local network gateway

1. In the portal, from **All resources**, click **+Add**. In the **Everything** blade search box, type **Local network gateway**, then click to search. This will return a list. Click **Local network gateway** to open the blade, then click **Create** to open the **Create local network gateway** blade.



2. On the **Create local network gateway blade**, specify a **Name** for your local network gateway object.
3. 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, this is 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, you will specify the public IP address that was assigned to the virtual network gateway for that VNet.
4. **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.
5. For **Subscription**, verify that the correct subscription is showing.
6. For **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.
7. For **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.
8. Click **Create** to create the local network gateway.

7. Configure your VPN device

To configure your VPN device, you'll need the public IP address of the virtual network gateway for configuring your on-premises VPN device. Work with your device manufacturer for specific configuration information and configure your device. Refer to the [VPN Devices](#) for more information about VPN devices that work well with Azure.

To find the public IP address of your virtual network gateway using PowerShell, use the following sample:

```
Get-AzureRmPublicIpAddress -Name GW1PublicIP -ResourceGroupName TestRG
```

You can also view the public IP address for your virtual network gateway by using the Azure portal. Navigate to **Virtual network gateways**, then click the name of your gateway.

8. Create a Site-to-Site VPN connection

Create the Site-to-Site VPN connection between your virtual network gateway and your VPN device. Be sure to replace the values with your own. The shared key must match the value you used for your VPN device configuration.

Before beginning this section, verify that your virtual network gateway and local network gateways have finished creating. If you are creating this configuration as an exercise, refer to these [values](#) when creating your connection.

To create the VPN connection

1. Locate your virtual network gateway and click **All settings** to open the **Settings** blade.
2. On the **Settings** blade, click **Connections**, and then click **Add** at the top of the blade to open the **Add connection** blade.



3. On the **Add connection** blade, **Name** your connection.
4. For **Connection type**, select **Site-to-site(IPSec)**.
5. For **Virtual network gateway**, the value is fixed because you are connecting from this gateway.
6. For **Local network gateway**, click **Choose a local network gateway** and select the local network gateway that you want to use.
7. For **Shared Key**, the value here must match the value that you are using for your local VPN device. If your VPN device on your local network doesn't provide a shared key, you can make one up and input it here and on your local device. The important thing is that they both match.

8. The remaining values for **Subscription**, **Resource Group**, and **Location** are fixed.
9. Click **OK** to create your connection. You'll see *Creating Connection* flash on the screen.
10. When the connection is complete, you'll see it appear in the **Connections** blade for your Gateway.

The screenshot shows the 'Connections' blade in the Azure portal. At the top, there's a search bar labeled 'Search connections'. Below it is a table with columns: NAME, STATUS, CONNECTION TYPE, and PEER. A single row is visible: 'VNet1s2s' under NAME, 'Succeeded' under STATUS, 'Site-to-site (IPsec)' under CONNECTION TYPE, and 'VNet1LocalNet' under PEER. There's also a '...' button next to the row.

9. Verify the VPN connection

You can verify your VPN connection either in the portal, or by using PowerShell.

To verify your connection by using PowerShell

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 created and want to test.

```
Get-AzureRmVirtualNetworkGatewayConnection -Name MyGWConnection -ResourceGroupName MyRG
```

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.

```
Body:
{
  "name": "MyGWConnection",
  "id": "/subscriptions/086cfcaa0-0d1d-4b1c-94544-f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/connections/MyGWConnection",
  "properties": {
    "provisioningState": "Succeeded",
    "resourceGuid": "1c484f82-23ec-47e2-8cd8-231107450446b",
    "virtualNetworkGateway1": {
      "id": "/subscriptions/086cfcaa0-0d1d-4b1c-94544-f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/virtualNetworkGateways/vnetgw1"
    },
    "localNetworkGateway2": {
      "id": "/subscriptions/086cfcaa0-0d1d-4b1c-94544-f8e3da2a0c7789/resourceGroups/MyRG/providers/Microsoft.Network/localNetworkGateways/LocalSite"
    },
    "connectionType": "IPsec",
    "routingWeight": 10,
    "sharedKey": "abc123",
    "connectionStatus": "Connected",
    "ingressBytesTransferred": 33509044,
    "egressBytesTransferred": 4142431
  }
}
```

To verify your connection by using the Azure portal

In the Azure portal, you can view the connection status by navigating to the connection. There are multiple ways to do this. 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 RG1	Data in 2.35 KB
Status Connected	Data out 3.14 KB
Location East US	Virtual network RMVNet
Subscription name Windows Azure Internal Consumption	Virtual network gateway RMGateway (40.114.5.29)
Subscription ID	Local network gateway Site2 (40.76.7.127)

Next steps

- Once your connection is complete, you can add virtual machines to your virtual networks. For more information, see [Virtual Machines](#).
- For information about BGP, see the [BGP Overview](#) and [How to configure BGP](#).

Connect a virtual network to an ExpressRoute circuit

1/17/2017 • 2 min to read • [Edit on GitHub](#)

This article will help you link virtual networks (VNets) to Azure ExpressRoute circuits by using the Resource Manager deployment model and the Azure portal. Virtual networks can either be in the same subscription, or they can be part of another subscription.

About Azure deployment models

It's important to know that Azure currently works with two deployment models: Resource Manager and classic. Before you begin your configuration, make sure that you understand the deployment models and tools. You'll need to know which model that you want to work in. Not all networking features are supported yet for both models. For information about the deployment models, see [Understanding Resource Manager deployment and classic deployment](#).

Configuration prerequisites

- Make sure that you have reviewed 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 [VPN gateway](#) (follow only steps 1-5).

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. You can link a virtual networks 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.

Connect a virtual network in the same subscription to a circuit

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 screenshot shows three windows side-by-side:

- Left Window:** Shows the 'Essentials' and 'Peering' sections for the 'ER-Demo-Ckt-SV' circuit.
- Middle Window:** The 'Settings' blade for the same circuit, with the 'Peering' section highlighted by a red box.
- Right Window:** The 'Peering' blade, showing a table of peering configurations. One row, 'Azure private | Enabled | 172.16.0.0/30 | 172.16.0.4/30', is highlighted with a red box.

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.

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** blade, and then configure the values. See the following reference example.

The screenshot shows three windows side-by-side:

- Left Window:** The 'Connections' blade for the 'ER-Demo-Ckt-SV' circuit, with the 'Add' button highlighted by a red box.
- Middle Window:** The 'Connections' blade, showing a table with no results.
- Right Window:** The 'Add connection' blade for 'ER-Demo-Ckt-SV', with the connection details highlighted by a red box.

3. After your connection has been successfully configured, your connection object will show the information for the connection.

The screenshot shows two windows side-by-side:

- Left Window:** The 'Connections' blade for the 'ER-Demo-Ckt-SV' circuit, showing the 'ER-VNet-Connection' entry.
- Right Window:** The 'ER-VNet-Connection' connection object, with the 'Virtual network' section highlighted by a red box.

To delete a connection

You can delete a connection by selecting the **Delete** icon on the blade for your connection.

Connect a virtual network in a different subscription to a circuit

At this time, you cannot connect virtual networks across subscriptions by using the Azure portal. However, you can use PowerShell to do this. See the [PowerShell](#) article for more information.

Next steps

For more information about ExpressRoute, see the [ExpressRoute FAQ](#).

Implementing a highly available hybrid network architecture

1/17/2017 • 5 min to read • [Edit on GitHub](#)

patterns & practices

proven practices for predictable results

This article describes how to connect an on-premises network to an Azure virtual network (VNet) using ExpressRoute, with a site-to-site virtual private network (VPN) as a failover connection.

Traffic flows between the on-premises network and the Azure VNet through an ExpressRoute connection. ExpressRoute connections are made using a private dedicated connection through a third-party connectivity provider. If there is a loss of connectivity in the ExpressRoute circuit, traffic will be routed through an IPSec VPN tunnel.

Note that if the ExpressRoute circuit is unavailable, the VPN route will only handle private peering connections. Public peering and Microsoft peering connections will pass over the Internet.

NOTE

Azure has two different deployment models: [Resource Manager](#) and classic. This blueprint uses Resource Manager, which Microsoft recommends for new deployments.

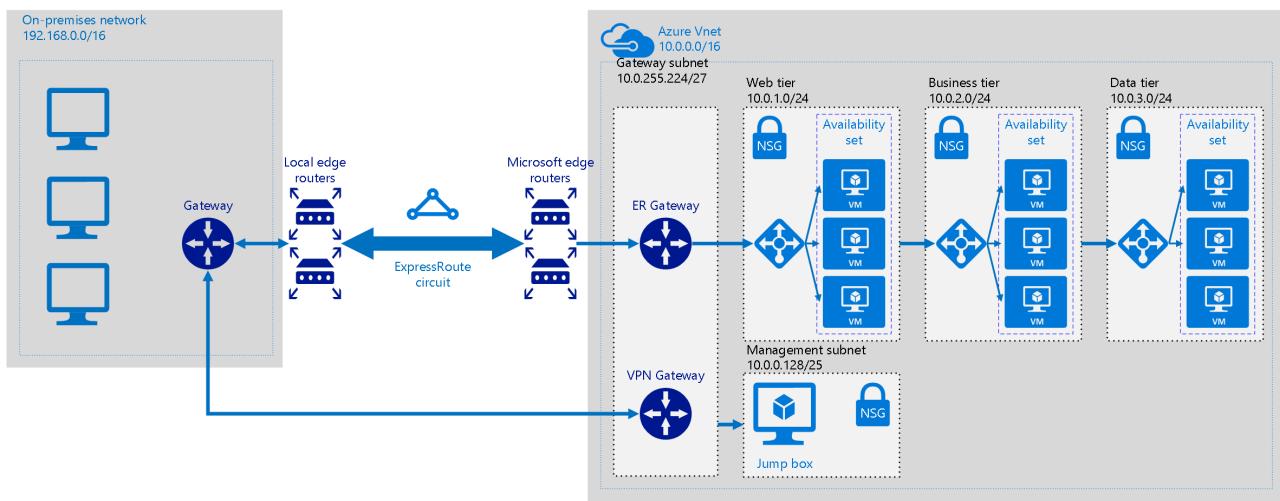
Typical use cases for this architecture include:

- Hybrid applications where workloads are distributed between an on-premises network and Azure.
- Applications running large-scale, mission-critical workloads that require a high degree of scalability.
- Large-scale backup and restore facilities for data that must be saved off-site.
- Handling big data workloads.
- Using Azure as a disaster-recovery site.

Architecture diagram

The following diagram highlights the important components in this architecture:

A Visio document that includes this architecture diagram is available for download from the [Microsoft download center](#). This diagram is on the "Hybrid network - ER-VPN" page.



- **On-premises network.** A private local-area network running within an organization.
- **VPN appliance.** A device or service that provides external connectivity to the on-premises network. The VPN appliance may be a hardware device, or it can be a software solution such as the Routing and Remote Access Service (RRAS) in Windows Server 2012.

NOTE

For a list of supported VPN appliances and information on configuring selected VPN appliances for connecting to Azure, see [About VPN devices for Site-to-Site VPN Gateway connections](#).

- **ExpressRoute circuit.** A layer 2 or layer 3 circuit supplied by the connectivity provider that joins the on-premises network with Azure through the edge routers. The circuit uses the hardware infrastructure managed by the connectivity provider.
- **ExpressRoute virtual network gateway.** The ExpressRoute virtual network gateway enables the VNet to connect to the ExpressRoute circuit used for connectivity with your on-premises network.
- **VPN virtual network gateway.** The VPN virtual network gateway enables the VNet to connect to the VPN appliance in the on-premises network. The VPN virtual network gateway is configured to accept requests from the on-premises network only through the VPN appliance. For more information, see [Connect an on-premises network to a Microsoft Azure virtual network](#).
- **VPN connection.** The connection has properties that specify the connection type (IPSec) and the key shared with the on-premises VPN appliance to encrypt traffic.
- **Azure Virtual Network (VNet).** Each VNet resides in a single Azure region, and can host multiple application tiers. Application tiers can be segmented using subnets in each VNet.
- **Gateway subnet.** The virtual network gateways are held in the same subnet.
- **Cloud application.** The application hosted in Azure. It might include multiple tiers, with multiple subnets connected through Azure load balancers. The traffic in each subnet may be subject to rules defined using [network security groups](#)(NSGs). For more information, see [Getting started with Microsoft Azure security](#).

Recommendations

The following recommendations apply for most scenarios. Follow these recommendations unless you have a specific requirement that overrides them.

VNet and GatewaySubnet

Create the ExpressRoute virtual network gateway and the VPN virtual network gateway in the same VNet. This

means that they should share the same subnet named *GatewaySubnet*.

If the VNet already includes a subnet named *GatewaySubnet*, ensure that it has a /27 or larger address space. If the existing subnet is too small, use the following PowerShell command to remove the subnet:

```
$vnet = Get-AzureRmVirtualNetworkGateway -Name <yourvnetname> -ResourceGroupName <yourresourcegroup>
Remove-AzureRmVirtualNetworkSubnetConfig -Name GatewaySubnet -VirtualNetwork $vnet
```

If the VNet does not contain a subnet named **GatewaySubnet**, create a new one using the following Powershell command:

```
$vnet = Get-AzureRmVirtualNetworkGateway -Name <yourvnetname> -ResourceGroupName <yourresourcegroup>
Add-AzureRmVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet -AddressPrefix
"10.200.255.224/27"
$vnet = Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

VPN and ExpressRoute gateways

Verify that your organization meets the [ExpressRoute prerequisite requirements](#) for connecting to Azure.

If you already have a VPN virtual network gateway in your Azure VNet, use the following Powershell command to remove it:

