

Contents

Azure Firewall Documentation

Overview

- What is Azure Firewall?

Tutorials

- Deploy and configure

- Deploy in hybrid network

- Filter inbound traffic with DNAT

- Monitor diagnostic logs

Samples

- Azure PowerShell

- Azure Monitor logs

Concepts

- FQDN tags

- Infrastructure FQDNs

- Logs and metrics

- Threat intelligence

- Rule processing logic

- Service tags

- IP Groups

- Forced tunneling

- Certifications

- Central management

How-to guides

- Deploy using Azure PowerShell

- Deploy in hybrid network - PowerShell

- Deploy using Azure CLI

- Deploy using a template

- Deploy with multiple public IP

- Deploy with Availability Zones

[Integrate with load balancer](#)

[Application rules with SQL FQDNs](#)

[SNAT private ranges](#)

[Create IP Groups](#)

Reference

[Azure CLI](#)

[Azure PowerShell](#)

[.NET](#)

[Java](#)

[Node.js](#)

[Python](#)

[REST](#)

[Azure Resource Manager](#)

Resources

[FAQ](#)

[Author templates](#)

[Azure Roadmap](#)

[Community templates](#)

[Pricing](#)

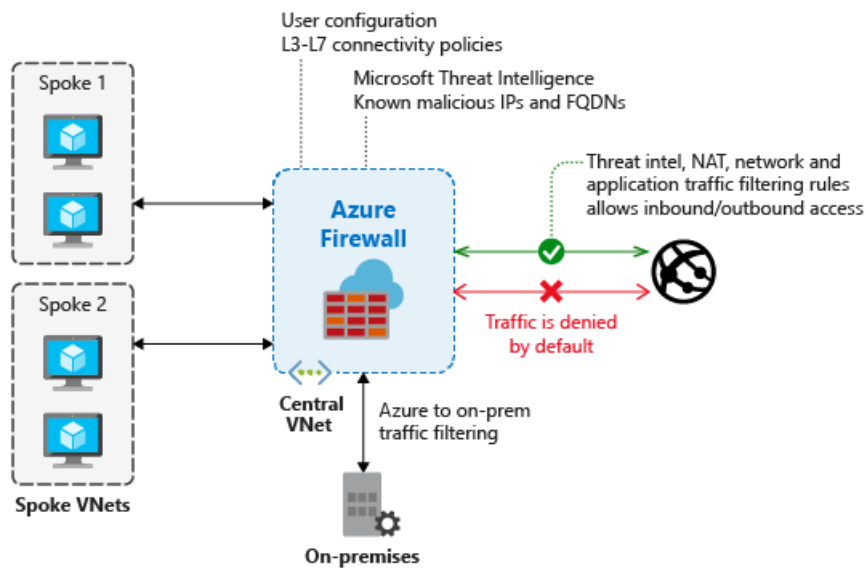
[Regional availability](#)

What is Azure Firewall?

2/26/2020 • 8 minutes to read • [Edit Online](#)



Azure Firewall is a managed, cloud-based network security service that protects your Azure Virtual Network resources. It's a fully stateful firewall as a service with built-in high availability and unrestricted cloud scalability.



You can centrally create, enforce, and log application and network connectivity policies across subscriptions and virtual networks. Azure Firewall uses a static public IP address for your virtual network resources allowing outside firewalls to identify traffic originating from your virtual network. The service is fully integrated with Azure Monitor for logging and analytics.

Azure Firewall offers the following features:

Built-in high availability

High availability is built in, so no additional load balancers are required and there's nothing you need to configure.

Availability Zones

Azure Firewall can be configured during deployment to span multiple Availability Zones for increased availability. With Availability Zones, your availability increases to 99.99% uptime. For more information, see the [Azure Firewall Service Level Agreement \(SLA\)](#). The 99.99% uptime SLA is offered when two or more Availability Zones are selected.

You can also associate Azure Firewall to a specific zone just for proximity reasons, using the service standard 99.95% SLA.

There's no additional cost for a firewall deployed in an Availability Zone. However, there are additional costs for inbound and outbound data transfers associated with Availability Zones. For more information, see [Bandwidth](#)

[pricing details.](#)

Azure Firewall Availability Zones are available in regions that support Availability Zones. For more information, see [What are Availability Zones in Azure?](#)

NOTE

Availability Zones can only be configured during deployment. You can't configure an existing firewall to include Availability Zones.

For more information about Availability Zones, see [What are Availability Zones in Azure?](#)

Unrestricted cloud scalability

Azure Firewall can scale up as much as you need to accommodate changing network traffic flows, so you don't need to budget for your peak traffic.

Application FQDN filtering rules

You can limit outbound HTTP/S traffic or Azure SQL traffic (preview) to a specified list of fully qualified domain names (FQDN) including wild cards. This feature doesn't require SSL termination.

Network traffic filtering rules

You can centrally create *allow* or *deny* network filtering rules by source and destination IP address, port, and protocol. Azure Firewall is fully stateful, so it can distinguish legitimate packets for different types of connections. Rules are enforced and logged across multiple subscriptions and virtual networks.

FQDN tags

FQDN tags make it easy for you to allow well known Azure service network traffic through your firewall. For example, say you want to allow Windows Update network traffic through your firewall. You create an application rule and include the Windows Update tag. Now network traffic from Windows Update can flow through your firewall.

Service tags

A service tag represents a group of IP address prefixes to help minimize complexity for security rule creation. You can't create your own service tag, nor specify which IP addresses are included within a tag. Microsoft manages the address prefixes encompassed by the service tag, and automatically updates the service tag as addresses change.

Threat intelligence

Threat intelligence-based filtering can be enabled for your firewall to alert and deny traffic from/to known malicious IP addresses and domains. The IP addresses and domains are sourced from the Microsoft Threat Intelligence feed.

Outbound SNAT support

All outbound virtual network traffic IP addresses are translated to the Azure Firewall public IP (Source Network Address Translation). You can identify and allow traffic originating from your virtual network to remote Internet destinations. Azure Firewall doesn't SNAT when the destination IP is a private IP range per [IANA RFC 1918](#).

If your organization uses a public IP address range for private networks, Azure Firewall will SNAT the traffic to one

of the firewall private IP addresses in AzureFirewallSubnet. You can configure Azure Firewall to **not** SNAT your public IP address range. For more information, see [Azure Firewall SNAT private IP address ranges](#).

Inbound DNAT support

Inbound Internet network traffic to your firewall public IP address is translated (Destination Network Address Translation) and filtered to the private IP addresses on your virtual networks.

Multiple public IP addresses

You can associate multiple public IP addresses (up to 100) with your firewall.

This enables the following scenarios:

- **DNAT** - You can translate multiple standard port instances to your backend servers. For example, if you have two public IP addresses, you can translate TCP port 3389 (RDP) for both IP addresses.
- **SNAT** - Additional ports are available for outbound SNAT connections, reducing the potential for SNAT port exhaustion. At this time, Azure Firewall randomly selects the source public IP address to use for a connection. If you have any downstream filtering on your network, you need to allow all public IP addresses associated with your firewall.

Azure Monitor logging

All events are integrated with Azure Monitor, allowing you to archive logs to a storage account, stream events to your Event Hub, or send them to Azure Monitor logs.

Certifications

Azure Firewall is Payment Card Industry (PCI), Service Organization Controls (SOC), International Organization for Standardization (ISO), and ICSA Labs compliant. For more information, see [Azure Firewall compliance certifications](#).

Known issues

Azure Firewall has the following known issues:

ISSUE	DESCRIPTION	MITIGATION
Network filtering rules for non-TCP/UDP protocols (for example ICMP) don't work for Internet bound traffic	Network filtering rules for non-TCP/UDP protocols don't work with SNAT to your public IP address. Non-TCP/UDP protocols are supported between spoke subnets and VNets.	Azure Firewall uses the Standard Load Balancer, which doesn't support SNAT for IP protocols today . We're exploring options to support this scenario in a future release.
Missing PowerShell and CLI support for ICMP	Azure PowerShell and CLI don't support ICMP as a valid protocol in network rules.	It's still possible to use ICMP as a protocol via the portal and the REST API. We're working to add ICMP in PowerShell and CLI soon.
FQDN tags require a protocol: port to be set	Application rules with FQDN tags require port: protocol definition.	You can use https as the port: protocol value. We're working to make this field optional when FQDN tags are used.

ISSUE	DESCRIPTION	MITIGATION
Moving a firewall to a different resource group or subscription isn't supported	Moving a firewall to a different resource group or subscription isn't supported.	Supporting this functionality is on our road map. To move a firewall to a different resource group or subscription, you must delete the current instance and recreate it in the new resource group or subscription.
Threat intelligence alerts may get masked	Network rules with destination 80/443 for outbound filtering masks threat intelligence alerts when configured to alert only mode.	Create outbound filtering for 80/443 using application rules. Or, change the threat intelligence mode to Alert and Deny .
Azure Firewall uses Azure DNS only for name resolution	Azure Firewall resolves FQDNs using Azure DNS only. A custom DNS server isn't supported. There's no impact on DNS resolution on other subnets.	We're working to relax this limitation.
Azure Firewall SNAT/DNAT doesn't work for private IP destinations	Azure Firewall SNAT/DNAT support is limited to Internet egress/ingress. SNAT/DNAT doesn't currently work for private IP destinations. For example, spoke to spoke.	This is a current limitation.
Can't remove first public IP configuration	Each Azure Firewall public IP address is assigned to an <i>IP configuration</i> . The first IP configuration is assigned during the firewall deployment, and typically also contains a reference to the firewall subnet (unless configured explicitly differently via a template deployment). You can't delete this IP configuration because it would de-allocate the firewall. You can still change or remove the public IP address associated with this IP configuration if the firewall has at least one other public IP address available to use.	This is by design.
Availability zones can only be configured during deployment.	Availability zones can only be configured during deployment. You can't configure Availability Zones after a firewall has been deployed.	This is by design.
SNAT on inbound connections	In addition to DNAT, connections via the firewall public IP address (inbound) are SNATed to one of the firewall private IPs. This requirement today (also for Active/Active NVAs) to ensure symmetric routing.	To preserve the original source for HTTP/S, consider using XFF headers. For example, use a service such as Azure Front Door or Azure Application Gateway in front of the firewall. You can also add WAF as part of Azure Front Door and chain to the firewall.

ISSUE	DESCRIPTION	MITIGATION
SQL FQDN filtering support only in proxy mode (port 1433)	<p>For Azure SQL Database, Azure SQL Data Warehouse, and Azure SQL Managed Instance:</p> <p>During the preview, SQL FQDN filtering is supported in proxy-mode only (port 1433).</p> <p>For Azure SQL IaaS:</p> <p>If you're using non-standard ports, you can specify those ports in the application rules.</p>	For SQL in redirect mode, which is the default if connecting from within Azure, you can instead filter access using the SQL service tag as part of Azure Firewall network rules.
Outbound traffic on TCP port 25 isn't allowed	Outbound SMTP connections that use TCP port 25 are blocked. Port 25 is primarily used for unauthenticated email delivery. This is the default platform behavior for virtual machines. For more information, see more Troubleshoot outbound SMTP connectivity issues in Azure . However, unlike virtual machines, it isn't currently possible to enable this functionality on Azure Firewall.	Follow the recommended method to send email as documented in the SMTP troubleshooting article. Or, exclude the virtual machine that needs outbound SMTP access from your default route to the firewall, and instead configure outbound access directly to the Internet.
Active FTP isn't supported	Active FTP is disabled on Azure Firewall to protect against FTP bounce attacks using the FTP PORT command.	You can use Passive FTP instead. You must still explicitly open TCP ports 20 and 21 on the firewall.

Next steps

- [Tutorial: Deploy and configure Azure Firewall using the Azure portal](#)
- [Deploy Azure Firewall using a template](#)
- [Create an Azure Firewall test environment](#)

Tutorial: Deploy and configure Azure Firewall using the Azure portal

2/21/2020 • 7 minutes to read • [Edit Online](#)

Controlling outbound network access is an important part of an overall network security plan. For example, you may want to limit access to web sites. Or, you may want to limit the outbound IP addresses and ports that can be accessed.

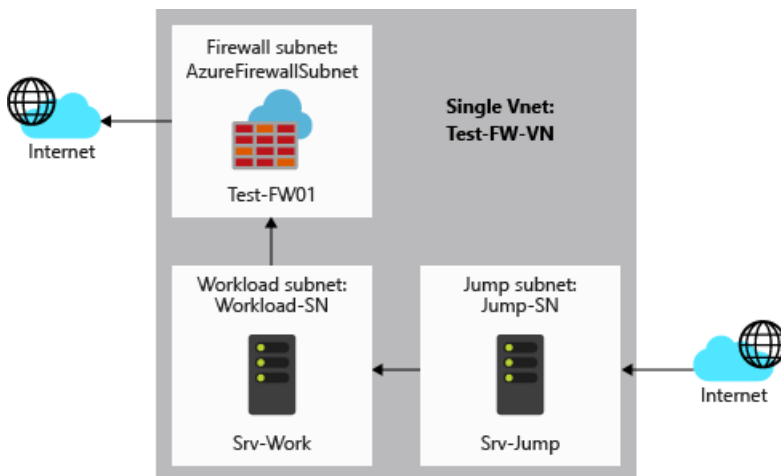
One way you can control outbound network access from an Azure subnet is with Azure Firewall. With Azure Firewall, you can configure:

- Application rules that define fully qualified domain names (FQDNs) that can be accessed from a subnet.
- Network rules that define source address, protocol, destination port, and destination address.

Network traffic is subjected to the configured firewall rules when you route your network traffic to the firewall as the subnet default gateway.

For this tutorial, you create a simplified single VNet with three subnets for easy deployment. For production deployments, a [hub and spoke model](#) is recommended. The firewall is in its own VNet. The workload servers are in peered VNets in the same region with one or more subnets.

- **AzureFirewallSubnet** - the firewall is in this subnet.
- **Workload-SN** - the workload server is in this subnet. This subnet's network traffic goes through the firewall.
- **Jump-SN** - The "jump" server is in this subnet. The jump server has a public IP address that you can connect to using Remote Desktop. From there, you can then connect to (using another Remote Desktop) the workload server.



In this tutorial, you learn how to:

- Set up a test network environment
- Deploy a firewall
- Create a default route
- Configure an application rule to allow access to www.google.com
- Configure a network rule to allow access to external DNS servers
- Test the firewall

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

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

Set up the network

First, create a resource group to contain the resources needed to deploy the firewall. Then create a VNet, subnets, and test servers.

Create a resource group

The resource group contains all the resources for the tutorial.

1. Sign in to the Azure portal at <https://portal.azure.com>.
2. On the Azure portal menu, select **Resource groups** or search for and select *Resource groups* from any page. Then select **Add**.
3. For **Resource group name**, enter *Test-FW-RG*.
4. For **Subscription**, select your subscription.
5. For **Resource group location**, select a location. All other resources that you create must be in the same location.
6. Select **Create**.

Create a VNet

This VNet will contain three subnets.

NOTE

The size of the AzureFirewallSubnet subnet is /26. For more information about the subnet size, see [Azure Firewall FAQ](#).

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Select **Networking** > **Virtual network**.
3. For **Name**, type **Test-FW-VN**.
4. For **Address space**, type **10.0.0.0/16**.
5. For **Subscription**, select your subscription.
6. For **Resource group**, select **Test-FW-RG**.
7. For **Location**, select the same location that you used previously.
8. Under **Subnet**, for **Name** type **AzureFirewallSubnet**. The firewall will be in this subnet, and the subnet name **must** be AzureFirewallSubnet.
9. For **Address range**, type **10.0.1.0/26**.
10. Accept the other default settings, and then select **Create**.

Create additional subnets

Next, create subnets for the jump server, and a subnet for the workload servers.

1. On the Azure portal menu, select **Resource groups** or search for and select *Resource groups* from any page. Then select **Test-FW-RG**.
2. Select the **Test-FW-VN** virtual network.
3. Select **Subnets** > **+Subnet**.
4. For **Name**, type **Workload-SN**.
5. For **Address range**, type **10.0.2.0/24**.
6. Select **OK**.

Create another subnet named **Jump-SN**, address range **10.0.3.0/24**.

Create virtual machines

Now create the jump and workload virtual machines, and place them in the appropriate subnets.

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Select **Compute** and then select **Windows Server 2016 Datacenter** in the Featured list.
3. Enter these values for the virtual machine:

SETTING	VALUE
Resource group	Test-FW-RG
Virtual machine name	Srv-Jump
Region	Same as previous
Administrator user name	azureuser
Password	Azure123456!

4. Under **Inbound port rules**, for **Public inbound ports**, select **Allow selected ports**.
5. For **Select inbound ports**, select **RDP (3389)**.
6. Accept the other defaults and select **Next: Disks**.
7. Accept the disk defaults and select **Next: Networking**.
8. Make sure that **Test-FW-VN** is selected for the virtual network and the subnet is **Jump-SN**.
9. For **Public IP**, accept the default new public ip address name (Srv-Jump-ip).
10. Accept the other defaults and select **Next: Management**.
11. Select **Off** to disable boot diagnostics. Accept the other defaults and select **Review + create**.
12. Review the settings on the summary page, and then select **Create**.

Use the information in the following table to configure another virtual machine named **Srv-Work**. The rest of the configuration is the same as the Srv-Jump virtual machine.

SETTING	VALUE
Subnet	Workload-SN
Public IP	None
Public inbound ports	None

Deploy the firewall

Deploy the firewall into the VNet.

1. On the Azure portal menu or from the **Home** page, select **Create a resource**.
2. Type **firewall** in the search box and press **Enter**.
3. Select **Firewall** and then select **Create**.

- On the **Create a Firewall** page, use the following table to configure the firewall:

SETTING	VALUE
Subscription	<your subscription>
Resource group	Test-FW-RG
Name	Test-FW01
Location	Select the same location that you used previously
Choose a virtual network	Use existing: Test-FW-VN
Public IP address	Add new. The Public IP address must be the Standard SKU type.

- Select **Review + create**.
- Review the summary, and then select **Create** to create the firewall.

This will take a few minutes to deploy.
- After deployment completes, go to the **Test-FW-RG** resource group, and select the **Test-FW01** firewall.
- Note the private IP address. You'll use it later when you create the default route.

Create a default route

For the **Workload-SN** subnet, configure the outbound default route to go through the firewall.

- On the Azure portal menu, select **All services** or search for and select *All services* from any page.
- Under **Networking**, select **Route tables**.
- Select **Add**.
- For **Name**, type **Firewall-route**.
- For **Subscription**, select your subscription.
- For **Resource group**, select **Test-FW-RG**.
- For **Location**, select the same location that you used previously.
- Select **Create**.
- Select **Refresh**, and then select the **Firewall-route** route table.
- Select **Subnets** and then select **Associate**.
- Select **Virtual network** > **Test-FW-VN**.
- For **Subnet**, select **Workload-SN**. Make sure that you select only the **Workload-SN** subnet for this route, otherwise your firewall won't work correctly.
- Select **OK**.
- Select **Routes** and then select **Add**.
- For **Route name**, type **fw-dg**.

16. For **Address prefix**, type **0.0.0.0/0**.
17. For **Next hop type**, select **Virtual appliance**.

Azure Firewall is actually a managed service, but virtual appliance works in this situation.
18. For **Next hop address**, type the private IP address for the firewall that you noted previously.
19. Select **OK**.

Configure an application rule

This is the application rule that allows outbound access to www.google.com.

1. Open the **Test-FW-RG**, and select the **Test-FW01** firewall.
2. On the **Test-FW01** page, under **Settings**, select **Rules**.
3. Select the **Application rule collection** tab.
4. Select **Add application rule collection**.
5. For **Name**, type **App-Coll01**.
6. For **Priority**, type **200**.
7. For **Action**, select **Allow**.
8. Under **Rules, Target FQDNs**, for **Name**, type **Allow-Google**.
9. For **Source type**, select **IP address**.
10. For **Source**, type **10.0.2.0/24**.
11. For **Protocol:port**, type **http, https**.
12. For **Target FQDNS**, type www.google.com
13. Select **Add**.

Azure Firewall includes a built-in rule collection for infrastructure FQDNs that are allowed by default. These FQDNs are specific for the platform and can't be used for other purposes. For more information, see [Infrastructure FQDNs](#).

Configure a network rule

This is the network rule that allows outbound access to two IP addresses at port 53 (DNS).

1. Select the **Network rule collection** tab.
2. Select **Add network rule collection**.
3. For **Name**, type **Net-Coll01**.
4. For **Priority**, type **200**.
5. For **Action**, select **Allow**.
6. Under **Rules, IP addresses**, for **Name**, type **Allow-DNS**.
7. For **Protocol**, select **UDP**.
8. For **Source type**, select **IP address**.
9. For **Source**, type **10.0.2.0/24**.
10. For **Destination address**, type **209.244.0.3,209.244.0.4**

These are public DNS servers operated by CenturyLink.

11. For **Destination Ports**, type **53**.

12. Select **Add**.

Change the primary and secondary DNS address for the Srv-Work network interface

For testing purposes in this tutorial, configure the server's primary and secondary DNS addresses. This isn't a general Azure Firewall requirement.

1. On the Azure portal menu, select **Resource groups** or search for and select *Resource groups* from any page. Select the **Test-FW-RG** resource group.
2. Select the network interface for the **Srv-Work** virtual machine.
3. Under **Settings**, select **DNS servers**.
4. Under **DNS servers**, select **Custom**.
5. Type **209.244.0.3** in the **Add DNS server** text box, and **209.244.0.4** in the next text box.
6. Select **Save**.
7. Restart the **Srv-Work** virtual machine.

Test the firewall

Now, test the firewall to confirm that it works as expected.

1. From the Azure portal, review the network settings for the **Srv-Work** virtual machine and note the private IP address.
2. Connect a remote desktop to **Srv-Jump** virtual machine, and sign in. From there, open a remote desktop connection to the **Srv-Work** private IP address.
3. Open Internet Explorer and browse to <https://www.google.com>.
4. Select **OK** > **Close** on the Internet Explorer security alerts.

You should see the Google home page.

5. Browse to <https://www.microsoft.com>.

You should be blocked by the firewall.

So now you've verified that the firewall rules are working:

- You can browse to the one allowed FQDN, but not to any others.
- You can resolve DNS names using the configured external DNS server.

Clean up resources

You can keep your firewall resources for the next tutorial, or if no longer needed, delete the **Test-FW-RG** resource group to delete all firewall-related resources.

Next steps

[Tutorial: Monitor Azure Firewall logs](#)

Tutorial: Deploy and configure Azure Firewall in a hybrid network using the Azure portal

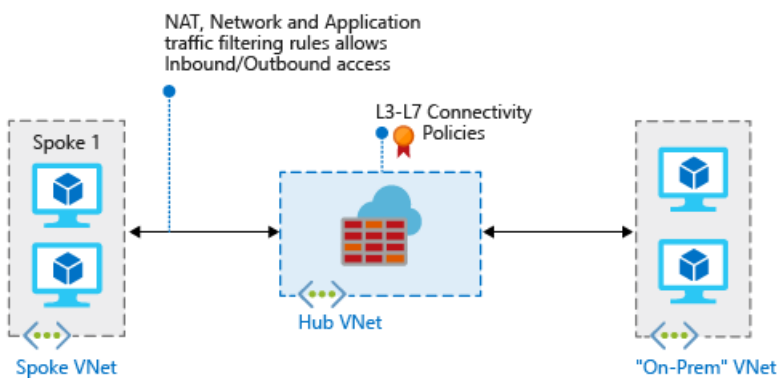
2/21/2020 • 13 minutes to read • [Edit Online](#)

When you connect your on-premises network to an Azure virtual network to create a hybrid network, the ability to control access to your Azure network resources is an important part of an overall security plan.

You can use Azure Firewall to control network access in a hybrid network using rules that define allowed and denied network traffic.

For this tutorial, you create three virtual networks:

- **VNet-Hub** - the firewall is in this virtual network.
- **VNet-Spoke** - the spoke virtual network represents the workload located on Azure.
- **VNet-Onprem** - The on-premises virtual network represents an on-premises network. In an actual deployment, it can be connected by either a VPN or ExpressRoute connection. For simplicity, this tutorial uses a VPN gateway connection, and an Azure-located virtual network is used to represent an on-premises network.



In this tutorial, you learn how to:

- Declare the variables
- Create the firewall hub virtual network
- Create the spoke virtual network
- Create the on-premises virtual network
- Configure and deploy the firewall
- Create and connect the VPN gateways
- Peer the hub and spoke virtual networks
- Create the routes
- Create the virtual machines
- Test the firewall

If you want to use Azure PowerShell instead to complete this procedure, see [Deploy and configure Azure Firewall in a hybrid network using Azure PowerShell](#).

Prerequisites

A hybrid network uses the hub-and-spoke architecture model to route traffic between Azure VNets and on-premise networks. The hub-and-spoke architecture has the following requirements:

- Set **AllowGatewayTransit** when peering VNet-Hub to VNet-Spoke. In a hub-and-spoke network architecture, a gateway transit allows the spoke virtual networks to share the VPN gateway in the hub, instead of deploying VPN gateways in every spoke virtual network.

Additionally, routes to the gateway-connected virtual networks or on-premises networks will automatically propagate to the routing tables for the peered virtual networks using the gateway transit. For more information, see [Configure VPN gateway transit for virtual network peering](#).

- Set **UseRemoteGateways** when you peer VNet-Spoke to VNet-Hub. If **UseRemoteGateways** is set and **AllowGatewayTransit** on remote peering is also set, the spoke virtual network uses gateways of the remote virtual network for transit.
- To route the spoke subnet traffic through the hub firewall, you need a User Defined route (UDR) that points to the firewall with the **Disable BGP route propagation** option set. The **Disable BGP route propagation** option prevents route distribution to the spoke subnets. This prevents learned routes from conflicting with your UDR.
- Configure a UDR on the hub gateway subnet that points to the firewall IP address as the next hop to the spoke networks. No UDR is required on the Azure Firewall subnet, as it learns routes from BGP.

See the [Create Routes](#) section in this tutorial to see how these routes are created.

NOTE

Azure Firewall must have direct Internet connectivity. If your AzureFirewallSubnet learns a default route to your on-premises network via BGP, you must override this with a 0.0.0.0/0 UDR with the **NextHopType** value set as **Internet** to maintain direct Internet connectivity.

Azure Firewall can be configured to support forced tunneling. For more information, see [Azure Firewall forced tunneling](#).

NOTE

Traffic between directly peered VNets is routed directly even if a UDR points to Azure Firewall as the default gateway. To send subnet to subnet traffic to the firewall in this scenario, a UDR must contain the target subnet network prefix explicitly on both subnets.

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

Create the firewall hub virtual network

First, create the resource group to contain the resources for this tutorial:

1. Sign in to the Azure portal at <https://portal.azure.com>.
2. On the Azure portal home page, select **Resource groups** > **Add**.
3. For **Resource group name**, type **FW-Hybrid-Test**.
4. For **Subscription**, select your subscription.
5. For **Region**, select **East US**. All resources that you create later must be in the same location.
6. Select **Review + Create**.
7. Select **Create**.

Now, create the VNet:

NOTE

The size of the AzureFirewallSubnet subnet is /26. For more information about the subnet size, see [Azure Firewall FAQ](#).

1. From the Azure portal home page, select **Create a resource**.
2. Under **Networking**, select **Virtual network**.
3. For **Name**, type **VNet-hub**.
4. For **Address space**, type **10.5.0.0/16**.
5. For **Subscription**, select your subscription.
6. For **Resource group**, select **FW-Hybrid-Test**.
7. For **Location**, select **East US**.
8. Under **Subnet**, for **Name** type **AzureFirewallSubnet**. The firewall will be in this subnet, and the subnet name **must** be AzureFirewallSubnet.
9. For **Address range**, type **10.5.0.0/26**.
10. Accept the other default settings, and then select **Create**.

Create the spoke virtual network

1. From the Azure portal home page, select **Create a resource**.
2. Under **Networking**, select **Virtual network**.
3. For **Name**, type **VNet-Spoke**.
4. For **Address space**, type **10.6.0.0/16**.
5. For **Subscription**, select your subscription.
6. For **Resource group**, select **FW-Hybrid-Test**.
7. For **Location**, select the same location that you used previously.
8. Under **Subnet**, for **Name** type **SN-Workload**.
9. For **Address range**, type **10.6.0.0/24**.
10. Accept the other default settings, and then select **Create**.

Create the on-premises virtual network

1. From the Azure portal home page, select **Create a resource**.
2. Under **Networking**, select **Virtual network**.
3. For **Name**, type **VNet-OnPrem**.
4. For **Address space**, type **192.168.0.0/16**.
5. For **Subscription**, select your subscription.
6. For **Resource group**, select **FW-Hybrid-Test**.
7. For **Location**, select the same location that you used previously.
8. Under **Subnet**, for **Name** type **SN-Corp**.
9. For **Address range**, type **192.168.1.0/24**.
10. Accept the other default settings, and then select **Create**.

Now create a second subnet for the gateway.

1. On the **VNet-Onprem** page, select **Subnets**.
2. Select **+Subnet**.
3. For **Name**, type **GatewaySubnet**.
4. For **Address range (CIDR block)** type **192.168.2.0/24**.
5. Select **OK**.

Create a public IP address

This is the public IP address used for the on-premises gateway.

1. From the Azure portal home page, select **Create a resource**.
2. In the search text box, type **public IP address** and press **Enter**.
3. Select **Public IP address** and then select **Create**.
4. For the name, type **VNet-Onprem-GW-pip**.
5. For the resource group, type **FW-Hybrid-Test**.
6. For **Location**, select the same location that you used previously.
7. Accept the other defaults, and then select **Create**.

Configure and deploy the firewall

Now deploy the firewall into the firewall hub virtual network.

1. From the Azure portal home page, select **Create a resource**.
2. In the left column, select **Networking**, and then select **Firewall**.
3. On the **Create a Firewall** page, use the following table to configure the firewall:

SETTING	VALUE
Subscription	<your subscription>
Resource group	FW-Hybrid-Test
Name	AzFW01
Location	Select the same location that you used previously
Choose a virtual network	Use existing: VNet-hub
Public IP address	Create new: Name - fw-pip.

4. Select **Review + create**.
5. Review the summary, and then select **Create** to create the firewall.

This takes a few minutes to deploy.

6. After deployment completes, go to the **FW-Hybrid-Test** resource group, and select the **AzFW01** firewall.
7. Note the private IP address. You'll use it later when you create the default route.

Configure network rules

First, add a network rule to allow web traffic.

1. On the **AzFW01** page, Select **Rules**.
2. Select the **Network rule collection** tab.
3. Select **Add network rule collection**.
4. For **Name**, type **RCNet01**.
5. For **Priority**, type **100**.
6. For **Action**, select **Allow**.

7. Under **Rules**, for **Name**, type **AllowWeb**.
8. For **Protocol**, select **TCP**.
9. For **Source type**, select **IP address**.
10. For **Source**, type **192.168.1.0/24**.
11. For **Destination address**, type **10.6.0.0/16**
12. For **Destination Ports**, type **80**.

Now add a rule to allow RDP traffic.

On the second rule row, type the following information:

1. **Name**, type **AllowRDP**.
2. For **Protocol**, select **TCP**.
3. For **Source type**, select **IP address**.
4. For **Source**, type **192.168.1.0/24**.
5. For **Destination address**, type **10.6.0.0/16**
6. For **Destination Ports**, type **3389**.
7. Select **Add**.

Create and connect the VPN gateways

The hub and on-premises virtual networks are connected via VPN gateways.

Create a VPN gateway for the hub virtual network

Now create the VPN gateway for the hub virtual network. Network-to-network configurations require a RouteBased VpnType. Creating a VPN gateway can often take 45 minutes or more, depending on the selected VPN gateway SKU.

1. From the Azure portal home page, select **Create a resource**.
2. In the search text box, type **virtual network gateway** and press **Enter**.
3. Select **Virtual network gateway**, and select **Create**.
4. For **Name**, type **GW-hub**.
5. For **Region**, select the same region that you used previously.
6. For **Gateway type**, select **VPN**.
7. For **VPN type**, select **Route-based**.
8. For **SKU**, select **Basic**.
9. For **Virtual network**, select **VNet-hub**.
10. For **Public IP address**, select **Create new**, and type **VNet-hub-GW-pip** for the name.
11. Accept the remaining defaults and then select **Review + create**.
12. Review the configuration, then select **Create**.

Create a VPN gateway for the on-premises virtual network

Now create the VPN gateway for the on-premises virtual network. Network-to-network configurations require a RouteBased VpnType. Creating a VPN gateway can often take 45 minutes or more, depending on the selected VPN gateway SKU.

1. From the Azure portal home page, select **Create a resource**.
2. In the search text box, type **virtual network gateway** and press **Enter**.
3. Select **Virtual network gateway**, and select **Create**.
4. For **Name**, type **GW-Onprem**.
5. For **Region**, select the same region that you used previously.

6. For **Gateway type**, select **VPN**.
7. For **VPN type**, select **Route-based**.
8. For **SKU**, select **Basic**.
9. For **Virtual network**, select **VNet-Onprem**.
10. For **Public IP address**, select **Create new**, and type **VNet-Onprem-GW-pip** for the name.
11. Accept the remaining defaults and then select **Review + create**.
12. Review the configuration, then select **Create**.

Create the VPN connections

Now you can create the VPN connections between the hub and on-premises gateways.

In this step, you create the connection from the hub virtual network to the on-premises virtual network. 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.

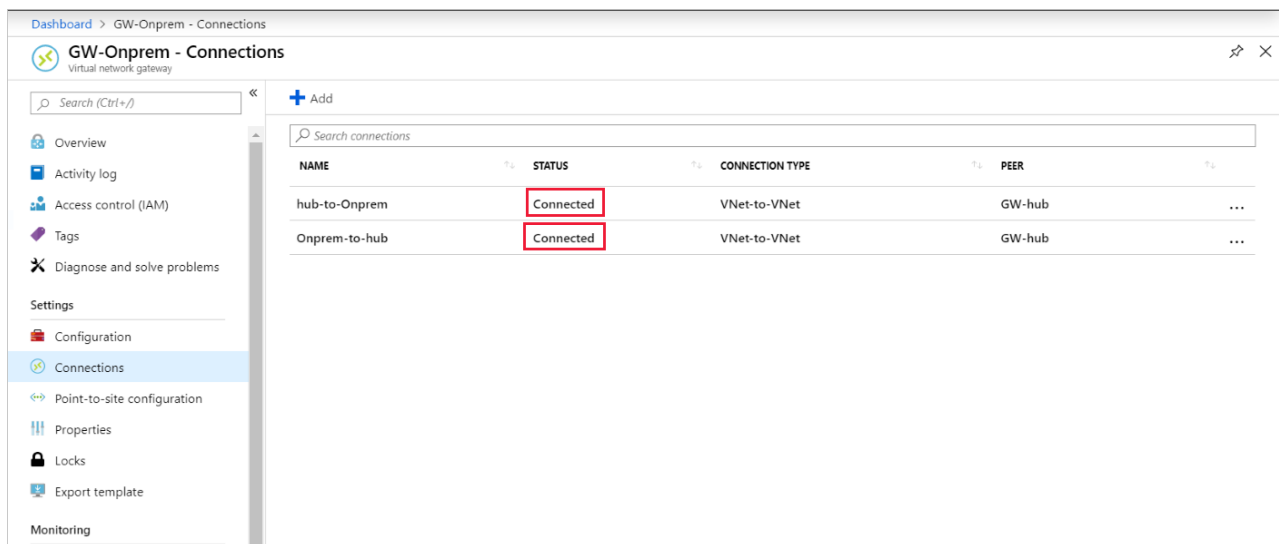
1. Open the **FW-Hybrid-Test** resource group and select the **GW-hub** gateway.
2. Select **Connections** in the left column.
3. Select **Add**.
4. The the connection name, type **Hub-to-Onprem**.
5. Select **VNet-to-VNet** for **Connection type**.
6. For the **Second virtual network gateway**, select **GW-Onprem**.
7. For **Shared key (PSK)**, type **AzureA1b2C3**.
8. Select **OK**.

Create the on-premises to hub virtual network connection. This step is similar to the previous one, except you create the connection from VNet-Onprem to VNet-hub. Make sure the shared keys match. The connection will be established after a few minutes.

1. Open the **FW-Hybrid-Test** resource group and select the **GW-Onprem** gateway.
2. Select **Connections** in the left column.
3. Select **Add**.
4. The the connection name, type **Onprem-to-Hub**.
5. Select **VNet-to-VNet** for **Connection type**.
6. For the **Second virtual network gateway**, select **GW-hub**.
7. For **Shared key (PSK)**, type **AzureA1b2C3**.
8. Select **OK**.

Verify the connection

After about five minutes or so, the status of both connections should be **Connected**.



Peer the hub and spoke virtual networks

Now peer the hub and spoke virtual networks.

1. Open the **FW-Hybrid-Test** resource group and select the **VNet-hub** virtual network.
2. In the left column, select **Peerings**.
3. Select **Add**.
4. For **Name**, type **HubtoSpoke**.
5. For the **Virtual network**, select **VNet-spoke**
6. For the name of the peering from VNetSpoke to VNet-hub, type **SpoketoHub**.
7. Select **Allow gateway transit**.
8. Select **OK**.

Configure additional settings for the SpoketoHub peering

You'll need to enable the **Allow forwarded traffic** on the SpoketoHub peering.

1. Open the **FW-Hybrid-Test** resource group and select the **VNet-Spoke** virtual network.
2. In the left column, select **Peerings**.
3. Select the **SpoketoHub** peering.
4. Under **Allow forwarded traffic from VNet-hub to VNet-Spoke**, select **Enabled**.
5. Select **Save**.

Create the routes

Next, create a couple routes:

- A route from the hub gateway subnet to the spoke subnet through the firewall IP address
 - A default route from the spoke subnet through the firewall IP address
1. From the Azure portal home page, select **Create a resource**.
 2. In the search text box, type **route table** and press **Enter**.
 3. Select **Route table**.
 4. Select **Create**.
 5. For the name, type **UDR-Hub-Spoke**.
 6. Select the **FW-Hybrid-Test** for the resource group.
 7. For **Location**, select the same location that you used previously.
 8. Select **Create**.

9. After the route table is created, select it to open the route table page.
10. Select **Routes** in the left column.
11. Select **Add**.
12. For the route name, type **ToSpoke**.
13. For the address prefix, type **10.6.0.0/16**.
14. For next hop type, select **Virtual appliance**.
15. For next hop address, type the firewall's private IP address that you noted earlier.
16. Select **OK**.

Now associate the route to the subnet.

1. On the **UDR-Hub-Spoke - Routes** page, select **Subnets**.
2. Select **Associate**.
3. Select **Choose a virtual network**.
4. Select **VNet-hub**.
5. Select **GatewaySubnet**.
6. Select **OK**.

Now create the default route from the spoke subnet.

1. From the Azure portal home page, select **Create a resource**.
2. In the search text box, type **route table** and press **Enter**.
3. Select **Route table**.
4. Select **Create**.
5. For the name, type **UDR-DG**.
6. Select the **FW-Hybrid-Test** for the resource group.
7. For **Location**, select the same location that you used previously.
8. For **Virtual network gateway route propagation**, select **Disabled**.
9. Select **Create**.
10. After the route table is created, select it to open the route table page.
11. Select **Routes** in the left column.
12. Select **Add**.
13. For the route name, type **ToHub**.
14. For the address prefix, type **0.0.0.0/0**.
15. For next hop type, select **Virtual appliance**.
16. For next hop address, type the firewall's private IP address that you noted earlier.
17. Select **OK**.

Now associate the route to the subnet.

1. On the **UDR-DG - Routes** page, select **Subnets**.
2. Select **Associate**.
3. Select **Choose a virtual network**.
4. Select **VNet-spoke**.
5. Select **SN-Workload**.
6. Select **OK**.

Create virtual machines

Now create the spoke workload and on-premises virtual machines, and place them in the appropriate subnets.

Create the workload virtual machine

Create a virtual machine in the spoke virtual network, running IIS, with no public IP address.

1. From the Azure portal home page, select **Create a resource**.
2. Under **Popular**, select **Windows Server 2016 Datacenter**.
3. Enter these values for the virtual machine:
 - **Resource group** - Select **FW-Hybrid-Test**.
 - **Virtual machine name**: *VM-Spoke-01*.
 - **Region** - Same region that you're used previously.
 - **User name**: *azureuser*.
 - **Password**: *Azure123456!*
4. Select **Next:Disks**.
5. Accept the defaults and select **Next: Networking**.
6. Select **VNet-Spoke** for the virtual network and the subnet is **SN-Workload**.
7. For **Public IP**, select **None**.
8. For **Public inbound ports**, select **Allow selected ports**, and then select **HTTP (80)**, and **RDP (3389)**
9. Select **Next:Management**.
10. For **Boot diagnostics**, Select **Off**.
11. Select **Review+Create**, review the settings on the summary page, and then select **Create**.

Install IIS

1. From the Azure portal, open the Cloud Shell and make sure that it's set to **PowerShell**.
2. Run the following command to install IIS on the virtual machine and change the location if necessary:

```
Set-AzVMExtension `
  -ResourceGroupName FW-Hybrid-Test `
  -ExtensionName IIS `
  -VMName VM-Spoke-01 `
  -Publisher Microsoft.Compute `
  -ExtensionType CustomScriptExtension `
  -TypeHandlerVersion 1.4 `
  -SettingString '{"commandToExecute":"powershell Add-WindowsFeature Web-Server; powershell
Add-Content -Path \"C:\\inetpub\\wwwroot\\Default.htm\" -Value $($env:computername)}' `
  -Location EastUS
```

Create the on-premises virtual machine

This is a virtual machine that you use to connect using Remote Desktop to the public IP address. From there, you then connect to the on-premises server through the firewall.

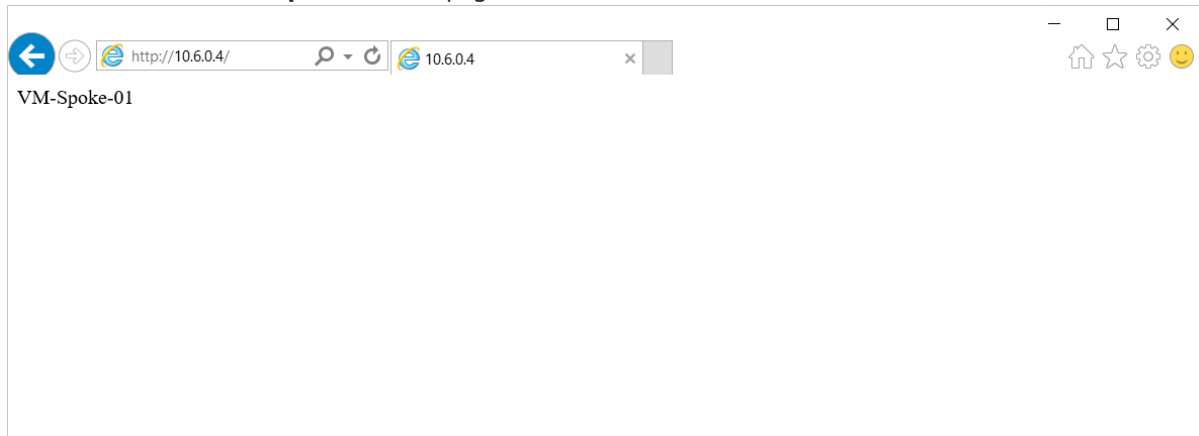
1. From the Azure portal home page, select **Create a resource**.
2. Under **Popular**, select **Windows Server 2016 Datacenter**.
3. Enter these values for the virtual machine:
 - **Resource group** - Select existing, and then select **FW-Hybrid-Test**.
 - **Virtual machine name** - *VM-Onprem*.
 - **Region** - Same region that you're used previously.
 - **User name**: *azureuser*.
 - **Password**: *Azure123456!*.
4. Select **Next:Disks**.
5. Accept the defaults and select **Next:Networking**.
6. Select **VNet-Onprem** for virtual network and the subnet is **SN-Corp**.
7. For **Public inbound ports**, select **Allow selected ports**, and then select **RDP (3389)**

8. Select **Next:Management**.
9. For **Boot diagnostics**, Select **Off**.
10. Select **Review + Create**, review the settings on the summary page, and then select **Create**.

Test the firewall

1. First, note the private IP address for **VM-spoke-01** virtual machine.
2. From the Azure portal, connect to the **VM-Onprem** virtual machine.
3. Open a web browser on **VM-Onprem**, and browse to `http://<VM-spoke-01 private IP>`.

You should see the **VM-spoke-01** web page:



4. From the **VM-Onprem** virtual machine, open a remote desktop to **VM-spoke-01** at the private IP address.

Your connection should succeed, and you should be able to sign in.

So now you've verified that the firewall rules are working:

- You can browse web server on the spoke virtual network.
- You can connect to the server on the spoke virtual network using RDP.

Next, change the firewall network rule collection action to **Deny** to verify that the firewall rules work as expected.

1. Select the **AzFW01** firewall.
2. Select **Rules**.
3. Select the **Network rule collection** tab and select the **RCNet01** rule collection.
4. For **Action**, select **Deny**.
5. Select **Save**.

Close any existing remote desktops before testing the changed rules. Now run the tests again. They should all fail this time.

Clean up resources

You can keep your firewall resources for the next tutorial, or if no longer needed, delete the **FW-Hybrid-Test** resource group to delete all firewall-related resources.

Next steps

Next, you can monitor the Azure Firewall logs.

[Tutorial: Monitor Azure Firewall logs](#)

Tutorial: Filter inbound Internet traffic with Azure Firewall DNAT using the Azure portal

2/26/2020 • 5 minutes to read • [Edit Online](#)

You can configure Azure Firewall Destination Network Address Translation (DNAT) to translate and filter inbound Internet traffic to your subnets. When you configure DNAT, the NAT rule collection action is set to **Dnat**. Each rule in the NAT rule collection can then be used to translate your firewall public IP and port to a private IP and port. DNAT rules implicitly add a corresponding network rule to allow the translated traffic. You can override this behavior by explicitly adding a network rule collection with deny rules that match the translated traffic. To learn more about Azure Firewall rule processing logic, see [Azure Firewall rule processing logic](#).

In this tutorial, you learn how to:

- Set up a test network environment
- Deploy a firewall
- Create a default route
- Configure a DNAT rule
- Test the firewall

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

For this tutorial, you create a two peered VNets:

- **VN-Hub** - the firewall is in this VNet.
- **VN-Spoke** - the workload server is in this VNet.

Create a resource group

1. Sign in to the Azure portal at <https://portal.azure.com>.
2. On the Azure portal home page, click **Resource groups**, then click **Add**.
3. For **Resource group name**, type **RG-DNAT-Test**.
4. For **Subscription**, select your subscription.
5. For **Resource group location**, select a location. All subsequent resources that you create must be in the same location.
6. Click **Create**.

Set up the network environment

First, create the VNets and then peer them.

Create the Hub VNet

1. From the Azure portal home page, click **All services**.
2. Under **Networking**, click **Virtual networks**.
3. Click **Add**.
4. For **Name**, type **VN-Hub**.
5. For **Address space**, type **10.0.0.0/16**.

6. For **Subscription**, select your subscription.
7. For **Resource group**, select **Use existing**, and then select **RG-DNAT-Test**.
8. For **Location**, select the same location that you used previously.
9. Under **Subnet**, for **Name** type **AzureFirewallSubnet**.

The firewall will be in this subnet, and the subnet name **must** be AzureFirewallSubnet.