```
Remove-AzureRmVirtualNetworkGateway -Name <yourgatewayname> -ResourceGroupName <yourresourcegroup>
```

Follow the instructions in [Implementing a hybrid network architecture with Azure ExpressRoute](#) to establish your ExpressRoute connection.

Follow the instructions in [Implementing a hybrid network architecture with Azure and On-premises VPN](#) to establish your VPN virtual network gateway connection.

After you have established the virtual network gateway connections, test the environment as follows:

1. Make sure you can connect from your on-premises network to your Azure VNet.
2. Contact your provider to stop ExpressRoute connectivity for testing.
3. Verify that you can still connect from your on-premises network to your Azure VNet using the VPN virtual network gateway connection.
4. Contact your provider to reestablish ExpressRoute connectivity.

Considerations

For ExpressRoute considerations, see the [Implementing a Hybrid Network Architecture with Azure ExpressRoute](#) guidance.

For site-to-site VPN considerations, see the [Implementing a Hybrid Network Architecture with Azure and On-premises VPN](#) guidance.

For general Azure security considerations, see [Microsoft cloud services and network security](#).

Solution Deployment

If you have an existing on-premises infrastructure already configured with a suitable network appliance, you can deploy the reference architecture by following these steps:

1. Right-click the button below and select either "Open link in new tab" or "Open link in new window":



2. Wait for the link to open in the Azure portal, then follow these steps:

- The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-hybrid-vpn-er-rg` in the text box.
 - Select the region from the **Location** drop down box.
 - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
 - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
 - Click the **Purchase** button.
3. Wait for the deployment to complete.

4. Right-click the button below and select either "Open link in new tab" or "Open link in new window":



5. Wait for the link to open in the Azure portal, then enter then follow these steps:

- Select **Use existing** in the **Resource group** section and enter `ra-hybrid-vpn-er-rg` in the text box.
- Select the region from the **Location** drop down box.
- Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
- Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
- Click the **Purchase** button.

Virtual appliance scenario

1/17/2017 • 8 min to read • [Edit on GitHub](#)

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.

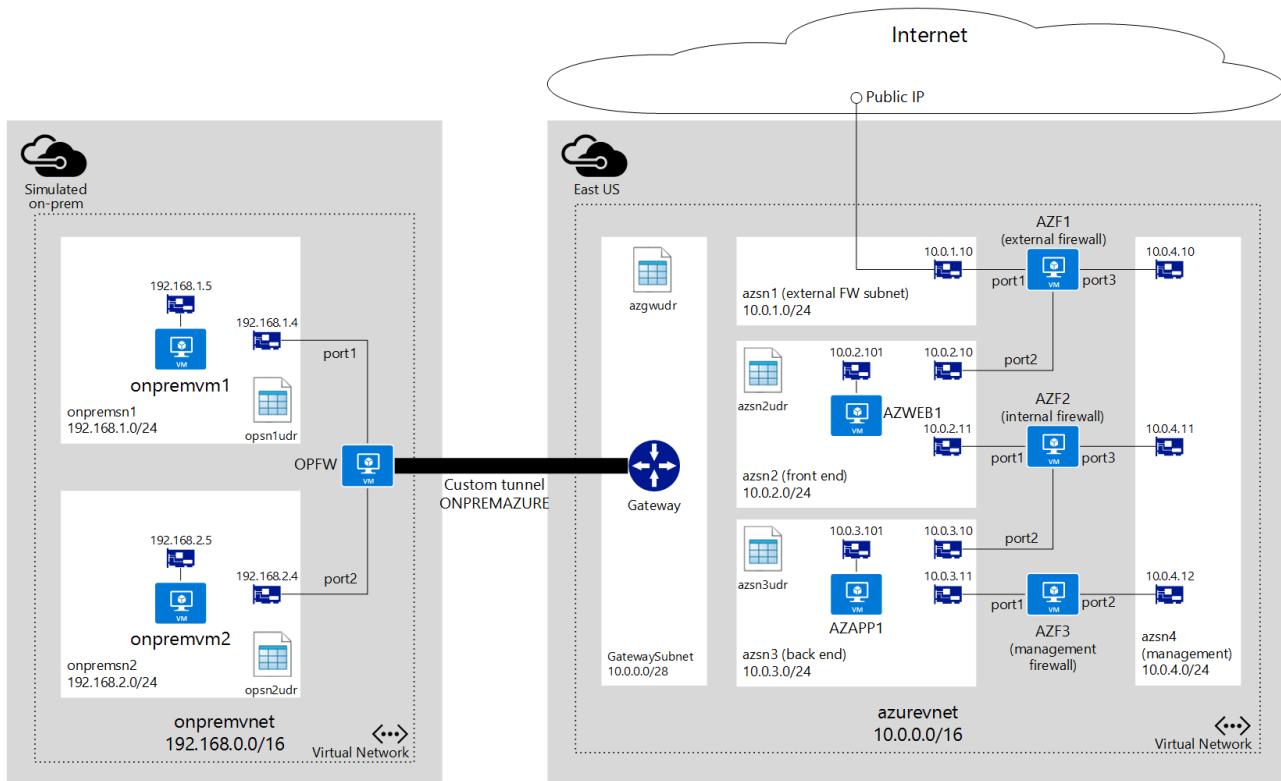
Considerations

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 **onpremnet** 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 **onpremnet** 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 **onpremvm2** and ping 10.0.3.101 to test connectivity between **onpremsn2** and **azsn3**.

Implementing a DMZ between Azure and the Internet

1/17/2017 • 6 min to read • [Edit on GitHub](#)

patterns & practices

proven practices for predictable results

This article describes best practices for implementing a secure hybrid network that extends your on-premises network and also accepts Internet traffic to Azure.

NOTE

Azure has two different deployment models: [Resource Manager](#) and classic. This reference architecture uses Resource Manager, which Microsoft recommends for new deployments.

This reference architecture extends the architecture described in [Implementing a DMZ between Azure and your on-premises datacenter](#). This reference architecture adds a public DMZ that handles Internet traffic, in addition to the private DMZ that handles traffic from the on-premises network

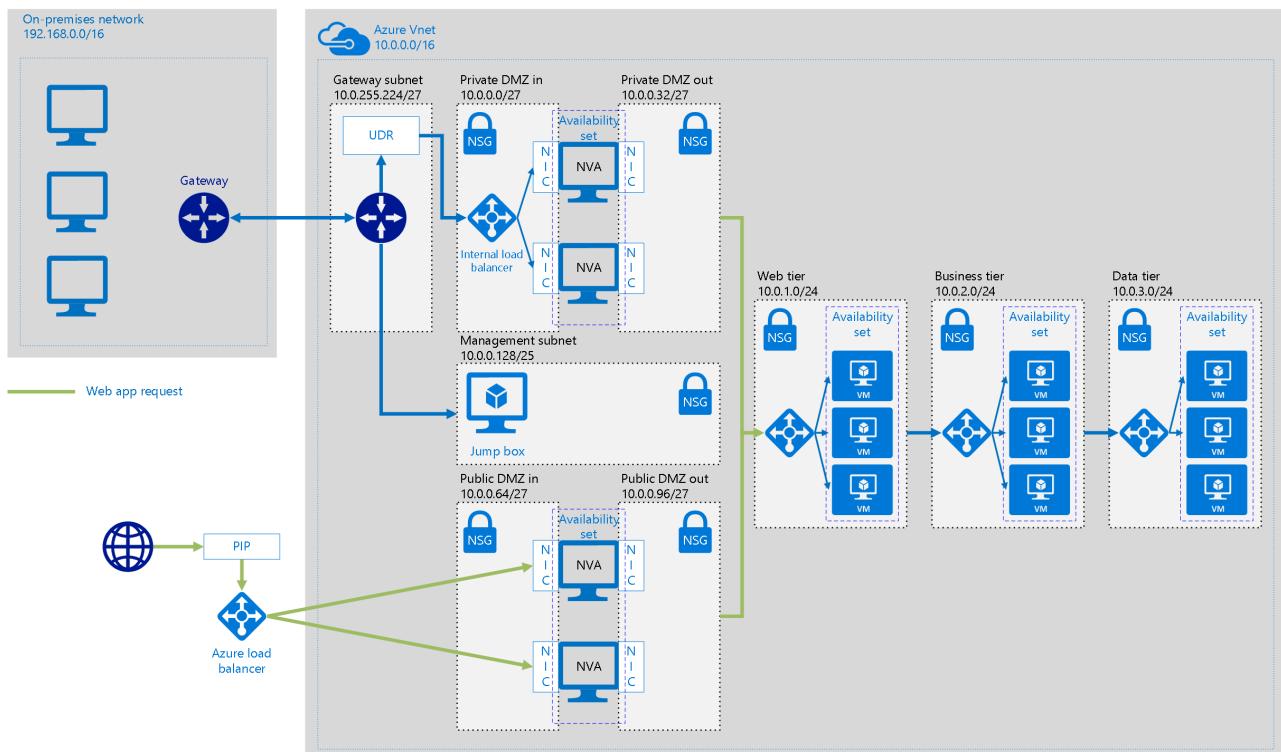
Typical uses for this architecture include:

- Hybrid applications where workloads run partly on-premises and partly in Azure.
- Azure infrastructure that routes incoming traffic from on-premises and the Internet.

Architecture diagram

The following diagram highlights the important components in this architecture:

A Visio document that includes this architecture diagram is available for download from the [Microsoft download center](#). This diagram is on the "DMZ - Public" page.



To enable Internet traffic to Azure, the architecture includes the following components:

- **Public IP address (PIP).** The IP address of the public endpoint. External users connected to the Internet can access the system through this address.
- **Network virtual appliance (NVA).** This architecture includes a separate pool of NVAs for traffic originating on the Internet.
- **Azure load balancer.** All incoming requests from the Internet pass through the load balancer and are distributed to the NVAs in the public DMZ.
- **Public DMZ inbound subnet.** This subnet accepts requests from the Azure load balancer. Incoming requests are passed to one of the NVAs in the public DMZ.
- **Public DMZ outbound subnet.** Requests that are approved by the NVA pass through this subnet to the internal load balancer for the web tier.

Recommendations

The following recommendations apply for most scenarios. Follow these recommendations unless you have a specific requirement that overrides them.

NVA recommendations

Use one set of NVAs for traffic originating on the Internet, and another for traffic originating on-premises. Using only one set of NVAs for both is a security risk, because it provides no security perimeter between the two sets of network traffic. Using separate NVAs reduces the complexity of checking security rules, and makes it clear which rules correspond to each incoming network request. One set of NVAs implements rules for Internet traffic only, while another set of NVAs implement rules for on-premises traffic only.

Include a layer-7 NVA to terminate application connections at the NVA level and maintain compatibility with the backend tiers. This guarantees symmetric connectivity where response traffic from the backend tiers returns through the NVA.

Public load balancer recommendations

For scalability and availability, deploy the public DMZ NVAs in an [availability set](#) and use an [Internet facing load balancer](#) to distribute Internet requests across the NVAs in the availability set.

Configure the load balancer to accept requests only on the ports necessary for Internet traffic. For example, restrict inbound HTTP requests to port 80 and inbound HTTPS requests to port 443.

Scalability considerations

Even if your architecture initially requires a single NVA in the public DMZ, we recommend putting a load balancer in front of the public DMZ from the beginning. That will make it easier to scale to multiple NVAs in the future, if needed.

Availability considerations

The Internet facing load balancer requires each NVA in the public DMZ inbound subnet to implement a [health probe](#). A health probe that fails to respond on this endpoint is considered to be unavailable, and the load balancer will direct requests to other NVAs in the same availability set. Note that if all NVAs fail to respond, your application will fail, so it's important to have monitoring configured to alert DevOps when the number of healthy NVA instances falls below a defined threshold.

Manageability considerations

All monitoring and management for the NVAs in the public DMZ should be performed by the jumpbox in the management subnet. As discussed in [Implementing a DMZ between Azure and your on-premises datacenter](#), define

a single network route from the on-premises network through the gateway to the jumpbox, in order to restrict access.

If gateway connectivity from your on-premises network to Azure is down, you can still reach the jumpbox by deploying a public IP address, adding it to the jumpbox, and logging in from the Internet.

Security considerations

This reference architecture implements multiple levels of security:

- The Internet facing load balancer directs requests to the NVAs in the inbound public DMZ subnet, and only on the ports necessary for the application.