NOTE

The size of the AzureFirewallSubnet subnet is /26. For more information about the subnet size, see [Azure Firewall FAQ](#).

10. For **Address range**, type **10.0.1.0/26**.
11. Use the other default settings, and then click **Create**.

Create a spoke VNet

1. From the Azure portal home page, click **All services**.
2. Under **Networking**, click **Virtual networks**.
3. Click **Add**.
4. For **Name**, type **VN-Spoke**.
5. For **Address space**, type **192.168.0.0/16**.
6. For **Subscription**, select your subscription.
7. For **Resource group**, select **Use existing**, and then select **RG-DNAT-Test**.
8. For **Location**, select the same location that you used previously.
9. Under **Subnet**, for **Name** type **SN-Workload**.

The server will be in this subnet.

10. For **Address range**, type **192.168.1.0/24**.
11. Use the other default settings, and then click **Create**.

Peer the VNets

Now peer the two VNets.

Hub to spoke

1. Click the **VN-Hub** virtual network.
2. Under **Settings**, click **Peerings**.
3. Click **Add**.
4. Type **Peer-HubSpoke** for the name.
5. Select **VN-Spoke** for the virtual network.
6. Click **OK**.

Spoke to hub

1. Click the **VN-Spoke** virtual network.
2. Under **Settings**, click **Peerings**.
3. Click **Add**.

4. Type **Peer-SpokeHub** for the name.
5. Select **VN-Hub** for the virtual network.
6. Click **Allow forwarded traffic**.
7. Click **OK**.

Create a virtual machine

Create a workload virtual machine, and place it in the **SN-Workload** subnet.

1. From the Azure portal home page, click **All services**.
2. Under **Compute**, click **Virtual machines**.
3. Click **Add**, and click **Windows Server**, click **Windows Server 2016 Datacenter**, and then click **Create**.

Basics

1. For **Name**, type **Srv-Workload**.
2. Type a username and password.
3. For **Subscription**, select your subscription.
4. For **Resource group**, click **Use existing**, and then select **RG-DNAT-Test**.
5. For **Location**, select the same location that you used previously.
6. Click **OK**.

Size

1. Choose an appropriate size for a test virtual machine running Windows Server. For example, **B2ms** (8 GB RAM, 16 GB storage).
2. Click **Select**.

Settings

1. Under **Network**, for **Virtual network**, select **VN-Spoke**.
2. For **Subnet**, select **SN-Workload**.
3. Click **Public IP address** and then click **None**.
4. For **Select public inbound ports**, select **No public inbound ports**.
5. Leave the other default settings and click **OK**.

Summary

Review the summary, and then click **Create**. This will take a few minutes to complete.

After deployment finishes, note the private IP address for the virtual machine. It will be used later when you configure the firewall. Click the virtual machine name, and under **Settings**, click **Networking** to find the private IP address.

Deploy the firewall

1. From the portal home page, click **Create a resource**.
2. Click **Networking**, and after **Featured**, click **See all**.
3. Click **Firewall**, and then click **Create**.
4. On the **Create a Firewall** page, use the following table to configure the firewall:

SETTING	VALUE
Name	FW-DNAT-test
Subscription	<your subscription>
Resource group	Use existing: RG-DNAT-Test
Location	Select the same location that you used previously
Choose a virtual network	Use existing: VN-Hub
Public IP address	Create new. The Public IP address must be the Standard SKU type.

- Click **Review + create**.
- Review the summary, and then click **Create** to create the firewall.

This will take a few minutes to deploy.
- After deployment completes, go to the **RG-DNAT-Test** resource group, and click the **FW-DNAT-test** firewall.
- Note the private IP address. You'll use it later when you create the default route.

Create a default route

For the **SN-Workload** subnet, you configure the outbound default route to go through the firewall.

- From the Azure portal home page, click **All services**.
- Under **Networking**, click **Route tables**.
- Click **Add**.
- For **Name**, type **RT-FWroute**.
- For **Subscription**, select your subscription.
- For **Resource group**, select **Use existing**, and select **RG-DNAT-Test**.
- For **Location**, select the same location that you used previously.
- Click **Create**.
- Click **Refresh**, and then click the **RT-FWroute** route table.
- Click **Subnets**, and then click **Associate**.
- Click **Virtual network**, and then select **VN-Spoke**.
- For **Subnet**, click **SN-Workload**.
- Click **OK**.
- Click **Routes**, and then click **Add**.
- For **Route name**, type **FW-DG**.
- For **Address prefix**, type **0.0.0.0/0**.

17. For **Next hop type**, select **Virtual appliance**.

Azure Firewall is actually a managed service, but virtual appliance works in this situation.

18. For **Next hop address**, type the private IP address for the firewall that you noted previously.

19. Click **OK**.

Configure a NAT rule

1. Open the **RG-DNAT-Test**, and click the **FW-DNAT-test** firewall.
2. On the **FW-DNAT-test** page, under **Settings**, click **Rules**.
3. Click **Add NAT rule collection**.
4. For **Name**, type **RC-DNAT-01**.
5. For **Priority**, type **200**.
6. Under **Rules**, for **Name**, type **RL-01**.
7. For **Protocol**, select **TCP**.
8. For **Source Addresses**, type *****.
9. For **Destination Addresses** type the firewall's public IP address.
10. For **Destination ports**, type **3389**.
11. For **Translated Address** type the private IP address for the Srv-Workload virtual machine.
12. For **Translated port**, type **3389**.
13. Click **Add**.

Test the firewall

1. Connect a remote desktop to firewall public IP address. You should be connected to the **Srv-Workload** virtual machine.
2. Close the remote desktop.

Clean up resources

You can keep your firewall resources for the next tutorial, or if no longer needed, delete the **RG-DNAT-Test** resource group to delete all firewall-related resources.

Next steps

In this tutorial, you learned how to:

- Set up a test network environment
- Deploy a firewall
- Create a default route
- Configure a DNAT rule
- Test the firewall

Next, you can monitor the Azure Firewall logs.

[Tutorial: Monitor Azure Firewall logs](#)

Tutorial: Monitor Azure Firewall logs and metrics

1/14/2020 • 4 minutes to read • [Edit Online](#)

You can monitor Azure Firewall using firewall logs. You can also use activity logs to audit operations on Azure Firewall resources. Using metrics, you can view performance counters in the portal.

You can access some of these logs through the portal. Logs can be sent to [Azure Monitor logs](#), Storage, and Event Hubs and analyzed in Azure Monitor logs or by different tools such as Excel and Power BI.

NOTE

This article was recently updated to use the term Azure Monitor logs instead of Log Analytics. Log data is still stored in a Log Analytics workspace and is still collected and analyzed by the same Log Analytics service. We are updating the terminology to better reflect the role of [logs in Azure Monitor](#). See [Azure Monitor terminology changes](#) for details.

In this tutorial, you learn how to:

- Enable logging through the Azure portal
- Enable logging with PowerShell
- View and analyze the activity log
- View and analyze the network and application rule logs
- View metrics

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Prerequisites

Before starting this tutorial, you should read [Azure Firewall logs and metrics](#) for an overview of the diagnostics logs and metrics available for Azure Firewall.

Enable diagnostic logging through the Azure portal

It can take a few minutes for the data to appear in your logs after you complete this procedure to turn on diagnostic logging. If you don't see anything at first, check again in a few more minutes.

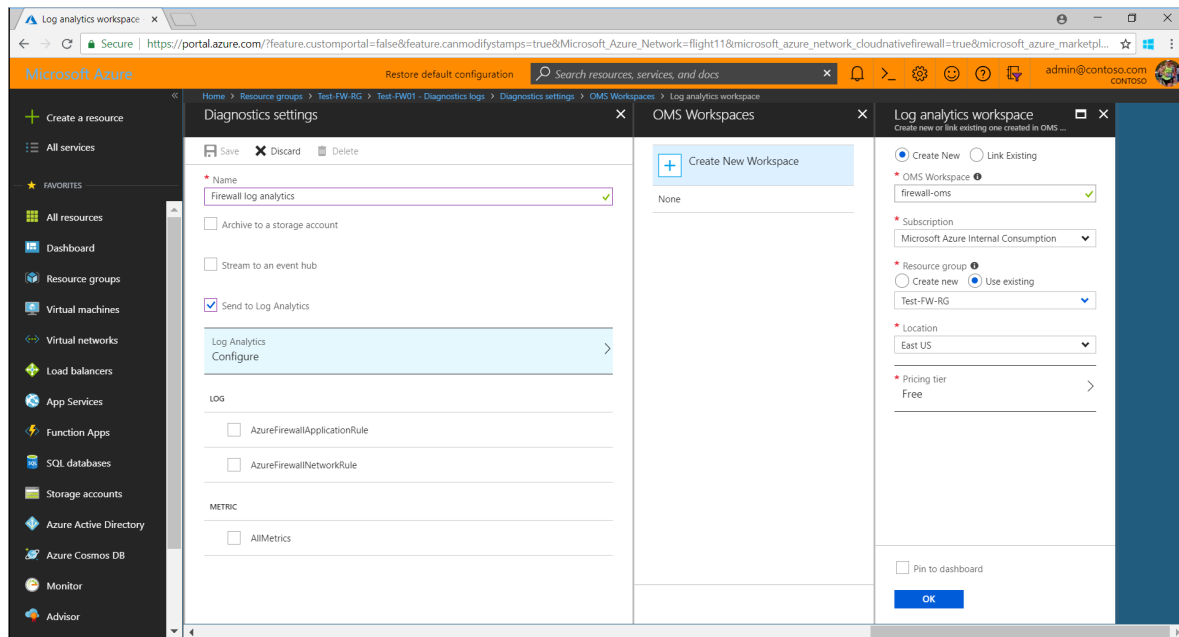
1. In the Azure portal, open your firewall resource group and click the firewall.
2. Under **Monitoring**, click **Diagnostic settings**.

For Azure Firewall, two service-specific logs are available:

- AzureFirewallApplicationRule
- AzureFirewallNetworkRule

3. To start collecting data, click **Turn on diagnostics**.
4. The **Diagnostics settings** page provides the settings for the diagnostic logs.

5. In this example, Azure Monitor logs stores the logs, so type **Firewall log analytics** for the name.
6. Click **Send to Log Analytics** to configure your workspace. You can also use event hubs and a storage account to save the diagnostic logs.
7. Under **Log Analytics**, click **Configure**.
8. In the Log Analytics workspaces page, click **Create New Workspace**.
9. On the **Log analytics workspace** page, type **firewall-oms** for the new **Log Analytics workspace** name.
10. Select your subscription, use the existing firewall resource group (**Test-FW-RG**), select **East US** for the location, and select the **Free** pricing tier.
11. Click **OK**.



OMS workspaces are now referred to as Log Analytics workspaces.

12. Under **Log**, click **AzureFirewallApplicationRule** and **AzureFirewallNetworkRule** to collect logs for application and network rules.

Home > Resource groups > Test-FW-RG > Test-FW01 - Diagnostics logs > Diagnostics settings

Diagnostics settings

Save Discard Delete

* Name
Firewall log analytics ✓

☐ Archive to a storage account

☐ Stream to an event hub

☒ Send to Log Analytics

Log Analytics
firewall-oms >

LOG

☒ AzureFirewallApplicationRule

☒ AzureFirewallNetworkRule

METRIC

☐ AllMetrics

13. Click **Save**.

Enable logging with PowerShell

Activity logging is automatically enabled for every Resource Manager resource. Diagnostic logging must be enabled to start collecting the data available through those logs.

To enable diagnostic logging, use the following steps:

1. Note your storage account's resource ID, where the log data is stored. This value is of the form:
`/subscriptions/<subscriptionId>/resourceGroups/<resource group name>/providers/Microsoft.Storage/storageAccounts/<storage account name>.`

You can use any storage account in your subscription. You can use the Azure portal to find this information. The information is located in the resource **Property** page.

2. Note your Firewall's resource ID for which logging is enabled. This value is of the form:
`/subscriptions/<subscriptionId>/resourceGroups/<resource group`

`name>/providers/Microsoft.Network/azureFirewalls/<Firewall name>.`

You can use the portal to find this information.

3. Enable diagnostic logging by using the following PowerShell cmdlet:

```
Set-AzDiagnosticSetting -ResourceId /subscriptions/<subscriptionId>/resourceGroups/<resource group name>/providers/Microsoft.Network/azureFirewalls/<Firewall name> `
-StorageAccountId /subscriptions/<subscriptionId>/resourceGroups/<resource group name>/providers/Microsoft.Storage/storageAccounts/<storage account name> `
-Enabled $true
```

TIP

Diagnostic logs do not require a separate storage account. The use of storage for access and performance logging incurs service charges.

View and analyze the activity log

You can view and analyze activity log data by using any of the following methods:

- **Azure tools:** Retrieve information from the activity log through Azure PowerShell, the Azure CLI, the Azure REST API, or the Azure portal. Step-by-step instructions for each method are detailed in the [Activity operations with Resource Manager](#) article.
- **Power BI:** If you don't already have a [Power BI](#) account, you can try it for free. By using the [Azure Activity Logs content pack for Power BI](#), you can analyze your data with preconfigured dashboards that you can use as is or customize.

View and analyze the network and application rule logs

[Azure Monitor logs](#) collects the counter and event log files. It includes visualizations and powerful search capabilities to analyze your logs.

For Azure Firewall log analytics sample queries, see [Azure Firewall log analytics samples](#).

You can also connect to your storage account and retrieve the JSON log entries for access and performance logs. After you download the JSON files, you can convert them to CSV and view them in Excel, Power BI, or any other data-visualization tool.

TIP

If you are familiar with Visual Studio and basic concepts of changing values for constants and variables in C#, you can use the [log converter tools](#) available from GitHub.

View metrics

Browse to an Azure Firewall, under **Monitoring** click **Metrics**. To view the available values, select the **METRIC** drop-down list.

Next steps

Now that you've configured your firewall to collect logs, you can explore Azure Monitor logs to view your data.

[Networking monitoring solutions in Azure Monitor logs](#)

Azure Firewall PowerShell samples

11/18/2019 • 2 minutes to read • [Edit Online](#)

The following table includes links to Azure PowerShell script samples that create firewalls:

Create an Azure Firewall and test infrastructure	Creates an Azure Firewall and a test network infrastructure.

Azure Firewall log analytics samples

1/23/2020 • 6 minutes to read • [Edit Online](#)

The following Azure Monitor logs samples can be used to analyze your Azure Firewall logs. The sample file is built in View Designer in Azure Monitor, the [View Designer in Azure Monitor](#) article has more information about the View Design concept.

NOTE

This article was recently updated to use the term Azure Monitor logs instead of Log Analytics. Log data is still stored in a Log Analytics workspace and is still collected and analyzed by the same Log Analytics service. We are updating the terminology to better reflect the role of [logs in Azure Monitor](#). See [Azure Monitor terminology changes](#) for details.

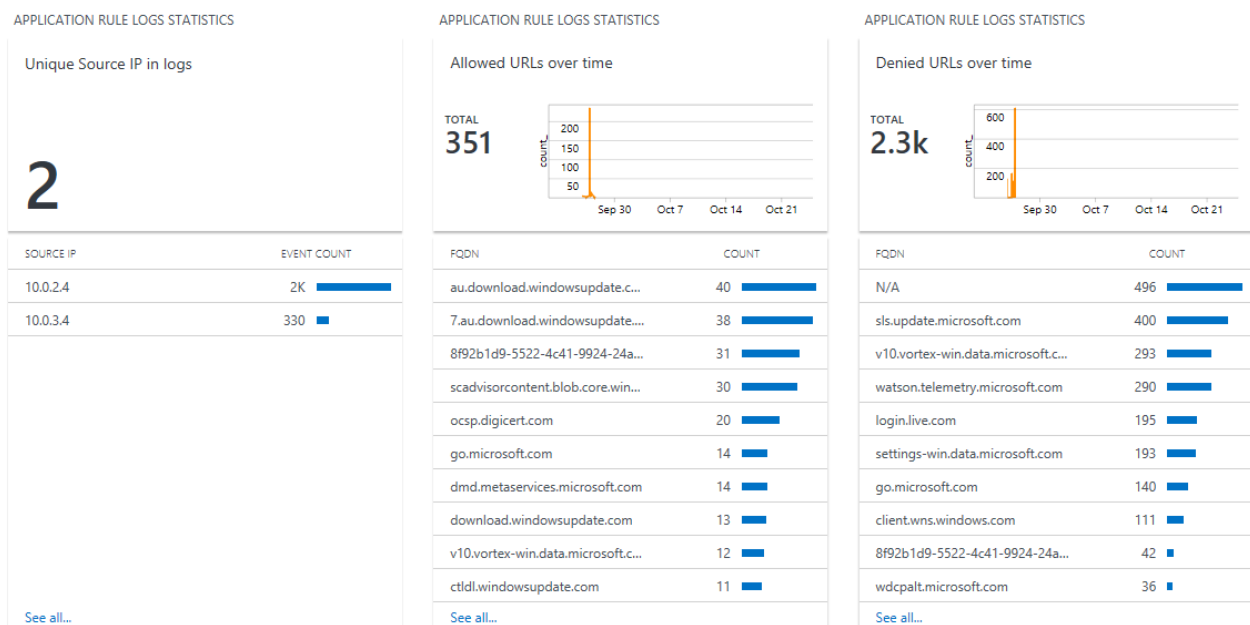
Azure Monitor logs view

Here's how you can configure an example Azure Monitor logs visualization. You can download the example visualization from the [azure-docs-json-samples](#) repository. The easiest way is to right-click the hyperlink on this page and choose *save as* and provide a name like **AzureFirewall.omsview**.

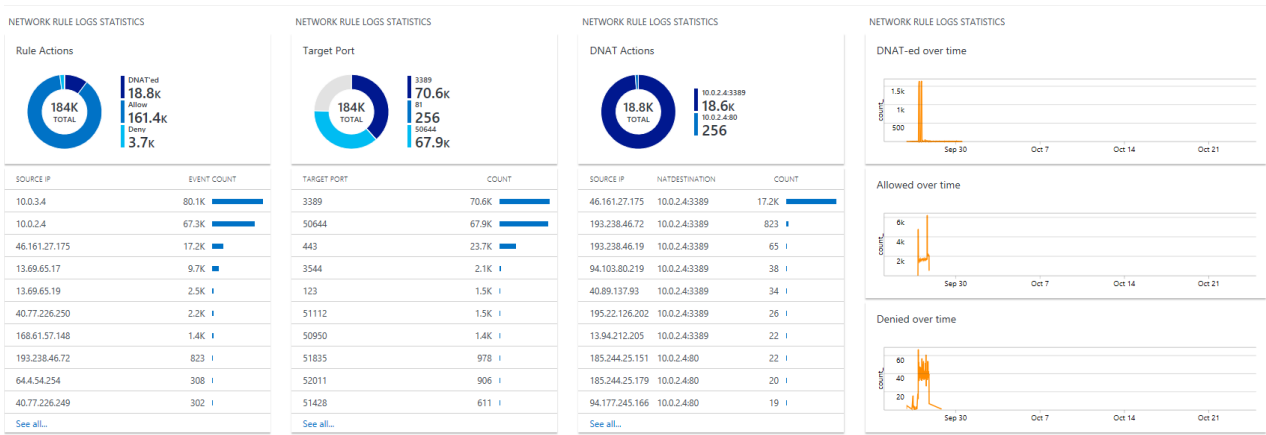
Execute the following steps to add the view to your Log Analytics workspace:

1. Open the Log Analytics workspace in the Azure portal.
2. Open **View Designer** below **General**.
3. Click **Import**.
4. Browse and select the **AzureFirewall.omsview** file you downloaded before.
5. Click **Save**.

Here's how the view looks for the application rule log data:



And for the network rule log data:



Azure Firewall logs data below AzureDiagnostics with Category as either **AzureFirewallApplicationRule** or **AzureFirewallNetworkRule**. The data containing the details is stored in the msg_s field. Using the [parse](#) operator we can extract the various interesting properties from the msg_s field. The queries below extract the information for both categories.

Application rules log data query

The query below parses the application rule log data. In the various comment lines there's some guidance as to how the query was built:

```
AzureDiagnostics
| where Category == "AzureFirewallApplicationRule"
//using :int makes it easier to pars but later we'll convert to string as we're not interested to do
//mathematical functions on these fields
//this first parse statement is valid for all entries as they all start with this format
| parse msg_s with Protocol " request from " SourceIP ":" SourcePortInt:int " " TempDetails
//case 1: for records that end with: "was denied. Reason: SNI TLS extension was missing."
| parse TempDetails with "was " Action1 ". Reason: " Rule1
//case 2: for records that end with
//"to ocsf.digicert.com:80. Action: Allow. Rule Collection: RC1. Rule: Rule1"
//"to v10.vortex-win.data.microsoft.com:443. Action: Deny. No rule matched. Proceeding with default action"
| parse TempDetails with "to " FQDN ":" TargetPortInt:int ". Action: " Action2 "." *
//case 2a: for records that end with:
//"to ocsf.digicert.com:80. Action: Allow. Rule Collection: RC1. Rule: Rule1"
| parse TempDetails with * ". Rule Collection: " RuleCollection2a ". Rule:" Rule2a
//case 2b: for records that end with:
//for records that end with: "to v10.vortex-win.data.microsoft.com:443. Action: Deny. No rule matched.
Proceeding with default action"
| parse TempDetails with * "Deny." RuleCollection2b ". Proceeding with" Rule2b
| extend
SourcePort = tostring(SourcePortInt)
| extend
TargetPort = tostring(TargetPortInt)
| extend
//make sure we only have Allowed / Deny in the Action Field
Action1 = case(Action1 == "Deny","Deny","Unknown Action")
| extend
Action = case(Action2 == "",Action1,Action2),
Rule = case(Rule2a == "",case(Rule1 == "",case(Rule2b == "", "N/A", Rule2b),Rule1),Rule2a),
RuleCollection = case(RuleCollection2b == "",case(RuleCollection2a == "", "No rule
matched",RuleCollection2a),RuleCollection2b),
FQDN = case(FQDN == "", "N/A", FQDN),
TargetPort = case(TargetPort == "", "N/A", TargetPort)
| project TimeGenerated, msg_s, Protocol, SourceIP, SourcePort, FQDN, TargetPort, Action ,RuleCollection, Rule
```

The same query in a more condensed format:

```

AzureDiagnostics
| where Category == "AzureFirewallApplicationRule"
| parse msg_s with Protocol " request from " SourceIP ":" SourcePortInt:int " " TempDetails
| parse TempDetails with "was " Action1 ". Reason: " Rule1
| parse TempDetails with "to " FQDN ":" TargetPortInt:int ". Action: " Action2 "." *
| parse TempDetails with * ". Rule Collection: " RuleCollection2a ". Rule:" Rule2a
| parse TempDetails with * "Deny." RuleCollection2b ". Proceeding with" Rule2b
| extend SourcePort = toString(SourcePortInt)
| extend TargetPort = toString(TargetPortInt)
| extend Action1 = case(Action1 == "Deny","Deny","Unknown Action")
| extend Action = case(Action2 == "",Action1,Action2),Rule = case(Rule2a == "", case(Rule1 == "",case(Rule2b
== "", "N/A", Rule2b),Rule1),Rule2a),
RuleCollection = case(RuleCollection2b == "",case(RuleCollection2a == "", "No rule matched",RuleCollection2a),
RuleCollection2b),FQDN = case(FQDN == "", "N/A", FQDN),TargetPort = case(TargetPort == "", "N/A", TargetPort)
| project TimeGenerated, msg_s, Protocol, SourceIP, SourcePort, FQDN, TargetPort, Action ,RuleCollection, Rule

```

Network rules log data query

The following query parses the network rule log data. In the various comment lines there's some guidance as to how the query was built:

```

AzureDiagnostics
| where Category == "AzureFirewallNetworkRule"
//using :int makes it easier to pars but later we'll convert to string as we're not interested to do
mathematical functions on these fields
//case 1: for records that look like this:
//TCP request from 10.0.2.4:51990 to 13.69.65.17:443. Action: Deny//Allow
//UDP request from 10.0.3.4:123 to 51.141.32.51:123. Action: Deny/Allow
//TCP request from 193.238.46.72:50522 to 40.119.154.83:3389 was DNAT'ed to 10.0.2.4:3389
| parse msg_s with Protocol " request from " SourceIP ":" SourcePortInt:int " to
" TargetIP ":" TargetPortInt:int *
//case 1a: for regular network rules
//TCP request from 10.0.2.4:51990 to 13.69.65.17:443. Action: Deny//Allow
//UDP request from 10.0.3.4:123 to 51.141.32.51:123. Action: Deny/Allow
| parse msg_s with * ". Action: " Action1a
//case 1b: for NAT rules
//TCP request from 193.238.46.72:50522 to 40.119.154.83:3389 was DNAT'ed to 10.0.2.4:3389
| parse msg_s with * " was " Action1b " to " NatDestination
//case 2: for ICMP records
//ICMP request from 10.0.2.4 to 10.0.3.4. Action: Allow
| parse msg_s with Protocol2 " request from " SourceIP2 " to " TargetIP2 ". Action: " Action2
| extend
SourcePort = toString(SourcePortInt),
TargetPort = toString(TargetPortInt)
| extend
    Action = case(Action1a == "", case(Action1b == "",Action2,Action1b), Action1a),
    Protocol = case(Protocol == "", Protocol2, Protocol),
    SourceIP = case(SourceIP == "", SourceIP2, SourceIP),
    TargetIP = case(TargetIP == "", TargetIP2, TargetIP),
    //ICMP records don't have port information
    SourcePort = case(SourcePort == "", "N/A", SourcePort),
    TargetPort = case(TargetPort == "", "N/A", TargetPort),
    //Regular network rules don't have a DNAT destination
    NatDestination = case(NatDestination == "", "N/A", NatDestination)
| project TimeGenerated, msg_s, Protocol, SourceIP,SourcePort,TargetIP,TargetPort,Action, NatDestination

```

The same query in a more condensed format:

```
AzureDiagnostics
| where Category == "AzureFirewallNetworkRule"
| parse msg_s with Protocol " request from " SourceIP ":" SourcePortInt:int " to
" TargetIP ":" TargetPortInt:int *
| parse msg_s with * ". Action: " Action1a
| parse msg_s with * " was " Action1b " to " NatDestination
| parse msg_s with Protocol2 " request from " SourceIP2 " to " TargetIP2 ". Action: " Action2
| extend SourcePort = toString(SourcePortInt),TargetPort = toString(TargetPortInt)
| extend Action = case(Action1a == "", case(Action1b == "",Action2,Action1b), Action1a),Protocol =
case(Protocol == "", Protocol2, Protocol),SourceIP = case(SourceIP == "", SourceIP2, SourceIP),TargetIP =
case(TargetIP == "", TargetIP2, TargetIP),SourcePort = case(SourcePort == "", "N/A", SourcePort),TargetPort =
case(TargetPort == "", "N/A", TargetPort),NatDestination = case(NatDestination == "", "N/A", NatDestination)
| project TimeGenerated, msg_s, Protocol, SourceIP,SourcePort,TargetIP,TargetPort,Action, NatDestination
```


Threat Intelligence log data query

The following query parses the Threat Intelligence rule log data:

```
AzureDiagnostics
| where OperationName == "AzureFirewallThreatIntelLog"
| parse msg_s with Protocol " request from " SourceIP ":" SourcePortInt:int " to " TargetIP ":"
TargetPortInt:int *
| parse msg_s with * ". Action: " Action "." Message
| parse msg_s with Protocol2 " request from " SourceIP2 " to " TargetIP2 ". Action: " Action2
| extend SourcePort = toString(SourcePortInt),TargetPort = toString(TargetPortInt)
| extend Protocol = case(Protocol == "", Protocol2, Protocol),SourceIP = case(SourceIP == "", SourceIP2,
SourceIP),TargetIP = case(TargetIP == "", TargetIP2, TargetIP),SourcePort = case(SourcePort == "", "N/A",
SourcePort),TargetPort = case(TargetPort == "", "N/A", TargetPort)
| sort by TimeGenerated desc | project TimeGenerated, msg_s, Protocol,
SourceIP,SourcePort,TargetIP,TargetPort,Action,Message
```

Sample logs

The following log samples show the data included in a log entry.

	1/21/2020, 8:41:49.493 PM	56512	ctldl.windowsupdate.com	80	Allow	AppC...
	TimeGenerated [UTC]	2020-01-21T20:41:49.493Z				
	msg_s	HTTP request from 10.1.2.4:56512 to ctldl.windowsupdate.com:80. Action: Allow. Rule Collection: AppCollection1. Rule: WindowsUpdate				
	Protocol	HTTP				
	SourceIP	10.1.2.4				
	SourcePort	56512				
	FQDN	ctldl.windowsupdate.com				
	TargetPort	80				
	Action	Allow				
	RuleCollection	AppCollection1				
	Rule	WindowsUpdate				

1/21/2020, 5:04:36.274 PM	TCP	10.6.0.4	52559	192.102.135.246	1688	Deny
TimeGenerated [UTC]	2020-01-21T17:04:36.274Z					
msg_s	TCP request from 10.6.0.4:52559 to 192.102.135.246:1688. Action: Deny					
Protocol	TCP					
SourceIP	10.6.0.4					
SourcePort	52559					
TargetIP	192.102.135.246					
TargetPort	1688					
Action	Deny					
NatDestination	N/A					

1/21/2020, 9:22:29.691 PM	TCP	192.57.40.46	48417	10.253.129.186	80	TCP request from 192.57.40.46:48417 to 10.253.129.186:80
TimeGenerated [UTC]	2020-01-21T17:22:29.691Z					
msg_s	TCP request from 192.57.40.46:48417 to 10.253.129.186:80. Action: Alert. ThreatIntel: Port Scan					
Protocol	TCP					
SourceIP	192.57.40.46					
SourcePort	48417					
TargetIP	10.253.129.186					
TargetPort	80					
Action	Alert					
Message	ThreatIntel: Port Scan					

Next steps

To learn about Azure Firewall monitoring and diagnostics, see [Tutorial: Monitor Azure Firewall logs and metrics](#).

FQDN tags overview

11/18/2019 • 2 minutes to read • [Edit Online](#)

An FQDN tag represents a group of fully qualified domain names (FQDNs) associated with well known Microsoft services. You can use an FQDN tag in application rules to allow the required outbound network traffic through your firewall.

For example, to manually allow Windows Update network traffic through your firewall, you need to create multiple application rules per the Microsoft documentation. Using FQDN tags, you can create an application rule, include the **Windows Updates** tag, and now network traffic to Microsoft Windows Update endpoints can flow through your firewall.

You can't create your own FQDN tags, nor can you specify which FQDNs are included within a tag. Microsoft manages the FQDNs encompassed by the FQDN tag, and updates the tag as FQDNs change.

The following table shows the current FQDN tags you can use. Microsoft maintains these tags and you can expect additional tags to be added periodically.

Current FQDN tags

FQDN TAG	DESCRIPTION
Windows Update	Allow outbound access to Microsoft Update as described in How to Configure a Firewall for Software Updates .
Windows Diagnostics	Allow outbound access to all Windows Diagnostics endpoints .
Microsoft Active Protection Service (MAPS)	Allow outbound access to MAPS .
App Service Environment (ASE)	<p>Allows outbound access to ASE platform traffic. This tag doesn't cover customer-specific Storage and SQL endpoints created by ASE. These should be enabled via Service Endpoints or added manually.</p> <p>For more information about integrating Azure Firewall with ASE, see Locking down an App Service Environment.</p>
Azure Backup	Allows outbound access to the Azure Backup services.
Azure HDInsight	Allows outbound access for HDInsight platform traffic. This tag doesn't cover customer-specific Storage or SQL traffic from HDInsight. Enable these using Service Endpoints or add them manually.

NOTE

When selecting FQDN Tag in an application rule, the protocol:port field must be set to **https**.

Next steps

To learn how to deploy an Azure Firewall, see [Tutorial: Deploy and configure Azure Firewall using the Azure portal](#).

Infrastructure FQDNs

11/18/2019 • 2 minutes to read • [Edit Online](#)

Azure Firewall includes a built-in rule collection for infrastructure FQDNs that are allowed by default. These FQDNs are specific for the platform and can't be used for other purposes.

The following services are included in the built-in rule collection:

- Compute access to storage Platform Image Repository (PIR)
- Managed disks status storage access
- Azure Diagnostics and Logging (MDS)

Overriding

You can override this built-in infrastructure rule collection by creating a deny all application rule collection that is processed last. It will always be processed before the infrastructure rule collection. Anything not in the infrastructure rule collection is denied by default.

Next steps

- Learn how to [deploy and configure an Azure Firewall](#).

Azure Firewall logs and metrics

1/21/2020 • 4 minutes to read • [Edit Online](#)

You can monitor Azure Firewall using firewall logs. You can also use activity logs to audit operations on Azure Firewall resources.

You can access some of these logs through the portal. Logs can be sent to [Azure Monitor logs](#), Storage, and Event Hubs and analyzed in Azure Monitor logs or by different tools such as Excel and Power BI.

Metrics are lightweight and can support near real-time scenarios making them useful for alerting and fast issue detection.

Diagnostic logs

The following diagnostic logs are available for Azure Firewall:

- **Application rule log**

The Application rule log is saved to a storage account, streamed to Event hubs and/or sent to Azure Monitor logs only if you've enabled it for each Azure Firewall. Each new connection that matches one of your configured application rules results in a log for the accepted/denied connection. The data is logged in JSON format, as shown in the following example:

```
Category: application rule logs.
Time: log timestamp.
Properties: currently contains the full message.
note: this field will be parsed to specific fields in the future, while maintaining backward
compatibility with the existing properties field.
```

```
{
  "category": "AzureFirewallApplicationRule",
  "time": "2018-04-16T23:45:04.8295030Z",
  "resourceId":
"/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/{resourceGroupName}/PROVIDERS/MICROSOFT.NETWORK/AZUREFIREWALLS/{resourceName}",
  "operationName": "AzureFirewallApplicationRuleLog",
  "properties": {
    "msg": "HTTPS request from 10.1.0.5:55640 to mydestination.com:443. Action: Allow. Rule
Collection: collection1000. Rule: rule1002"
  }
}
```

- **Network rule log**

The Network rule log is saved to a storage account, streamed to Event hubs and/or sent to Azure Monitor logs only if you've enabled it for each Azure Firewall. Each new connection that matches one of your configured network rules results in a log for the accepted/denied connection. The data is logged in JSON format, as shown in the following example:

Category: network rule logs.

Time: log timestamp.

Properties: currently contains the full message.

note: this field will be parsed to specific fields in the future, while maintaining backward compatibility with the existing properties field.

```
{
  "category": "AzureFirewallNetworkRule",
  "time": "2018-06-14T23:44:11.0590400Z",
  "resourceId":
"/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/{resourceGroupName}/PROVIDERS/MICROSOFT.NETWORK/AZUREFIREWALLS/{resourceName}",
  "operationName": "AzureFirewallNetworkRuleLog",
  "properties": {
    "msg": "TCP request from 111.35.136.173:12518 to 13.78.143.217:2323. Action: Deny"
  }
}
```

You have three options for storing your logs:

- **Storage account:** Storage accounts are best used for logs when logs are stored for a longer duration and reviewed when needed.
- **Event hubs:** Event hubs are a great option for integrating with other security information and event management (SEIM) tools to get alerts on your resources.
- **Azure Monitor logs:** Azure Monitor logs is best used for general real-time monitoring of your application or looking at trends.

Activity logs

Activity log entries are collected by default, and you can view them in the Azure portal.

You can use [Azure activity logs](#) (formerly known as operational logs and audit logs) to view all operations submitted to your Azure subscription.

Metrics

Metrics in Azure Monitor are numerical values that describe some aspect of a system at a particular time. Metrics are collected every minute, and are useful for alerting because they can be sampled frequently. An alert can be fired quickly with relatively simple logic.

The following metrics are available for Azure Firewall:

- **Application rules hit count** - The number of times an application rule has been hit.

Unit: count

- **Network rules hit count** - The number of times a network rule has been hit.

Unit: count

- **Data processed** - Amount of data traversing the firewall.

Unit: bytes

- **Firewall health state** - Indicates the health of the firewall based on SNAT port availability.

Unit: percent

This metric has two dimensions:

- Status: Possible values are *Healthy*, *Degraded*, *Unhealthy*.
- Reason: Indicates the reason for the corresponding status of the firewall.

If SNAT ports are used > 95%, they are considered exhausted and the health is 50% with status=**Degraded** and reason=**SNAT port**. The firewall keeps processing traffic and existing connections are not affected. However, new connections may not be established intermittently.

If SNAT ports are used < 95%, then firewall is considered healthy and health is shown as 100%.

If no SNAT ports usage is reported, health is shown as 0%.

- **SNAT port utilization** - The percentage of SNAT ports that have been utilized by the firewall.

Unit: percent

When you add more public IP addresses to your firewall, more SNAT ports are available, reducing the SNAT ports utilization. Additionally, when the firewall scales out for different reasons (for example, CPU or throughput) additional SNAT ports also become available. So effectively, a given percentage of SNAT ports utilization may go down without you adding any public IP addresses, just because the service scaled out. You can directly control the number of public IP addresses available to increase the ports available on your firewall. But, you can't directly control firewall scaling. Currently, SNAT ports are added only for the first five public IP addresses.

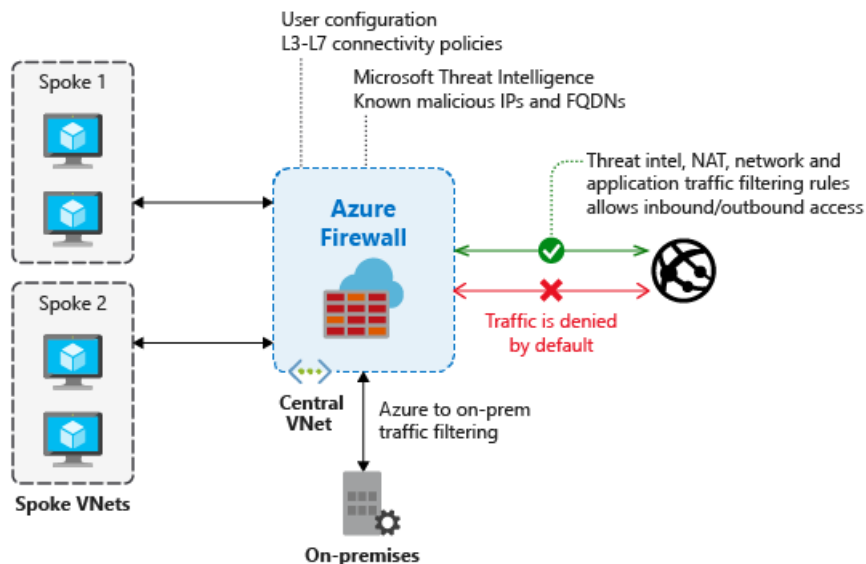
Next steps

- To learn how to monitor Azure Firewall logs and metrics, see [Tutorial: Monitor Azure Firewall logs](#).
- To learn more about metrics in Azure Monitor, see [Metrics in Azure Monitor](#).

Azure Firewall threat intelligence-based filtering

11/18/2019 • 2 minutes to read • [Edit Online](#)

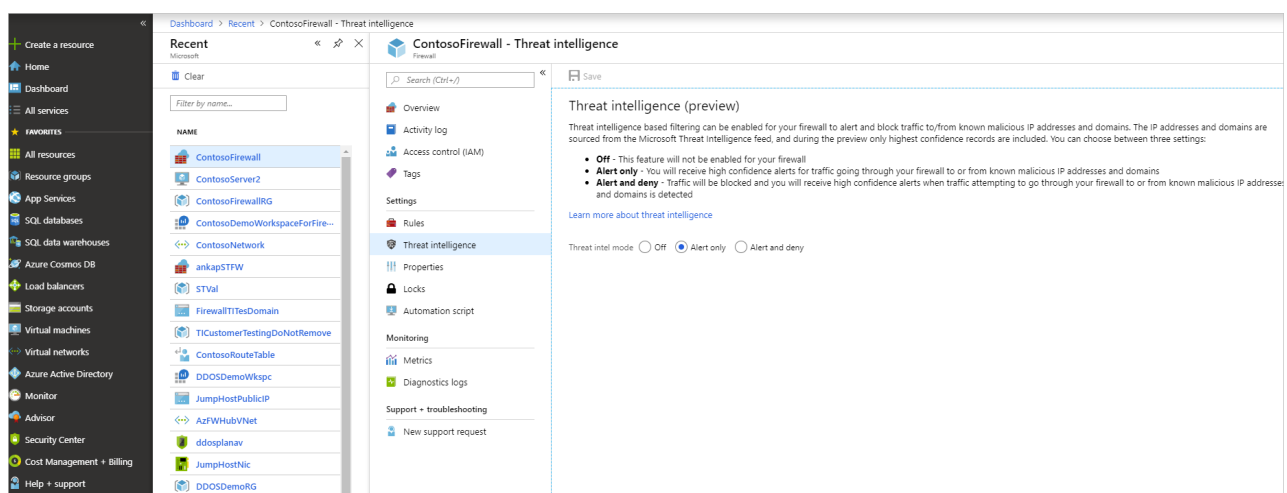
Threat intelligence-based filtering can be enabled for your firewall to alert and deny traffic from/to known malicious IP addresses and domains. The IP addresses and domains are sourced from the Microsoft Threat Intelligence feed. [Intelligent Security Graph](#) powers Microsoft threat intelligence and is used by multiple services including Azure Security Center.



If you've enabled threat intelligence-based filtering, the associated rules are processed before any of the NAT rules, network rules, or application rules.

You can choose to just log an alert when a rule is triggered, or you can choose alert and deny mode.

By default, threat intelligence-based filtering is enabled in alert mode. You can't turn off this feature or change the mode until the portal interface becomes available in your region.



Logs

The following log excerpt shows a triggered rule:

```
{
  "category": "AzureFirewallNetworkRule",
  "time": "2018-04-16T23:45:04.8295030Z",
  "resourceId":
"/SUBSCRIPTIONS/{subscriptionId}/RESOURCEGROUPS/{resourceGroupName}/PROVIDERS/MICROSOFT.NETWORK/AZUREFIREWALLS/
{resourceName}",
  "operationName": "AzureFirewallThreatIntelLog",
  "properties": {
    "msg": "HTTP request from 10.0.0.5:54074 to somemaliciousdomain.com:80. Action: Alert. ThreatIntel:
Bot Networks"
  }
}
```

Testing

- **Outbound testing** - Outbound traffic alerts should be a rare occurrence, as it means that your environment has been compromised. To help test outbound alerts are working, a test FQDN has been created that triggers an alert. Use **testmaliciousdomain.eastus.cloudapp.azure.com** for your outbound tests.
- **Inbound testing** - You can expect to see alerts on incoming traffic if DNAT rules are configured on the firewall. This is true even if only specific sources are allowed on the DNAT rule and traffic is otherwise denied. Azure Firewall doesn't alert on all known port scanners; only on scanners that are known to also engage in malicious activity.

Next steps

- See [Azure Firewall Log Analytics samples](#)
- Learn how to [deploy and configure an Azure Firewall](#)
- Review the [Microsoft Security intelligence report](#)

Azure Firewall rule processing logic

2/26/2020 • 2 minutes to read • [Edit Online](#)

You can configure NAT rules, network rules, and applications rules on Azure Firewall. The rules are processed according to the rule type.

NOTE

If you enable threat intelligence-based filtering, those rules are highest priority and are always processed first. Threat-intelligence filtering may deny traffic before any configured rules are processed. For more information, see [Azure Firewall threat intelligence-based filtering](#).

Outbound

Network rules and applications rules

If you configure network rules and application rules, then network rules are applied in priority order before application rules. The rules are terminating. So if a match is found in a network rule, no other rules are processed. If there is no network rule match, and if the protocol is HTTP,HTTPS, or MSSQL, the the packet is then evaluated by the application rules in priority order. If still no match is found, then the packet is evaluated against the [infrastructure rule collection](#). If there is still no match, then the packet is denied by default.

Inbound

NAT rules

Inbound Internet connectivity can be enabled by configuring Destination Network Address Translation (DNAT) as described in [Tutorial: Filter inbound traffic with Azure Firewall DNAT using the Azure portal](#). NAT rules are applied in priority before network rules. If a match is found, an implicit corresponding network rule to allow the translated traffic is added. You can override this behavior by explicitly adding a network rule collection with deny rules that match the translated traffic.

Application rules are not applied for inbound connections. So if you want to filter inbound HTTP/S traffic, you should use Web Application Firewall (WAF). For more information, see [What is Azure Web Application Firewall?](#)

Examples

The following examples show the results of some of these rule combinations.

Example 1

Connection to google.com is allowed because of a matching network rule.

Network rule

- Action: Allow

NAME	PROTOCOL	SOURCE TYPE	SOURCE	DESTINATION TYPE	DESTINATION ADDRESS	DESTINATION PORTS
Allow-web	TCP	IP address	*	IP address	*	80,443

Application rule

- Action: Deny

NAME	SOURCE TYPE	SOURCE	PROTOCOL:PORT	TARGET FQDNS
Deny-google	IP address	*	http:80,https:443	google.com

Result

The connection to google.com is allowed because the packet matches the *Allow-web* network rule. Rule processing stops at this point.

Example 2

SSH traffic is denied because a higher priority *Deny* network rule collection blocks it.

Network rule collection 1

- Name: Allow-collection
- Priority: 200
- Action: Allow

NAME	PROTOCOL	SOURCE TYPE	SOURCE	DESTINATION TYPE	DESTINATION ADDRESS	DESTINATION PORTS
Allow-SSH	TCP	IP address	*	IP address	*	22

Network rule collection 2

- Name: Deny-collection
- Priority: 100
- Action: Deny

NAME	PROTOCOL	SOURCE TYPE	SOURCE	DESTINATION TYPE	DESTINATION ADDRESS	DESTINATION PORTS
Deny-SSH	TCP	IP address	*	IP address	*	22

Result

SSH connections are denied because a higher priority network rule collection blocks it. Rule processing stops at this point.

Next steps

- Learn how to [deploy and configure an Azure Firewall](#).

Azure Firewall service tags

11/18/2019 • 2 minutes to read • [Edit Online](#)

A service tag represents a group of IP address prefixes to help minimize complexity for security rule creation. You cannot create your own service tag, nor specify which IP addresses are included within a tag. Microsoft manages the address prefixes encompassed by the service tag, and automatically updates the service tag as addresses change.

Azure Firewall service tags can be used in the network rules destination field. You can use them in place of specific IP addresses.

Supported service tags

See [Security groups](#) for a list of service tags that are available for use in Azure firewall network rules.

Next steps

To learn more about Azure Firewall rules, see [Azure Firewall rule processing logic](#).

IP Groups (preview) in Azure Firewall

2/18/2020 • 2 minutes to read • [Edit Online](#)

IMPORTANT

This public preview is provided without a service level agreement and should not be used for production workloads. Certain features may not be supported, may have constrained capabilities, or may not be available in all Azure locations. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

IP Groups allow you to group and manage IP addresses for Azure Firewall rules in the following ways:

- As a source address in DNAT rules
- As a source or destination address in network rules
- As a source address in application rules

An IP Group can have a single IP address, multiple IP addresses, or one or more IP address ranges.

IP Groups can be reused in Azure Firewall DNAT, network, and application rules for multiple firewalls across regions and subscriptions in Azure. Group names must be unique. You can configure an IP Group in the Azure portal, Azure CLI, or REST API. A sample template is provided to help you get started.