- The NSG rules for the inbound and outbound public DMZ subnets prevent the NVAs from being compromised, by blocking requests that fall outside of the NSG rules.
- The NAT routing configuration for the NVAs directs incoming requests on port 80 and port 443 to the web tier load balancer, but ignores requests on all other ports.

You should log all incoming requests on all ports. Regularly audit the logs, paying attention to requests that fall outside of expected parameters, as these may indicate intrusion attempts.

Solution deployment

A deployment for a reference architecture that implements these recommendations is available on [GitHub](#). The reference architecture can be deployed either with Windows or Linux VMs by following the directions below:

1. Right-click the button below and select either "Open link in new tab" or "Open link in new window":



2. Once the link has opened in the Azure portal, you must enter values for some of the settings:

- The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-public-dmz-network-rg` in the text box.
- Select the region from the **Location** drop down box.
- Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
- Select the **Os Type** from the drop down box, **windows** or **linux**.
- Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
- Click the **Purchase** button.

3. Wait for the deployment to complete.

4. Right-click the button below and select either "Open link in new tab" or "Open link in new window":



5. Once the link has opened in the Azure portal, you must enter values for some of the settings:

- The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-public-dmz-wl-rg` in the text box.
- Select the region from the **Location** drop down box.
- Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
- Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
- Click the **Purchase** button.

6. Wait for the deployment to complete.

7. Right-click the button below and select either "Open link in new tab" or "Open link in new window":



8. Once the link has opened in the Azure portal, you must enter values for some of the settings:
 - The **Resource group** name is already defined in the parameter file, so select **Use Existing** and enter `ra-public-dmz-network-rg` in the text box.
 - Select the region from the **Location** drop down box.
 - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
 - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
 - Click the **Purchase** button.
9. Wait for the deployment to complete.
10. The parameter files include hard-coded administrator user name and password for all VMs, and it is strongly recommended that you immediately change both. For each VM in the deployment, select it in the Azure portal and then click **Reset password** in the **Support + troubleshooting** blade. Select **Reset password** in the **Mode** drop down box, then select a new **User name** and **Password**. Click the **Update** button to save.

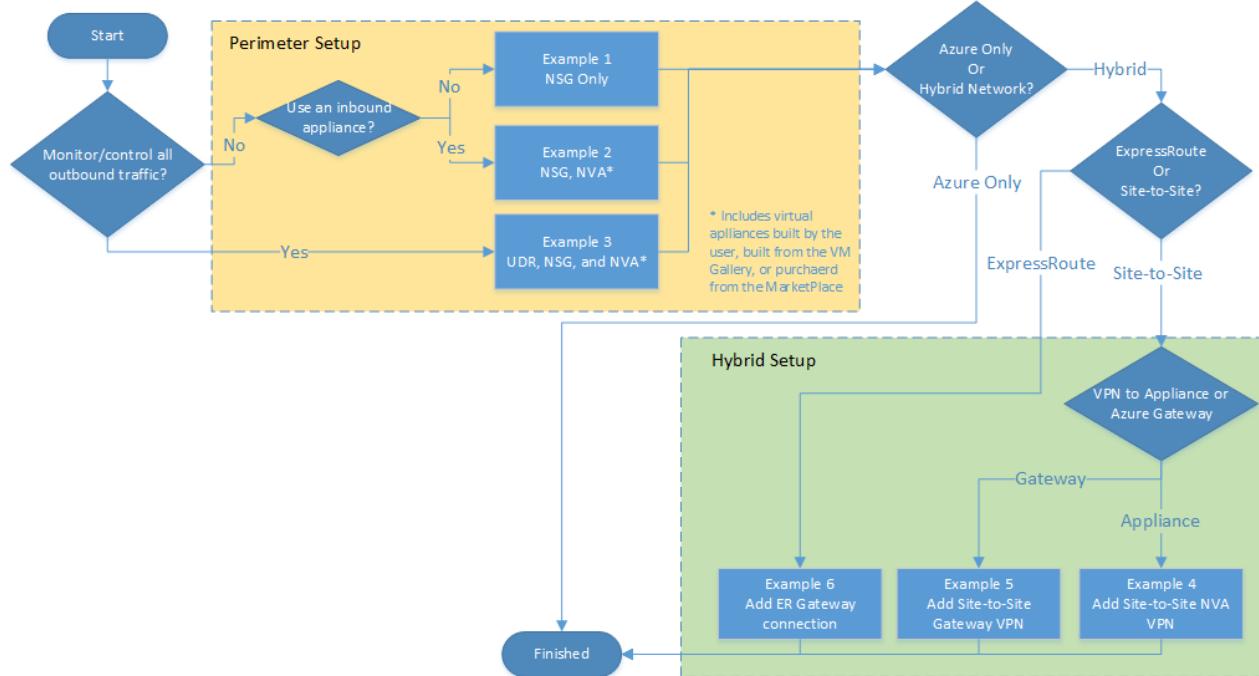
Microsoft cloud services and network security

1/17/2017 • 37 min to read • [Edit on GitHub](#)

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.

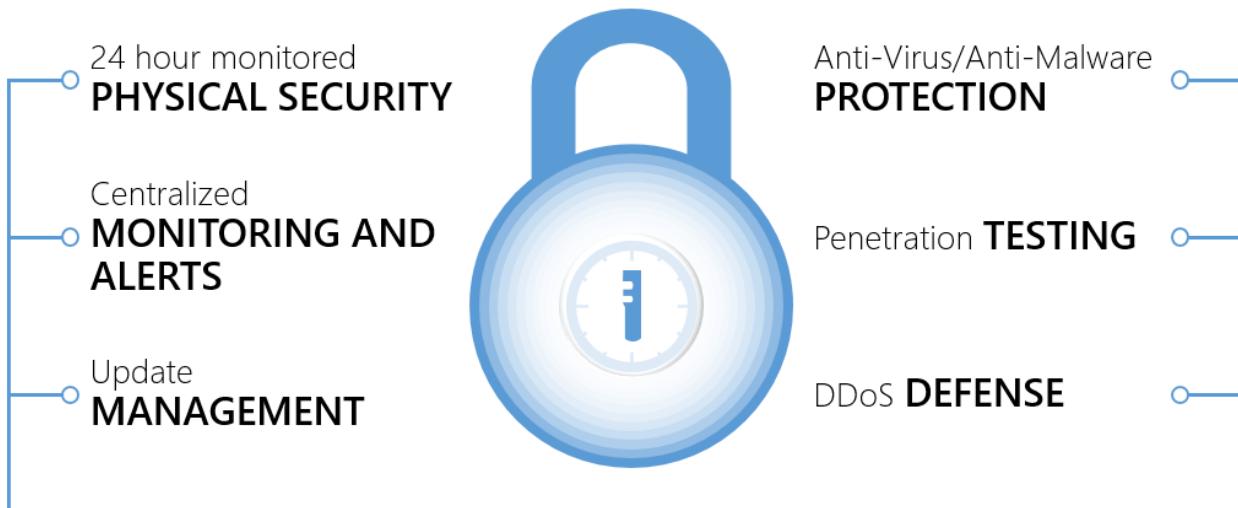
Examples for adding connections between virtual networks, high availability, and service chaining will be added to this document over the next few months.

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.

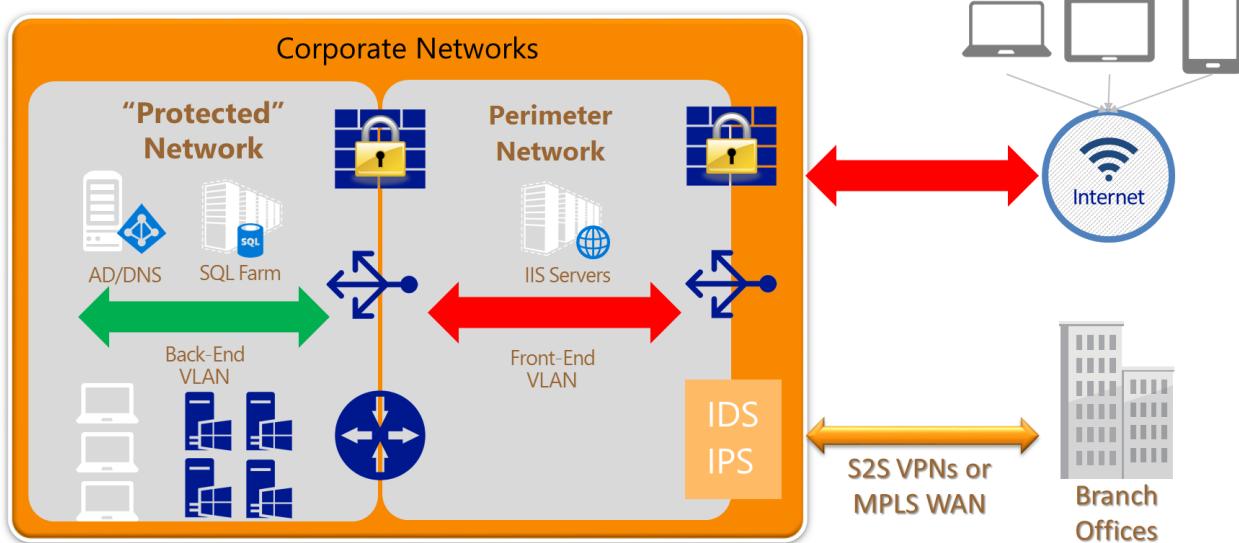


This approach provides a more secure foundation for customers to deploy their services in the Microsoft cloud. The next step is for customers to design and create a security architecture to protect these services.

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.

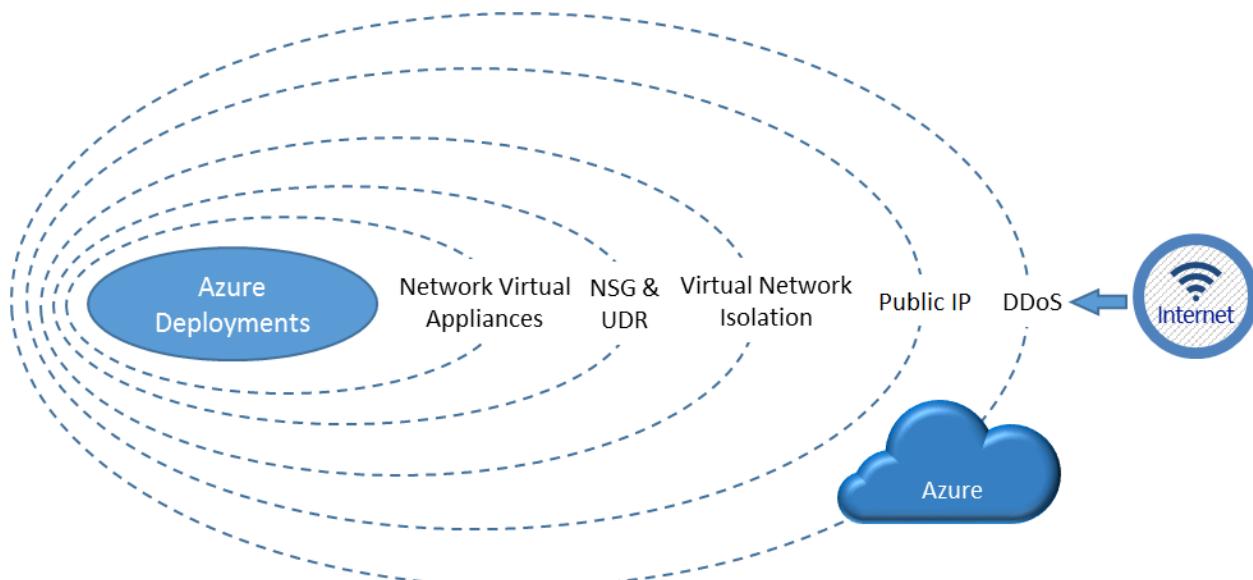


There are many architectures used to implement a perimeter network. These architectures can range from a simple load balancer to a multiple-subnet perimeter network with varied mechanisms at each boundary to block traffic and protect the deeper layers of the corporate network. How the perimeter network 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:

1. **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.
2. **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:

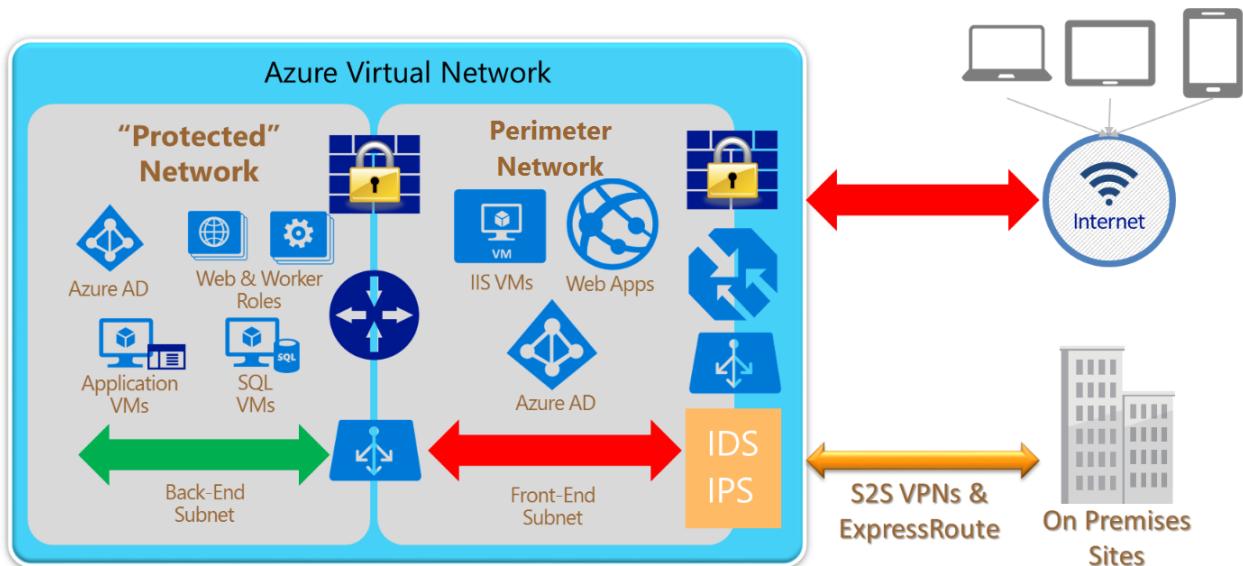
- **Traffic isolation:** A virtual network is the traffic isolation boundary on the Azure platform. Virtual machines (VMs) in one virtual network cannot communicate directly to VMs in a different virtual network, even if both virtual networks are created by the same customer. Isolation is a critical 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?

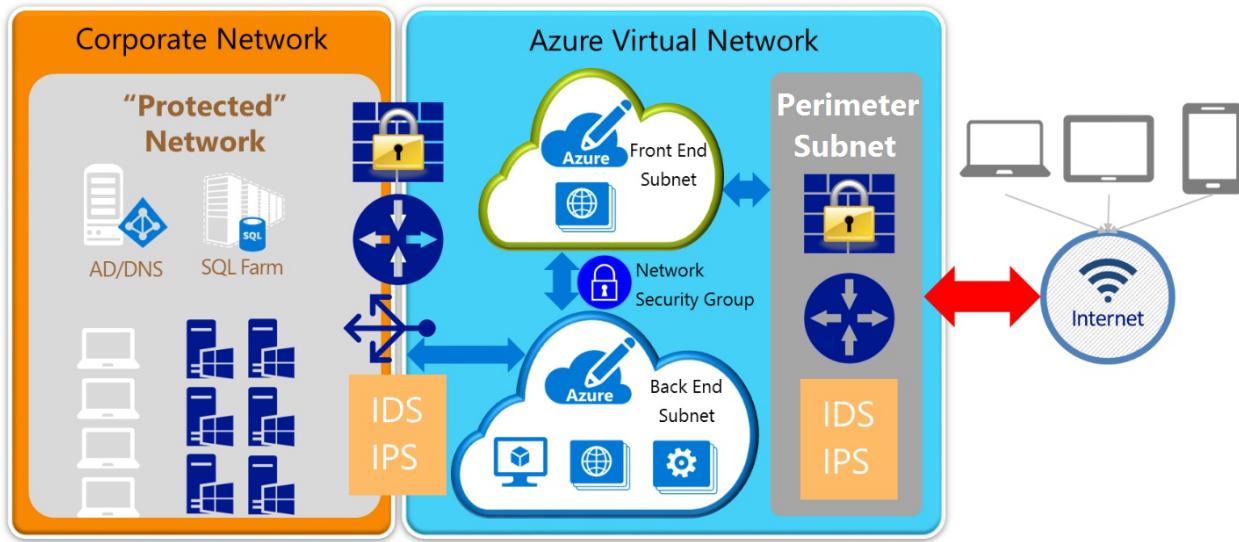
The first decision point is to decide how many security boundaries are needed in a given scenario:

- 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).

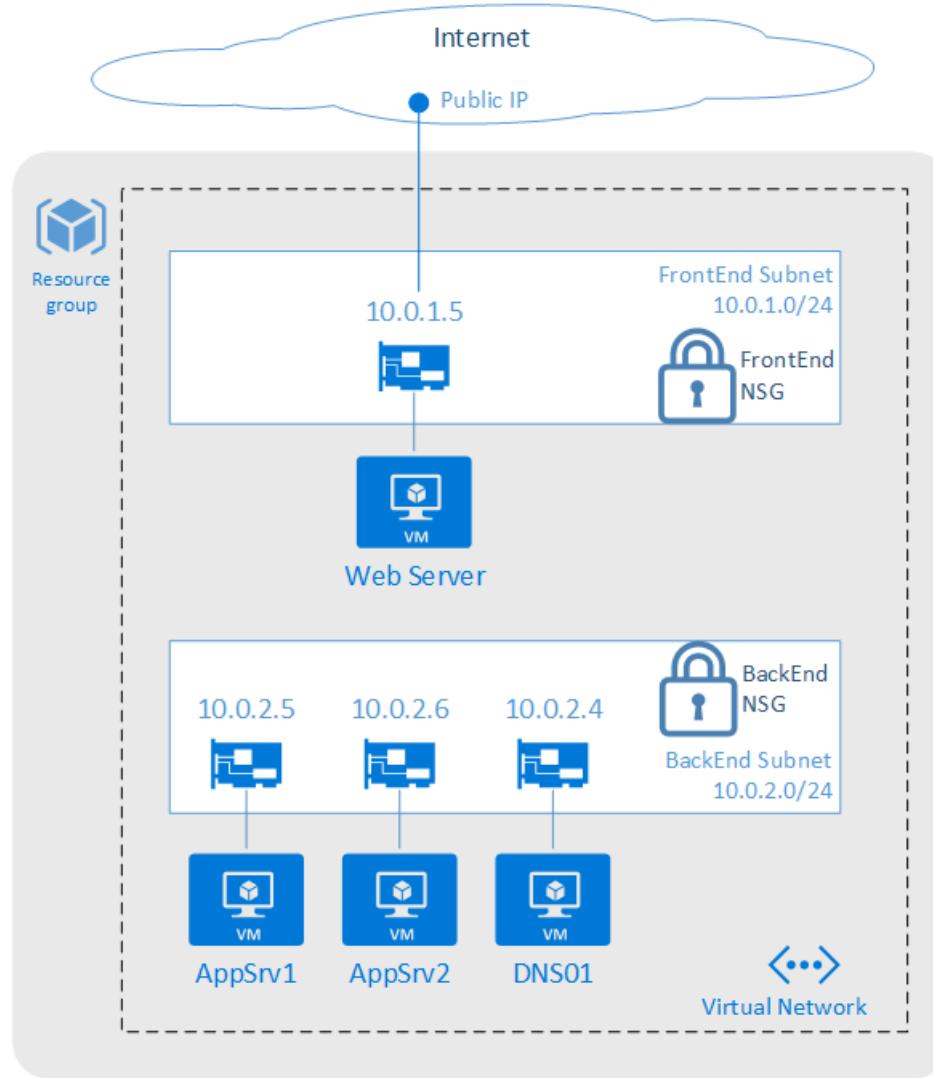
Additionally, traffic between virtual networks within Azure may be needed. These scenarios will be added in the future.

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).

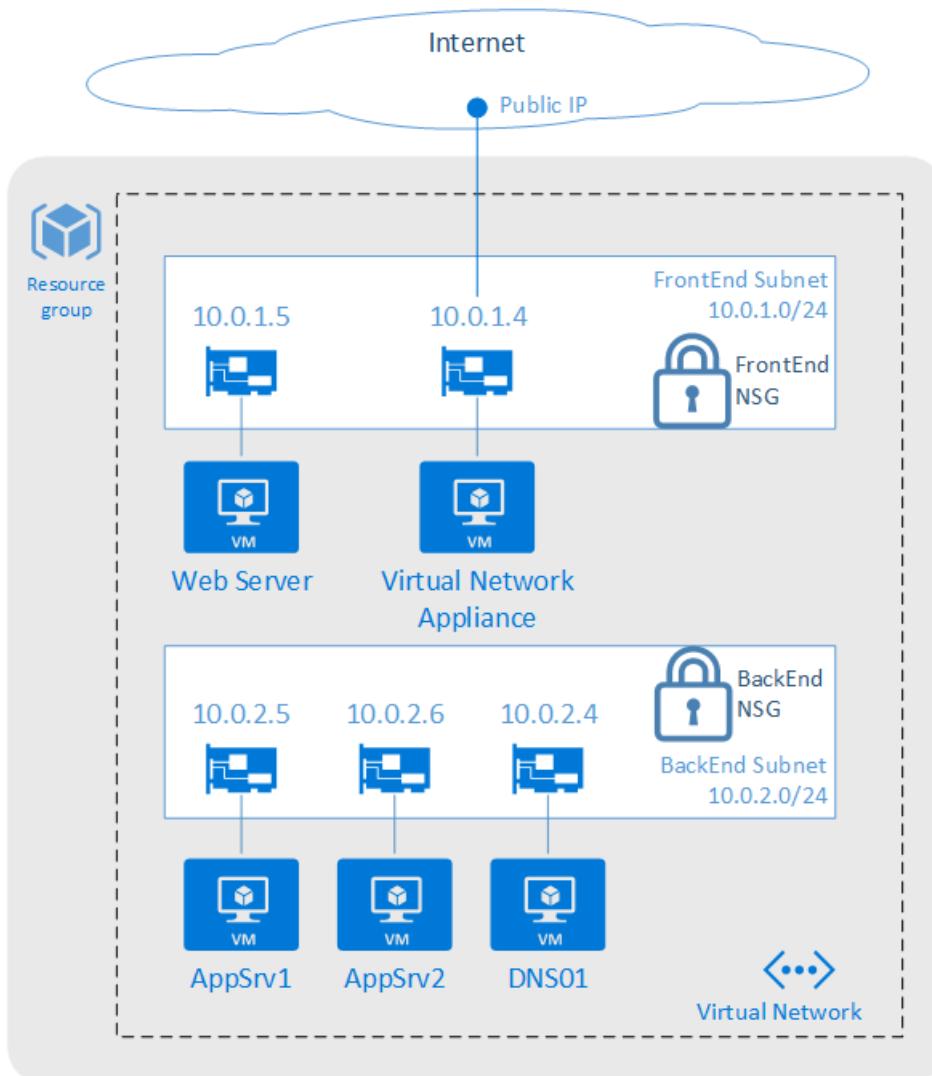
Conclusion

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.

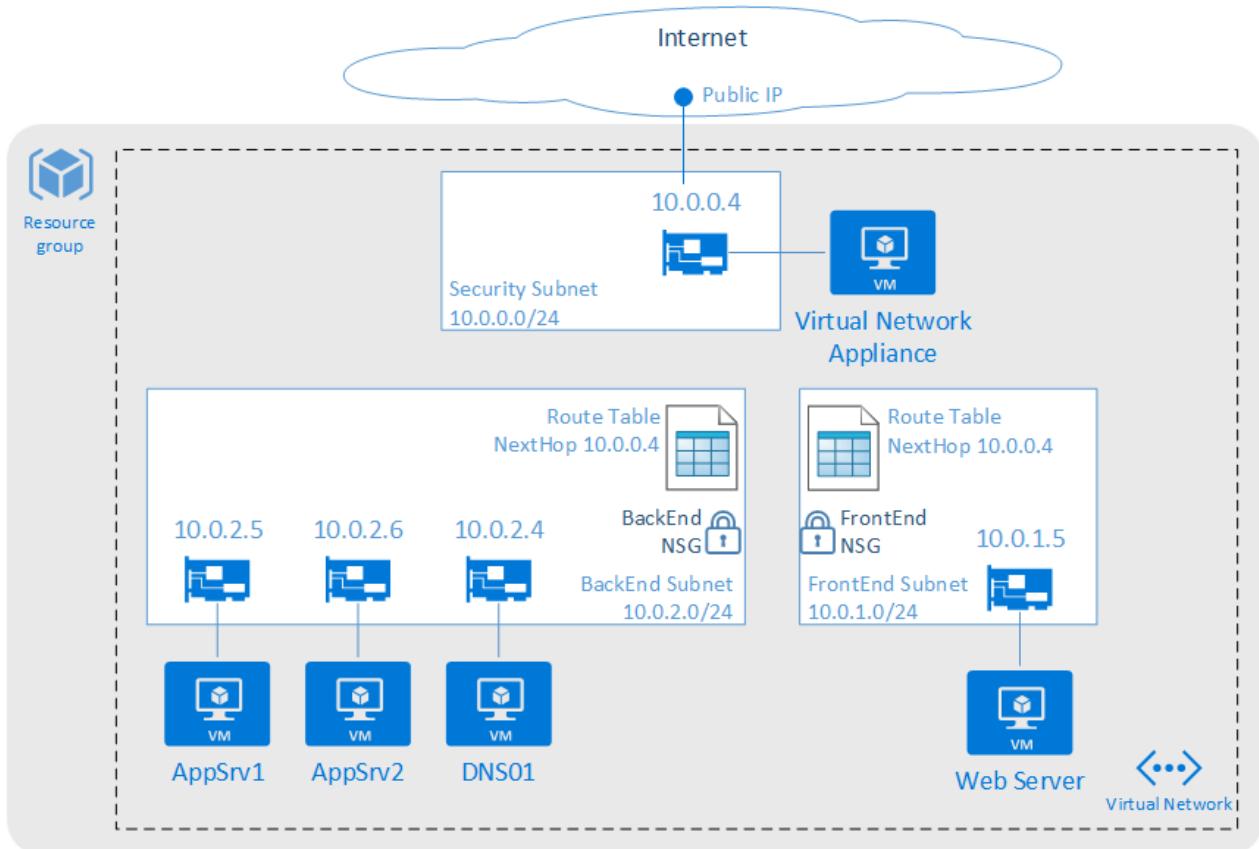
Conclusion

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:

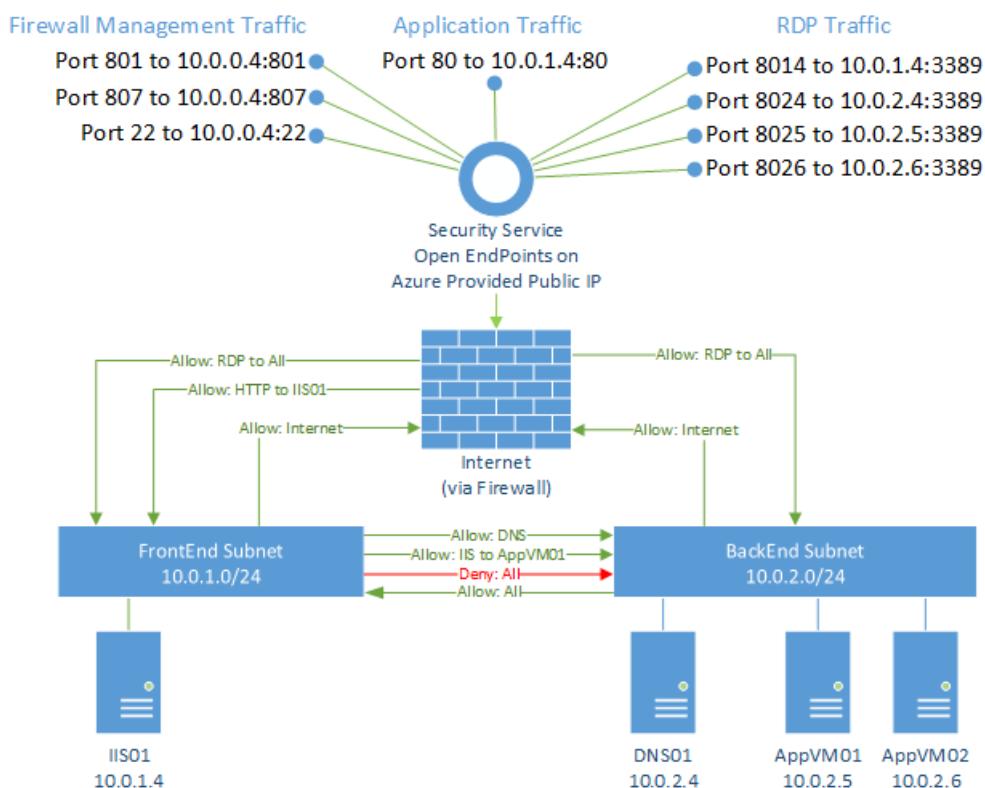
- No Public IP addresses on any FrontEnd or BackEnd NICs.

- 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).
 - 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.
 - 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.

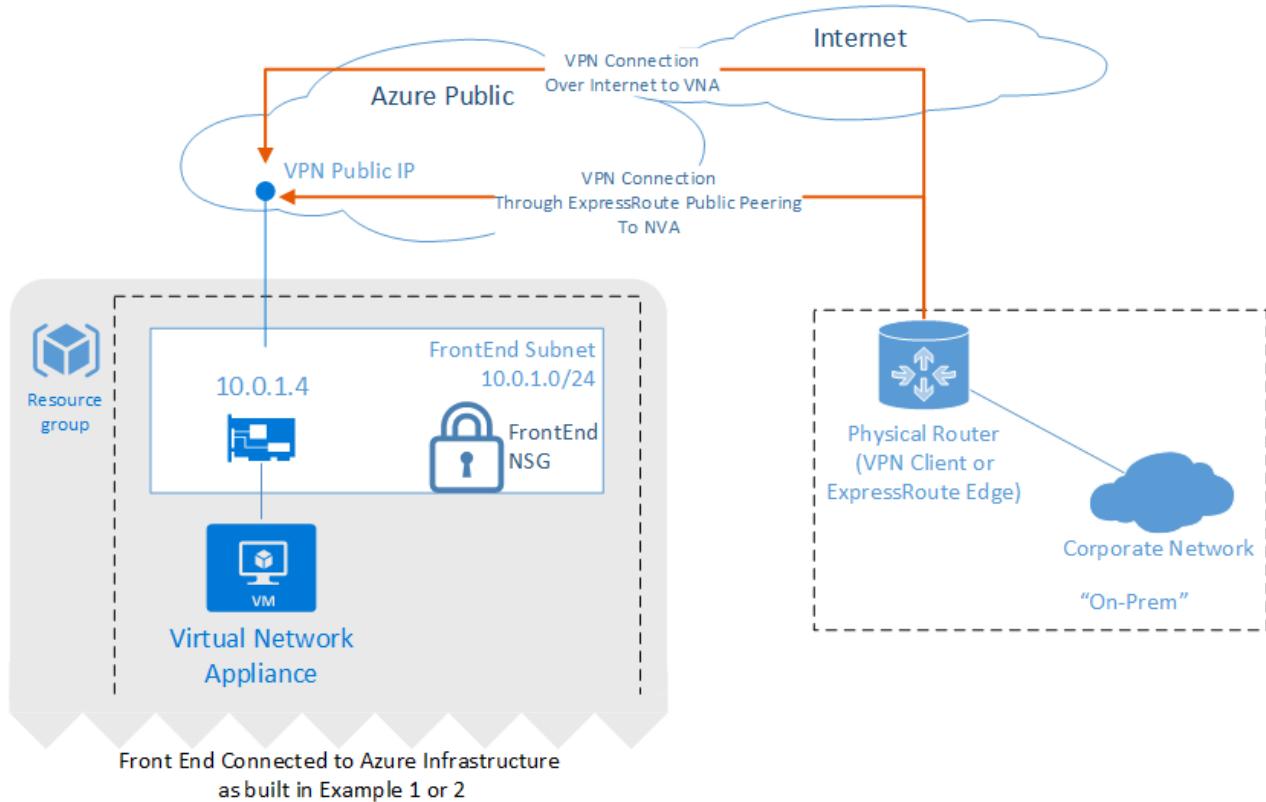
Once the previous rules are created, it's important to review the priority of each rule to ensure traffic is allowed or denied as desired. For this example, the rules are in priority order.

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.

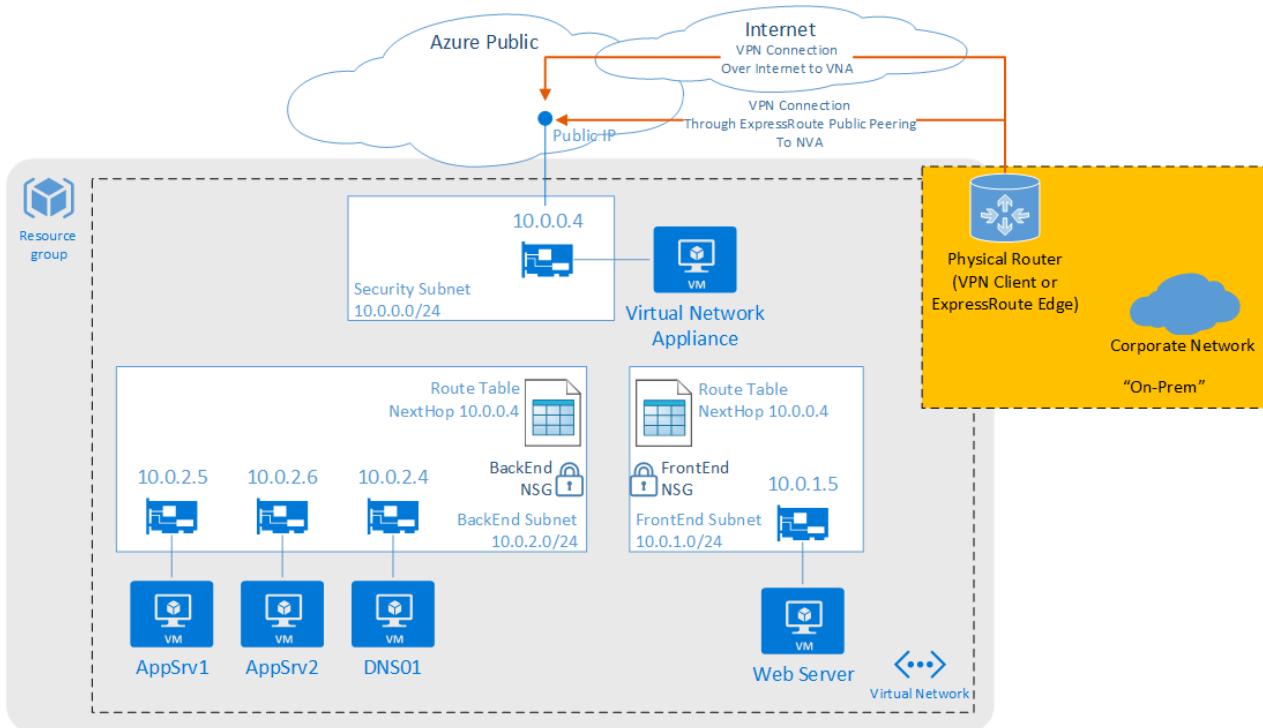
NOTE

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).

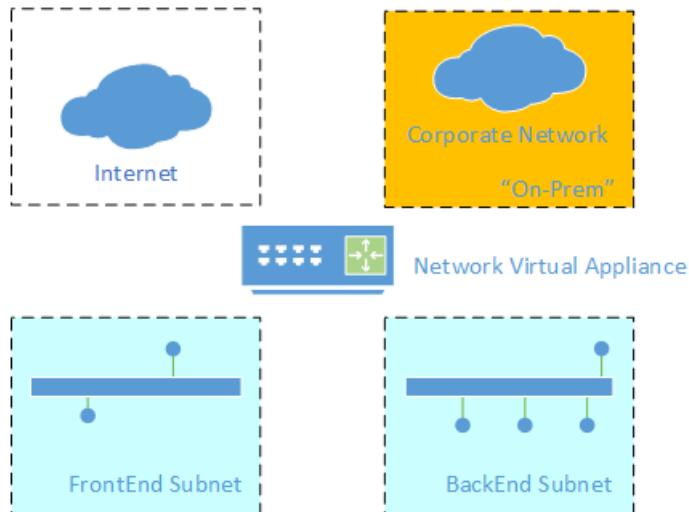
Traffic flows should be considered carefully, as they can be optimized or degraded by this design pattern, depending on the specific use case.

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:



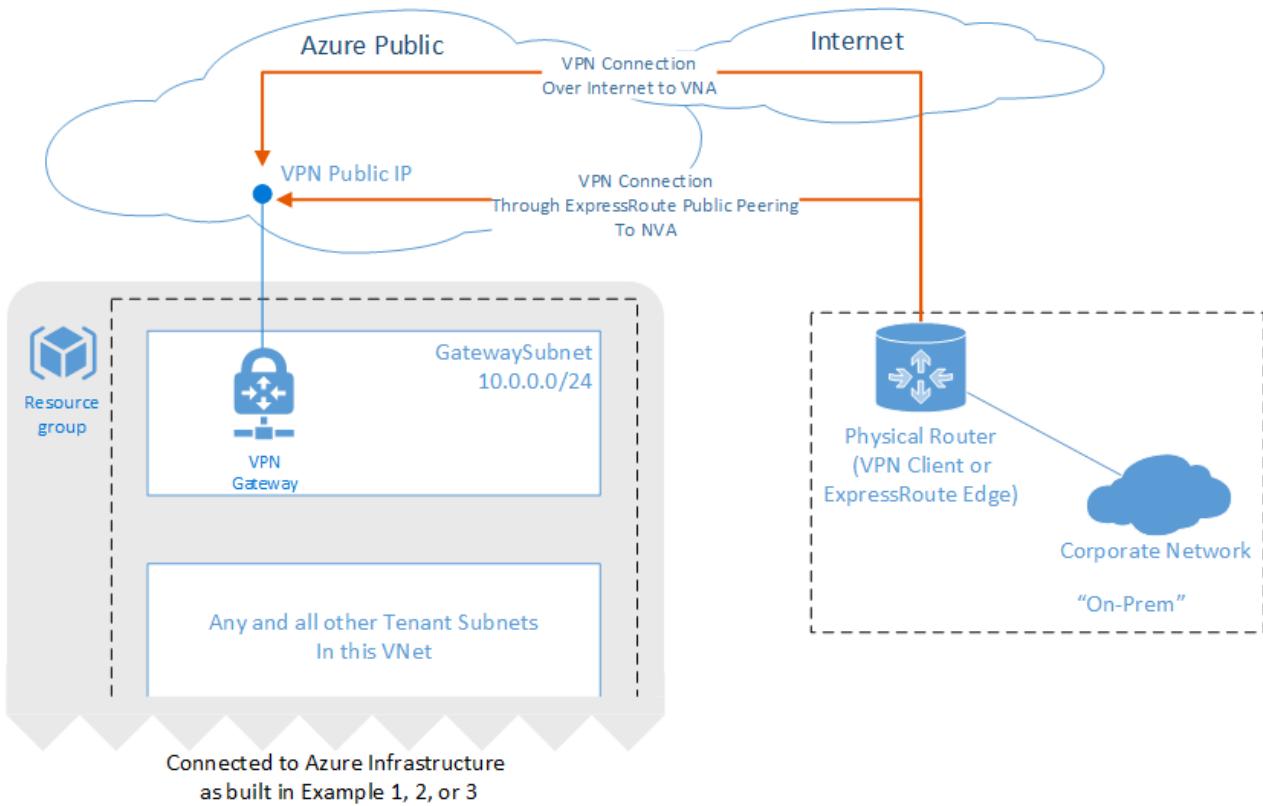
Conclusion

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.

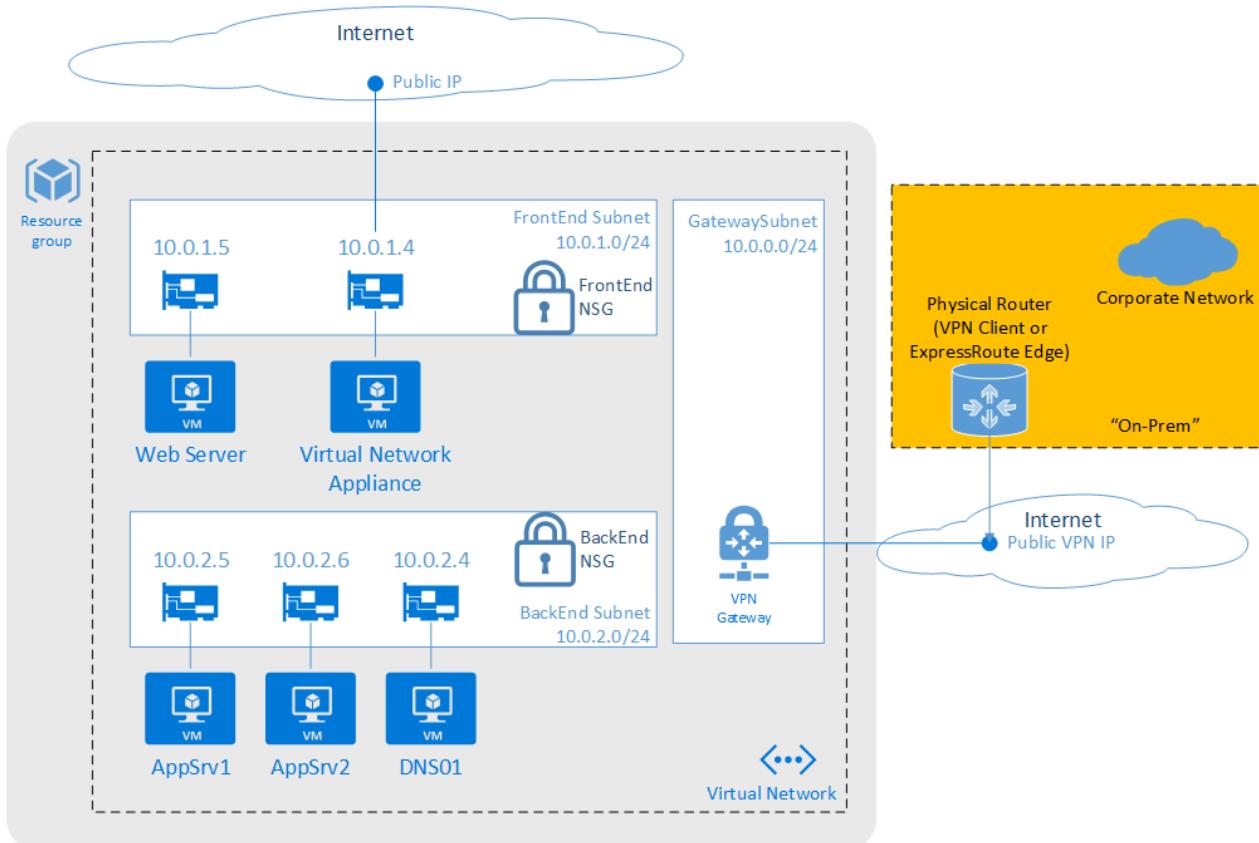
NOTE

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:



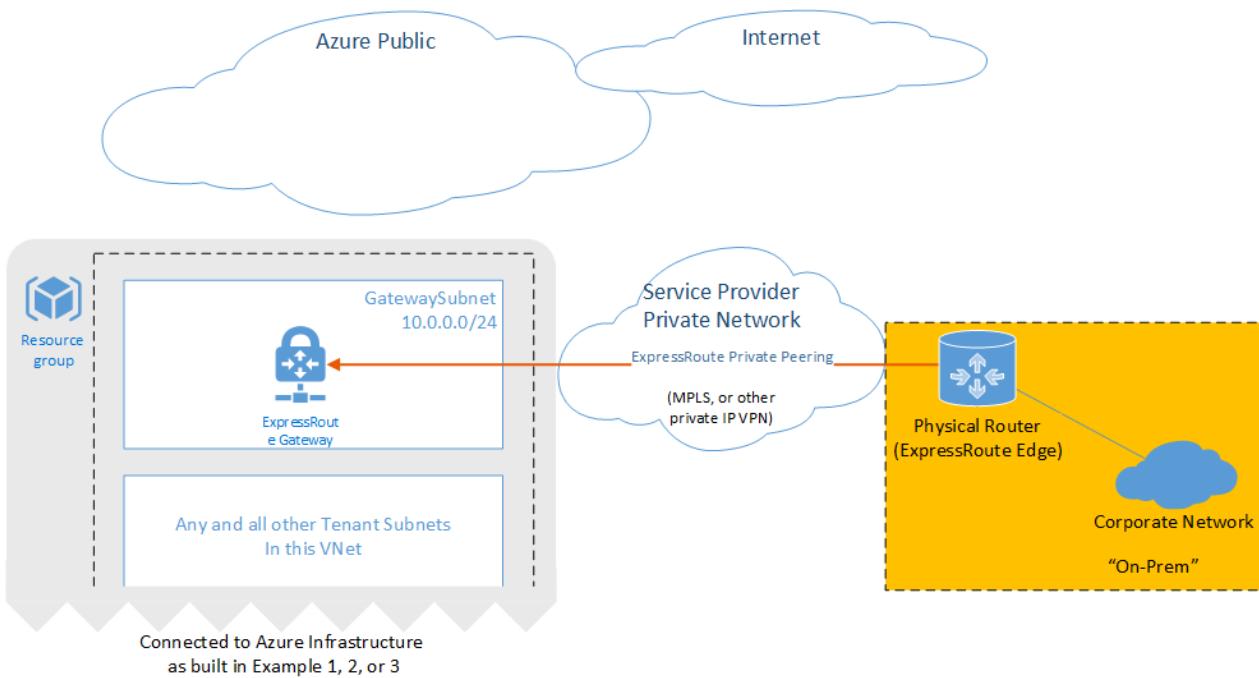
Conclusion

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.

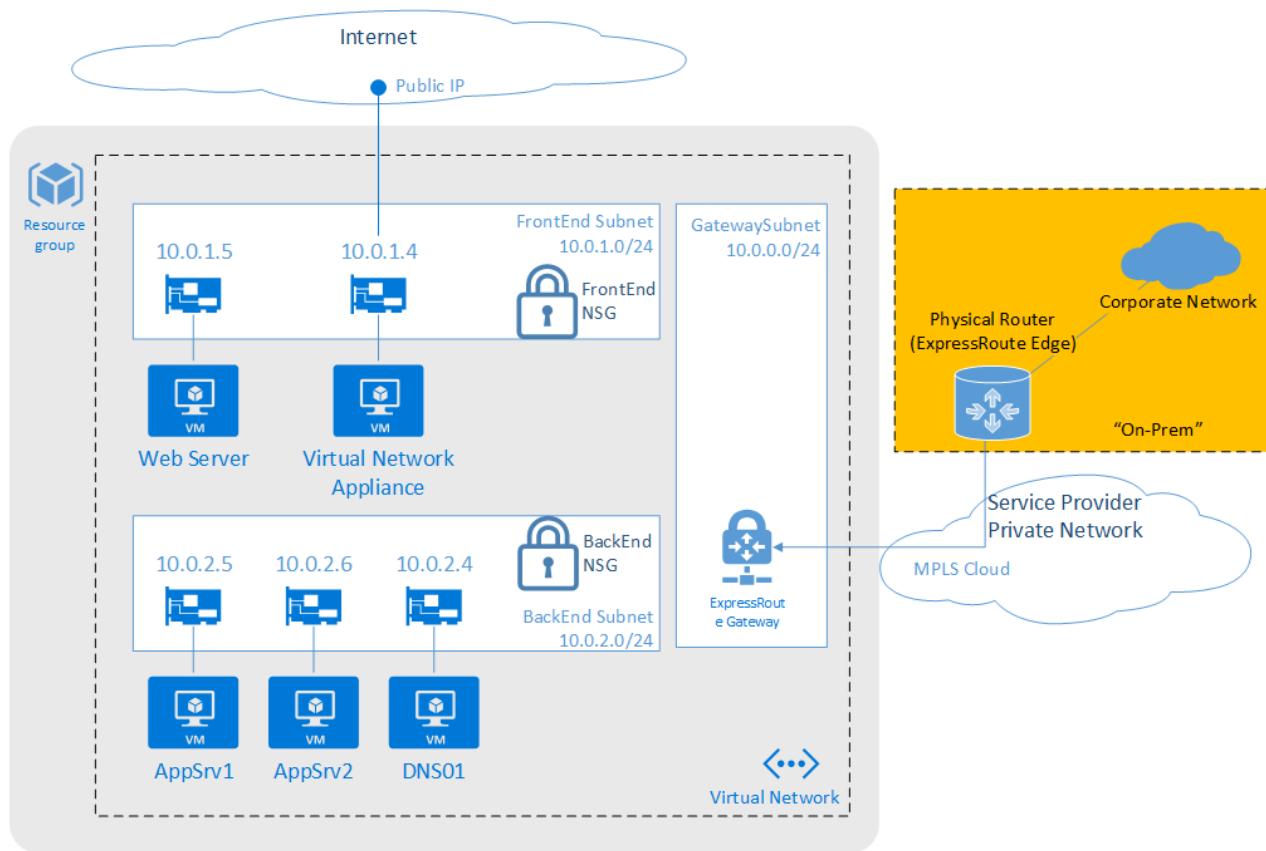
TIP

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

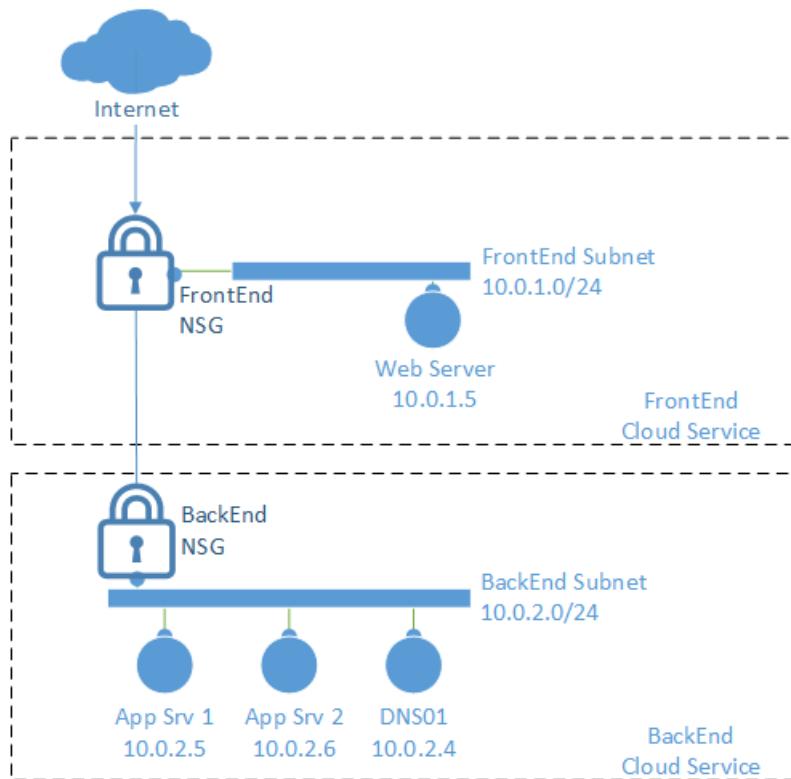
- Access Azure with Azure Resource Manager:
- Accessing Azure with PowerShell: <https://docs.microsoft.com/powershell/azureps-cmdlets-docs/>
- Virtual networking documentation: <https://docs.microsoft.com/azure/virtual-network/>
- Network security group documentation: <https://docs.microsoft.com/azure/virtual-network/virtual-networks-nsg>
- 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

1/17/2017 • 20 min to read • [Edit on GitHub](#)

[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 $FEPrefix -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}}
}

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>
                <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](#)

Next steps

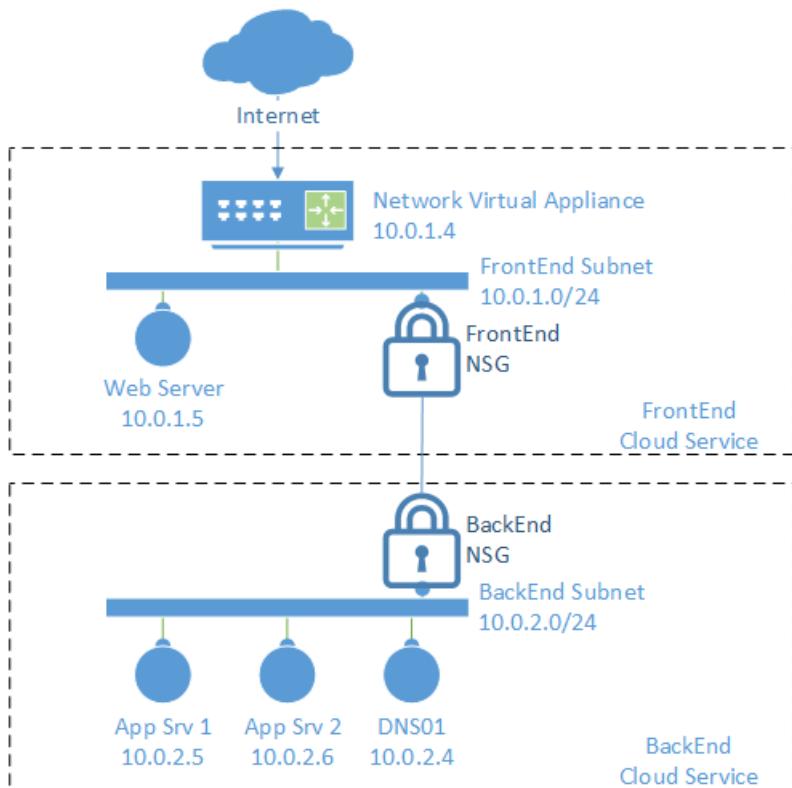
- Update and save XML file
- Run the PowerShell script to build the environment
- Install the sample application
- Test different traffic flows through this DMZ

Example 2 – Build a DMZ to protect applications with a Firewall and NSGs

1/17/2017 • 22 min to read • [Edit on GitHub](#)

[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 be 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.

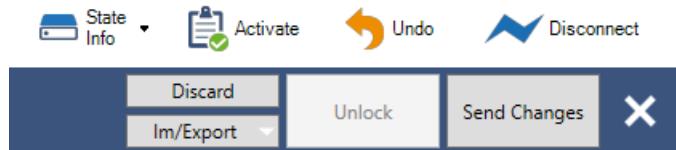
Although only one rule has been created it's important that its priority is set correctly. If in the grid of all rules on

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
# 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"