Sample format

The following IPv4 address format examples are valid to use in IP Groups:

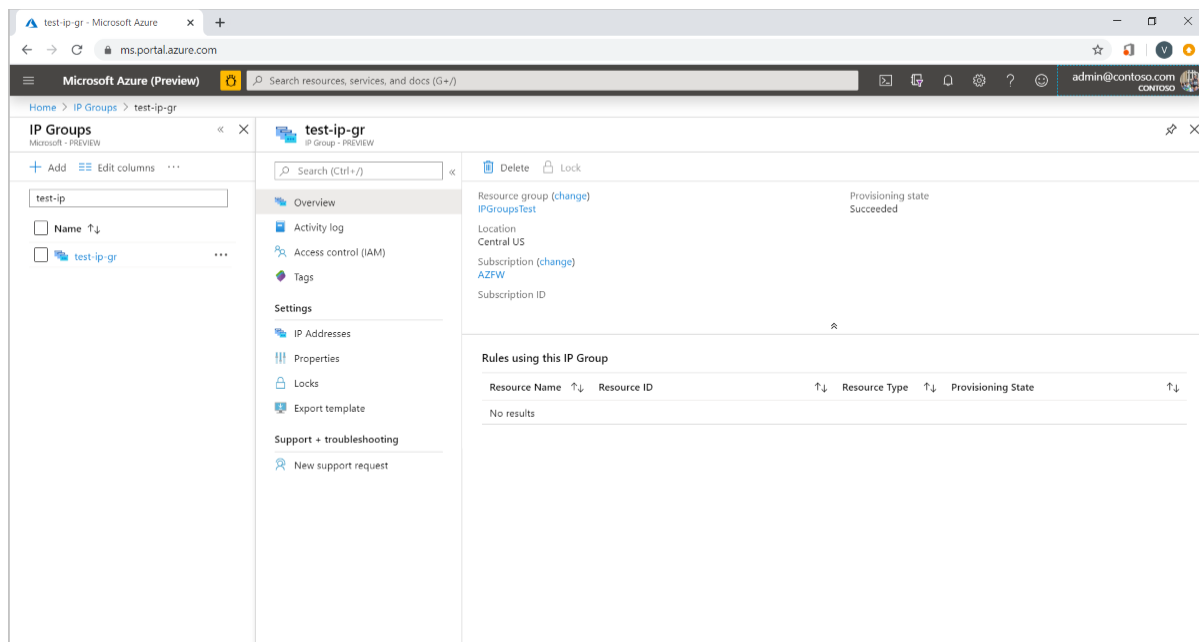
- Single address: 10.0.0.0
- CIDR notation: 10.1.0.0/32
- Address range: 10.2.0.0-10.2.0.31

Create an IP Group

An IP Group can be created using the Azure portal, Azure CLI, or REST API. For more information, see [Create an IP Group \(preview\)](#).

Browse IP Groups

1. In the Azure portal search bar, type **IP Groups** and select it. You can see the list of the IP Groups, or you can select **Add** to create a new IP Group.
2. Select an IP Group to open the overview page. You can edit, add, or delete IP addresses or IP Groups.



Manage an IP Group

You can see all the IP addresses in the IP Group and the rules or resources that are associated with it. To delete an IP Group, you must first dissociate the IP Group from the resource that is using it.

1. To view or edit the IP addresses, select **IP Addresses** under **Settings** on the left pane.
2. To add a single or multiple IP address(es), select **Add IP Addresses**. This opens the **Drag or Browse** page for an upload, or you can enter the address manually.
3. Selecting the ellipses (...) to the right to edit or delete IP addresses. To edit or delete multiple IP addresses, select the boxes and select **Edit** or **Delete** at the top.
4. Finally, can export the file in the CSV file format.

NOTE

If you delete all the IP addresses in an IP Group while it is still in use in a rule, that rule is skipped.

Use an IP Group

You can now select **IP Group** as a **Source type** or **Destination type** for the IP address(es) when you create Azure Firewall DNAT, application, or network rules.

NOTE

IP Groups are not supported in Firewall Policy. It is currently only supported with traditional firewall rules.

Add application rule collection

Name *

Contoso-RC

✓

Priority *

200

✓

Action *

Allow

▼

Rules

FQDN tags

name	Source type	Source	FQDN tags
	IP address ▼	*, 192.168.10.1, 192.168.10.0/24, 192.1...	0 selected ▼

i FQDN tags may require additional configuration. [Learn more.](#)

Target FQDNs

name	Source type	Source	Protocol:Port	Target FQDNs
Allowed-search ✓	IP Group ▼	test-ip-gr ▼	http,https ✓	bing.com,google.com ✓ ...
	IP address ▼	*, 192.168.10.1, 192.168.10.0/...	http, http:8080, https, mssql:1...	www.microsoft.com, *.micros...

i mssql(Preview): SQL should be enabled in proxy mode. This may require additional configuration. [Learn more.](#)

Add

Region availability

IP Groups are currently available in the following regions:

- West US
- West US 2
- East US
- East US 2
- Central US
- North Central US
- West Central US
- South Central US
- Canada Central
- North Europe
- West Europe
- France Central
- UK South
- Australia East
- Australia Central
- Australia Southeast

Related Azure PowerShell cmdlets

The following Azure PowerShell cmdlets can be used to create and manage IP Groups:

- [New-AzIpGroup](#)
- [Remove-AzIPGroup](#)
- [Get-AzIpGroup](#)
- [Set-AzIpGroup](#)
- [New-AzFirewallNetworkRule](#)

- [New-AzFirewallApplicationRule](#)
- [New-AzFirewallNatRule](#)

Next steps

- Learn how to [deploy and configure an Azure Firewall](#).

Azure Firewall forced tunneling (preview)

2/25/2020 • 2 minutes to read • [Edit Online](#)

You can configure Azure Firewall to route all Internet-bound traffic to a designated next hop instead of going directly to the Internet. For example, you may have an on-premises edge firewall or other network virtual appliance (NVA) to process network traffic before it's passed to the Internet.

IMPORTANT

Azure Firewall forced tunneling is currently in public preview.

This public preview is provided without a service-level agreement and shouldn't be used for production workloads. Certain features might not be supported, might have constrained capabilities, or might not be available in all Azure locations. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

By default, forced tunneling isn't allowed on Azure Firewall to ensure all its outbound Azure dependencies are met. User Defined Route (UDR) configurations on the *AzureFirewallSubnet* that have a default route not going directly to the Internet are disabled.


Forced tunneling configuration

To support forced tunneling, service management traffic is separated from customer traffic. An additional dedicated subnet named *AzureFirewallManagementSubnet* (minimum subnet size /26) is required with its own associated public IP address. The only route allowed on this subnet is a default route to the Internet, and BGP route propagation must be disabled.

If you have a default route advertised via BGP to force traffic to on-premises, you must create the *AzureFirewallSubnet* and *AzureFirewallManagementSubnet* before deploying your firewall and have a UDR with a default route to the Internet, and **Virtual network gateway route propagation** disabled.

Within this configuration, the *AzureFirewallSubnet* can now include routes to any on-premise firewall or NVA to process traffic before it's passed to the Internet. You can also publish these routes via BGP to *AzureFirewallSubnet* if **Virtual network gateway route propagation** is enabled on this subnet.

For example, you can create a default route on the *AzureFirewallSubnet* with your VPN gateway as the next hop to get to your on-premise device. Or you can enable **Virtual network gateway route propagation** to get the appropriate routes to the on-premise network.



RT01 - Configuration

Route table

Save

Discard

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Configuration

Virtual network gateway route propagation *

Disabled

Enabled

Once you configure Azure Firewall to support forced tunneling, you can't undo the configuration. If you remove all other IP configurations on your firewall, the management IP configuration is removed as well and the firewall is deallocated. The public IP address assigned to the management IP configuration can't be removed, but you can assign a different public IP address.

Next steps

- [Tutorial: Deploy and configure Azure Firewall in a hybrid network using the Azure portal](#)

Azure Firewall certifications

2/20/2020 • 2 minutes to read • [Edit Online](#)

Azure Firewall is Payment Card Industry (PCI), Service Organization Controls (SOC), International Organization for Standardization (ISO), ICSA Labs, and HITRUST compliant.

The following certifications are for global Azure and Azure Government.

Global Azure certifications

The following Azure Firewall certifications are for global Azure:

- 23 NYCRR 500
- AFM and DNB (Netherlands)
- AMF and ACPR (France)
- APRA(Australia)
- Argentina PDPA
- Australia IRAP
- CDSA
- CFTC 1.31
- CSA STAR Attestation
- CSA STAR Certification
- CSA STAR Self-Assessment
- Canadian Privacy Laws
- DPP(UK)
- EU ENISA IAF
- EU Model Clauses
- European Banking Authority
- FCA and PRA (UK)
- FERPA (US)
- FFIEC(US)
- FINMA (Switzerland)
- FSA (Denmark)
- GLBA (US)
- Germany C5
- GxP (FDA 21 CFR Part 11)
- HITRUST
- ISO 20000-1:2011
- ISO 22301:2012
- ISO 27001:2013
- ISO 27017:2015
- ISO 27018:2014
- ISO 9001:2015
- Japan My Number Act
- K-ISMS
- KNF(Poland)

- MAS and ABS (Singapore)
- MPAA(US)
- NBB and FSMA (Belgium)
- NEN 7510:2011 (Netherlands)
- NHS IG Toolkit (UK)
- Netherlands BIR 2012
- OSFI(Canada)
- PCI DSS Level 1
- RBI and IRDAI (India)
- SOC 1 Type 2
- SOC 2 Type 2
- SOC 3
- SOX (US)
- Spain DPA
- TISAX
- TruSight
- UK G-Cloud
- WCAG 2.0

Azure Government certifications

The following Azure Firewall certifications are for Azure Government:

- CJIS
- CNSSI 1253
- CSA STAR Attestation
- DFARS
- DoD DISA SRG Level 2
- DoE 10 CFR Part 810
- EAR
- FIPS 140-2
- FedRAMP High
- HITRUST
- IRS 1075
- ITAR
- MARS-E (US)
- NERC
- NIST Cybersecurity Framework
- NIST SP 800-171
- SOC 1 Type 2
- SOC 2 Type 2
- SOC 3
- SOX (US)
- Section 508 VPATs

ICSA Labs Corporate Firewall Certification



ICSA Labs is a leading vendor in third-party testing and certification of security and health IT products, as well as network-connected devices. They measure product compliance, reliability, and performance for most of the world's top technology vendors.

Azure Firewall is the first cloud firewall service to attain the ICSA Labs Corporate Firewall Certification. For the Azure Firewall certification report, see the [ICSA Labs Certification Testing and Audit Report](#). For more information, see the [ICSA Labs Firewall Certification Program](#) page.

Next steps

For more information about Microsoft compliance, see the following information.

- [Microsoft Compliance Guide](#)
- [Overview of Microsoft Azure compliance](#)

Azure Firewall central management

2/18/2020 • 2 minutes to read • [Edit Online](#)

IMPORTANT

Azure Firewall Manager is currently a managed public preview.

This public preview is provided without a service-level agreement and shouldn't be used for production workloads. Certain features might not be supported, might have constrained capabilities, or might not be available in all Azure locations. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

If you manage multiple firewalls, you know that continuously changing firewall rules make it difficult to keep them in sync. Central IT teams need a way to define base firewall policies and enforce them across multiple business units. At the same time, DevOps teams want to create their own local derived firewall policies for better agility.

Azure Firewall Manager Preview can help solve these problems.

Azure Firewall Manager Preview

Azure Firewall Manager Preview is a network security management service that provides central security policy and route management for cloud-based security perimeters. It makes it easy for Enterprise IT teams to centrally define network and application level rules for traffic filtering across multiple Azure Firewall instances. You can span different Azure regions and subscriptions in hub and spoke architectures for traffic governance and protection. It also provides DevOps better agility with derived local firewall security policies that are implemented across organizations.

Firewall policy

A Firewall policy is an Azure resource that contains NAT, network, and application rule collections and Threat Intelligence settings. It's a global resource that can be used across multiple Azure Firewall instances in *Secured Virtual Hubs* and *Hub Virtual Networks*. New policies can be created from scratch or inherited from existing policies. Inheritance allows DevOps to create local firewall policies on top of organization mandated base policy. Policies work across regions and subscriptions.

You can create Firewall Policy and associations with Azure Firewall Manager. However, you can also create and manage a policy using REST API, templates, Azure PowerShell, and CLI. Once you create a policy, you can associate it with a firewall in a virtual WAN hub making it a *Secured Virtual Hub* and/or a firewall in a virtual network making it *Hub Virtual Network*.

Pricing

Policies are billed based on firewall associations. A policy with zero or one firewall association is free of charge. A policy with multiple firewall associations is billed at a fixed rate. For more information, see [Azure Firewall Manager Pricing](#).

Azure Firewall Management partners

The following leading third-party solutions support Azure Firewall central management using standard Azure REST APIs. Each of these solutions has its own unique characteristics and features:

- [AlgoSec CloudFlow](#)
- [Barracuda Cloud Security Guardian](#)

- [Tufin Orca](#)

Next steps

For more information about Azure Firewall Manager Preview, see [What is Azure Firewall Manager Preview?](#)

Deploy and configure Azure Firewall using Azure PowerShell

11/8/2019 • 5 minutes to read • [Edit Online](#)

Controlling outbound network access is an important part of an overall network security plan. For example, you may want to limit access to web sites. Or, you may want to limit the outbound IP addresses and ports that can be accessed.

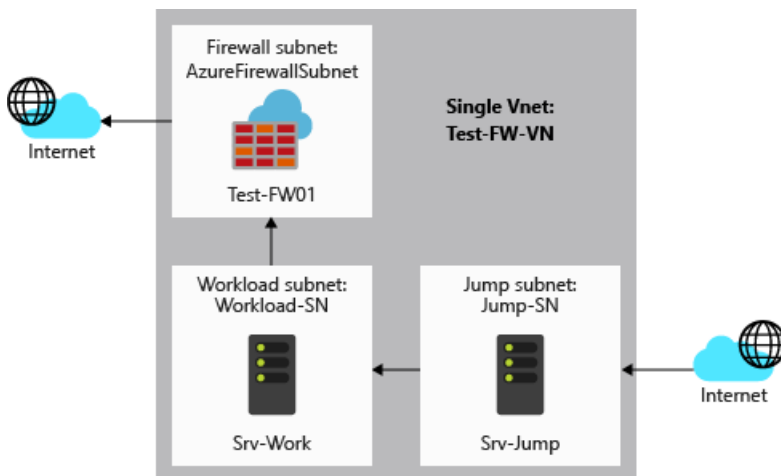
One way you can control outbound network access from an Azure subnet is with Azure Firewall. With Azure Firewall, you can configure:

- Application rules that define fully qualified domain names (FQDNs) that can be accessed from a subnet.
- Network rules that define source address, protocol, destination port, and destination address.

Network traffic is subjected to the configured firewall rules when you route your network traffic to the firewall as the subnet default gateway.

For this article, you create a simplified single VNet with three subnets for easy deployment. For production deployments, a [hub and spoke model](#) is recommended, where the firewall is in its own VNet. The workload servers are in peered VNets in the same region with one or more subnets.

- **AzureFirewallSubnet** - the firewall is in this subnet.
- **Workload-SN** - the workload server is in this subnet. This subnet's network traffic goes through the firewall.
- **Jump-SN** - The "jump" server is in this subnet. The jump server has a public IP address that you can connect to using Remote Desktop. From there, you can then connect to (using another Remote Desktop) the workload server.



In this article, you learn how to:

- Set up a test network environment
- Deploy a firewall
- Create a default route
- Configure an application rule to allow access to www.google.com
- Configure a network rule to allow access to external DNS servers
- Test the firewall

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

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

Prerequisites

This procedure requires that you run PowerShell locally. You must have the Azure PowerShell module installed.

Run `Get-Module -ListAvailable Az` to find the version. If you need to upgrade, see [Install Azure PowerShell module](#). After you verify the PowerShell version, run `Connect-AzAccount` to create a connection with Azure.

Set up the network

First, create a resource group to contain the resources needed to deploy the firewall. Then create a VNet, subnets, and test servers.

Create a resource group

The resource group contains all the resources for the deployment.

```
New-AzResourceGroup -Name Test-FW-RG -Location "East US"
```

Create a VNet

This virtual network has three subnets:

NOTE

The size of the AzureFirewallSubnet subnet is /26. For more information about the subnet size, see [Azure Firewall FAQ](#).

```
$FWsub = New-AzVirtualNetworkSubnetConfig -Name AzureFirewallSubnet -AddressPrefix 10.0.1.0/26
$Worksub = New-AzVirtualNetworkSubnetConfig -Name Workload-SN -AddressPrefix 10.0.2.0/24
$Jumpsub = New-AzVirtualNetworkSubnetConfig -Name Jump-SN -AddressPrefix 10.0.3.0/24
```

Now, create the virtual network:

```
$testVnet = New-AzVirtualNetwork -Name Test-FW-VN -ResourceGroupName Test-FW-RG `
-Location "East US" -AddressPrefix 10.0.0.0/16 -Subnet $FWsub, $Worksub, $Jumpsub
```

Create virtual machines

Now create the jump and workload virtual machines, and place them in the appropriate subnets. When prompted, type a user name and password for the virtual machine.

Create the Srv-Jump virtual machine.

```
New-AzVm `
  -ResourceGroupName Test-FW-RG `
  -Name "Srv-Jump" `
  -Location "East US" `
  -VirtualNetworkName Test-FW-VN `
  -SubnetName Jump-SN `
  -OpenPorts 3389 `
  -Size "Standard_DS2"
```

Create a workload virtual machine with no public IP address. When prompted, type a user name and password for the virtual machine.

```
#Create the NIC
$NIC = New-AzNetworkInterface -Name Srv-work -ResourceGroupName Test-FW-RG `
    -Location "East US" -Subnetid $testVnet.Subnets[1].Id

#Define the virtual machine
$VirtualMachine = New-AzVMConfig -VMName Srv-Work -VMSize "Standard_DS2"
$VirtualMachine = Set-AzVMOperatingSystem -VM $VirtualMachine -Windows -ComputerName Srv-Work -
    ProvisionVMAgent -EnableAutoUpdate
$VirtualMachine = Add-AzVMNetworkInterface -VM $VirtualMachine -Id $NIC.Id
$VirtualMachine = Set-AzVMSourceImage -VM $VirtualMachine -PublisherName 'MicrosoftWindowsServer' -Offer
    'WindowsServer' -Skus '2016-Datacenter' -Version latest

#Create the virtual machine
New-AzVM -ResourceGroupName Test-FW-RG -Location "East US" -VM $VirtualMachine -Verbose
```

Deploy the firewall

Now deploy the firewall into the virtual network.

```
# Get a Public IP for the firewall
$FWpip = New-AzPublicIpAddress -Name "fw-pip" -ResourceGroupName Test-FW-RG `
    -Location "East US" -AllocationMethod Static -Sku Standard
# Create the firewall
$Azfw = New-AzFirewall -Name Test-FW01 -ResourceGroupName Test-FW-RG -Location "East US" -VirtualNetworkName
    Test-FW-VN -PublicIpName fw-pip

#Save the firewall private IP address for future use

$AzfwPrivateIP = $Azfw.IpConfigurations.privateipaddress
$AzfwPrivateIP
```

Note the private IP address. You'll use it later when you create the default route.

Create a default route

Create a table, with BGP route propagation disabled

```
$routeTableDG = New-AzRouteTable `
    -Name Firewall-rt-table `
    -ResourceGroupName Test-FW-RG `
    -location "East US" `
    -DisableBgpRoutePropagation

#Create a route
Add-AzRouteConfig `
    -Name "DG-Route" `
    -RouteTable $routeTableDG `
    -AddressPrefix 0.0.0.0/0 `
    -NextHopType "VirtualAppliance" `
    -NextHopIpAddress $AzfwPrivateIP `
    | Set-AzRouteTable

#Associate the route table to the subnet

Set-AzVirtualNetworkSubnetConfig `
    -VirtualNetwork $testVnet `
    -Name Workload-SN `
    -AddressPrefix 10.0.2.0/24 `
    -RouteTable $routeTableDG | Set-AzVirtualNetwork
```

Configure an application rule

The application rule allows outbound access to www.google.com.

```
$AppRule1 = New-AzFirewallApplicationRule -Name Allow-Google -SourceAddress 10.0.2.0/24 `
  -Protocol http, https -TargetFqdn www.google.com

$AppRuleCollection = New-AzFirewallApplicationRuleCollection -Name App-Coll01 `
  -Priority 200 -ActionType Allow -Rule $AppRule1

$Azfw.ApplicationRuleCollections = $AppRuleCollection

Set-AzFirewall -AzureFirewall $Azfw
```

Azure Firewall includes a built-in rule collection for infrastructure FQDNs that are allowed by default. These FQDNs are specific for the platform and can't be used for other purposes. For more information, see [Infrastructure FQDNs](#).

Configure a network rule

The network rule allows outbound access to two IP addresses at port 53 (DNS).

```
$NetRule1 = New-AzFirewallNetworkRule -Name "Allow-DNS" -Protocol UDP -SourceAddress 10.0.2.0/24 `
  -DestinationAddress 209.244.0.3,209.244.0.4 -DestinationPort 53

$NetRuleCollection = New-AzFirewallNetworkRuleCollection -Name RCNet01 -Priority 200 `
  -Rule $NetRule1 -ActionType "Allow"

$Azfw.NetworkRuleCollections = $NetRuleCollection

Set-AzFirewall -AzureFirewall $Azfw
```

Change the primary and secondary DNS address for the Srv-Work network interface

For testing purposes in this procedure, configure the server's primary and secondary DNS addresses. This isn't a general Azure Firewall requirement.

```
$NIC.DnsSettings.DnsServers.Add("209.244.0.3")
$NIC.DnsSettings.DnsServers.Add("209.244.0.4")
$NIC | Set-AzNetworkInterface
```

Test the firewall

Now, test the firewall to confirm that it works as expected.

1. Note the private IP address for the **Srv-Work** virtual machine:

```
$NIC.IpConfigurations.PrivateIpAddress
```

2. Connect a remote desktop to **Srv-Jump** virtual machine, and sign in. From there, open a remote desktop connection to the **Srv-Work** private IP address and sign in.
3. On **SRV-Work**, open a PowerShell window and run the following commands:

```
nslookup www.google.com
nslookup www.microsoft.com
```

Both commands should return answers, showing that your DNS queries are getting through the firewall.

4. Run the following commands:

```
Invoke-WebRequest -Uri https://www.google.com
Invoke-WebRequest -Uri https://www.google.com

Invoke-WebRequest -Uri https://www.microsoft.com
Invoke-WebRequest -Uri https://www.microsoft.com
```

The `www.google.com` requests should succeed, and the `www.microsoft.com` requests should fail. This demonstrates that your firewall rules are operating as expected.

So now you've verified that the firewall rules are working:

- You can resolve DNS names using the configured external DNS server.
- You can browse to the one allowed FQDN, but not to any others.

Clean up resources

You can keep your firewall resources for the next tutorial, or if no longer needed, delete the **Test-FW-RG** resource group to delete all firewall-related resources:

```
Remove-AzResourceGroup -Name Test-FW-RG
```

Next steps

- [Tutorial: Monitor Azure Firewall logs](#)

Deploy and configure Azure Firewall in a hybrid network using Azure PowerShell

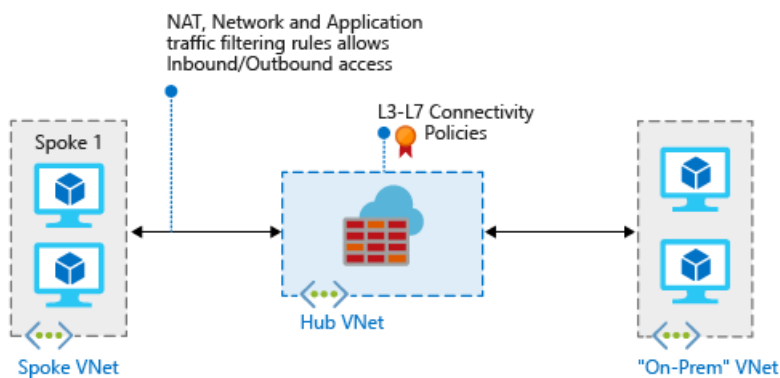
2/19/2020 • 12 minutes to read • [Edit Online](#)

When you connect your on-premises network to an Azure virtual network to create a hybrid network, the ability to control access to your Azure network resources is an important part of an overall security plan.

You can use Azure Firewall to control network access in a hybrid network using rules that define allowed and denied network traffic.

For this article, you create three virtual networks:

- **VNet-Hub** - the firewall is in this virtual network.
- **VNet-Spoke** - the spoke virtual network represents the workload located on Azure.
- **VNet-Onprem** - The on-premises virtual network represents an on-premises network. In an actual deployment, it can be connected by either a VPN or ExpressRoute connection. For simplicity, this article uses a VPN gateway connection, and an Azure-located virtual network is used to represent an on-premises network.



In this article, you learn how to:

- Declare the variables
- Create the firewall hub virtual network
- Create the spoke virtual network
- Create the on-premises virtual network
- Configure and deploy the firewall
- Create and connect the VPN gateways
- Peer the hub and spoke virtual networks
- Create the routes
- Create the virtual machines
- Test the firewall

If you want to use Azure portal instead to complete this tutorial, see [Tutorial: Deploy and configure Azure Firewall in a hybrid network using the Azure portal](#).

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Prerequisites

This article requires that you run PowerShell locally. You must have the Azure PowerShell module installed. Run `Get-Module -ListAvailable Az` to find the version. If you need to upgrade, see [Install Azure PowerShell module](#). After you verify the PowerShell version, run `Login-AzAccount` to create a connection with Azure.

There are three key requirements for this scenario to work correctly:

- A User Defined Route (UDR) on the spoke subnet that points to the Azure Firewall IP address as the default gateway. Virtual network gateway route propagation must be **Disabled** on this route table.
- A UDR on the hub gateway subnet must point to the firewall IP address as the next hop to the spoke networks.

No UDR is required on the Azure Firewall subnet, as it learns routes from BGP.

- Make sure to set **AllowGatewayTransit** when peering VNet-Hub to VNet-Spoke and **UseRemoteGateways** when peering VNet-Spoke to VNet-Hub.

See the [Create Routes](#) section in this article to see how these routes are created.

NOTE

Azure Firewall must have direct Internet connectivity. If your AzureFirewallSubnet learns a default route to your on-premises network via BGP, you must override this with a 0.0.0.0/0 UDR with the **NextHopType** value set as **Internet** to maintain direct Internet connectivity.

Azure Firewall can be configured to support forced tunneling. For more information, see [Azure Firewall forced tunneling](#).

NOTE

Traffic between directly peered VNets is routed directly even if a UDR points to Azure Firewall as the default gateway. To send subnet to subnet traffic to the firewall in this scenario, a UDR must contain the target subnet network prefix explicitly on both subnets.

To review the related Azure PowerShell reference documentation, see [Azure PowerShell Reference](#).

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

Declare the variables

The following example declares the variables using the values for this article. In some cases, you might need to replace some values with your own to work in your subscription. Modify the variables if needed, then copy and paste them into your PowerShell console.

```

$RG1 = "FW-Hybrid-Test"
$Location1 = "East US"

# Variables for the firewall hub VNet

$VNetnameHub = "VNet-hub"
$SNnameHub = "AzureFirewallSubnet"
$VNetHubPrefix = "10.5.0.0/16"
$SNHubPrefix = "10.5.0.0/24"
$SNGWHubPrefix = "10.5.1.0/24"
$GWHubName = "GW-hub"
$GWHubpipName = "VNet-hub-GW-pip"
$GWIPconfNameHub = "GW-ipconf-hub"
$ConnectionNameHub = "hub-to-Onprem"

# Variables for the spoke virtual network

$VnetNameSpoke = "VNet-Spoke"
$SNnameSpoke = "SN-Workload"
$VNetSpokePrefix = "10.6.0.0/16"
$SNSpokePrefix = "10.6.0.0/24"
$SNSpokeGWPrefix = "10.6.1.0/24"

# Variables for the on-premises virtual network

$VNetnameOnprem = "Vnet-Onprem"
$SNNameOnprem = "SN-Corp"
$VNetOnpremPrefix = "192.168.0.0/16"
$SNOntremPrefix = "192.168.1.0/24"
$SNGWOnpremPrefix = "192.168.2.0/24"
$GWOnpremName = "GW-Onprem"
$GWIPconfNameOnprem = "GW-ipconf-Onprem"
$ConnectionNameOnprem = "Onprem-to-hub"
$GWOnprempipName = "VNet-Onprem-GW-pip"

$SNnameGW = "GatewaySubnet"

```

Create the firewall hub virtual network

First, create the resource group to contain the resources for this article:

```
New-AzResourceGroup -Name $RG1 -Location $Location1
```

Define the subnets to be included in the virtual network:

```

$FWsub = New-AzVirtualNetworkSubnetConfig -Name $SNnameHub -AddressPrefix $SNHubPrefix
$GWsub = New-AzVirtualNetworkSubnetConfig -Name $SNnameGW -AddressPrefix $SNGWHubPrefix

```

Now, create the firewall hub virtual network:

```

$VNetHub = New-AzVirtualNetwork -Name $VNetnameHub -ResourceGroupName $RG1 `
-Location $Location1 -AddressPrefix $VNetHubPrefix -Subnet $FWsub,$GWsub

```

Request a public IP address to be allocated to the VPN gateway you'll create for your virtual network. Notice that the *AllocationMethod* is **Dynamic**. You can't specify the IP address that you want to use. It's dynamically allocated to your VPN gateway.

```
$gwpip1 = New-AzPublicIpAddress -Name $GWHubpipName -ResourceGroupName $RG1 `
-Location $Location1 -AllocationMethod Dynamic
```

Create the spoke virtual network

Define the subnets to be included in the spoke virtual network:

```
$Spokesub = New-AzVirtualNetworkSubnetConfig -Name $SNnameSpoke -AddressPrefix $SNSpokePrefix
$GWsubSpoke = New-AzVirtualNetworkSubnetConfig -Name $SNnameGW -AddressPrefix $SNSpokeGWPrefix
```

Create the spoke virtual network:

```
$VNetSpoke = New-AzVirtualNetwork -Name $VnetNameSpoke -ResourceGroupName $RG1 `
-Location $Location1 -AddressPrefix $VNetSpokePrefix -Subnet $Spokesub,$GWsubSpoke
```

Create the on-premises virtual network

Define the subnets to be included in the virtual network:

```
$Onpremsub = New-AzVirtualNetworkSubnetConfig -Name $SNNameOnprem -AddressPrefix $SNOntremPrefix
$GWOnpremsub = New-AzVirtualNetworkSubnetConfig -Name $SNnameGW -AddressPrefix $SNGWOnpremPrefix
```

Now, create the on-premises virtual network:

```
$VNetOnprem = New-AzVirtualNetwork -Name $VNetnameOnprem -ResourceGroupName $RG1 `
-Location $Location1 -AddressPrefix $VNetOnpremPrefix -Subnet $Onpremsub,$GWOnpremsub
```

Request a public IP address to be allocated to the gateway you'll create for the virtual network. Notice that the *AllocationMethod* is **Dynamic**. You can't specify the IP address that you want to use. It's dynamically allocated to your gateway.

```
$gwOnprempip = New-AzPublicIpAddress -Name $GWOnprempipName -ResourceGroupName $RG1 `
-Location $Location1 -AllocationMethod Dynamic
```

Configure and deploy the firewall

Now deploy the firewall into the hub virtual network.

```
# Get a Public IP for the firewall
$FWpip = New-AzPublicIpAddress -Name "fw-pip" -ResourceGroupName $RG1 `
-Location $Location1 -AllocationMethod Static -Sku Standard
# Create the firewall
$Azfw = New-AzFirewall -Name AzFW01 -ResourceGroupName $RG1 -Location $Location1 -VirtualNetworkName
$VNetnameHub -PublicIpName fw-pip

#Save the firewall private IP address for future use

$AzfwPrivateIP = $Azfw.IpConfigurations.privateipaddress
$AzfwPrivateIP
```

Configure network rules

```
$Rule1 = New-AzFirewallNetworkRule -Name "AllowWeb" -Protocol TCP -SourceAddress $SNOnpremPrefix `
-DestinationAddress $VNetSpokePrefix -DestinationPort 80

$Rule2 = New-AzFirewallNetworkRule -Name "AllowRDP" -Protocol TCP -SourceAddress $SNOnpremPrefix `
-DestinationAddress $VNetSpokePrefix -DestinationPort 3389

$NetRuleCollection = New-AzFirewallNetworkRuleCollection -Name RCNet01 -Priority 100 `
-Rule $Rule1,$Rule2 -ActionType "Allow"
$Azfw.NetworkRuleCollections = $NetRuleCollection
Set-AzFirewall -AzureFirewall $Azfw
```

Create and connect the VPN gateways

The hub and on-premises virtual networks are connected via VPN gateways.

Create a VPN gateway for the hub virtual network

Create the VPN gateway configuration. The VPN gateway configuration defines the subnet and the public IP address to use.

```
$vnet1 = Get-AzVirtualNetwork -Name $VNetnameHub -ResourceGroupName $RG1
$subnet1 = Get-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet1
$gwipconf1 = New-AzVirtualNetworkGatewayIpConfig -Name $GWIPconfNameHub `
-Subnet $subnet1 -PublicIpAddress $gwpip1
```

Now create the VPN gateway for the hub virtual network. Network-to-network configurations require a RouteBased VpnType. Creating a VPN gateway can often take 45 minutes or more, depending on the selected VPN gateway SKU.

```
New-AzVirtualNetworkGateway -Name $GWHubName -ResourceGroupName $RG1 `
-Location $Location1 -IpConfigurations $gwipconf1 -GatewayType Vpn `
-VpnType RouteBased -GatewaySku basic
```

Create a VPN gateway for the on-premises virtual network

Create the VPN gateway configuration. The VPN gateway configuration defines the subnet and the public IP address to use.

```
$vnet2 = Get-AzVirtualNetwork -Name $VNetnameOnprem -ResourceGroupName $RG1
$subnet2 = Get-AzVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet2
$gwipconf2 = New-AzVirtualNetworkGatewayIpConfig -Name $GWIPconfNameOnprem `
-Subnet $subnet2 -PublicIpAddress $gwOnprempip
```

Now create the VPN gateway for the on-premises virtual network. Network-to-network configurations require a RouteBased VpnType. Creating a VPN gateway can often take 45 minutes or more, depending on the selected VPN gateway SKU.

```
New-AzVirtualNetworkGateway -Name $GWOnpremName -ResourceGroupName $RG1 `
-Location $Location1 -IpConfigurations $gwipconf2 -GatewayType Vpn `
-VpnType RouteBased -GatewaySku basic
```

Create the VPN connections

Now you can create the VPN connections between the hub and on-premises gateways

Get the VPN gateways

```
$vnetHubgw = Get-AzVirtualNetworkGateway -Name $GWHubName -ResourceGroupName $RG1
$vnetOnpremgw = Get-AzVirtualNetworkGateway -Name $GWOnpremName -ResourceGroupName $RG1
```

Create the connections

In this step, you create the connection from the hub virtual network to the on-premises virtual network. You'll see a shared key referenced in the examples. You can use your own values for the shared key. The important thing is that the shared key must match for both connections. Creating a connection can take a short while to complete.

```
New-AzVirtualNetworkGatewayConnection -Name $ConnectionNameHub -ResourceGroupName $RG1 `
-VirtualNetworkGateway1 $vnetHubgw -VirtualNetworkGateway2 $vnetOnpremgw -Location $Location1 `
-ConnectionType Vnet2Vnet -SharedKey 'AzureA1b2C3'
```

Create the on-premises to hub virtual network connection. This step is similar to the previous one, except you create the connection from VNet-Onprem to VNet-hub. Make sure the shared keys match. The connection will be established after a few minutes.

```
New-AzVirtualNetworkGatewayConnection -Name $ConnectionNameOnprem -ResourceGroupName $RG1 `
-VirtualNetworkGateway1 $vnetOnpremgw -VirtualNetworkGateway2 $vnetHubgw -Location $Location1 `
-ConnectionType Vnet2Vnet -SharedKey 'AzureA1b2C3'
```

Verify the connection

You can verify a successful connection by using the *Get-AzVirtualNetworkGatewayConnection* cmdlet, with or without *-Debug*. Use the following cmdlet example, configuring the values to match your own. If prompted, select **A** to run **All**. In the example, *-Name* refers to the name of the connection that you want to test.

```
Get-AzVirtualNetworkGatewayConnection -Name $ConnectionNameHub -ResourceGroupName $RG1
```

After the cmdlet finishes, view the values. In the following example, the connection status shows as *Connected* and you can see ingress and egress bytes.

```
"connectionStatus": "Connected",
"ingressBytesTransferred": 33509044,
"egressBytesTransferred": 4142431
```

Peer the hub and spoke virtual networks

Now peer the hub and spoke virtual networks.

```
# Peer hub to spoke
Add-AzVirtualNetworkPeering -Name HubtoSpoke -VirtualNetwork $VNetHub -RemoteVirtualNetworkId $VNetSpoke.Id -
AllowGatewayTransit

# Peer spoke to hub
Add-AzVirtualNetworkPeering -Name SpoketoHub -VirtualNetwork $VNetSpoke -RemoteVirtualNetworkId $VNetHub.Id -
AllowForwardedTraffic -UseRemoteGateways
```

Create the routes

Next, create a couple routes:

- A route from the hub gateway subnet to the spoke subnet through the firewall IP address
- A default route from the spoke subnet through the firewall IP address

```

#Create a route table
$routeTableHubSpoke = New-AzRouteTable `
  -Name 'UDR-Hub-Spoke' `
  -ResourceGroupName $RG1 `
  -location $Location1

#Create a route
Get-AzRouteTable `
  -ResourceGroupName $RG1 `
  -Name UDR-Hub-Spoke `
  | Add-AzRouteConfig `
  -Name "ToSpoke" `
  -AddressPrefix $VNetSpokePrefix `
  -NextHopType "VirtualAppliance" `
  -NextHopIpAddress $AzfwPrivateIP `
  | Set-AzRouteTable

#Associate the route table to the subnet

Set-AzVirtualNetworkSubnetConfig `
  -VirtualNetwork $VNetHub `
  -Name $SNnameGW `
  -AddressPrefix $SNGWHubPrefix `
  -RouteTable $routeTableHubSpoke | `
Set-AzVirtualNetwork

#Now create the default route

#Create a table, with BGP route propagation disabled. The property is now called "Virtual network gateway
route propagation," but the API still refers to the parameter as "DisableBgpRoutePropagation."
$routeTableSpokeDG = New-AzRouteTable `
  -Name 'UDR-DG' `
  -ResourceGroupName $RG1 `
  -location $Location1 `
  -DisableBgpRoutePropagation

#Create a route
Get-AzRouteTable `
  -ResourceGroupName $RG1 `
  -Name UDR-DG `
  | Add-AzRouteConfig `
  -Name "ToFirewall" `
  -AddressPrefix 0.0.0.0/0 `
  -NextHopType "VirtualAppliance" `
  -NextHopIpAddress $AzfwPrivateIP `
  | Set-AzRouteTable

#Associate the route table to the subnet

Set-AzVirtualNetworkSubnetConfig `
  -VirtualNetwork $VNetSpoke `
  -Name $SNnameSpoke `
  -AddressPrefix $SNSpokePrefix `
  -RouteTable $routeTableSpokeDG | `
Set-AzVirtualNetwork

```

Create virtual machines

Now create the spoke workload and on-premises virtual machines, and place them in the appropriate subnets.

Create the workload virtual machine

Create a virtual machine in the spoke virtual network, running IIS, with no public IP address, and allows pings in. When prompted, type a user name and password for the virtual machine.

```
# Create an inbound network security group rule for ports 3389 and 80
$nsgRuleRDP = New-AzNetworkSecurityRuleConfig -Name Allow-RDP -Protocol Tcp `
-Direction Inbound -Priority 200 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix
$SNSpokePrefix -DestinationPortRange 3389 -Access Allow
$nsgRuleWeb = New-AzNetworkSecurityRuleConfig -Name Allow-web -Protocol Tcp `
-Direction Inbound -Priority 202 -SourceAddressPrefix * -SourcePortRange * -DestinationAddressPrefix
$SNSpokePrefix -DestinationPortRange 80 -Access Allow

# Create a network security group
$nsg = New-AzNetworkSecurityGroup -ResourceGroupName $RG1 -Location $Location1 -Name NSG-Spoke02 -
SecurityRules $nsgRuleRDP,$nsgRuleWeb

#Create the NIC
$NIC = New-AzNetworkInterface -Name spoke-01 -ResourceGroupName $RG1 -Location $Location1 -SubnetId
$VnetSpoke.Subnets[0].Id -NetworkSecurityGroupId $nsg.Id

#Define the virtual machine
$VirtualMachine = New-AzVMConfig -VMName VM-Spoke-01 -VMSize "Standard_DS2"
$VirtualMachine = Set-AzVMOperatingSystem -VM $VirtualMachine -Windows -ComputerName Spoke-01 -
ProvisionVMAgent -EnableAutoUpdate
$VirtualMachine = Add-AzVMNetworkInterface -VM $VirtualMachine -Id $NIC.Id
$VirtualMachine = Set-AzVMSourceImage -VM $VirtualMachine -PublisherName 'MicrosoftWindowsServer' -Offer
'WindowsServer' -Skus '2016-Datacenter' -Version latest

#Create the virtual machine
New-AzVM -ResourceGroupName $RG1 -Location $Location1 -VM $VirtualMachine -Verbose

#Install IIS on the VM
Set-AzVMExtension `
-ResourceGroupName $RG1 `
-ExtensionName IIS `
-VMName VM-Spoke-01 `
-Publisher Microsoft.Compute `
-ExtensionType CustomScriptExtension `
-TypeHandlerVersion 1.4 `
-SettingString '{"commandToExecute":"powershell Add-WindowsFeature Web-Server"}' `
-Location $Location1
```

Create the on-premises virtual machine

This is a simple virtual machine that you use to connect using Remote Desktop to the public IP address. From there, you then connect to the on-premises server through the firewall. When prompted, type a user name and password for the virtual machine.

```
New-AzVm `
-ResourceGroupName $RG1 `
-Name "VM-Onprem" `
-Location $Location1 `
-VirtualNetworkName $VNetnameOnprem `
-SubnetName $SNNNameOnprem `
-OpenPorts 3389 `
-Size "Standard_DS2"
```

Test the firewall

First, get and then note the private IP address for **VM-spoke-01** virtual machine.

```
$NIC.IpConfigurations.privateipaddress
```

From the Azure portal, connect to the **VM-Onprem** virtual machine.

Open a web browser on **VM-Onprem**, and browse to `http://<VM-spoke-01 private IP>`.

You should see the Internet Information Services default page.

From **VM-Onprem**, open a remote desktop to **VM-spoke-01** at the private IP address.

Your connection should succeed, and you should be able to sign in using your chosen username and password.

So now you've verified that the firewall rules are working:

- You can browse web server on the spoke virtual network.
- You can connect to the server on the spoke virtual network using RDP.

Next, change the firewall network rule collection action to **Deny** to verify that the firewall rules work as expected. Run the following script to change the rule collection action to **Deny**.

```
$rcNet = $azfw.GetNetworkRuleCollectionByName("RCNet01")
$rcNet.action.type = "Deny"

Set-AzFirewall -AzureFirewall $azfw
```

Now run the tests again. They should all fail this time. Close any existing remote desktops before testing the changed rules.

Clean up resources

You can keep your firewall resources for the next tutorial, or if no longer needed, delete the **FW-Hybrid-Test** resource group to delete all firewall-related resources.

Next steps

Next, you can monitor the Azure Firewall logs.

[Tutorial: Monitor Azure Firewall logs](#)

Deploy and configure Azure Firewall using Azure CLI

11/8/2019 • 6 minutes to read • [Edit Online](#)

Controlling outbound network access is an important part of an overall network security plan. For example, you may want to limit access to web sites. Or, you may want to limit the outbound IP addresses and ports that can be accessed.

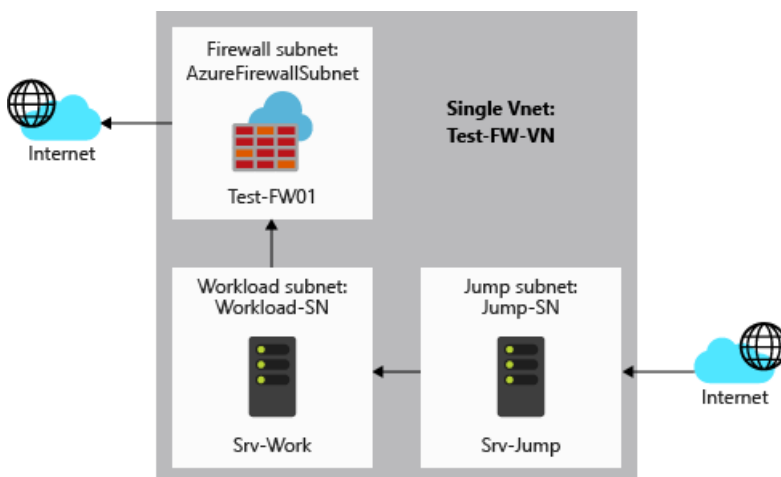
One way you can control outbound network access from an Azure subnet is with Azure Firewall. With Azure Firewall, you can configure:

- Application rules that define fully qualified domain names (FQDNs) that can be accessed from a subnet. The FQDN can also [include SQL instances](#).
- Network rules that define source address, protocol, destination port, and destination address.

Network traffic is subjected to the configured firewall rules when you route your network traffic to the firewall as the subnet default gateway.

For this article, you create a simplified single VNet with three subnets for easy deployment. For production deployments, a [hub and spoke model](#) is recommended. The firewall is in its own VNet. The workload servers are in peered VNets in the same region with one or more subnets.

- **AzureFirewallSubnet** - the firewall is in this subnet.
- **Workload-SN** - the workload server is in this subnet. This subnet's network traffic goes through the firewall.
- **Jump-SN** - The "jump" server is in this subnet. The jump server has a public IP address that you can connect to using Remote Desktop. From there, you can then connect to (using another Remote Desktop) the workload server.



In this article, you learn how to:

- Set up a test network environment
- Deploy a firewall
- Create a default route
- Configure an application rule to allow access to www.google.com
- Configure a network rule to allow access to external DNS servers
- Test the firewall




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

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

Use Azure Cloud Shell

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

To start Azure Cloud Shell:

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

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

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

Prerequisites

Azure CLI

If you choose to install and use the CLI locally, run Azure CLI version 2.0.4 or later. To find the version, run **az --version**. For information about installing or upgrading, see [Install Azure CLI](#).

Install the Azure Firewall extension:

```
az extension add -n azure-firewall
```

Set up the network

First, create a resource group to contain the resources needed to deploy the firewall. Then create a VNet, subnets, and test servers.

Create a resource group

The resource group contains all the resources for the deployment.

```
az group create --name Test-FW-RG --location eastus
```

Create a VNet

This virtual network has three subnets.

NOTE

The size of the AzureFirewallSubnet subnet is /26. For more information about the subnet size, see [Azure Firewall FAQ](#).

```
az network vnet create \  
  --name Test-FW-VN \  
  --resource-group Test-FW-RG \  
  --location eastus \  
  --address-prefix 10.0.0.0/16 \  
  --subnet-name AzureFirewallSubnet \  
  --subnet-prefix 10.0.1.0/26  
az network vnet subnet create \  
  --name Workload-SN \  
  --resource-group Test-FW-RG \  
  --vnet-name Test-FW-VN \  
  --address-prefix 10.0.2.0/24  
az network vnet subnet create \  
  --name Jump-SN \  
  --resource-group Test-FW-RG \  
  --vnet-name Test-FW-VN \  
  --address-prefix 10.0.3.0/24
```

Create virtual machines

Now create the jump and workload virtual machines, and place them in the appropriate subnets. When prompted, type a password for the virtual machine.