```

```

# 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 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 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

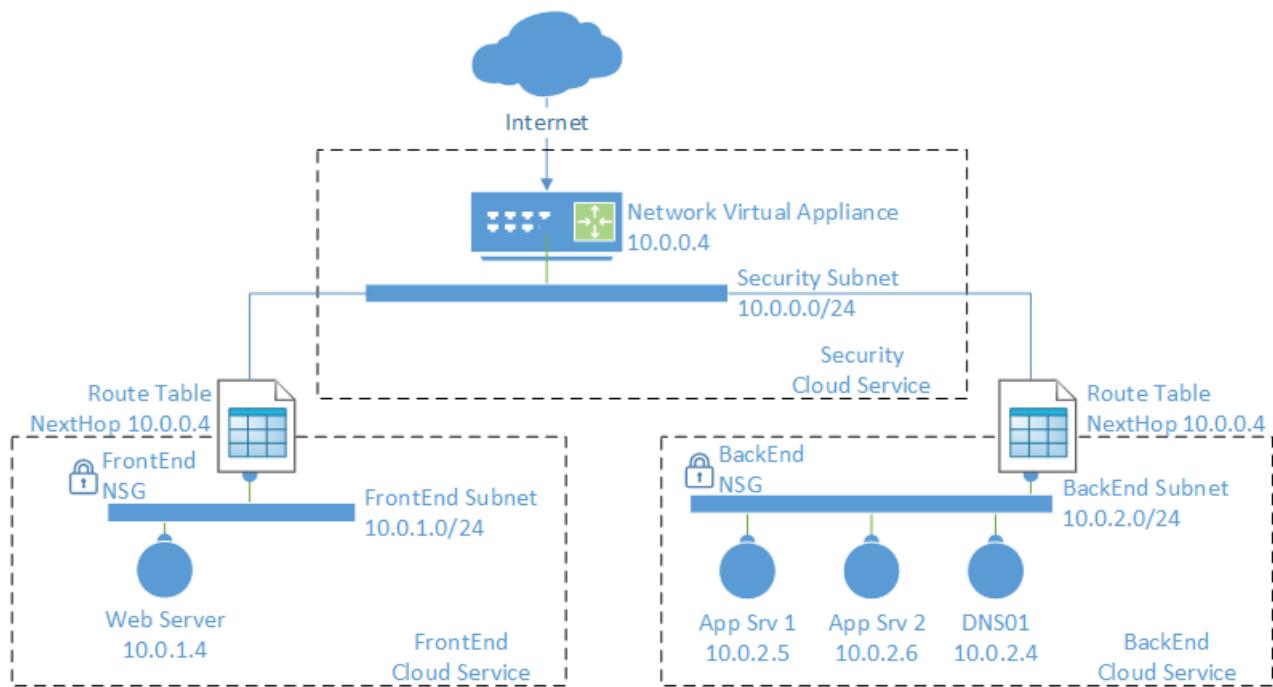
If you wish to install a sample application for this, and other DMZ Examples, one has been provided at the following link: [Sample Application Script](#)

Example 3 – Build a DMZ to Protect Networks with a Firewall, UDR, and NSG

1/17/2017 • 48 min to read • [Edit on GitHub](#)

[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-premise 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:

- 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.

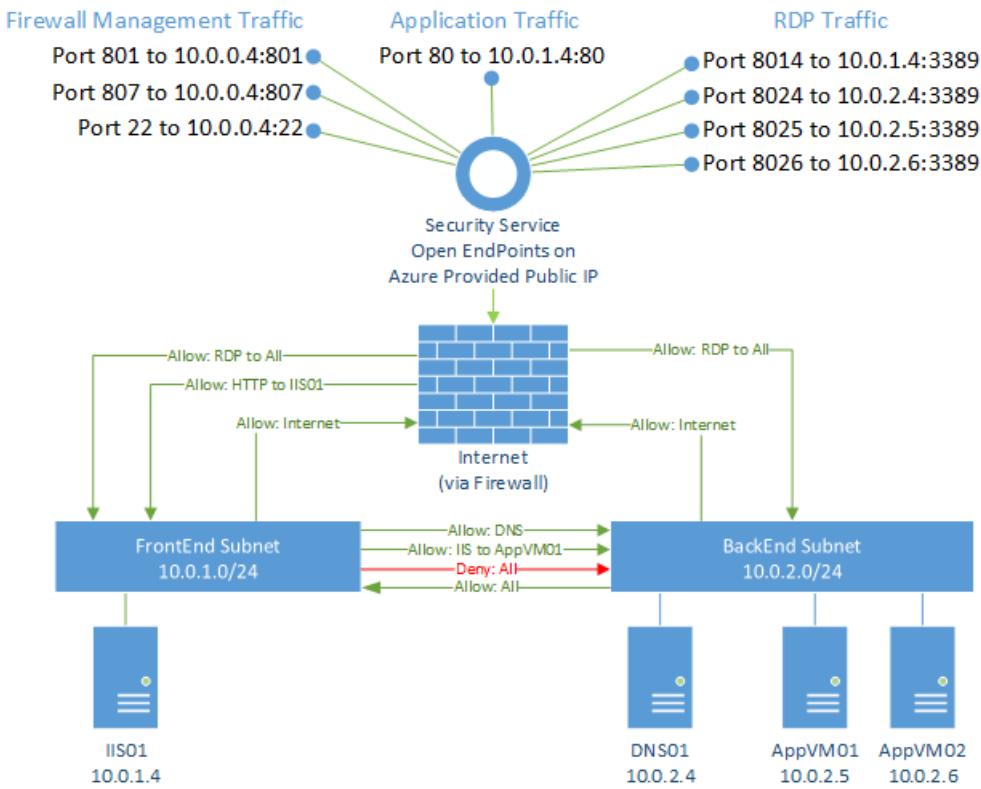
```
Get-AzureNetworkSecurityGroup -Name $NSGName |  
Set-AzureNetworkSecurityRule -Name "Isolate the $VNetName VNet "  
from the Internet"  
-Type Inbound -Priority 100 -Action Deny  
-SourceAddressPrefix INTERNET -SourcePortRange '*'  
-DestinationAddressPrefix VIRTUAL_NETWORK  
-DestinationPortRange '*'  
-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. secsv001.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
```

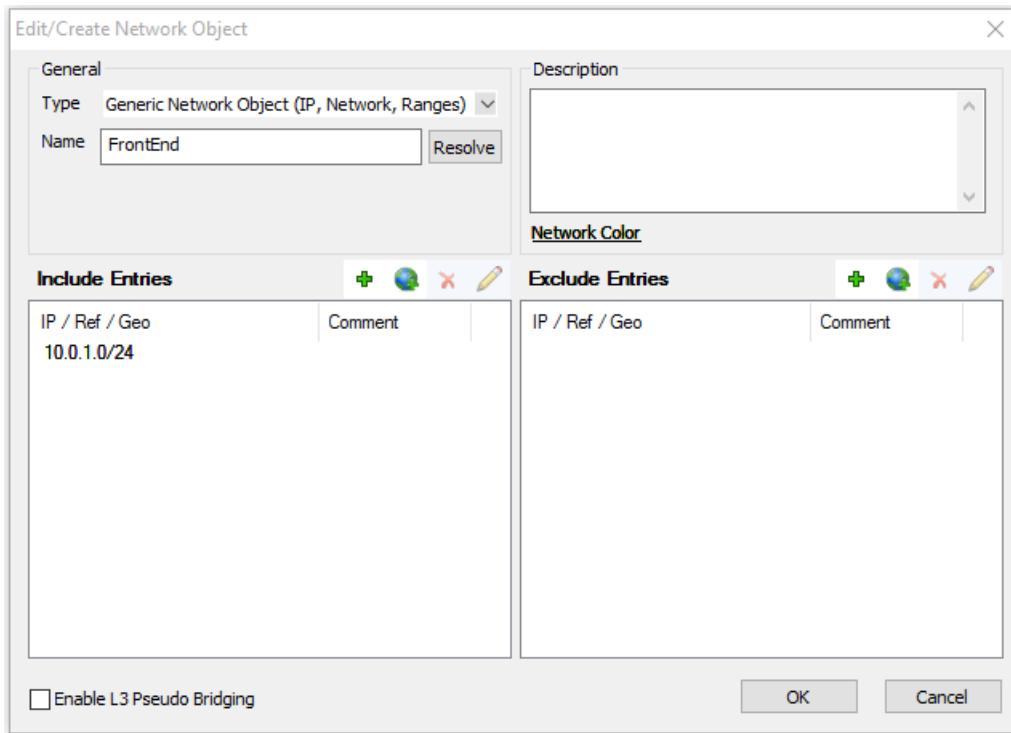
Although not clearly shown here due to the use of variables, but endpoints are **only** opened on the Security Cloud Service. This is to ensure that all inbound traffic is handled (routed, NAT'd, dropped) by the firewall.

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](#)

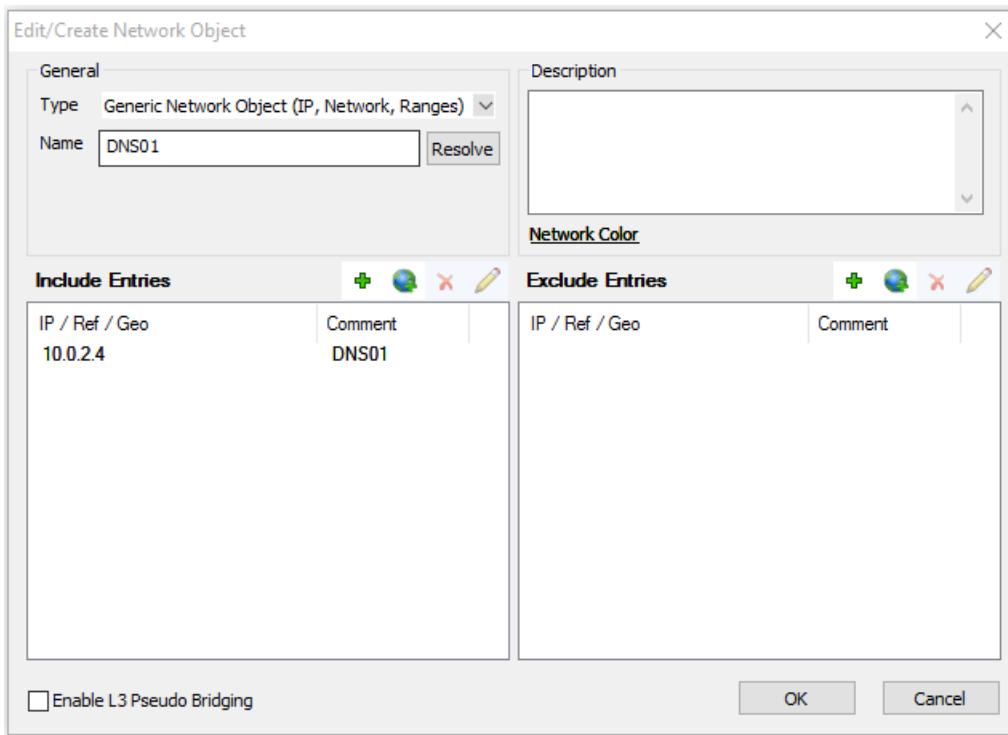
Once logged onto the firewall but before creating firewall rules, there are two prerequisite object classes that can make creating the rules easier; Network & Service objects.

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:



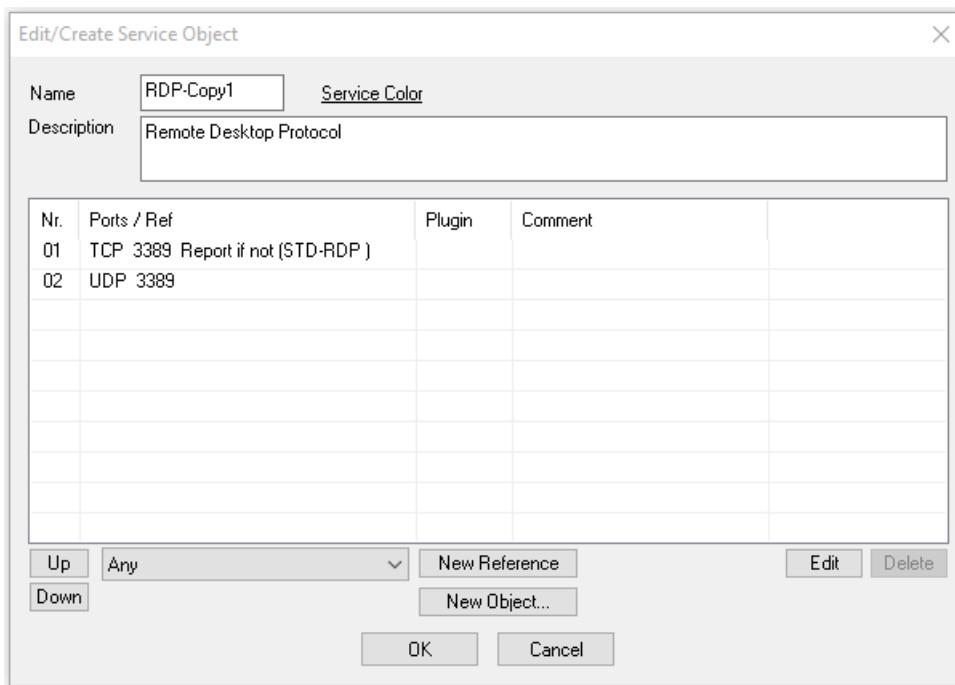
This will create a named network for the FrontEnd subnet, a similar object should be created for the BackEnd subnet as well. Now the subnets can be more easily referenced by name in the firewall rules.

For the DNS Server Object:

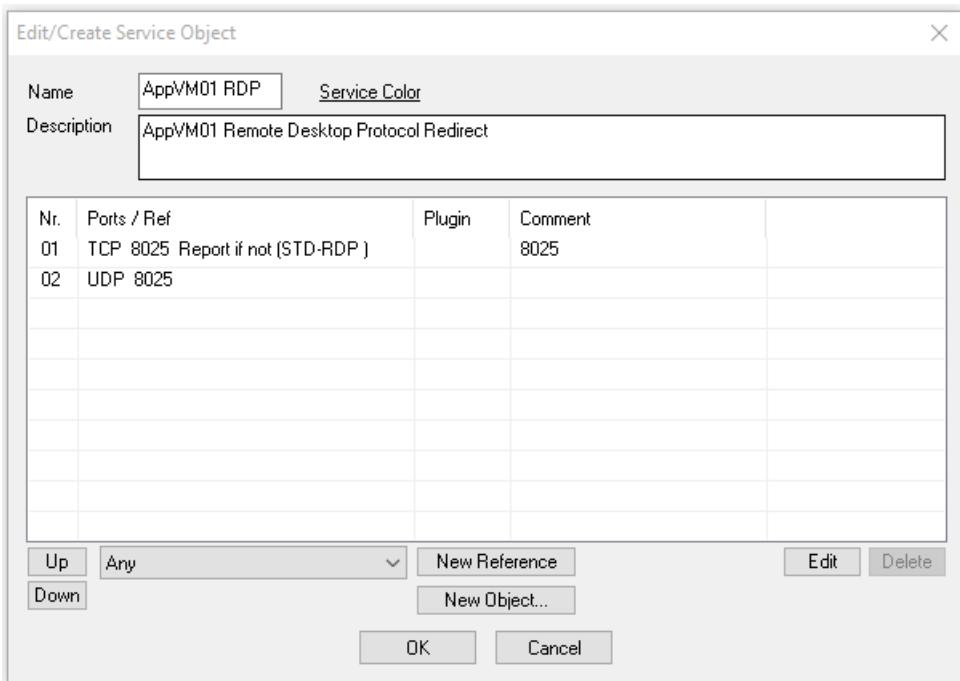


This single IP address reference will be utilized in a DNS rule later in the document.

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 configuration of an App Redirect rule. The rule is named "SETUP-MGMT-ACCESS". It is set to Bi-Directional and is a Dynamic Rule. The Source is set to "Any" with the value "0.0.0.0/0". The Service is set to "NGF-MGMT-BOX" with a dropdown menu showing options like "Ref: NGF-MGMT-CONF", "Ref: NGF-MGMT-CTRL", etc. The Destination is set to "DHCP1 Local IP". In the Redirection section, the 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 configuration of a Destination NAT rule. The rule is named "RDP-to-AppVM01". It is set to Bi-Directional and is a Dynamic Rule. The Source is set to "Any" with the value "0.0.0.0/0". The Service is set to "AppVM01 RDP" with a dropdown menu showing "TCP 8025 Report if not (STD-RDP)" and "UDP 8025". The Destination is set to "DHCP1 Local IP". In the Redirection section, the Target List is set to "10.0.2.5:3389".

A total of four RDP rules will need to be created:

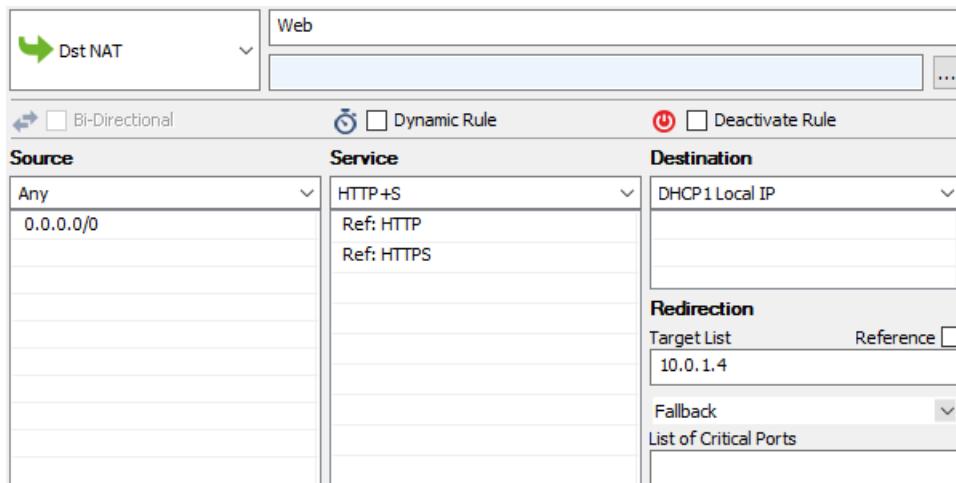
RULE NAME	SERVER	SERVICE	TARGET LIST
RDP-to-IIS01	IIS01	IIS01 RDP	10.0.1.4:3389
RDP-to-DNS01	DNS01	DNS01 RDP	10.0.2.4:3389
RDP-to-AppVM01	AppVM01	AppVM01 RDP	10.0.2.5:3389
RDP-to-AppVM02	AppVM02	AppVm02 RDP	10.0.2.6:3389

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).

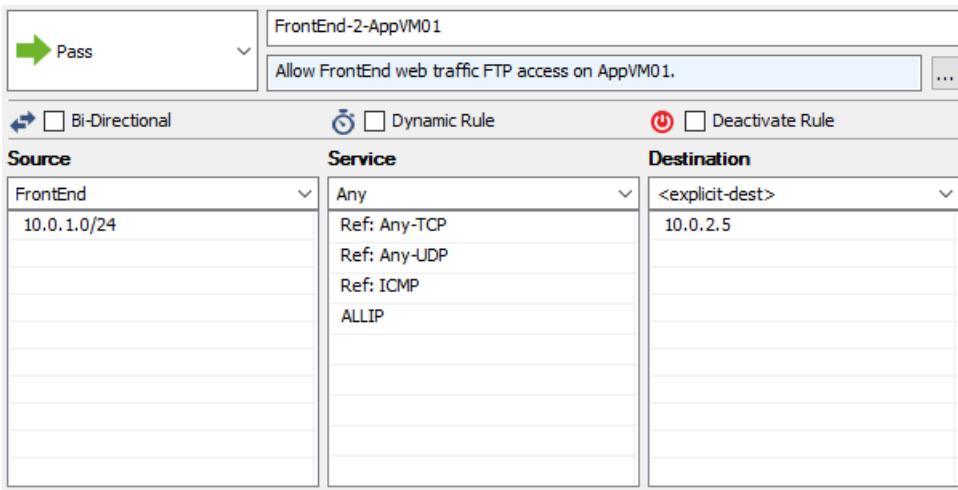
First discussed is the front end rule for web traffic:



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. secsvc001.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"
$FEPPrefix = "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 Appliaction 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 diffrent 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}