Create the Srv-Jump virtual machine.

```
az vm create \  
  --resource-group Test-FW-RG \  
  --name Srv-Jump \  
  --location eastus \  
  --image win2016datacenter \  
  --vnet-name Test-FW-VN \  
  --subnet Jump-SN \  
  --admin-username azureadmin  
az vm open-port --port 3389 --resource-group Test-FW-RG --name Srv-Jump
```

Create a NIC for Srv-Work with specific DNS server IP addresses and no public IP address to test with.

```
az network nic create \  
  -g Test-FW-RG \  
  -n Srv-Work-NIC \  
  --vnet-name Test-FW-VN \  
  --subnet Workload-SN \  
  --public-ip-address "" \  
  --dns-servers 209.244.0.3 209.244.0.4
```

Now create the workload virtual machine. When prompted, type a password for the virtual machine.

```
az vm create \  
  --resource-group Test-FW-RG \  
  --name Srv-Work \  
  --location eastus \  
  --image win2016datacenter \  
  --nics Srv-Work-NIC \  
  --admin-username azureadmin
```

Deploy the firewall

Now deploy the firewall into the virtual network.

```
az network firewall create \  
  --name Test-FW01 \  
  --resource-group Test-FW-RG \  
  --location eastus  
az network public-ip create \  
  --name fw-pip \  
  --resource-group Test-FW-RG \  
  --location eastus \  
  --allocation-method static \  
  --sku standard  
az network firewall ip-config create \  
  --firewall-name Test-FW01 \  
  --name FW-config \  
  --public-ip-address fw-pip \  
  --resource-group Test-FW-RG \  
  --vnet-name Test-FW-VN  
az network firewall update \  
  --name Test-FW01 \  
  --resource-group Test-FW-RG  
az network public-ip show \  
  --name fw-pip \  
  --resource-group Test-FW-RG  
fwprivaddr="$(az network firewall ip-config list -g Test-FW-RG -f Test-FW01 --query "[?name=='FW-config'].privateIpAddress" --output tsv)"
```

Note the private IP address. You'll use it later when you create the default route.

Create a default route

Create a table, with BGP route propagation disabled

```
az network route-table create \  
  --name Firewall-rt-table \  
  --resource-group Test-FW-RG \  
  --location eastus \  
  --disable-bgp-route-propagation true
```

Create the route.

```
az network route-table route create \  
  --resource-group Test-FW-RG \  
  --name DG-Route \  
  --route-table-name Firewall-rt-table \  
  --address-prefix 0.0.0.0/0 \  
  --next-hop-type VirtualAppliance \  
  --next-hop-ip-address $fwprivaddr
```

Associate the route table to the subnet

```
az network vnet subnet update \  
  -n Workload-SN \  
  -g Test-FW-RG \  
  --vnet-name Test-FW-VN \  
  --address-prefixes 10.0.2.0/24 \  
  --route-table Firewall-rt-table
```

Configure an application rule

The application rule allows outbound access to www.google.com.

```
az network firewall application-rule create \  
  --collection-name App-Coll01 \  
  --firewall-name Test-FW01 \  
  --name Allow-Google \  
  --protocols Http=80 Https=443 \  
  --resource-group Test-FW-RG \  
  --target-fqdns www.google.com \  
  --source-addresses 10.0.2.0/24 \  
  --priority 200 \  
  --action Allow
```

Azure Firewall includes a built-in rule collection for infrastructure FQDNs that are allowed by default. These FQDNs are specific for the platform and can't be used for other purposes. For more information, see [Infrastructure FQDNs](#).

Configure a network rule

The network rule allows outbound access to two IP addresses at port 53 (DNS).

```
az network firewall network-rule create \  
  --collection-name Net-Coll01 \  
  --destination-addresses 209.244.0.3 209.244.0.4 \  
  --destination-ports 53 \  
  --firewall-name Test-FW01 \  
  --name Allow-DNS \  
  --protocols UDP \  
  --resource-group Test-FW-RG \  
  --priority 200 \  
  --source-addresses 10.0.2.0/24 \  
  --action Allow
```

Test the firewall

Now, test the firewall to confirm that it works as expected.

1. Note the private IP address for the **Srv-Work** virtual machine:

```
az vm list-ip-addresses \  
-g Test-FW-RG \  
-n Srv-Work
```

2. Connect a remote desktop to **Srv-Jump** virtual machine, and sign in. From there, open a remote desktop connection to the **Srv-Work** private IP address and sign in.
3. On **SRV-Work**, open a PowerShell window and run the following commands:

```
nslookup www.google.com  
nslookup www.microsoft.com
```

Both commands should return answers, showing that your DNS queries are getting through the firewall.

4. Run the following commands:

```
Invoke-WebRequest -Uri https://www.google.com
Invoke-WebRequest -Uri https://www.google.com

Invoke-WebRequest -Uri https://www.microsoft.com
Invoke-WebRequest -Uri https://www.microsoft.com
```

The `www.google.com` requests should succeed, and the `www.microsoft.com` requests should fail. This demonstrates that your firewall rules are operating as expected.

So now you've verified that the firewall rules are working:

- You can resolve DNS names using the configured external DNS server.
- You can browse to the one allowed FQDN, but not to any others.

Clean up resources

You can keep your firewall resources for the next tutorial, or if no longer needed, delete the **Test-FW-RG** resource group to delete all firewall-related resources:

```
az group delete \
-n Test-FW-RG
```

Next steps

- [Tutorial: Monitor Azure Firewall logs](#)

Deploy Azure Firewall using a template

11/18/2019 • 2 minutes to read • [Edit Online](#)

The [Create AzureFirewall sandbox setup template](#) creates a test network environment with a firewall. The network has one virtual network (VNet) with three subnets: *AzureFirewallSubnet*, *ServersSubnet*, and *JumpboxSubnet*. The *ServersSubnet* and *JumpboxSubnet* subnet each have a single, two-core Windows Server virtual machine.

The firewall is in the *AzureFirewallSubnet* subnet, and has an application rule collection with a single rule that allows access to `www.microsoft.com`.

A user-defined route points network traffic from the *ServersSubnet* subnet through the firewall, where the firewall rules are applied.

For more information about Azure Firewall, see [Deploy and configure Azure Firewall using the Azure portal](#).

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see [Introducing the new Azure PowerShell Az module](#). For Az module installation instructions, see [Install Azure PowerShell](#).

Use the template to deploy Azure Firewall

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

To install and deploy Azure Firewall by using the template:

1. Access the template at <https://github.com/Azure/azure-quickstart-templates/tree/master/101-azurefirewall-with-zones-sandbox>.
2. Read the introduction, and when ready to deploy, select **Deploy to Azure**.
3. Sign in to the Azure portal if necessary.
4. In the portal, on the **Create a sandbox setup of AzureFirewall** page, type or select the following values:
 - **Resource group:** Select **Create new**, type a name for the resource group, and select **OK**.
 - **Virtual Network Name:** Type a name for the new VNet.
 - **Admin Username:** Type a username for the administrator user account.
 - **Admin Password:** Type an administrator password.
5. Read the terms and conditions, and then select **I agree to the terms and conditions stated above**.
6. Select **Purchase**.

It will take a few minutes to create the resources.

7. Explore the resources that were created with the firewall.

To learn about the JSON syntax and properties for a firewall in a template, see [Microsoft.Network/azureFirewalls](#).

Clean up resources

When you no longer need them, you can remove the resource group, firewall, and all related resources by running

the [Remove-AzResourceGroup](#) PowerShell command. To remove a resource group named *MyResourceGroup*, run:

```
Remove-AzResourceGroup -Name MyResourceGroup
```

Don't remove the resource group and firewall yet, if you plan to continue on to the firewall monitoring tutorial.

Next steps

Next, you can monitor the Azure Firewall logs:

[Tutorial: Monitor Azure Firewall logs](#)

Deploy an Azure Firewall with multiple public IP addresses using Azure PowerShell

11/19/2019 • 2 minutes to read • [Edit Online](#)

This feature enables the following scenarios:

- **DNAT** - You can translate multiple standard port instances to your backend servers. For example, if you have two public IP addresses, you can translate TCP port 3389 (RDP) for both IP addresses.
- **SNAT** - Additional ports are available for outbound SNAT connections, reducing the potential for SNAT port exhaustion. At this time, Azure Firewall randomly selects the source public IP address to use for a connection. If you have any downstream filtering on your network, you need to allow all public IP addresses associated with your firewall.

Azure Firewall with multiple public IP addresses is available via the Azure portal, Azure PowerShell, Azure CLI, REST, and templates. You can deploy an Azure Firewall with up to 100 public IP addresses.

The following Azure PowerShell examples show how you can configure, add, and remove public IP addresses for Azure Firewall.

NOTE

You can't remove the first ipConfiguration from the Azure Firewall public IP address configuration page. If you want to modify the IP address, you can use Azure PowerShell.

Create a firewall with two or more public IP addresses

This example creates a firewall attached to virtual network *vnet* with two public IP addresses.

```
$rgName = "resourceGroupName"

$vnet = Get-AzVirtualNetwork `
    -Name "vnet" `
    -ResourceGroupName $rgName

$pip1 = New-AzPublicIpAddress `
    -Name "AzFwPublicIp1" `
    -ResourceGroupName "rg" `
    -Sku "Standard" `
    -Location "centralus" `
    -AllocationMethod Static

$pip2 = New-AzPublicIpAddress `
    -Name "AzFwPublicIp2" `
    -ResourceGroupName "rg" `
    -Sku "Standard" `
    -Location "centralus" `
    -AllocationMethod Static

New-AzFirewall `
    -Name "azFw" `
    -ResourceGroupName $rgName `
    -Location centralus `
    -VirtualNetwork $vnet `
    -PublicIpAddress @($pip1, $pip2)
```

Add a public IP address to an existing firewall

In this example, the public IP address *azFwPublicIp1* is attached to the firewall.

```
$pip = New-AzPublicIpAddress `
-Name "azFwPublicIp1" `
-ResourceGroupName "rg" `
-Sku "Standard" `
-Location "centralus" `
-AllocationMethod Static

$azFw = Get-AzFirewall `
-Name "AzureFirewall" `
-ResourceGroupName "rg"

$azFw.AddPublicIpAddress($pip)

$azFw | Set-AzFirewall
```

Remove a public IP address from an existing firewall

In this example, the public IP address *azFwPublicIp1* is detached from the firewall.

```
$pip = Get-AzPublicIpAddress `
-Name "azFwPublicIp1" `
-ResourceGroupName "rg"

$azFw = Get-AzFirewall `
-Name "AzureFirewall" `
-ResourceGroupName "rg"

$azFw.RemovePublicIpAddress($pip)

$azFw | Set-AzFirewall
```

Next steps

- [Tutorial: Monitor Azure Firewall logs](#)

Deploy an Azure Firewall with Availability Zones using Azure PowerShell

11/19/2019 • 2 minutes to read • [Edit Online](#)

Azure Firewall can be configured during deployment to span multiple Availability Zones for increased availability.

This feature enables the following scenarios:

- You can increase availability to 99.99% uptime. For more information, see the Azure Firewall [Service Level Agreement \(SLA\)](#). The 99.99% uptime SLA is offered when two or more Availability Zones are selected.
- You can also associate Azure Firewall to a specific zone just for proximity reasons, using the service standard 99.95% SLA.

For more information about Azure Firewall Availability Zones, see [What is Azure Firewall?](#)

The following Azure PowerShell example shows how you can deploy an Azure Firewall with Availability Zones.

Create a firewall with Availability Zones

This example creates a firewall in zones 1, 2, and 3.

When the standard public IP address is created, no specific zone is specified. This creates a zone-redundant IP address by default. Standard public IP addresses can be configured either in all zones, or a single zone.

It's important to know, because you can't have a firewall in zone 1 and an IP address in zone 2. But you can have a firewall in zone 1 and IP address in all zones, or a firewall and an IP address in the same single zone for proximity purposes.

```
$rgName = "resourceGroupName"

$vnet = Get-AzVirtualNetwork `
  -Name "vnet" `
  -ResourceGroupName $rgName

$pip1 = New-AzPublicIpAddress `
  -Name "AzFwPublicIp1" `
  -ResourceGroupName "rg" `
  -Sku "Standard" `
  -Location "centralus" `
  -AllocationMethod Static

New-AzFirewall `
  -Name "azFw" `
  -ResourceGroupName $rgName `
  -Location centralus `
  -VirtualNetwork $vnet `
  -PublicIpAddress @( $pip1 )
  -Zone 1,2,3
```

Next steps

- [Tutorial: Monitor Azure Firewall logs](#)

Integrate Azure Firewall with Azure Standard Load Balancer

11/18/2019 • 2 minutes to read • [Edit Online](#)

You can integrate an Azure Firewall into a virtual network with an Azure Standard Load Balancer (either public or internal).

The preferred design is to integrate an internal load balancer with your Azure firewall, as this is a much simpler design. You can use a public load balancer if you already have one deployed and you want to keep it in place. However, you need to be aware of an asymmetric routing issue that can break functionality with the public load balancer scenario.

For more information about Azure Load Balancer, see [What is Azure Load Balancer?](#)

Public load balancer

With a public load balancer, the load balancer is deployed with a public frontend IP address.

Asymmetric routing

Asymmetric routing is where a packet takes one path to the destination and takes another path when returning to the source. This issue occurs when a subnet has a default route going to the firewall's private IP address and you're using a public load balancer. In this case, the incoming load balancer traffic is received via its public IP address, but the return path goes through the firewall's private IP address. Since the firewall is stateful, it drops the returning packet because the firewall is not aware of such an established session.

Fix the routing issue

When you deploy an Azure Firewall into a subnet, one step is to create a default route for the subnet directing packets through the firewall's private IP address located on the AzureFirewallSubnet. For more information, see [Tutorial: Deploy and configure Azure Firewall using the Azure portal](#).

When you introduce the firewall into your load balancer scenario, you want your Internet traffic to come in through your firewall's public IP address. From there, the firewall applies its firewall rules and NATs the packets to your load balancer's public IP address. This is where the problem occurs. Packets arrive on the firewall's public IP address, but return to the firewall via the private IP address (using the default route). To avoid this problem, create an additional host route for the firewall's public IP address. Packets going to the firewall's public IP address are routed via the Internet. This avoids taking the default route to the firewall's private IP address.

Configure Azure Firewall application rules with SQL FQDNs

7/18/2019 • 2 minutes to read • [Edit Online](#)

IMPORTANT

Azure Firewall application rules with SQL FQDNs is currently in public preview. This preview version is provided without a service level agreement, and it's not recommended for production workloads. Certain features might not be supported or might have constrained capabilities. For more information, see [Supplemental Terms of Use for Microsoft Azure Previews](#).

You can now configure Azure Firewall application rules with SQL FQDNs. This allows you to limit access from your virtual networks to only the specified SQL server instances.

With SQL FQDNs, you can filter traffic:

- From your VNets to an Azure SQL Database or an Azure SQL Data Warehouse. For example: Only allow access to *sql-server1.database.windows.net*.
- From on-premises to Azure SQL Managed Instances or SQL IaaS running in your VNets.
- From spoke-to-spoke to Azure SQL Managed Instances or SQL IaaS running in your VNets.

During the public preview, SQL FQDN filtering is supported in [proxy-mode](#) only (port 1433). If you use SQL in the default redirect mode, you can filter access using the SQL service tag as part of [network rules](#). If you use non-default ports for SQL IaaS traffic, you can configure those ports in the firewall application rules.

Application rules with SQL FQDNs is currently available in all regions via the Azure portal, Azure CLI, REST, and templates.

Configure using Azure CLI

1. Deploy an [Azure Firewall using Azure CLI](#).
2. If you filter traffic to Azure SQL Database, SQL Data Warehouse, or SQL Managed Instance, ensure the SQL connectivity mode is set to **Proxy**. To learn how to switch SQL connectivity mode, see [Azure SQL Connectivity Architecture](#).

NOTE

SQL *proxy* mode can result in more latency compared to *redirect*. If you want to continue using redirect mode, which is the default for clients connecting within Azure, you can filter access using the SQL [service tag](#) in firewall [network rules](#).

3. Configure an application rule with SQL FQDN to allow access to a SQL server:

```
az extension add -n azure-firewall

az network firewall application-rule create \
-g FWRG \
-f azfirewall \
-c FWAppRules \
-n srule \
--protocols mssql=1433 \
--source-addresses 10.0.0.0/24 \
--target-fqdns sql-srv1.database.windows.net
```

Configure using the Azure portal

1. Deploy an [Azure Firewall using Azure CLI](#).
2. If you filter traffic to Azure SQL Database, SQL Data Warehouse, or SQL Managed Instance, ensure the SQL connectivity mode is set to **Proxy**. To learn how to switch SQL connectivity mode, see [Azure SQL Connectivity Architecture](#).

NOTE

SQL *proxy* mode can result in more latency compared to *redirect*. If you want to continue using redirect mode, which is the default for clients connecting within Azure, you can filter access using the SQL [service tag](#) in firewall [network rules](#).

3. Add the application rule with the appropriate protocol, port, and SQL FQDN and then select **Save**.

Edit application rule collection

Name

* Priority

* Action

Rules

FQDN tags

NAME	SOURCE ADDRESSES	FQDN TAGS
<input type="text"/>	<input type="text" value="*, 192.168.10.1, 192.168.10.0/24, 192.168.10.2 - 192.168..."/>	<input type="text" value="0 selected"/>

i FQDN tags may require additional configuration. [Learn more.](#)

Target FQDNs

NAME	SOURCE ADDRESSES	PROTOCOL:PORT	TARGET FQDNS
<input type="text" value="rc1"/> ✓	<input type="text" value="*"/> ✓	<input type="text" value="mssql:1433"/> ✓	<input type="text" value="sql-srv123.database.windows.net"/> ✓
<input type="text"/>	<input type="text" value="*, 192.168.10.1, 192.168.10.0/24, 192.16..."/>	<input type="text" value="http, http:8080, https, mssql:1433"/>	<input type="text" value="www.microsoft.com, *.microsoft.com"/>

i mssql(Preview): SQL should be enabled in proxy mode. This may require additional configuration. [Learn more.](#)

4. Access SQL from a virtual machine in a VNet that filters the traffic through the firewall.
5. Validate that [Azure Firewall logs](#) show the traffic is allowed.

Next steps

To learn about SQL proxy and redirect modes, see [Azure SQL database connectivity architecture](#).

Azure Firewall SNAT private IP address ranges

2/18/2020 • 2 minutes to read • [Edit Online](#)

Azure Firewall doesn't SNAT when the destination IP address is in a private IP address range per [IANA RFC 1918](#).

If your organization uses a public IP address range for private networks, Azure Firewall will SNAT the traffic to one of the firewall private IP addresses in AzureFirewallSubnet. However, you can configure Azure Firewall to **not** SNAT your public IP address range.

Configure SNAT private IP address ranges

You can use Azure PowerShell to specify an IP address range that the firewall won't SNAT.

New firewall

For a new firewall, the Azure PowerShell command is:

```
New-AzFirewall -Name $GatewayName -ResourceGroupName $RG -Location $Location -VirtualNetworkName $vnet.Name -
PublicIpAddress $LBPIP.Name -PrivateRange @"(\"IANAPrivateRanges\", \"IPRange1\", \"IPRange2\")
```

NOTE

IANAPrivateRanges is expanded to the current defaults on Azure Firewall while the other ranges are added to it.

For more information, see [New-AzFirewall](#).

Existing firewall

To configure an existing firewall, use the following Azure PowerShell commands:

```
$azfw = Get-AzFirewall -ResourceGroupName "Firewall Resource Group name"
$azfw.PrivateRange = @"(\"IANAPrivateRanges\", \"IPRange1\", \"IPRange2\")
Set-AzFirewall -AzureFirewall $azfw
```

Templates

You can add the following to the `additionalProperties` section:

```
"additionalProperties": {
  "Network.SNAT.PrivateRanges": "IANAPrivateRanges , IPRange1, IPRange2"
},
```

Next steps

- Learn how to [deploy and configure an Azure Firewall](#).

Create IP Groups (preview)

2/18/2020 • 2 minutes to read • [Edit Online](#)

IMPORTANT

This public preview is provided without a service level agreement and should not be used for production workloads. Certain features may not be supported, may have constrained capabilities, or may not be available in all Azure locations. See the [Supplemental Terms of Use for Microsoft Azure Previews](#) for details.

IP Groups allow you to group and manage IP addresses for Azure Firewall rules. They can have a single IP address, multiple IP addresses, or one or more IP address ranges.

Create an IP Group

1. From the Azure portal home page, select **Create a resource**.
2. Type **IP Groups** in the search text box, then select **IP Groups**.
3. Select **Create**.
4. Select your subscription.
5. Select a resource group or create a new one.
6. Type a unique name for your IP Group, and then select a region.
7. Select **Next: IP addresses**.
8. Type an IP address, multiple IP addresses, or IP address ranges.

There are two ways to enter IP addresses:

- You can manually enter them
- You can import them from a file

To import from a file, select **Import from a file**. You may either drag your file to the box or select **Browse for files**. If necessary, you can review and edit your uploaded IP addresses.

When you type an IP address, the portal validates it to check for overlapping, duplicates, and formatting issues.

9. When finished, select **Review + Create**.
10. Select **Create**.

Next steps

- [Learn more about IP Groups](#)

Azure Resource Manager overview

12/23/2019 • 5 minutes to read • [Edit Online](#)

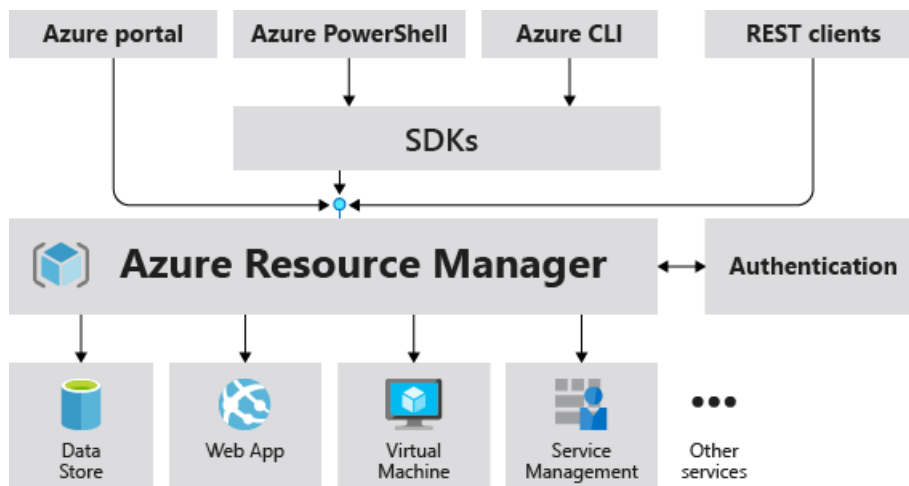
Azure Resource Manager is the deployment and management service for Azure. It provides a management layer that enables you to create, update, and delete resources in your Azure subscription. You use management features, like access control, locks, and tags, to secure and organize your resources after deployment.

To learn about Azure Resource Manager templates, see [Template deployment overview](#).

Consistent management layer

When a user sends a request from any of the Azure tools, APIs, or SDKs, Resource Manager receives the request. It authenticates and authorizes the request. Resource Manager sends the request to the Azure service, which takes the requested action. Because all requests are handled through the same API, you see consistent results and capabilities in all the different tools.

The following image shows the role Azure Resource Manager plays in handling Azure requests.