```

```

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 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-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 $FEPrefix `

-NextHopType VNETLocal

# Assoicate 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

```

```

Write-Host "Writing rules into the NSG - Foreground Color 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

1/17/2017 • 5 min to read • [Edit on GitHub](#)

[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-Asp-Net45, Web-Mgmt-Tools, Web-Mgmt-Console

# Create Web App Pages
# Add-Content "C:\inetpub\wwwroot\index.html" -Value "Hello from IIS01!" -Force
```

```

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>
                </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>
</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

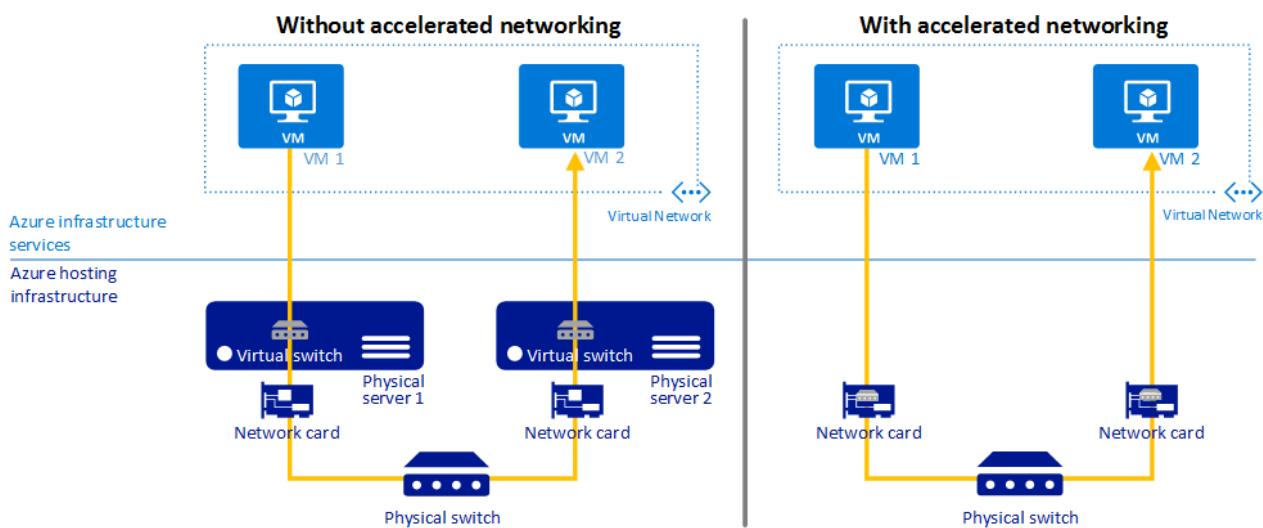
- Run the IIS01 script on an IIS server
- Run File Server script on AppVM01
- Browse to the Public IP on IIS01 to validate your build

Accelerated networking for a virtual machine

1/17/2017 • 4 min to read • [Edit on GitHub](#)

Accelerated Networking enables Single Root I/O Virtualization (SR-IOV) to a virtual machine (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. This article explains how to use the Azure Portal to configure Accelerated Networking in the Azure Resource Manager deployment model. You can also create a VM with Accelerated Networking using Azure PowerShell. To learn how, click the PowerShell box at the top of this article.

The following picture shows communication between two virtual machines (VM) 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, read the [Hyper-V Network Virtualization and Virtual Switch](#) article.

With Accelerated Networking, network traffic arrives at the network card (NIC) and is then forwarded to the VM. All network policies that the virtual switch applies without Accelerated Networking are 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 VNet. When communicating across VNets or connecting on-premises, this feature has a minimal impact to overall latency.

WARNING

This feature is currently in preview release and may not have the same level of availability and reliability as features that are in general availability release. The feature is not supported, may have constrained capabilities, and may not be available in all Azure locations. For the most up-to-date notifications on availability and status of this feature, check the [Azure Virtual Network updates](#) page.

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

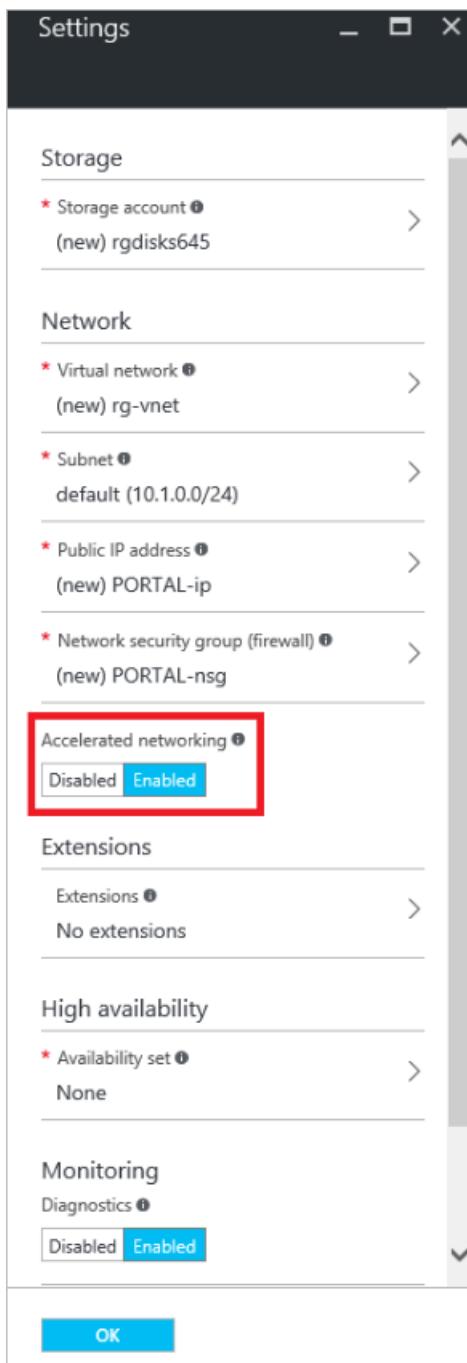
The following limitations exist when using this capability:

- **Network interface creation:** Accelerated networking can only be enabled for a new network interface. It cannot be enabled on an existing network interface.
- **VM creation:** A network interface with accelerated networking enabled can only be attached to a VM when the VM is created. The network interface cannot be attached to an existing VM.
- **Regions:** Offered in the West Central US and West Europe Azure regions only. The set of regions will expand in the future.
- **Supported operating system:** Microsoft Windows Server 2012 R2 and Windows Server 2016 Technical Preview 5. Linux and Windows Server 2012 support will be added soon.
- **VM Size:** Standard_D15_v2 and Standard_DS15_v2 are the only supported VM instance sizes. For more information, see the [Windows VM sizes](#) article. The set of supported VM instance sizes will expand in the future.

Changes to these limitations will be announced through the [Azure Virtual Networking updates](#) page.

Create a Windows VM with Accelerated Networking

1. Register for the preview by sending an email to [Accelerated Networking Subscriptions](#) with your subscription ID and intended use. Do not complete the remaining steps until after you receive an e-mail notifying you that you've been accepted into the preview.
2. Login to the Azure Portal at <http://portal.azure.com>.
3. Create a VM by completing the steps in the [Create a Windows VM](#) article selecting the following options:
 - Select an operating system listed in the Limitations section of this article.
 - Select a location (region) listed in the Limitations section of this article.
 - Select a VM size listed in the Limitations section of this article. If one of the supported sizes isn't listed, click **View all** in the **Choose a size** blade to select a size from an expanded list.
 - In the **Settings** blade, click *Enabled* for **Accelerated networking**, as shown in the following picture:

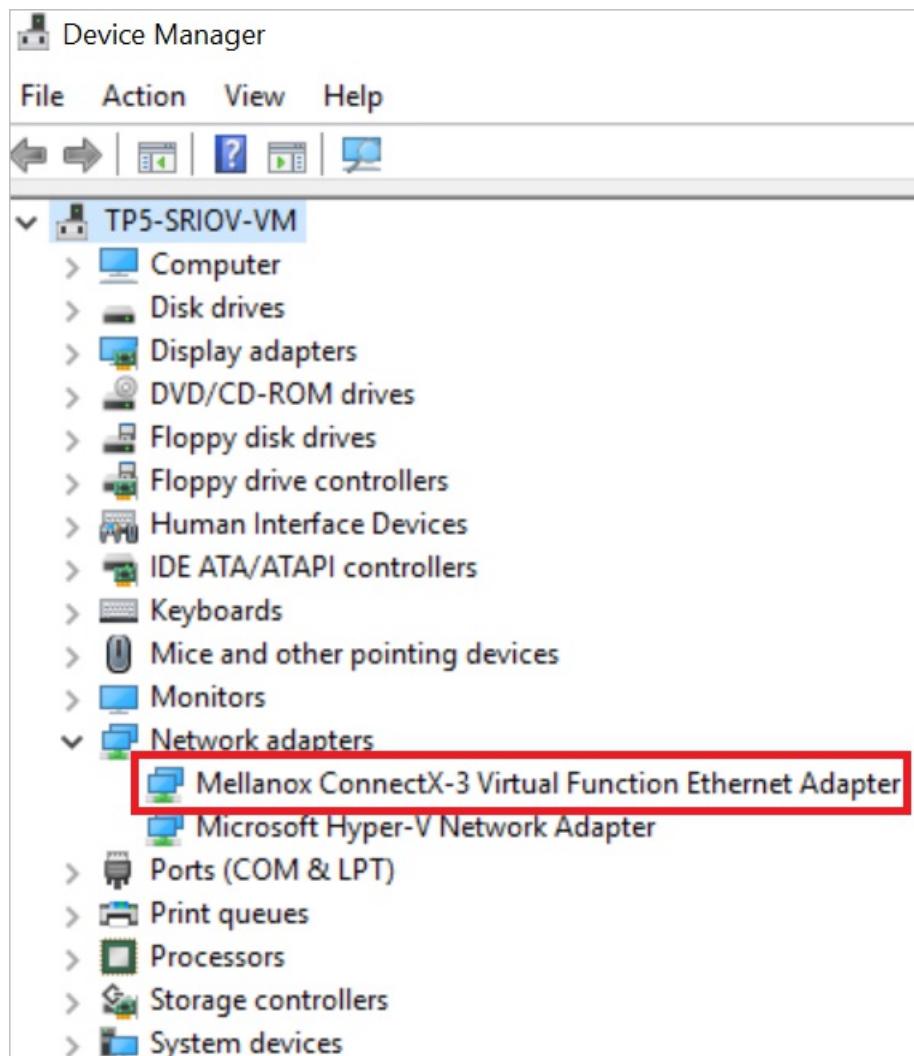


NOTE

The Accelerated Networking option will only be visible if you've:

- Been accepted into the preview
- Selected supported operating system, location, and VM sizes mentioned in the Limitations section of this article.

4. Once the VM is created, download the [Accelerated Networking driver](#), connect and login to the VM, and run the driver installer inside the VM.
5. Right-click the Windows button and click **Device Manager**. Verify that the **Mellanox ConnectX-3 Virtual Function Ethernet Adapter** appears under the **Network** option when expanded, as shown in the following picture:

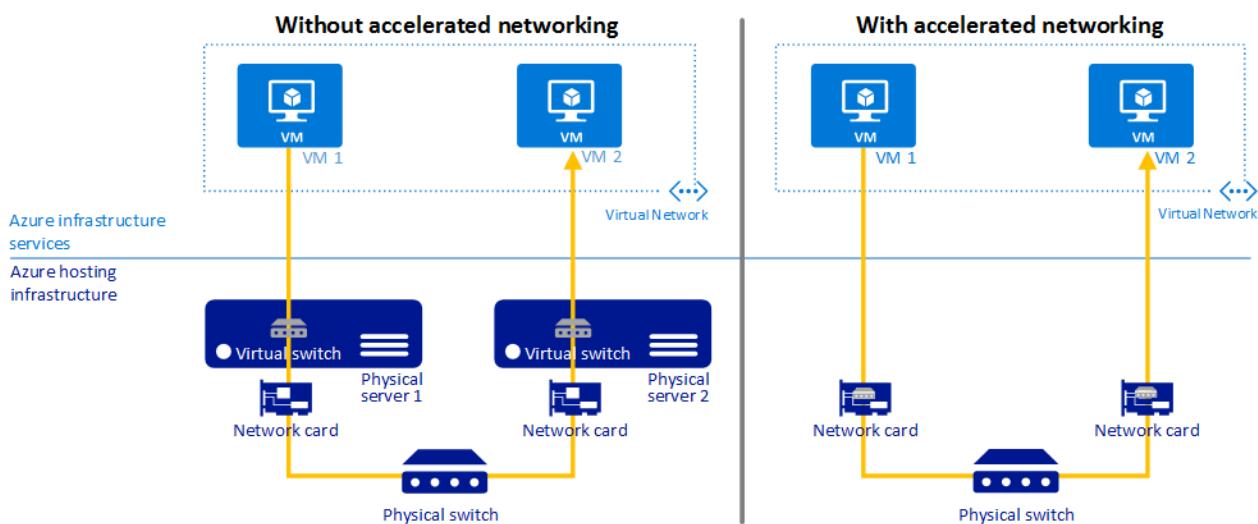


Accelerated networking for a virtual machine

1/17/2017 • 7 min to read • [Edit on GitHub](#)

Accelerated Networking enables Single Root I/O Virtualization (SR-IOV) to a virtual machine (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. This article explains how to use Azure PowerShell to configure Accelerated Networking in the Azure Resource Manager deployment model. You can also create a VM with Accelerated Networking using the Azure Portal. To learn how, click the Azure Portal box at the top of this article.

The following picture shows communication between two virtual machines (VM) 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, read the [Hyper-V Network Virtualization and Virtual Switch](#) article.

With Accelerated Networking, network traffic arrives at the network card (NIC) and is then forwarded to the VM. All network policies that the virtual switch applies without Accelerated Networking are 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 VNet. When communicating across VNets or connecting on-premises, this feature has a minimal impact to overall latency.

WARNING

This feature is currently in preview release and may not have the same level of availability and reliability as features that are in general availability release. The feature is not supported, may have constrained capabilities, and may not be available in all Azure locations. For the most up-to-date notifications on availability and status of this feature, check the [Azure Virtual Network updates](#) page.

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

The following limitations exist when using this capability:

- **Network interface creation:** Accelerated networking can only be enabled for a new network interface. It cannot be enabled on an existing network interface.
- **VM creation:** A network interface with accelerated networking enabled can only be attached to a VM when the VM is created. The network interface cannot be attached to an existing VM.
- **Regions:** Offered in the West Central US and West Europe Azure regions only. The set of regions will expand in the future.
- **Supported operating system:** Microsoft Windows Server 2012 R2 and Windows Server 2016 Technical Preview 5. Linux and Windows Server 2012 support will be added soon.
- **VM Size:** Standard_D15_v2 and Standard_DS15_v2 are the only supported VM instance sizes. For more information, see the [Windows VM sizes](#) article. The set of supported VM instance sizes will expand in the future.

Changes to these limitations will be announced through the [Azure Virtual Networking updates](#) page.

Create a Windows VM with Accelerated Networking

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. To register for the preview, send an email to [Accelerated Networking Subscriptions](#) with your subscription ID and intended use. Do not complete the remaining steps until after you receive an e-mail notifying you that you've been accepted into the preview.
3. Register the capability with your subscription by entering the following commands:

```
Register-AzureRmProviderFeature -FeatureName AllowAcceleratedNetworkingFeature -ProviderNamespace Microsoft.Network
Register-AzureRmResourceProvider -ProviderNamespace Microsoft.Network
```

4. Replace *westcentralus* with the name of another location supported by this capability listed in the [Limitations](#) section of this article (if desired). Enter the following command to set a variable for the location:

```
$locName = "westcentralus"
```

5. Replace *RG1* with a name for the resource group that will contain the new network interface and enter the following commands to create it:

```
$rgName = "RG1"
New-AzureRmResourceGroup -Name $rgName -Location $locName
```

6. Change *VM1-NIC1* to what you want to name the network interface and then enter the following command:

```
$NICName = "VM1-NIC1"
```

7. The network interface must be connected to a subnet within an existing Azure Virtual Network (VNet) in the same location and [subscription](#) as the network interface. Learn more about VNets by reading the [Virtual network overview](#) article if you're not familiar with them. To create a VNet, complete the steps in the [Create a VNet](#) article. Change the "values" of the following \$Variables to the name of the VNet and subnet you want to connect the network interface to.

```
$VNetName = "VNet1"  
$SubnetName = "Subnet1"
```

If you don't know the name of an existing VNet in the location you chose in step 3, enter the following commands:

```
$VirtualNetworks = Get-AzureRmVirtualNetwork  
$VirtualNetworks | Where-Object { $_.Location -eq $locName } | Format-Table Name, Location
```

If the list returned is empty, you need to create a VNet in the location. To create a VNet, complete the steps in the [Create a virtual network](#) article.

To get the name of the subnets within the VNet, type the following commands and replace *Subnet1* above with the name of a subnet:

```
$VNet = Get-AzureRmVirtualNetwork -Name $VNetName -ResourceGroupName $rgName  
$VNet.Subnets | Format-Table Name, AddressPrefix
```

8. Enter the following commands to retrieve the VNet and subnet and assign them to variables.

```
$VNet = Get-AzureRmVirtualNetwork -Name $VNetName -ResourceGroupName $rgName  
$Subnet = $VNet.Subnets | Where-Object { $_.Name -eq $SubnetName }
```

9. Identify an existing public IP address resource that can be associated to the network interface so you can connect to it over the Internet. If you don't want to access the VM with the network interface over the Internet, you can skip this step. Without a public IP address, you must connect to the VM from another VM connected to the same VNet.

Change *PIP1* to the name of an existing public IP address resource that exists in the location you're creating the network interface in and that isn't currently associated with another network interface. If necessary, change \$rgName to the name of the resource group the public IP address resource exists in and enter the following command:

```
$PIP1 = Get-AzureRmPublicIPAddress -Name "PIP1" -ResourceGroupName $rgName
```

Enter the following commands if you don't know the name of an existing public IP address resource:

```
$PublicIPAddresses = Get-AzureRmPublicIPAddress  
$PublicIPAddresses | Where-Object { $_.Location -eq $locName } | Format-Table Name, Location, IPAddress,  
IpConfiguration
```

If the **IPConfiguration** column has no value in the output returned, the public IP address resource is not associated with an existing network interface and can be used. If the list is blank, or there are no available public IP address resources, you can create one using the New-AzureRmPublicIPAddress command.

NOTE

Public IP addresses have a nominal fee. Learn more about pricing by reading the [IP address pricing](#) page.

10. If you chose not to add a public IP address resource to the interface, remove `-PublicIpAddress $PIP1` at the end of the command that follows. Create the network interface with accelerated networking by entering the following command:

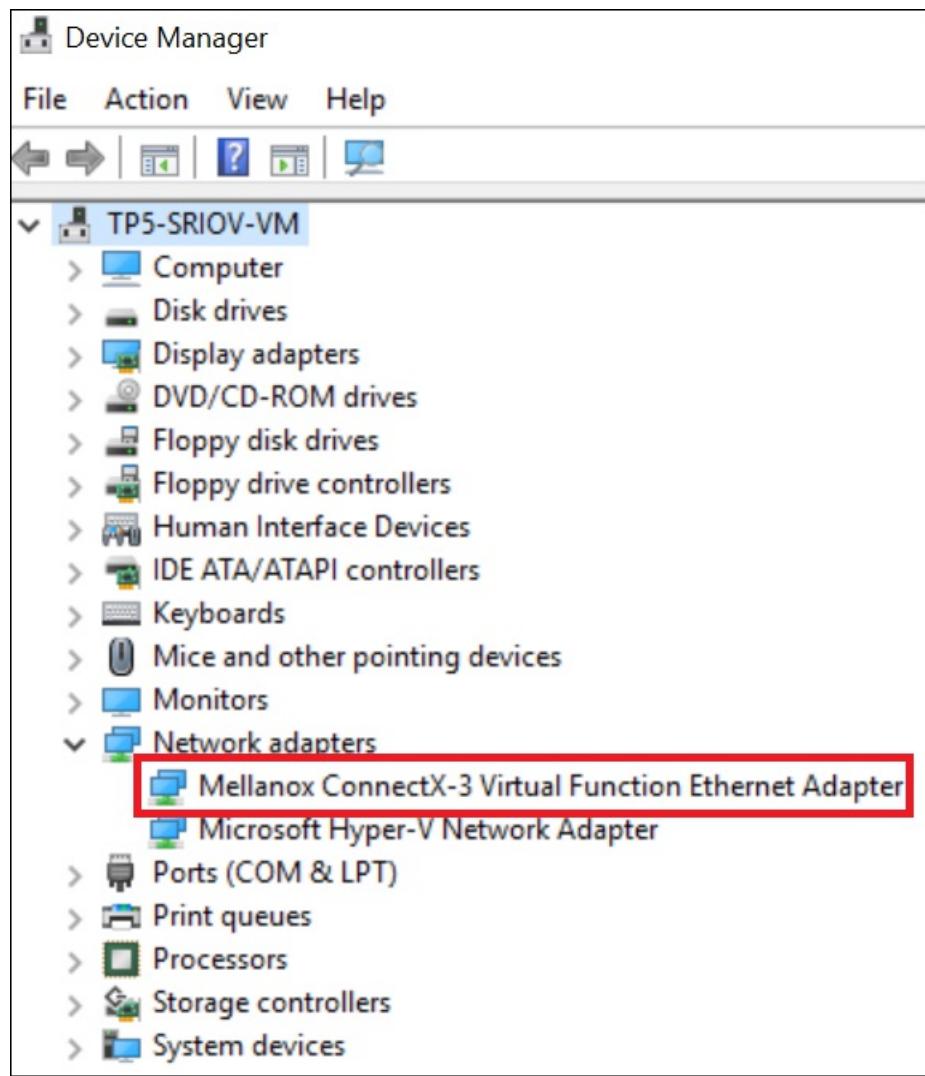
```
$nic = New-AzureRmNetworkInterface -Location $locName -Name $NICName -ResourceGroupName $rgName -Subnet  
$Subnet -EnableAcceleratedNetworking -PublicIpAddress $PIP1
```

11. Assign the network interface to a VM when creating the VM by following the instructions in steps 3 and 6 of the [Create a VM](#) article. In step 6-2, replace `Standard_A1` with one of the VM sizes listed in the [Limitations](#) section of this article.

NOTE

If you changed the *name* of the `$locName`, `$rgName`, or `$nic` variables in this article, step 6 in the Create a VM article will fail. You can however, change the *values* of the variables.

12. Once the VM is created, download the [Accelerated Networking driver](#), connect to the VM, and run the driver installer inside the VM.
13. Right-click the Windows button and click **Device Manager**. Verify that the **Mellanox ConnectX-3 Virtual Function Ethernet Adapter** appears under the **Network** option when expanded, as shown in the following picture:



What is an endpoint Access Control List (ACLs)?

1/17/2017 • 4 min to read • [Edit on GitHub](#)

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.

IMPORTANT

It is recommended to use Network Security Groups (NSGs) instead of ACLs whenever possible. To learn more about NSGs, see [What is a Network Security Group?](#).

ACLs can be configured by using either PowerShell or the Management Portal. To configure a network ACL by using PowerShell, see [Managing Access Control Lists \(ACLs\) for Endpoints by using PowerShell](#). To configure a network ACL by using the Management 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
- Specify up to 50 ACL 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.

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

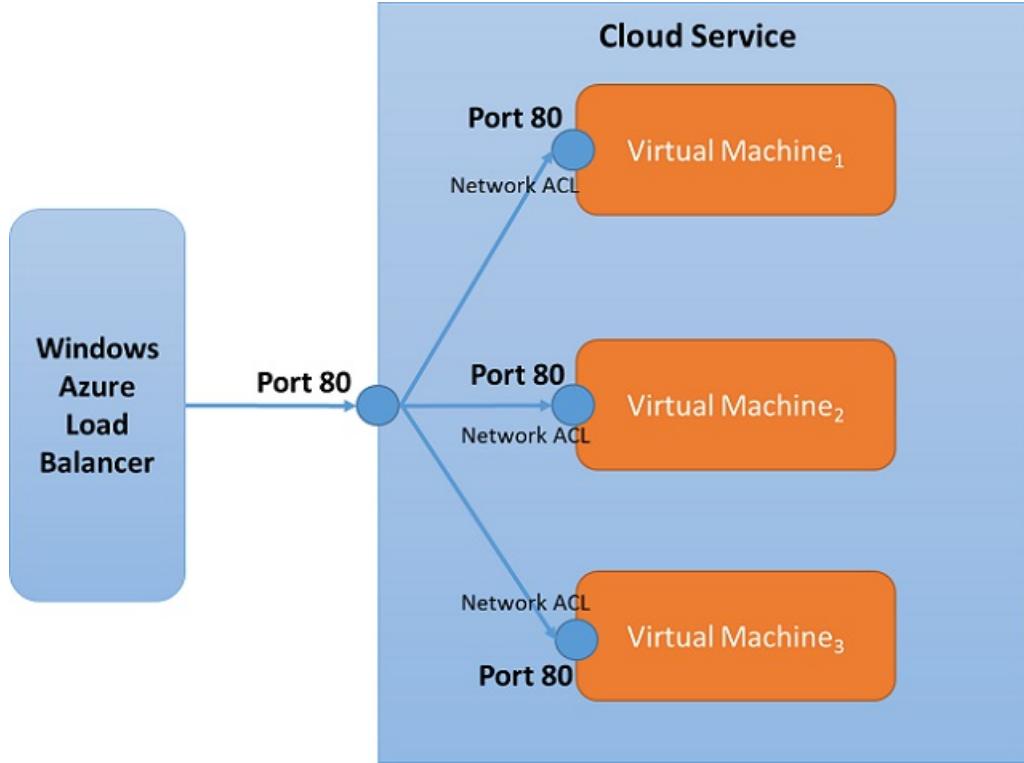
Because multiple rules can be specified for an endpoint, there must be a way to organize rules in order to determine which rule takes precedence. The rule order specifies precedence. Network ACLs follow a *lowest takes precedence* rule order. In the example below, the endpoint on port 80 is selectively 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 (LB Set) endpoint. If an ACL is specified for a LB Set, the Network ACL is applied to all Virtual Machines in that LB Set. For example, if a LB Set is created with "Port 80" and the LB Set contains 3 VMs, the Network ACL created on endpoint "Port 80" of one VM will automatically apply to the other VMs.



Next Steps

[How to manage Access Control Lists \(ACLs\) for Endpoints by using PowerShell](#)

How to manage Access Control Lists (ACLs) for Endpoints by using PowerShell

1/17/2017 • 3 min to read • [Edit on GitHub](#)

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 \(ACL\)?](#)

Manage NSGs using the portal

1/17/2017 • 5 min to read • [Edit on GitHub](#)

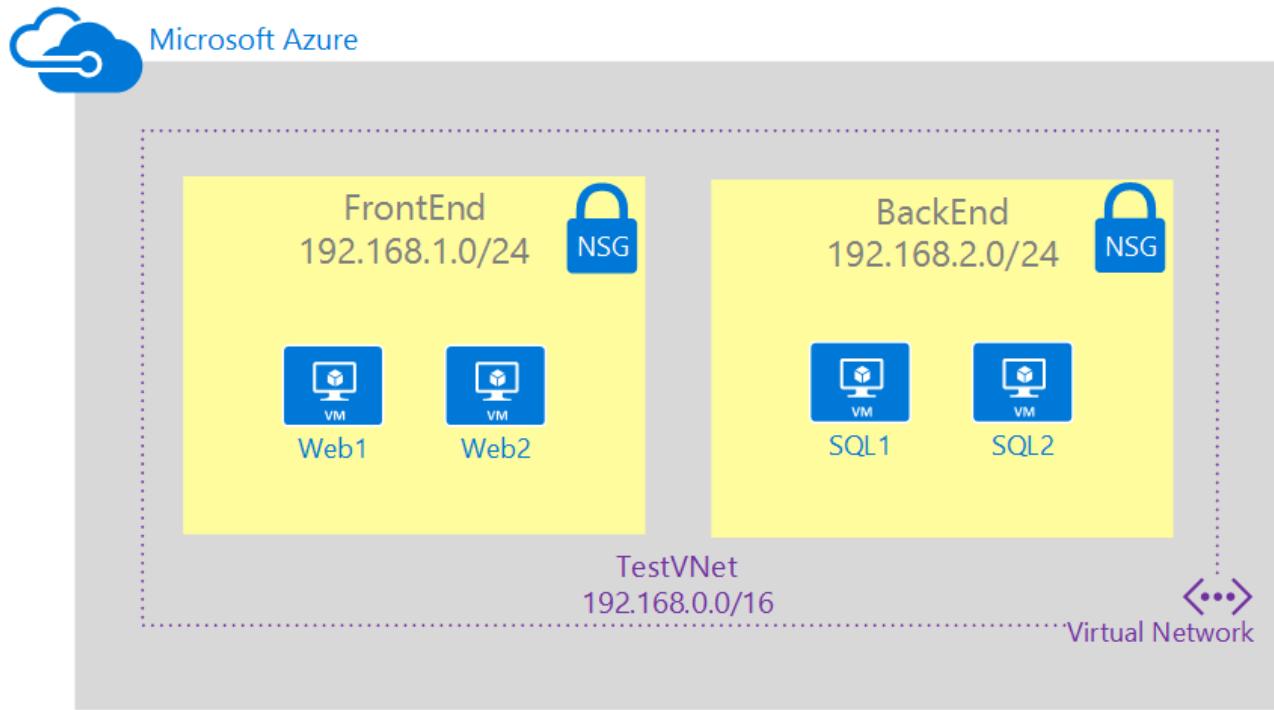
After you create one or more Network Security Groups (NSGs), you need to be able to retrieve information about your NSGs, add and remove rules, edit existing rules, associate or dissociate NSGs, and delete NSGs. In this article, you will learn how to execute each of these tasks. Before you can manage NSGs, it's important to know [how NSGs work](#).

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.

Sample Scenario

To better illustrate how to manage NSGs, this article uses the scenario below.



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

- **NSG-FrontEnd**. The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule**. This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule**. This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd**. The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule**. This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule**. This 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 traffic 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.

To deploy the scenario described above, follow [this link](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal. In the sample instructions below, the template was used to deploy a resource group names **RG-NSG**.

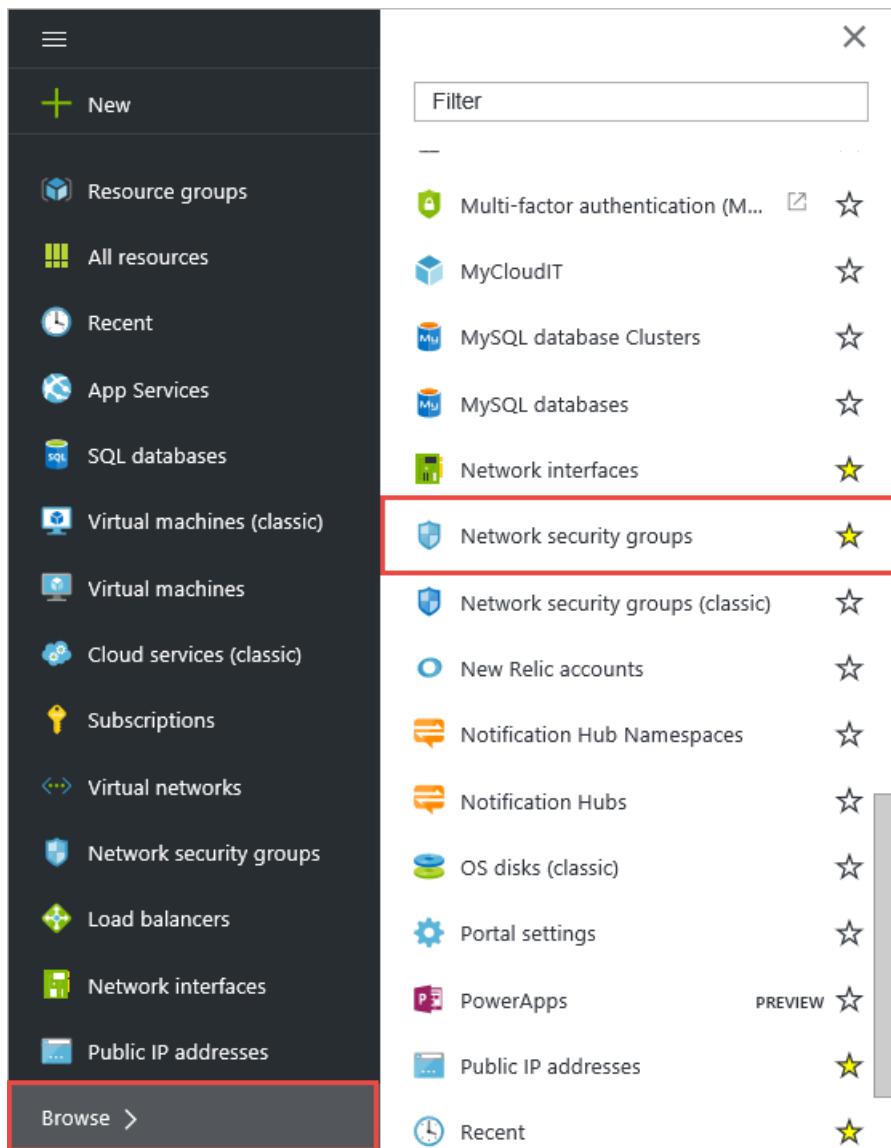
Retrieve Information

You can view your existing NSGs, retrieve rules for an existing NSG, and find out what resources an NSG is associated to.

View existing NSGs

To view all existing NSGs in a subscription, complete the following steps:

1. From a browser, navigate to <http://portal.azure.com> and, if necessary, sign in with your Azure account.
2. Click **Browse > > Network security groups**.



3. Check the list of NSGs in the **Network security groups** blade.

Network security groups			
Microsoft			
Add		Columns	Refresh
<input type="text"/> All subscriptions			▼
NAME			
NSG-Backend	RG-NSG	West US	Microsoft Azure Internal Consum... ...
NSG-FrontEnd	RG-NSG	West US	Microsoft Azure Internal Consum... ...

View NSGs in a resource group

To view the list of NSGs in the **RG-NSG** resource group, complete the following steps:

1. Click **Resource groups > > RG-NSG > ...**

The screenshot shows two windows side-by-side. The left window is the 'Resource groups' blade, with 'Resource groups' selected in the sidebar. It lists several resource groups, with 'RG-NSG' highlighted and selected. The right window is the 'RG-NSG' blade, which shows details about the resource group like subscription name, ID, last deployment, and location. Below this, the 'Resources' section lists various resources, including 'sqlAvSet', 'webAvSet', 'SQL1', 'SQL2', 'Web1', 'Web2', 'TestNICSQL1', 'TestNICSQL2', and 'TestNICWeb1'. The 'RG-NSG' item is also listed here.

2. In the list of resources, look for items displaying the NSG icon, as shown in the **Resources** blade below.

Resources

RG-NSG

Add Columns Refresh

NAME	RESOURCE GROUP	LOCATION	SUBSCRIPTION
sqlAvSet	RG-NSG	West US	Microsoft Azure Interna...
webAvSet	RG-NSG	West US	Microsoft Azure Interna...
SQL1	RG-NSG	West US	Microsoft Azure Interna...
SQL2	RG-NSG	West US	Microsoft Azure Interna...
Web1	RG-NSG	West US	Microsoft Azure Interna...
Web2	RG-NSG	West US	Microsoft Azure Interna...
TestNICSQL1	RG-NSG	West US	Microsoft Azure Interna...
TestNICSQL2	RG-NSG	West US	Microsoft Azure Interna...
TestNICWeb1	RG-NSG	West US	Microsoft Azure Interna...
TestNICWeb2	RG-NSG	West US	Microsoft Azure Interna...
NSG-Backend	RG-NSG	West US	Microsoft Azure Interna...
NSG-FrontEnd	RG-NSG	West US	Microsoft Azure Interna...
TestPIPSQL1	RG-NSG	West US	Microsoft Azure Interna...
TestPIPSQL2	RG-NSG	West US	Microsoft Azure Interna...

List all rules for an NSG

To view the rules of an NSG named **NSG-FrontEnd**, complete the following steps:

- From the **Network security groups** blade, or the **Resources** blade shown above, click **NSG-FrontEnd**.
- In the **Settings** tab, click **Inbound security rules**.

The screenshot shows two windows side-by-side. The left window is titled "NSG-FrontEnd" and "Network security group". It has "Settings" and "Delete" buttons at the top. Below is a table with "Essentials" and "All settings" buttons. The table includes columns for Resource group (RG-NSG), Security rules (0 inbound, 0 outbound), Location (West US), Associated with (1 subnets, 0 network interfaces), and Subscription name (Microsoft Azure Internal Consumption). The "All settings" button is highlighted with a blue box. The right window is titled "Settings" and "NSG-FrontEnd". It has a search bar and sections for SUPPORT & TROUBLESHOOTING (Audit logs), GENERAL (Properties, Inbound security rules, Outbound security rules), and a sidebar with a red box around the "Inbound security rules" link.

- The **Inbound security rules** blade is displayed as shown below.

PRIORITY	NAME	SOURCE	DESTINATION	SERVICE	ACTION
100	web-rule	Any	Any	TCP/80	Allow ...
101	rdp-rule	Any	Any	TCP/3389	Allow ...

- In the **Settings** tab, click **Outbound security rules** to see the outbound rules.

NOTE

To view default rules, click the **Default rules** icon at the top of the blade that displays the rules.

View NSGs associations

To view what resources the **NSG-FrontEnd** NSG is associate with, complete the following steps:

- From the **Network security groups** blade, or the **Resources** blade shown above, click **NSG-FrontEnd**.
- In the **Settings** tab, click **Subnets** to view what subnets are associated to the NSG.

NAME	ADDRESS RANGE	VIRTUAL NETWORK	...
FrontEnd	192.168.1.0/24	TestVNet	...

- In the **Settings** tab, click **Network interfaces** to view what NICs are associated to the NSG.

Manage rules

You can add rules to an existing NSG, edit existing rules, and remove rules.

Add a rule

To add a rule allowing **inbound** traffic to port **443** from any machine to the **NSG-FrontEnd** NSG, complete the following steps:

- From the **Network security groups** blade, or the **Resources** blade shown above, click **NSG-FrontEnd**.
- In the **Settings** tab, click **Inbound security rules**.
- In the **Inbound security rules** blade, click **Add**. Then, in the **Add inbound security rule** blade, fill the

values as shown below, and then click **OK**.

The screenshot shows two windows side-by-side. On the left is the 'Inbound security rules' blade for the 'NSG-FrontEnd' network security group. It lists two existing rules: 'web-rule' (Priority 100, Allow, TCP/80) and 'rdp-rule' (Priority 101, Allow, TCP/3389). A red box highlights the 'Add' button. On the right is the 'Add inbound security rule' dialog box. It has fields for 'Name' (set to 'allow-https'), 'Priority' (set to 201), 'Source' (set to 'Any'), 'Protocol' (set to 'TCP'), 'Source port range' (set to '*'), 'Destination' (set to 'Any'), 'Destination port range' (set to '443'), and 'Action' (set to 'Allow'). A red box highlights the 'OK' button at the bottom.

After a few seconds, notice the new rule in the **Inbound security rules** blade.

The screenshot shows the 'Inbound security rules' blade for 'NSG-FrontEnd'. It displays three rules: 'web-rule' (Priority 100, Allow, TCP/80), 'rdp-rule' (Priority 101, Allow, TCP/3389), and the newly created 'allow-https' rule (Priority 201, Allow, TCP/443). A red box highlights the 'allow-https' rule in the list.

Change a rule

To change the rule created above to allow inbound traffic from the **Internet** only, complete the following steps:

1. From the **Network security groups** blade, or the **Resources** blade shown above, click **NSG-FrontEnd**.
2. In the **Settings** tab, click the rule created above.
3. In the **allow-https** blade, change the **Source** property as shown below, and then click **Save**.

The screenshot shows two windows side-by-side. The left window is titled 'Inbound security rules' under 'NSG-FrontEnd'. It lists three rules: 'web-rule' (Priority 100, Any to Any, TCP/80, Allow), 'rdp-rule' (Priority 101, Any to Any, TCP/3389, Allow), and 'allow-https' (Priority 201, Any to Any, TCP/443, Allow). The 'allow-https' rule is highlighted. The right window is titled 'allow-https' and shows its configuration details. It includes fields for Name ('allow-https'), Priority (201), Source (Any, CIDR block, Tag 'Internet'), Protocol (TCP selected), Source port range (*), Destination (Any, CIDR block, Tag 'Internet'), Destination port range (443), and Action (Allow selected).

Delete a rule

To delete the rule created above, complete the following steps:

1. From the **Network security groups** blade, or the **Resources** blade shown above, click **NSG-FrontEnd**.
2. In the **Settings** tab, click the rule created above.
3. In the **allow-https** blade, click **Delete**, and then click **Yes**.

The screenshot shows two windows side-by-side. The left window is the same 'Inbound security rules' blade as before, with the 'allow-https' rule still highlighted. The right window is a confirmation dialog titled 'allow-https' with the message 'Delete security rule' and 'Do you want to delete the security rule "allow-https"?'. It has 'Yes' and 'No' buttons, with 'Yes' highlighted. Below the buttons are fields for Source (Any, CIDR block, Tag 'Internet'), Protocol (TCP selected), Source port range (*), Destination (Any, CIDR block, Tag 'Internet'), Destination port range (443), and Action (Allow selected).

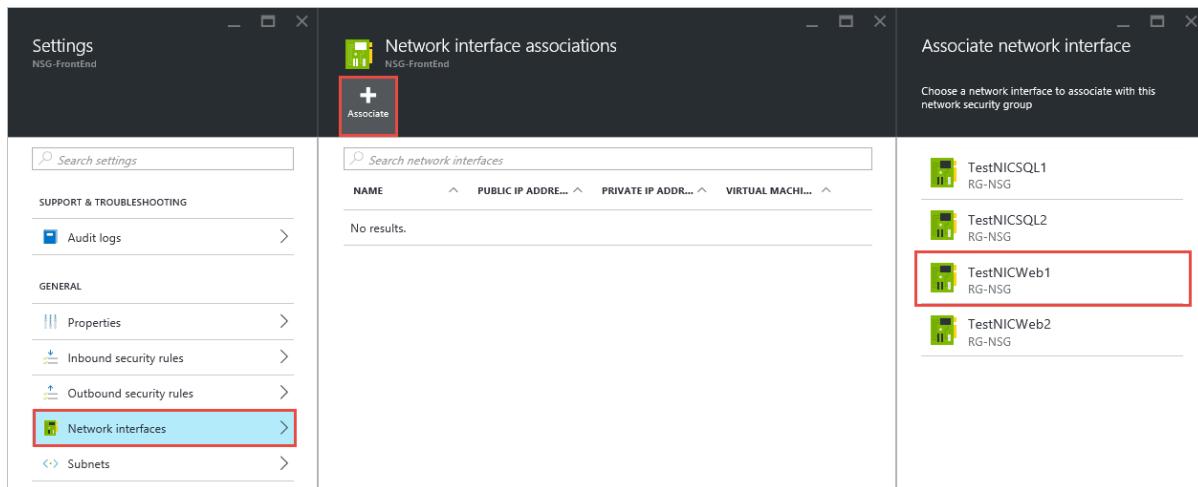
Manage associations

You can associate an NSG to subnets and NICs. You can also dissociate an NSG from any resource it's associated to.

Associate an NSG to a NIC

To associate the **NSG-FrontEnd** NSG to the **TestNICWeb1** NIC, complete the following steps:

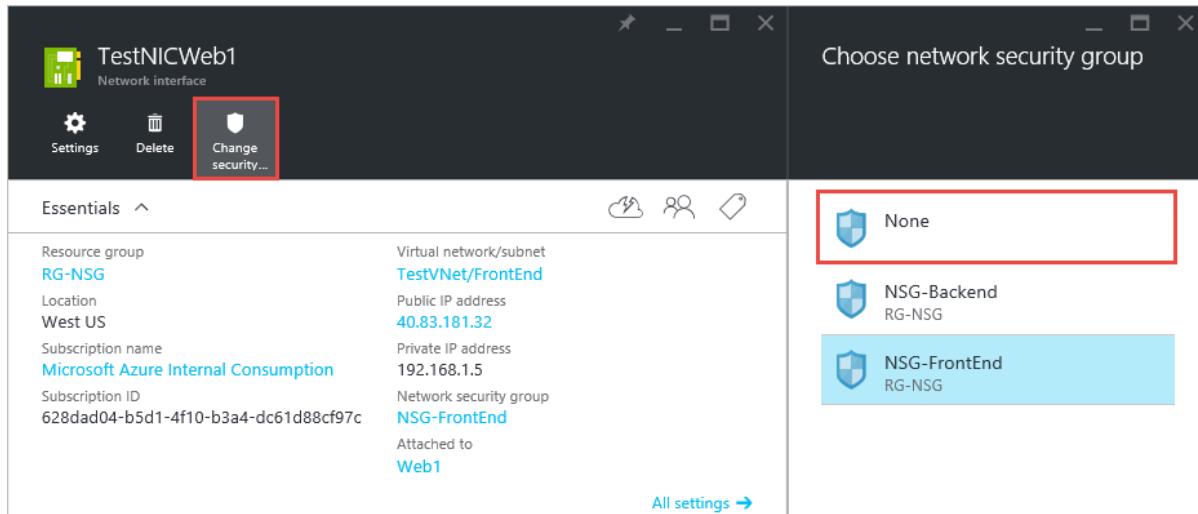
1. From the **Network security groups** blade, or the **Resources** blade shown above, click **NSG-FrontEnd**.
2. In the **Settings** tab, click **Network interfaces** > **Associate** > **TestNICWeb1**.



Dissociate an NSG from a NIC

To dissociate the **NSG-FrontEnd** NSG from the **TestNICWeb1** NIC, complete the following steps:

1. From the Azure portal, click **Resource groups** > > **RG-NSG** > ... > **TestNICWeb1**.
2. In the **TestNICWeb1** blade, click **Change security...** > **None**.



NOTE

You can also use this blade to associate the NIC to any existing NSG.

Dissociate an NSG from a subnet

To dissociate the **NSG-FrontEnd** NSG from the **FrontEnd** subnet, complete the following steps:

1. From the Azure portal, click **Resource groups** > > **RG-NSG** > ... > **TestVNet**.
2. In the **Settings** blade, click **Subnets** > **FrontEnd** > **Network security group** > **None**.

- In the **FrontEnd** blade, click **Save**.

Associate an NSG to a subnet

To associate the **NSG-FrontEnd** NSG to the **FrontEnd** subnet again, complete the following steps:

- From the Azure portal, click **Resource groups > > RG-NSG > ... > TestVNet**.
- In the **Settings** blade, click **Subnets > FrontEnd > Network security group > NSG-FrontEnd**.
- In the **FrontEnd** blade, click **Save**.

NOTE

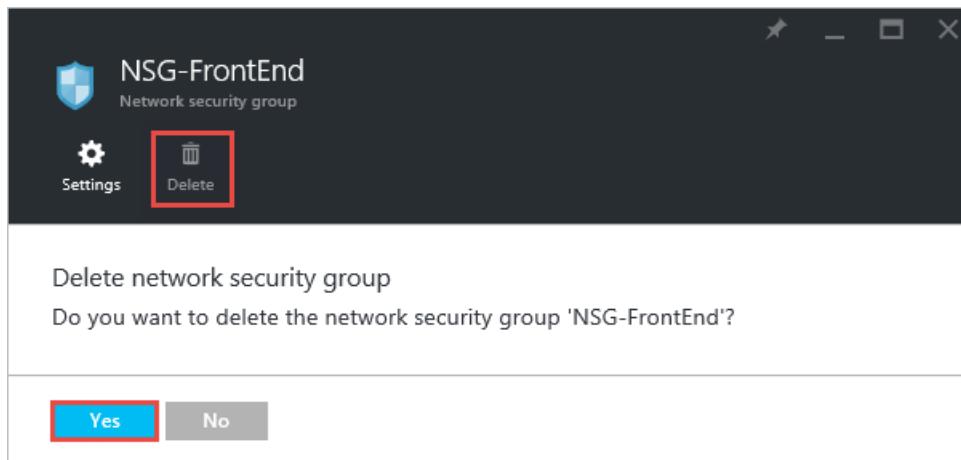
You can also associate an NSG to a subnet from the NSG's **Settings** blade.

Delete an NSG

You can only delete an NSG if it's not associated to any resource. To delete an NSG, complete the following steps:

- From the Azure portal, click **Resource groups > > RG-NSG > ... > NSG-FrontEnd**.
- In the **Settings** blade, click **Network interfaces**.
- If there are any NICs listed, click the NIC, and follow step 2 in [Dissociate an NSG from a NIC](#).
- Repeat step 3 for each NIC.

5. In the **Settings** blade, click **Subnets**.
6. If there are any subnets listed, click the subnet and follow steps 2 and 3 in [Dissociate an NSG from a subnet](#).
7. Scrolls left to the **NSG-FrontEnd** blade, then click **Delete** > **Yes**.



Next steps

- [Enable logging for NSGs](#).

Manage NSGs using PowerShell

1/17/2017 • 8 min to read • [Edit on GitHub](#)

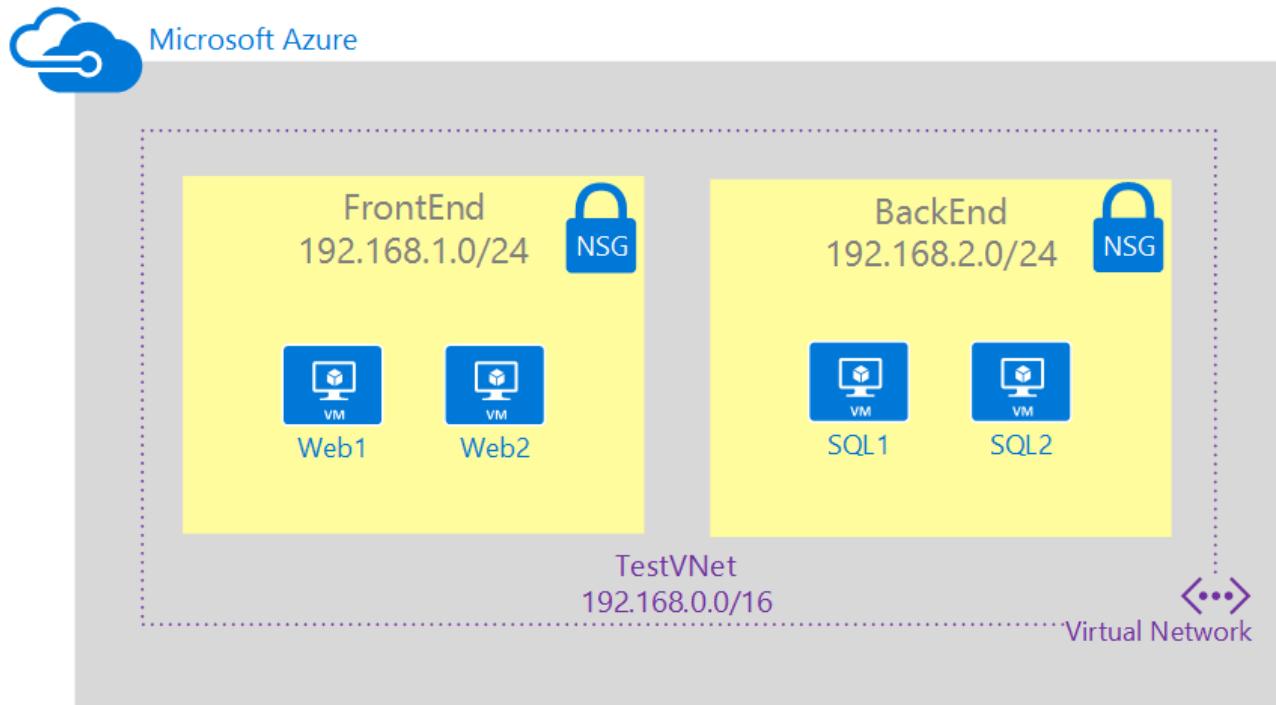
After you create one or more Network Security Groups (NSGs), you need to be able to retrieve information about your NSGs, add and remove rules, edit existing rules, associate or dissociate NSGs, and delete NSGs. In this article, you will learn how to execute each of these tasks. Before you can manage NSGs, it's important to know [how NSGs work](#).

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.

Sample Scenario

To better illustrate how to manage NSGs, this article uses the scenario below.



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

- **NSG-FrontEnd**. The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule**. This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule**. This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd**. The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule**. This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule**. This 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 traffic 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.

To deploy the scenario described above, follow [this link](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal. In the sample instructions below, the template was used to deploy a resource group names **RG-NSG**.

Prerequisite: Install the Azure PowerShell Module

To perform the steps in this article, you'll need to [to install and configure Azure PowerShell](#) and follow the instructions all the way to the end to sign into Azure and select your subscription.

NOTE

If you don't have an Azure account, you'll need one. Go sign up for a [free trial here](#).

Retrieve Information

You can view your existing NSGs, retrieve rules for an existing NSG, and find out what resources an NSG is associated to.

View existing NSGs

To view all existing NSGs in a subscription, run the `Get-AzureRmNetworkSecurityGroup` cmdlet.

Expected result:

```

Name : NSG-BackEnd
ResourceGroupName : RG-NSG
Location : westus
Id : /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/
      Microsoft.Network/networkSecurityGroups/NSG-BackEnd
Etag : W/"[Id]"
ResourceGuid : [Id]
ProvisioningState : Succeeded
Tags :
SecurityRules : [...]
DefaultSecurityRules : [...]
NetworkInterfaces : [...]
Subnets : [...]

Name : NSG-FrontEnd
ResourceGroupName : RG-NSG
Location : eastus
Id : /subscriptions/[Subscription Id]/resourceGroups/NRP-RG/providers/
      Microsoft.Network/networkSecurityGroups/NSG-FrontEnd
Etag : W/"[Id]"
ResourceGuid : [Id]
ProvisioningState : Succeeded
Tags :
SecurityRules : [...]
DefaultSecurityRules : [...]
NetworkInterfaces : [...]
Subnets : [...]

Name : WEB1
ResourceGroupName : RG101
Location : eastus2
Id : /subscriptions/[Subscription Id]/resourceGroups/RG101/providers/M
      icrosoft.Network/networkSecurityGroups/WEB1
Etag : W/"[Id]"
ResourceGuid : [Id]
ProvisioningState : Succeeded
Tags :
SecurityRules : [...]
DefaultSecurityRules : [...]
NetworkInterfaces : [...]
Subnets : [...]

```

To view the list of NSGs in a specific resource group, run the `Get-AzureRmNetworkSecurityGroup` cmdlet.

Expected output:

```

Name : NSG-BackEnd
ResourceGroupName : RG-NSG
Location : westus
Id : /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/
      Microsoft.Network/networkSecurityGroups/NSG-BackEnd
Etag : W/"[Id]"
ResourceGuid : [Id]
ProvisioningState : Succeeded
Tags :
SecurityRules : [...]
DefaultSecurityRules : [...]
NetworkInterfaces : [...]
Subnets : [...]

Name : NSG-FrontEnd
ResourceGroupName : RG-NSG
Location : eastus
Id : /subscriptions/[Subscription Id]/resourceGroups/NRP-RG/providers/
      Microsoft.Network/networkSecurityGroups/NSG-FrontEnd
Etag : W/"[Id]"
ResourceGuid : [Id]
ProvisioningState : Succeeded
Tags :
SecurityRules : [...]
DefaultSecurityRules : [...]
NetworkInterfaces : [...]
Subnets : [...]

```

List all rules for an NSG

To view the rules of an NSG named **NSG-FrontEnd**, enter the following command:

```
Get-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name NSG-FrontEnd | Select SecurityRules -ExpandProperty SecurityRules
```

Expected output:

```

Name          : rdp-rule
Id           : /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/
Microsoft.Network/networkSecurityGroups/NSG-FrontEnd/securityRules/rdp-rule
Etag         : W/"[Id]"
ProvisioningState : Succeeded
Description    : Allow RDP
Protocol       : Tcp
SourcePortRange: *
DestinationPortRange: 3389
SourceAddressPrefix: Internet
DestinationAddressPrefix: *
Access         : Allow
Priority       : 100
Direction      : Inbound

Name          : web-rule
Id           : /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/
Microsoft.Network/networkSecurityGroups/NSG-FrontEnd/securityRules/web-rule
Etag         : W/"[Id]"
ProvisioningState : Succeeded
Description    : Allow HTTP
Protocol       : Tcp
SourcePortRange: *
DestinationPortRange: 80
SourceAddressPrefix: Internet
DestinationAddressPrefix: *
Access         : Allow
Priority       : 101
Direction      : Inbound

```

NOTE

You can also use

```
Get-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name "NSG-FrontEnd" | Select
DefaultSecurityRules -ExpandProperty DefaultSecurityRules
```

to list the default rules from the **NSG-FrontEnd** NSG.

View NSGs associations

To view what resources the **NSG-FrontEnd** NSG is associate with, run the following command:

```
Get-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name NSG-FrontEnd
```

Look for the **NetworkInterfaces** and **Subnets** properties as shown below:

```

NetworkInterfaces   : []
Subnets            : [
    {
        "Id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/RG-
NSG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd",
        "IpConfigurations": []
    }
]

```

In the previous example, the NSG is not associated to any network interfaces (NICs); it is associated to a subnet named **FrontEnd**.

Manage rules

You can add rules to an existing NSG, edit existing rules, and remove rules.

Add a rule

To add a rule allowing **inbound** traffic to port **443** from any machine to the **NSG-FrontEnd** NSG, complete the following steps:

1. Run the following command to retrieve the existing NSG and store it in a variable:

```
$nsg = Get-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name NSG-FrontEnd
```

2. Run the following command to add a rule to the NSG:

```
Add-AzureRmNetworkSecurityRuleConfig -NetworkSecurityGroup $nsg `  
-Name https-rule `  
-Description "Allow HTTPS" `  
-Access Allow `  
-Protocol Tcp `  
-Direction Inbound `  
-Priority 102 `  
-SourceAddressPrefix * `  
-SourcePortRange * `  
-DestinationAddressPrefix * `  
-DestinationPortRange 443
```

3. To save the changes made to the NSG, run the following command:

```
Set-AzureRmNetworkSecurityGroup -NetworkSecurityGroup $nsg
```

Expected output showing only the security rules:

```
Name          : NSG-FrontEnd  
...  
SecurityRules : [  
    {  
        "Name": "rdp-rule",  
        ...  
    },  
    {  
        "Name": "web-rule",  
        ...  
    },  
    {  
        "Name": "https-rule",  
        "Etag": "W/[Id]",  
        "Id": "/subscriptions/[Subscription Id]/resourceGroups/RG-  
NSG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd/securityRules/https-rule",  
        "Description": "Allow HTTPS",  
        "Protocol": "Tcp",  
        "SourcePortRange": "*",  
        "DestinationPortRange": "443",  
        "SourceAddressPrefix": "*",  
        "DestinationAddressPrefix": "*",  
        "Access": "Allow",  
        "Priority": 102,  
        "Direction": "Inbound",  
        "ProvisioningState": "Succeeded"  
    }  
]
```

Change a rule

To change the rule created above to allow inbound traffic from the **Internet** only, follow the steps below.

1. Run the following command to retrieve the existing NSG and store it in a variable:

```
$nsg = Get-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name NSG-FrontEnd
```

- Run the following command with the new rule settings:

```
Set-AzureRmNetworkSecurityRuleConfig -NetworkSecurityGroup $nsg ` 
-Name https-rule ` 
-Description "Allow HTTPS" ` 
-Access Allow ` 
-Protocol Tcp ` 
-Direction Inbound ` 
-Priority 102 ` 
-SourceAddressPrefix * ` 
-SourcePortRange Internet ` 
-DestinationAddressPrefix * ` 
-DestinationPortRange 443
```

- To save the changes made to the NSG, run the following command:

```
Set-AzureRmNetworkSecurityGroup -NetworkSecurityGroup $nsg
```

Expected output showing only the security rules:

```
Name          : NSG-FrontEnd
...
SecurityRules : [
    {
        "Name": "rdp-rule",
        ...
    },
    {
        "Name": "web-rule",
        ...
    },
    {
        "Name": "https-rule",
        "Etag": "W/\"[Id]\"",
        "Id": "/subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd/securityRules/https-rule",
        "Description": "Allow HTTPS",
        "Protocol": "Tcp",
        "SourcePortRange": "*",
        "DestinationPortRange": "443",
        "SourceAddressPrefix": "Internet",
        "DestinationAddressPrefix": "*",
        "Access": "Allow",
        "Priority": 102,
        "Direction": "Inbound",
        "ProvisioningState": "Succeeded"
    }
]
```

Delete a rule

- Run the following command to retrieve the existing NSG and store it in a variable:

```
$nsg = Get-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name NSG-FrontEnd
```

- Run the following command to remove the rule from the NSG:

```
Remove-AzureRmNetworkSecurityRuleConfig -NetworkSecurityGroup $nsg -Name https-rule
```

- Save the changes made to the NSG, by running the following command:

```
Set-AzureRmNetworkSecurityGroup -NetworkSecurityGroup $nsg
```

Expected output showing only the security rules, notice the **https-rule** is no longer listed:

```
Name          : NSG-FrontEnd
...
SecurityRules : [
    {
        "Name": "rdp-rule",
        ...
    },
    {
        "Name": "web-rule",
        ...
    }
]
```

Manage associations

You can associate an NSG to subnets and NICs. You can also dissociate an NSG from any resource it's associated to.

Associate an NSG to a NIC

To associate the **NSG-FrontEnd** NSG to the **TestNICWeb1** NIC, complete the following steps:

- Run the following command to retrieve the existing NSG and store it in a variable:

```
$nsg = Get-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name NSG-FrontEnd
```

- Run the following command to retrieve the existing NIC and store it in a variable:

```
$nic = Get-AzureRmNetworkInterface -ResourceGroupName RG-NSG -Name TestNICWeb1
```

- Set the **NetworkSecurityGroup** property of the **NIC** variable to the value of the **NSG** variable, by entering the following command:

```
$nic.NetworkSecurityGroup = $nsg
```

- To save the changes made to the NIC, run the following command:

```
Set-AzureRmNetworkInterface -NetworkInterface $nic
```

Expected output showing only the **NetworkSecurityGroup** property:

```
NetworkSecurityGroup : {
    "SecurityRules": [],
    "DefaultSecurityRules": [],
    "NetworkInterfaces": [],
    "Subnets": [],
    "Id": "/subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd"
}
```

Dissociate an NSG from a NIC

To dissociate the **NSG-FrontEnd** NSG from the **TestNICWeb1** NIC, complete the following steps:

1. Run the following command to retrieve the existing NIC and store it in a variable:

```
$nic = Get-AzureRmNetworkInterface -ResourceGroupName RG-NSG -Name TestNICWeb1
```

2. Set the **NetworkSecurityGroup** property of the **NIC** variable to **\$null** by running the following command:

```
$nic.NetworkSecurityGroup = $null
```

3. To save the changes made to the NIC, run the following command:

```
Set-AzureRmNetworkInterface -NetworkInterface $nic
```

Expected output showing only the **NetworkSecurityGroup** property:

```
NetworkSecurityGroup : null
```

Dissociate an NSG from a subnet

To dissociate the **NSG-FrontEnd** NSG from the **FrontEnd** subnet, complete the following steps:

1. Run the following command to retrieve the existing VNet and store it in a variable:

```
$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName RG-NSG -Name TestVNet
```

2. Run the following command to retrieve the **FrontEnd** subnet and store it in a variable:

```
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet -Name FrontEnd
```

3. Set the **NetworkSecurityGroup** property of the **subnet** variable to **\$null** by entering the following command:

```
$subnet.NetworkSecurityGroup = $null
```

4. To save the changes made to the subnet, run the following command:

```
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

Expected output showing only the properties of the **FrontEnd** subnet. Notice there isn't a property for **NetworkSecurityGroup**:

```

...
Subnets      : [
    {
        "Name": "FrontEnd",
        "Etag": "W/[Id]",
        "Id": "/subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd",
        "AddressPrefix": "192.168.1.0/24",
        "IpConfigurations": [
            {
                "Id": "/subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/networkInterfaces/TestNICWeb2/ipConfigurations/ipconfig1"
            },
            {
                "Id": "/subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/networkInterfaces/TestNICWeb1/ipConfigurations/ipconfig1"
            }
        ],
        "ProvisioningState": "Succeeded"
    },
    ...
]

```

Associate an NSG to a subnet

To associate the **NSG-FrontEnd** NSG to the **FrontEnd** subnet again, complete the following steps:

1. Run the following command to retrieve the existing VNet and store it in a variable:

```
$vnet = Get-AzureRmVirtualNetwork -ResourceGroupName RG-NSG -Name TestVNet
```

2. Run the following command to retrieve the **FrontEnd** subnet and store it in a variable:

```
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -VirtualNetwork $vnet -Name FrontEnd
```

3. Run the following command to retrieve the existing NSG and store it in a variable:

```
$nsg = Get-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name NSG-FrontEnd
```

4. Set the **NetworkSecurityGroup** property of the **subnet** variable to **\$null** by running the following command:

```
$subnet.NetworkSecurityGroup = $nsg
```

5. To save the changes made to the subnet, run the following command:

```
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

Expected output showing only the **NetworkSecurityGroup** property of the **FrontEnd** subnet:

```
...
"NetworkSecurityGroup": {
    "SecurityRules": [],
    "DefaultSecurityRules": [],
    "NetworkInterfaces": [],
    "Subnets": [],
    "Id": "/subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd"
}
...
```

Delete an NSG

You can only delete an NSG if it's not associated to any resource. To delete an NSG, follow the steps below.

1. To check the resources associated to an NSG, run the `azure network nsg show` as shown in [View NSGs associations](#).
2. If the NSG is associated to any NICs, run the `azure network nic set` as shown in [Dissociate an NSG from a NIC](#) for each NIC.
3. If the NSG is associated to any subnet, run the `azure network vnet subnet set` as shown in [Dissociate an NSG from a subnet](#) for each subnet.
4. To delete the NSG, run the following command:

```
Remove-AzureRmNetworkSecurityGroup -ResourceGroupName RG-NSG -Name NSG-FrontEnd -Force
```

NOTE

The `-Force` parameter ensures you don't need to confirm the deletion.

Next steps

- [Enable logging for NSGs](#).

Manage NSGs using the Azure CLI

1/17/2017 • 9 min to read • [Edit on GitHub](#)

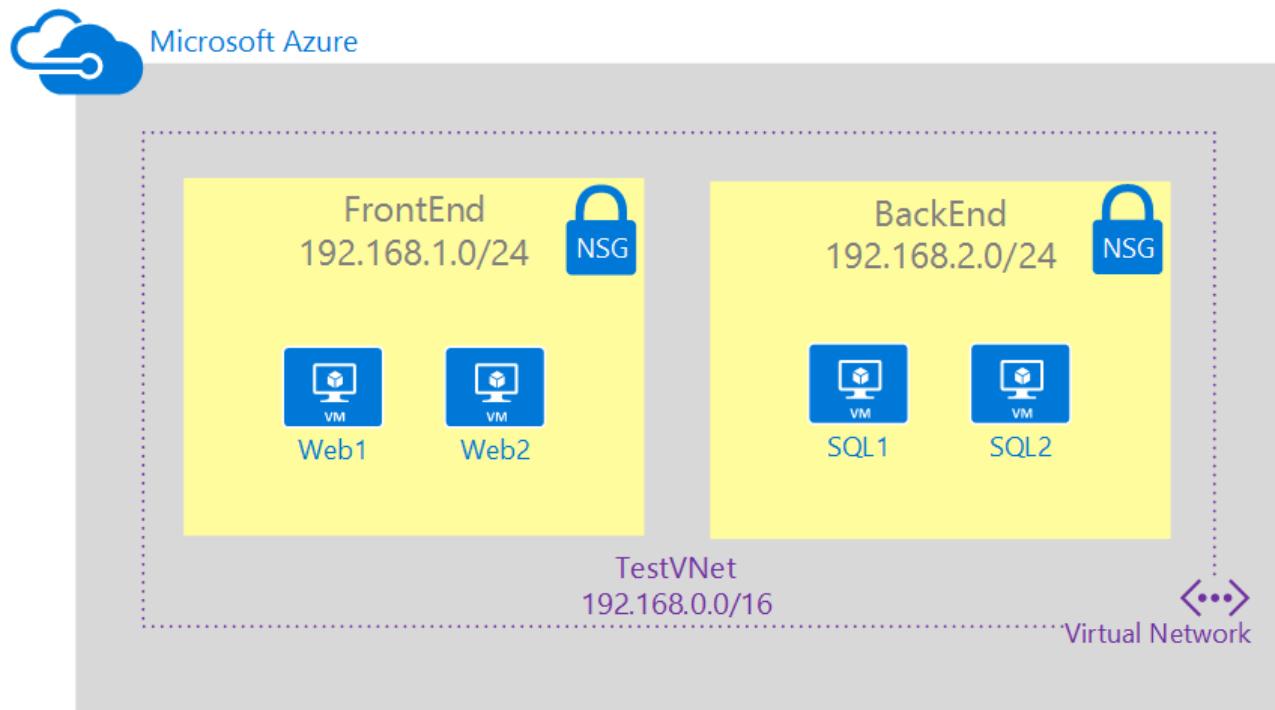
After you create one or more Network Security Groups (NSGs), you need to be able to retrieve information about your NSGs, add and remove rules, edit existing rules, associate or dissociate NSGs, and delete NSGs. In this article, you will learn how to execute each of these tasks. Before you can manage NSGs, it's important to know [how NSGs work](#).

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.

Sample Scenario

To better illustrate how to manage NSGs, this article uses the scenario below.



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

- **NSG-FrontEnd.** The front end NSG will be applied to the *FrontEnd* subnet, and contain two rules:
 - **rdp-rule.** This rule will allow RDP traffic to the *FrontEnd* subnet.
 - **web-rule.** This rule will allow HTTP traffic to the *FrontEnd* subnet.
- **NSG-BackEnd.** The back end NSG will be applied to the *BackEnd* subnet, and contain two rules:
 - **sql-rule.** This rule allows SQL traffic only from the *FrontEnd* subnet.
 - **web-rule.** This 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 traffic 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.

To deploy the scenario described above, follow [this link](#), click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal. In the sample instructions below, the template was used to deploy a resource group names **RG-NSG**.

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.

Retrieve Information

You can view your existing NSGs, retrieve rules for an existing NSG, and find out what resources an NSG is associated to.

View existing NSGs

To view the list of NSGs in a specific resource group, run the `azure network nsg list` command as shown below.

```
azure network nsg list --resource-group RG-NSG
```

Expected output:

```
info: Executing command network nsg list
+ Getting the network security groups
data: Name          Location
data: -----
data: NSG-BackEnd   westus
data: NSG-FrontEnd  westus
info: network nsg list command OK
```

List all rules for an NSG

To view the rules of an NSG named **NSG-FrontEnd**, run the `azure network nsg show` command as shown below.

```
azure network nsg show --resource-group RG-NSG --name NSG-FrontEnd
```

Expected output:

```

info: Executing command network nsg show
+ Looking up the network security group "NSG-FrontEnd"
data: Id : /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd
data: Name : NSG-FrontEnd
data: Type : Microsoft.Network/networkSecurityGroups
data: Location : westus
data: Provisioning state : Succeeded
data: Tags : displayName=NSG - Front End
data: Security group rules:
data: Name Source IP Source Port Destination IP Destination Port
Protocol Direction Access Priority
data: -----
-- -----
data: rdp-rule Internet * * 3389 Tcp
Inbound Allow 100
data: web-rule Internet * * 80 Tcp
Inbound Allow 101
data: AllowVnetInBound VirtualNetwork * VirtualNetwork * *
Inbound Allow 65000
data: AllowAzureLoadBalancerInBound AzureLoadBalancer * * * *
Inbound Allow 65001
data: DenyAllInBound * * * * *
Inbound Deny 65500
data: AllowVnetOutBound VirtualNetwork * VirtualNetwork * *
Outbound Allow 65000
data: AllowInternetOutBound * * Internet * *
Outbound Allow 65001
data: DenyAllOutBound * * * * *
Outbound Deny 65500
info: network nsg show command OK

```

NOTE

You can also use `azure network nsg rule list --resource-group RG-NSG --nsg-name NSG-FrontEnd` to list the rules from the **NSG-FrontEnd** NSG.

View NSG associations

To view what resources the **NSG-FrontEnd** NSG is associate with, run the `azure network nsg show` command as shown below. Notice that the only difference is the use of the **--json** parameter.

```
azure network nsg show --resource-group RG-NSG --name NSG-FrontEnd --json
```

Look for the **networkInterfaces** and **subnets** properties as shown below:

```

"networkInterfaces": [],
...
"subnets": [
  {
    "id": "/subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd"
  }
],
...

```

In the example above, the NSG is not associated to any network interfaces (NICs), and it is associated to a subnet named **FrontEnd**.

Manage rules

You can add rules to an existing NSG, edit existing rules, and remove rules.

Add a rule

To add a rule allowing **inbound** traffic to port **443** from any machine to the **NSG-FrontEnd** NSG, enter the following command:

```
azure network nsg rule create --resource-group RG-NSG \
--nsg-name NSG-FrontEnd \
--name allow-https \
--description "Allow access to port 443 for HTTPS" \
--protocol Tcp \
--source-address-prefix * \
--source-port-range * \
--destination-address-prefix * \
--destination-port-range 443 \
--access Allow \
--priority 102 \
--direction Inbound
```

Expected output:

```
info: Executing command network nsg rule create
+ Looking up the network security rule "allow-https"
+ Creating a network security rule "allow-https"
+ Looking up the network security group "NSG-FrontEnd"
data: Id : /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd/securityRules/allow-https
data: Name : allow-https
data: Type : Microsoft.Network/networkSecurityGroups/securityRules
data: Provisioning state : Succeeded
data: Description : Allow access to port 443 for HTTPS
data: Source IP : *
data: Source Port : *
data: Destination IP : *
data: Destination Port : 443
data: Protocol : Tcp
data: Direction : Inbound
data: Access : Allow
data: Priority : 102
info: network nsg rule create command OK
```

Change a rule

To change the rule created above to allow inbound traffic from the **Internet** only, run the following command:

```
azure network nsg rule set --resource-group RG-NSG \
--nsg-name NSG-FrontEnd \
--name allow-https \
--source-address-prefix Internet
```

Expected output:

```
info: Executing command network nsg rule set
+ Looking up the network security group "NSG-FrontEnd"
+ Setting a network security rule "allow-https"
+ Looking up the network security group "NSG-FrontEnd"
data: Id : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxx/resourceGroups/RG-NSG/providers/Microsoft.Network/networkSecurityGroups/NSG-
FrontEnd/securityRules/allow-https
data: Name : allow-https
data: Type : Microsoft.Network/networkSecurityGroups/securityRules
data: Provisioning state : Succeeded
data: Description : Allow access to port 443 for HTTPS
data: Source IP : Internet
data: Source Port : *
data: Destination IP : *
data: Destination Port : 443
data: Protocol : Tcp
data: Direction : Inbound
data: Access : Allow
data: Priority : 102
info: network nsg rule set command OK
```

Delete a rule

To delete the rule created above, run the following command:

```
azure network nsg rule delete --resource-group RG-NSG \
--nsg-name NSG-FrontEnd \
--name allow-https \
--quiet
```

NOTE

The `--quiet` parameter ensures you don't need to confirm the deletion.

Expected output:

```
info: Executing command network nsg rule delete
+ Looking up the network security group "NSG-FrontEnd"
+ Deleting network security rule "allow-https"
info: network nsg rule delete command OK
```

Manage associations

You can associate an NSG to subnets and NICs. You can also dissociate an NSG from any resource it's associated to.

Associate an NSG to a NIC

To associate the **NSG-FrontEnd** NSG to the **TestNICWeb1** NIC, run the following command:

```
azure network nic set --resource-group RG-NSG \
--name TestNICWeb1 \
--network-security-group-name NSG-FrontEnd
```

Expected output:

```
info: Executing command network nic set
+ Looking up the network interface "TestNICWeb1"
+ Looking up the network security group "NSG-FrontEnd"
+ Updating network interface "TestNICWeb1"
+ Looking up the network interface "TestNICWeb1"
data: Id : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/RG-NSG/providers/Microsoft.Network/networkInterfaces/TestNICWeb1
data: Name : TestNICWeb1
data: Type : Microsoft.Network/networkInterfaces
data: Location : westus
data: Provisioning state : Succeeded
data: MAC address : 00-0D-3A-30-A1-F8
data: Enable IP forwarding : false
data: Tags : displayName=NetworkInterfaces - Web
data: Network security group : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/RG-NSG/providers/Microsoft.Network/networkSecurityGroups/NSG-FrontEnd
data: Virtual machine : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/RG-NSG/providers/Microsoft.Compute/virtualMachines/Web1
data: IP configurations:
data: Name : ipconfig1
data: Provisioning state : Succeeded
data: Public IP address : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/RG-NSG/providers/Microsoft.Network/publicIPAddresses/TestPIPWeb1
data: Private IP address : 192.168.1.5
data: Private IP Allocation Method : Dynamic
data: Subnet : /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx/resourceGroups/RG-NSG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data:
info: network nic set command OK
```

Dissociate an NSG from a NIC

To dissociate the **NSG-FrontEnd** NSG from the **TestNICWeb1** NIC, run the following command:

```
azure network nic set --resource-group RG-NSG --name TestNICWeb1 --network-security-group-id ""
```

NOTE

Notice the "" (empty) value for the `network-security-group-id` parameter. That is how you remove an association to an NSG. You can't do the same with the `network-security-group-name` parameter.

Expected result:

```

info:  Executing command network nic set
+ Looking up the network interface "TestNICWeb1"
+ Updating network interface "TestNICWeb1"
+ Looking up the network interface "TestNICWeb1"
data:  Id                      : /subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/networkInterfaces/TestNICWeb1
data:  Name                     : TestNICWeb1
data:  Type                      : Microsoft.Network/networkInterfaces
data:  Location                  : westus
data:  Provisioning state       : Succeeded
data:  MAC address               : 00-0D-3A-30-A1-F8
data:  Enable IP forwarding     : false
data:  Tags                      : displayName=NetworkInterfaces - Web
data:  Virtual machine           : /subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Compute/virtualMachines/Web1
data:  IP configurations:
data:    Name                    : ipconfig1
data:    Provisioning state     : Succeeded
data:    Public IP address       : /subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/publicIPAddresses/TestPIPWeb1
data:    Private IP address      : 192.168.1.5
data:    Private IP Allocation Method : Dynamic
data:    Subnet                   : /subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data:
info:  network nic set command OK

```

Dissociate an NSG from a subnet

To dissociate the **NSG-FrontEnd** NSG from the **FrontEnd** subnet, run the following command:

```

azure network vnet subnet set --resource-group RG-NSG \
  --vnet-name TestVNet \
  --name FrontEnd \
  --network-security-group-id ""

```

Expected output:

```

info:  Executing command network vnet subnet set
+ Looking up the subnet "FrontEnd"
+ Setting subnet "FrontEnd"
+ Looking up the subnet "FrontEnd"
data:  Id                      : /subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data:  Type                     : Microsoft.Network/virtualNetworks/subnets
data:  ProvisioningState        : Succeeded
data:  Name                      : FrontEnd
data:  Address prefix            : 192.168.1.0/24
data:  IP configurations:
data:    /subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/networkInterfaces/TestNICWeb2/ipConfigurations/ipconfig1
data:    /subscriptions/[Subscription Id]/resourceGroups/RG-
NSG/providers/Microsoft.Network/networkInterfaces/TestNICWeb1/ipConfigurations/ipconfig1
data:
info:  network vnet subnet set command OK

```

Associate an NSG to a subnet

To associate the **NSG-FrontEnd** NSG to the **FrontEnd** subnet again, run the following command:

```

azure network vnet subnet set --resource-group RG-NSG \
  --vnet-name TestVNet \
  --name FrontEnd \
  --network-security-group-name NSG-FronEnd

```

NOTE

The command above only works because the **NSG-FrontEnd** NSG is in the same resource group as the virtual network **TestVNet**. If the NSG is in a different resource group, you need to use the `--network-security-group-id` parameter instead, and provide the full id for the NSG. You can retrieve the id by running

```
azure network nsg show --resource-group RG-NSG --name NSG-FrontEnd --json
```

 and looking for the **id** property.

Expected output:

```
info: Executing command network vnet subnet set
+ Looking up the subnet "FrontEnd"
+ Looking up the network security group "NSG-FrontEnd"
+ Setting subnet "FrontEnd"
+ Looking up the subnet "FrontEnd"
data: Id : /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd
data: Type : Microsoft.Network/virtualNetworks/subnets
data: ProvisioningState : Succeeded
data: Name : FrontEnd
data: Address prefix : 192.168.1.0/24
data: Network security group : [object Object]
data: IP configurations:
data:   /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/networkInterfaces/TestNICWeb2/ipConfigurations/ipconfig1
data:   /subscriptions/[Subscription Id]/resourceGroups/RG-NSG/providers/Microsoft.Network/networkInterfaces/TestNICWeb1/ipConfigurations/ipconfig1
data:
info: network vnet subnet set command OK
```

Delete an NSG

You can only delete an NSG if it's not associated to any resource. To delete an NSG, follow the steps below.

1. To check the resources associated to an NSG, run the `azure network nsg show` as shown in [View NSGs associations](#).
2. If the NSG is associated to any NICs, run the `azure network nic set` as shown in [Dissociate an NSG from a NIC](#) for each NIC.
3. If the NSG is associated to any subnet, run the `azure network vnet subnet set` as shown in [Dissociate an NSG from a subnet](#) for each subnet.
4. To delete the NSG, run the following command:

```
azure network nsg delete --resource-group RG-NSG --name NSG-FrontEnd --quiet
```

Expected output:

```
info: Executing command network nsg delete
+ Looking up the network security group "NSG-FrontEnd"
+ Deleting network security group "NSG-FrontEnd"
info: network nsg delete command OK
```

Next steps

- [Enable logging for NSGs](#).

Log analytics for network security groups (NSGs)

1/17/2017 • 3 min to read • [Edit on GitHub](#)

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, and 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 **More 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 Networking Analytics](#) (preview) solution for enhanced insights.

Troubleshoot Network Security Groups using the Azure Portal

1/17/2017 • 8 min to read • [Edit on GitHub](#)

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.

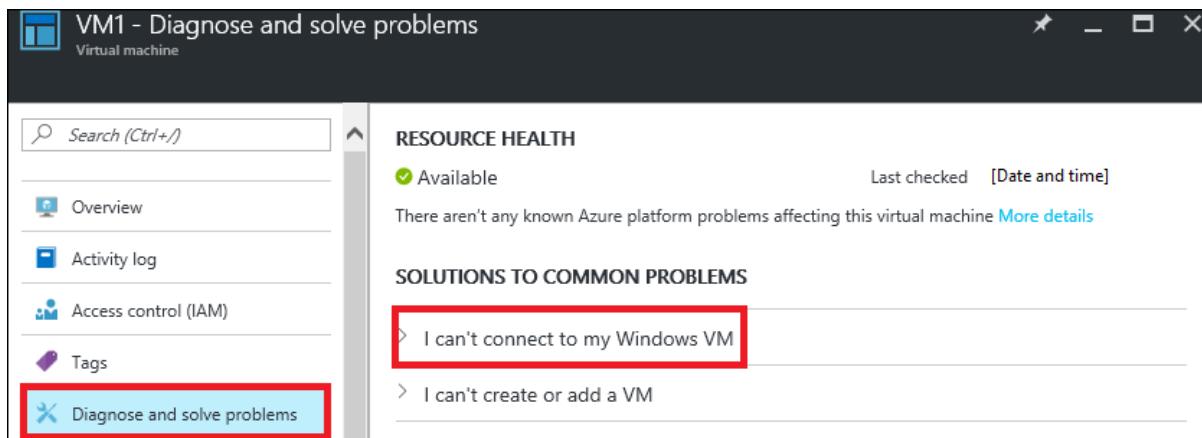
While this example uses TCP port 3389, the following steps can be used to determine inbound and outbound connection failures over any port.

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>.
2. Click **More 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:

The screenshot shows the 'Diagnose and solve problems' blade for VM1. It displays the 'RESOURCE HEALTH' section with a green checkmark and the 'SOLUTIONS TO COMMON PROBLEMS' section. Under 'I can't connect to my Windows VM', there's a 'Recommended steps' section. One of the steps, 'To connect to your VM via RDP, please review [effective security group rules](#) to ensure inbound "Allow" NSG rule exists for RDP port(3389)', is highlighted with a red box.

Diagnose and solve problems
vm1

RESOURCE HEALTH
Available Last checked [Date and time]
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

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

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:

Effective security rules

Showing only top 50 security rules in each grid, click Download above to see all.

Select a network interface below to see the effective security rules and network security groups associated with it.

Scope: Virtual machine (VM1)

Network interface: VM1-NIC1

Associated NSGs: VM1-nsg (Network interface), Subnet1-NSG (Subnet)

Click on a rule row to see the expanded list of prefixes.

VM1-nsg Subnet1-NSG

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

Address prefixes

AllowVnetInBound

Source	Destination
10.9.0.0/16	
168.63.129.16/32	
10.0.0.0/16	
10.1.0.0/16	

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:

VM1-nsg Subnet1-NSG

Inbound rules

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
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

Address prefixes

AllowVnetInBound

Source	Destination
10.9.0.0/16	
168.63.129.16/32	
10.0.0.0/16	
10.1.0.0/16	

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.

- 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**.

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

- After removing the *denyRDP* inbound rule in the **Subnet1-NSG** and adding an *allowRDP* rule, the effective rules list looks like the following picture:

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 **More 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 'Effective security rules' blade for a Network Interface (VM1-NIC1). The blade has a left sidebar with various options like Overview, Activity log, Access control (IAM), Tags, IP configurations, DNS servers, Network security group, Effective security rules (which is selected and highlighted in blue), Effective routes, Properties, Locks, and Automation script. Below the sidebar, there's a 'SUPPORT + TROUBLESHOOTING' section with a 'New support request' link. The main content area has a search bar and a note: 'Showing only top 50 security rules in each grid, click Download above to see all.' It also shows the 'Scope' is set to 'Network interface (VM1-NIC1)' and 'Associated NSGs' are 'VM1-nsg (Network interface), Subnet1-NSG (Subnet)'. There are two tabs: 'VM1-nsg' (selected) and 'Subnet1-NSG'. Under 'VM1-nsg', there are two sections: 'Inbound rules' and 'Outbound rules', each with a table of security rules. The 'Inbound rules' table has columns: NAME, PRIORITY, SOURCE, SOURCE PORTS, DESTINATION, DESTINATION PORTS, PROTOCOL, and ACCESS. The 'Outbound rules' table has similar columns.

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

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 **More 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 left sidebar has a tree view with 'Effective security rules' selected. The main area shows the scope as 'Network security group (VM1-nsg)', 'Virtual machine' as 'VM1', and 'Network interface' as 'VM1-NIC1'. Under 'Associated NSGs', it lists 'VM1-nsg (Network interface)' and 'Subnet1-NSG (Subnet)'. Below this, there are two tables: 'Inbound rules' and 'Outbound rules', each with columns for Name, Priority, Source, Source Ports, Destination, Destination Ports, and Protocol.

NAME	PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL
allowRDP	1000	Internet (76 prefixes)	0-65535	0.0.0.0/0	3389-3389	TCP
AllowVnetInBound	65000	VirtualNetwork (4 prefixes)	0-65535	VirtualNetwork (4 prefixes)	0-65535	All
AllowAzureLoadBalancer...	65001	AzureLoadBalancer (1 prefixes)	0-65535	0.0.0.0/0	0-65535	All
DenyAllInBound	65500	0.0.0.0/0	0-65535	0.0.0.0/0	0-65535	All

NAME	PRIORITY	SOURCE	SOURCE PORTS	DESTINATION	DESTINATION PORTS	PROTOCOL
AllowVnetOutBound	65000	VirtualNetwork (4 prefixes)	0-65535	VirtualNetwork (4 prefixes)	0-65535	All
AllowInternetOutBound	65001	0.0.0.0/0	0-65535	Internet (76 prefixes)	0-65535	All

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

1/17/2017 • 6 min to read • [Edit on GitHub](#)

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.
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:

Microsoft-Azure-Compute

```

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:

\$NSGs:EffectiveSecurityRules Sort-Object Direction, Access, Priority Out-GridView											
Filter											
and DestinationPortRange contains 3389											
Add criteria											
Clear All											
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	0	0	Allow	100	Inbound	
securityRules/Deny-All-RDP	Tcp	0-65535	3389-3389	0.0.0.0/0	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

1/17/2017 • 7 min to read • [Edit on GitHub](#)

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.

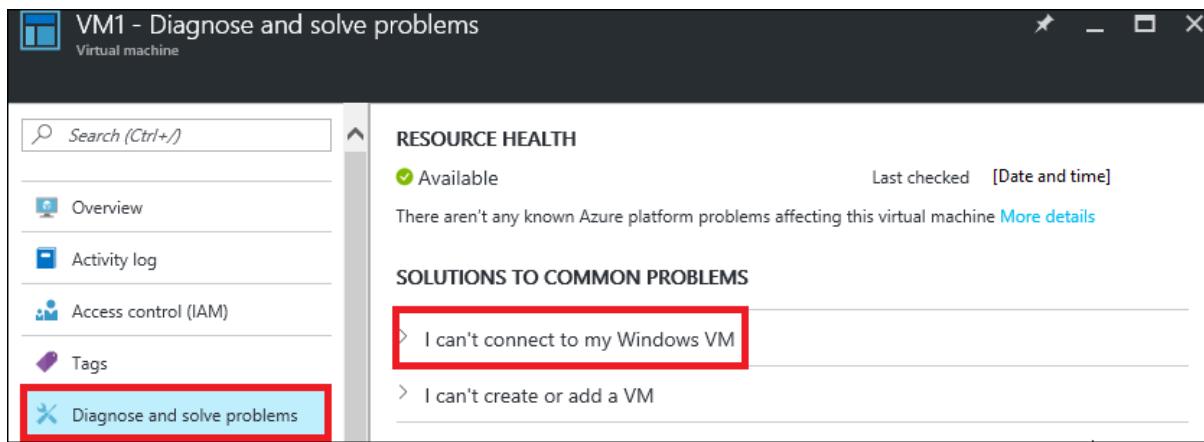
NOTE

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>.
2. Click **More 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 'Diagnose and solve problems' blade for the same virtual machine. The 'SOLUTIONS TO COMMON PROBLEMS' section now includes a expanded item 'I can't connect to my Windows VM'. Under this item, there's a heading 'Recommended steps' followed by a numbered list of troubleshooting steps. Step 6, which mentions 'effective routes', is highlighted with a red box.

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

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 **More 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.

Scope	Network interface (VM1-NIC1)		
SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE
Default	Active	10.9.0.0/16	VnetLocal
Default	Active	10.10.0.0/16	VNetPeering
Default	Active	0.0.0.0/0	Internet

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 (*UDRRoute*) 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 for managing network interfaces. On the left, the 'Network interfaces' blade lists 'Subscriptions: All 3 selected'. A red box highlights the 'VM1-NIC1' entry in the 'NAME' column. On the right, the 'VM1-NIC1 - Effective routes' blade is open. The sidebar includes links for Overview, Activity log, Access control (IAM), Tags, IP configurations, DNS servers, Network security group, Effective security rules, and Effective routes. A red box highlights the 'Effective routes' link under the 'SETTINGS' section.

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 **More 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 for managing route tables. On the left, the 'Route tables' blade lists 'Subscriptions: All 3 selected' and shows 'UDRouteTable' selected. A red box highlights the 'UDRouteTable' entry in the 'NAME' column. On the right, the 'UDRouteTable - Effective routes' blade is open. The sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Routes, Subnets, and Effective routes. A red box highlights the 'Effective routes' link under the 'SETTINGS' section.

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.

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

1/17/2017 • 5 min to read • [Edit on GitHub](#)

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.
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.

Search peerings				
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.

Viewing and modifying hostnames

1/17/2017 • 2 min to read • [Edit on GitHub](#)

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](#)

How to move a VM or role instance to a different subnet

1/17/2017 • 1 min to read • [Edit on GitHub](#)

You can use PowerShell to move your VMs from one subnet to another in the same virtual network (VNet). Role instances can be moved by editing the CSCFG, rather than using PowerShell.

NOTE

This article contains information that is relative to Azure classic deployments 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-AzureVMGet-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>
```