All capabilities that are available in the portal are also available through PowerShell, Azure CLI, REST APIs, and client SDKs. Functionality initially released through APIs will be represented in the portal within 180 days of initial release.

Terminology

If you're new to Azure Resource Manager, there are some terms you might not be familiar with.

- **resource** - A manageable item that is available through Azure. Virtual machines, storage accounts, web apps, databases, and virtual networks are examples of resources.
- **resource group** - A container that holds related resources for an Azure solution. The resource group includes those resources that you want to manage as a group. You decide which resources belong in a resource group based on what makes the most sense for your organization. See [Resource groups](#).
- **resource provider** - A service that supplies Azure resources. For example, a common resource provider is Microsoft.Compute, which supplies the virtual machine resource. Microsoft.Storage is another common resource provider. See [Resource providers and types](#).
- **Resource Manager template** - A JavaScript Object Notation (JSON) file that defines one or more resources to deploy to a resource group or subscription. The template can be used to deploy the resources consistently and repeatedly. See [Template deployment overview](#).

- **declarative syntax** - Syntax that lets you state "Here is what I intend to create" without having to write the sequence of programming commands to create it. The Resource Manager template is an example of declarative syntax. In the file, you define the properties for the infrastructure to deploy to Azure. See [Template deployment overview](#).

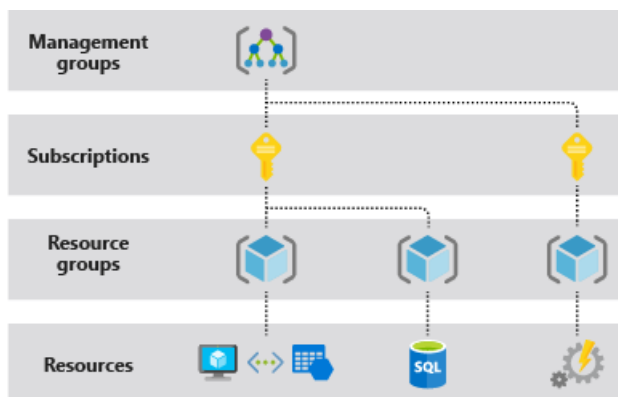
The benefits of using Resource Manager

With Resource Manager, you can:

- Manage your infrastructure through declarative templates rather than scripts.
- Deploy, manage, and monitor all the resources for your solution as a group, rather than handling these resources individually.
- Redeploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state.
- Define the dependencies between resources so they're deployed in the correct order.
- Apply access control to all services in your resource group because Role-Based Access Control (RBAC) is natively integrated into the management platform.
- Apply tags to resources to logically organize all the resources in your subscription.
- Clarify your organization's billing by viewing costs for a group of resources sharing the same tag.

Understand scope

Azure provides four levels of scope: [management groups](#), subscriptions, [resource groups](#), and resources. The following image shows an example of these layers.



You apply management settings at any of these levels of scope. The level you select determines how widely the setting is applied. Lower levels inherit settings from higher levels. For example, when you apply a [policy](#) to the subscription, the policy is applied to all resource groups and resources in your subscription. When you apply a policy on the resource group, that policy is applied the resource group and all its resources. However, another resource group doesn't have that policy assignment.

You can deploy templates to management groups, subscriptions, or resource groups.

Resource groups

There are some important factors to consider when defining your resource group:

- All the resources in your group should share the same lifecycle. You deploy, update, and delete them together. If one resource, such as a database server, needs to exist on a different deployment cycle it should be in another resource group.

- Each resource can only exist in one resource group.
- You can add or remove a resource to a resource group at any time.
- You can move a resource from one resource group to another group. For more information, see [Move resources to new resource group or subscription](#).
- A resource group can contain resources that are located in different regions.
- A resource group can be used to scope access control for administrative actions.
- A resource can interact with resources in other resource groups. This interaction is common when the two resources are related but don't share the same lifecycle (for example, web apps connecting to a database).

When creating a resource group, you need to provide a location for that resource group. You may be wondering, "Why does a resource group need a location? And, if the resources can have different locations than the resource group, why does the resource group location matter at all?" The resource group stores metadata about the resources. When you specify a location for the resource group, you're specifying where that metadata is stored. For compliance reasons, you may need to ensure that your data is stored in a particular region.

If the resource group's region is temporarily unavailable, you can't update resources in the resource group because the metadata is unavailable. The resources in other regions will still function as expected, but you can't update them. For more information about building reliable applications, see [Designing reliable Azure applications](#).

Resiliency of Azure Resource Manager

The Azure Resource Manager service is designed for resiliency and continuous availability. Resource Manager and control plane operations (requests sent to management.azure.com) in the REST API are:

- Distributed across regions. Some services are regional.
- Distributed across Availability Zones (as well regions) in locations that have multiple Availability Zones.
- Not dependent on a single logical data center.
- Never taken down for maintenance activities.

This resiliency applies to services that receive requests through Resource Manager. For example, Key Vault benefits from this resiliency.

Next steps

- For all the operations offered by resource providers, see the [Azure REST APIs](#).
- To learn about moving resources, see [Move resources to new resource group or subscription](#).
- To learn about tagging resources, see [Use tags to organize your Azure resources](#).
- To learn about locking resources, see [Lock resources to prevent unexpected changes](#).
- For information about creating templates for deployments, see [Template deployment overview](#).

Azure Firewall FAQ

2/26/2020 • 9 minutes to read • [Edit Online](#)

What is Azure Firewall?

Azure Firewall is a managed, cloud-based network security service that protects your Azure Virtual Network resources. It's a fully stateful firewall-as-a-service with built-in high availability and unrestricted cloud scalability. You can centrally create, enforce, and log application and network connectivity policies across subscriptions and virtual networks.

What capabilities are supported in Azure Firewall?

- Stateful firewall as a service
- Built-in high availability with unrestricted cloud scalability
- FQDN filtering
- FQDN tags
- Network traffic filtering rules
- Outbound SNAT support
- Inbound DNAT support
- Centrally create, enforce, and log application and network connectivity policies across Azure subscriptions and VNets
- Fully integrated with Azure Monitor for logging and analytics

What is the typical deployment model for Azure Firewall?

You can deploy Azure Firewall on any virtual network, but customers typically deploy it on a central virtual network and peer other virtual networks to it in a hub-and-spoke model. You can then set the default route from the peered virtual networks to point to this central firewall virtual network. Global VNet peering is supported, but it isn't recommended because of potential performance and latency issues across regions. For best performance, deploy one firewall per region.

The advantage of this model is the ability to centrally exert control on multiple spoke VNets across different subscriptions. There are also cost savings as you don't need to deploy a firewall in each VNet separately. The cost savings should be measured versus the associated peering cost based on the customer traffic patterns.

How can I install the Azure Firewall?

You can set up Azure Firewall by using the Azure portal, PowerShell, REST API, or by using templates. See [Tutorial: Deploy and configure Azure Firewall using the Azure portal](#) for step-by-step instructions.

What are some Azure Firewall concepts?

Azure Firewall supports rules and rule collections. A rule collection is a set of rules that share the same order and priority. Rule collections are executed in order of their priority. Network rule collections are higher priority than application rule collections, and all rules are terminating.

There are three types of rule collections:

- *Application rules*: Configure fully qualified domain names (FQDNs) that can be accessed from a subnet.
- *Network rules*: Configure rules that contain source addresses, protocols, destination ports, and destination

addresses.

- *NAT rules*: Configure DNAT rules to allow incoming Internet connections.

Does Azure Firewall support inbound traffic filtering?

Azure Firewall supports inbound and outbound filtering. Inbound protection is typically used for non-HTTP/S protocols. For example RDP, SSH, and FTP protocols. For best inbound HTTP/S protection, use a web application firewall such as [Azure Web Application Firewall \(WAF\)](#).

Which logging and analytics services are supported by the Azure Firewall?

Azure Firewall is integrated with Azure Monitor for viewing and analyzing firewall logs. Logs can be sent to Log Analytics, Azure Storage, or Event Hubs. They can be analyzed in Log Analytics or by different tools such as Excel and Power BI. For more information, see [Tutorial: Monitor Azure Firewall logs](#).

How does Azure Firewall work differently from existing services such as NVAs in the marketplace?

Azure Firewall is a basic firewall service that can address certain customer scenarios. It's expected that you'll have a mix of third-party NVAs and Azure Firewall. Working better together is a core priority.

What is the difference between Application Gateway WAF and Azure Firewall?

The Web Application Firewall (WAF) is a feature of Application Gateway that provides centralized inbound protection of your web applications from common exploits and vulnerabilities. Azure Firewall provides inbound protection for non-HTTP/S protocols (for example, RDP, SSH, FTP), outbound network-level protection for all ports and protocols, and application-level protection for outbound HTTP/S.

What is the difference between Network Security Groups (NSGs) and Azure Firewall?

The Azure Firewall service complements network security group functionality. Together, they provide better "defense-in-depth" network security. Network security groups provide distributed network layer traffic filtering to limit traffic to resources within virtual networks in each subscription. Azure Firewall is a fully stateful, centralized network firewall as-a-service, which provides network- and application-level protection across different subscriptions and virtual networks.

Are Network Security Groups (NSGs) supported on the Azure Firewall subnet?

Azure Firewall is a managed service with multiple protection layers, including platform protection with NIC level NSGs (not viewable). Subnet level NSGs aren't required on the Azure Firewall subnet, and are disabled to ensure no service interruption.

How do I set up Azure Firewall with my service endpoints?

For secure access to PaaS services, we recommend service endpoints. You can choose to enable service endpoints in the Azure Firewall subnet and disable them on the connected spoke virtual networks. This way you benefit from both features-- service endpoint security and central logging for all traffic.

What is the pricing for Azure Firewall?

See [Azure Firewall Pricing](#).

How can I stop and start Azure Firewall?

You can use Azure PowerShell *deallocate* and *allocate* methods.

For example:

```
# Stop an existing firewall

$azfw = Get-AzFirewall -Name "FW Name" -ResourceGroupName "RG Name"
$azfw.Deallocate()
Set-AzFirewall -AzureFirewall $azfw
```

```
# Start a firewall

$azfw = Get-AzFirewall -Name "FW Name" -ResourceGroupName "RG Name"
$vnet = Get-AzVirtualNetwork -ResourceGroupName "RG Name" -Name "VNet Name"
$publicip = Get-AzPublicIpAddress -Name "Public IP Name" -ResourceGroupName "RG Name"
$azfw.Allocate($vnet,$publicip)
Set-AzFirewall -AzureFirewall $azfw
```

NOTE

You must reallocate a firewall and public IP to the original resource group and subscription.

What are the known service limits?

For Azure Firewall service limits, see [Azure subscription and service limits, quotas, and constraints](#).

Can Azure Firewall in a hub virtual network forward and filter network traffic between two spoke virtual networks?

Yes, you can use Azure Firewall in a hub virtual network to route and filter traffic between two spoke virtual network. Subnets in each of the spoke virtual networks must have a UDR pointing to the Azure Firewall as a default gateway for this scenario to work properly.

Can Azure Firewall forward and filter network traffic between subnets in the same virtual network or peered virtual networks?

Yes. However, configuring the UDRs to redirect traffic between subnets in the same VNET requires additional attention. While using the VNET address range as a target prefix for the UDR is sufficient, this also routes all traffic from one machine to another machine in the same subnet through the Azure Firewall instance. To avoid this, include a route for the subnet in the UDR with a next hop type of **VNET**. Managing these routes might be cumbersome and prone to error. The recommended method for internal network segmentation is to use Network Security Groups, which don't require UDRs.

Does Azure Firewall outbound SNAT between private networks?

Azure Firewall doesn't SNAT when the destination IP address is a private IP range per [IANA RFC 1918](#). If your organization uses a public IP address range for private networks, Azure Firewall SNATs the traffic to one of the

firewall private IP addresses in AzureFirewallSubnet. You can configure Azure Firewall to **not** SNAT your public IP address range. For more information, see [Azure Firewall SNAT private IP address ranges](#).

Is forced tunneling/chaining to a Network Virtual Appliance supported?

Forced tunneling is supported. For more information, see [Azure Firewall forced tunneling \(preview\)](#).

Azure Firewall must have direct Internet connectivity. If your AzureFirewallSubnet learns a default route to your on-premises network via BGP, you must override this with a 0.0.0.0/0 UDR with the **NextHopType** value set as **Internet** to maintain direct Internet connectivity.

If your configuration requires forced tunneling to an on-premises network and you can determine the target IP prefixes for your Internet destinations, you can configure these ranges with the on-premises network as the next hop via a user defined route on the AzureFirewallSubnet. Or, you can use BGP to define these routes.

Are there any firewall resource group restrictions?

Yes. The firewall, VNet, and the public IP address all must be in the same resource group.

When configuring DNAT for inbound Internet network traffic, do I also need to configure a corresponding network rule to allow that traffic?

No. NAT rules implicitly add a corresponding network rule to allow the translated traffic. You can override this behavior by explicitly adding a network rule collection with deny rules that match the translated traffic. To learn more about Azure Firewall rule processing logic, see [Azure Firewall rule processing logic](#).

How do wildcards work in an application rule target FQDN?

If you configure ***.contoso.com**, it allows *anyvalue*.contoso.com, but not contoso.com (the domain apex). If you want to allow the domain apex, you must explicitly configure it as a target FQDN.

What does *Provisioning state: Failed* mean?

Whenever a configuration change is applied, Azure Firewall attempts to update all its underlying backend instances. In rare cases, one of these backend instances may fail to update with the new configuration and the update process stops with a failed provisioning state. Your Azure Firewall is still operational, but the applied configuration may be in an inconsistent state, where some instances have the previous configuration where others have the updated rule set. If this happens, try updating your configuration one more time until the operation succeeds and your Firewall is in a *Succeeded* provisioning state.

How does Azure Firewall handle planned maintenance and unplanned failures?

Azure Firewall consists of several backend nodes in an active-active configuration. For any planned maintenance, we have connection draining logic to gracefully update nodes. Updates are planned during non-business hours for each of the Azure regions to further limit risk of disruption. For unplanned issues, we instantiate a new node to replace the failed node. Connectivity to the new node is typically reestablished within 10 seconds from the time of the failure.

Is there a character limit for a firewall name?

Yes. There's a 50 character limit for a firewall name.

Why does Azure Firewall need a /26 subnet size?

Azure Firewall must provision more virtual machine instances as it scales. A /26 address space ensures that the

firewall has enough IP addresses available to accommodate the scaling.

Does the firewall subnet size need to change as the service scales?

No. Azure Firewall doesn't need a subnet bigger than /26.

How can I increase my firewall throughput?

Azure Firewall's initial throughput capacity is 2.5 - 3 Gbps and it scales out to 30 Gbps. It scales out based on CPU usage and throughput. Contact Support to increase your firewall's throughput capacity if your firewall isn't scaling out to meet your needs and you need higher throughput capacity.

How long does it take for Azure Firewall to scale out?

It takes from five to seven minutes for Azure Firewall to scale out. Contact Support to increase your firewall's initial throughput capacity if you have bursts that require a faster autoscale.

Does Azure Firewall allow access to Active Directory by default?

No. Azure Firewall blocks Active Directory access by default. To allow access, configure the AzureActiveDirectory service tag. For more information, see [Azure Firewall service tags](#).

Understand the structure and syntax of Azure Resource Manager templates

2/26/2020 • 13 minutes to read • [Edit Online](#)

This article describes the structure of an Azure Resource Manager template. It presents the different sections of a template and the properties that are available in those sections.

This article is intended for users who have some familiarity with Resource Manager templates. It provides detailed information about the structure of the template. For a step-by-step tutorial that guides you through the process of creating a template, see [Tutorial: Create and deploy your first Azure Resource Manager template](#).

Template format

In its simplest structure, a template has the following elements:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "",
  "apiProfile": "",
  "parameters": { },
  "variables": { },
  "functions": [ ],
  "resources": [ ],
  "outputs": { }
}
```

ELEMENT NAME	REQUIRED	DESCRIPTION
\$schema	Yes	<p>Location of the JSON schema file that describes the version of the template language.</p> <p>For resource group deployments, use:</p> <pre>https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#</pre> <p>For subscription deployments, use:</p> <pre>https://schema.management.azure.com/schemas/2015-01-01/subscriptionDeploymentTemplate.json#</pre>
contentVersion	Yes	<p>Version of the template (such as 1.0.0.0). You can provide any value for this element. Use this value to document significant changes in your template. When deploying resources using the template, this value can be used to make sure that the right template is being used.</p>

ELEMENT NAME	REQUIRED	DESCRIPTION
apiProfile	No	<p>An API version that serves as a collection of API versions for resource types. Use this value to avoid having to specify API versions for each resource in the template. When you specify an API profile version and don't specify an API version for the resource type, Resource Manager uses the API version for that resource type that is defined in the profile.</p> <p>The API profile property is especially helpful when deploying a template to different environments, such as Azure Stack and global Azure. Use the API profile version to make sure your template automatically uses versions that are supported in both environments. For a list of the current API profile versions and the resources API versions defined in the profile, see API Profile.</p> <p>For more information, see Track versions using API profiles.</p>
parameters	No	Values that are provided when deployment is executed to customize resource deployment.
variables	No	Values that are used as JSON fragments in the template to simplify template language expressions.
functions	No	User-defined functions that are available within the template.
resources	Yes	Resource types that are deployed or updated in a resource group or subscription.
outputs	No	Values that are returned after deployment.

Each element has properties you can set. This article describes the sections of the template in greater detail.

Parameters

In the parameters section of the template, you specify which values you can input when deploying the resources. You're limited to 256 parameters in a template. You can reduce the number of parameters by using objects that contain multiple properties.

The available properties for a parameter are:

```

"parameters": {
  "<parameter-name>" : {
    "type" : "<type-of-parameter-value>",
    "defaultValue": "<default-value-of-parameter>",
    "allowedValues": [ "<array-of-allowed-values>" ],
    "minValue": <minimum-value-for-int>,
    "maxValue": <maximum-value-for-int>,
    "minLength": <minimum-length-for-string-or-array>,
    "maxLength": <maximum-length-for-string-or-array-parameters>,
    "metadata": {
      "description": "<description-of-the parameter>"
    }
  }
}

```

ELEMENT NAME	REQUIRED	DESCRIPTION
parameter-name	Yes	Name of the parameter. Must be a valid JavaScript identifier.
type	Yes	Type of the parameter value. The allowed types and values are string , securestring , int , bool , object , secureObject , and array . See Data types .
defaultValue	No	Default value for the parameter, if no value is provided for the parameter.
allowedValues	No	Array of allowed values for the parameter to make sure that the right value is provided.
minValue	No	The minimum value for int type parameters, this value is inclusive.
maxValue	No	The maximum value for int type parameters, this value is inclusive.
minLength	No	The minimum length for string, secure string, and array type parameters, this value is inclusive.
maxLength	No	The maximum length for string, secure string, and array type parameters, this value is inclusive.
description	No	Description of the parameter that is displayed to users through the portal. For more information, see Comments in templates .

For examples of how to use parameters, see [Parameters in Azure Resource Manager templates](#).

Data types

For integers passed as inline parameters, the range of values may be limited by the SDK or command-line tool you use for deployment. For example, when using PowerShell to deploy a template, integer types can range from -2147483648 to 2147483647. To avoid this limitation, specify large integer values in a [parameter file](#). Resource types apply their own limits for integer properties.

When specifying boolean and integer values in your template, don't surround the value with quotation marks. Start and end string values with double quotation marks.

Objects start with a left brace and end with a right brace. Arrays start with a left bracket and end with a right bracket.

Secure strings and secure objects can't be read after resource deployment.

For samples of formatting data types, see [Parameter type formats](#).

Variables

In the variables section, you construct values that can be used throughout your template. You don't need to define variables, but they often simplify your template by reducing complex expressions.

The following example shows the available options for defining a variable:

```
"variables": {
  "<variable-name>": "<variable-value>",
  "<variable-name>": {
    <variable-complex-type-value>
  },
  "<variable-object-name>": {
    "copy": [
      {
        "name": "<name-of-array-property>",
        "count": <number-of-iterations>,
        "input": <object-or-value-to-repeat>
      }
    ]
  },
  "copy": [
    {
      "name": "<variable-array-name>",
      "count": <number-of-iterations>,
      "input": <object-or-value-to-repeat>
    }
  ]
}
```

For information about using `copy` to create several values for a variable, see [Variable iteration](#).

For examples of how to use variables, see [Variables in Azure Resource Manager template](#).

Functions

Within your template, you can create your own functions. These functions are available for use in your template. Typically, you define complicated expressions that you don't want to repeat throughout your template. You create the user-defined functions from expressions and [functions](#) that are supported in templates.

When defining a user function, there are some restrictions:

- The function can't access variables.
- The function can only use parameters that are defined in the function. When you use the [parameters function](#) within a user-defined function, you're restricted to the parameters for that function.
- The function can't call other user-defined functions.
- The function can't use the [reference function](#).
- Parameters for the function can't have default values.

```

"functions": [
  {
    "namespace": "<namespace-for-functions>",
    "members": {
      "<function-name>": {
        "parameters": [
          {
            "name": "<parameter-name>",
            "type": "<type-of-parameter-value>"
          }
        ],
        "output": {
          "type": "<type-of-output-value>",
          "value": "<function-return-value>"
        }
      }
    }
  }
],

```

ELEMENT NAME	REQUIRED	DESCRIPTION
namespace	Yes	Namespace for the custom functions. Use to avoid naming conflicts with template functions.
function-name	Yes	Name of the custom function. When calling the function, combine the function name with the namespace. For example, to call a function named <code>uniqueName</code> in the namespace <code>contoso</code> , use <code>"[contoso.uniqueName()]"</code> .
parameter-name	No	Name of the parameter to be used within the custom function.
parameter-value	No	Type of the parameter value. The allowed types and values are string , securestring , int , bool , object , secureObject , and array .
output-type	Yes	Type of the output value. Output values support the same types as function input parameters.
output-value	Yes	Template language expression that is evaluated and returned from the function.

For examples of how to use custom functions, see [User-defined functions in Azure Resource Manager template](#).

Resources

In the resources section, you define the resources that are deployed or updated.

You define resources with the following structure:

```

"resources": [
{
  "condition": "<true-to-deploy-this-resource>",
  "type": "<resource-provider-namespace/resource-type-name>",
  "apiVersion": "<api-version-of-resource>",
  "name": "<name-of-the-resource>",
  "comments": "<your-reference-notes>",
  "location": "<location-of-resource>",
  "dependsOn": [
    "<array-of-related-resource-names>"
  ],
  "tags": {
    "<tag-name1>": "<tag-value1>",
    "<tag-name2>": "<tag-value2>"
  },
  "sku": {
    "name": "<sku-name>",
    "tier": "<sku-tier>",
    "size": "<sku-size>",
    "family": "<sku-family>",
    "capacity": "<sku-capacity>"
  },
  "kind": "<type-of-resource>",
  "copy": {
    "name": "<name-of-copy-loop>",
    "count": "<number-of-iterations>",
    "mode": "<serial-or-parallel>",
    "batchSize": "<number-to-deploy-serially>"
  },
  "plan": {
    "name": "<plan-name>",
    "promotionCode": "<plan-promotion-code>",
    "publisher": "<plan-publisher>",
    "product": "<plan-product>",
    "version": "<plan-version>"
  },
  "properties": {
    "<settings-for-the-resource>",
    "copy": [
      {
        "name": ,
        "count": ,
        "input": {}
      }
    ]
  },
  "resources": [
    "<array-of-child-resources>"
  ]
}
]

```

ELEMENT NAME	REQUIRED	DESCRIPTION
condition	No	Boolean value that indicates whether the resource will be provisioned during this deployment. When <code>true</code> , the resource is created during deployment. When <code>false</code> , the resource is skipped for this deployment. See condition .

ELEMENT NAME	REQUIRED	DESCRIPTION
type	Yes	Type of the resource. This value is a combination of the namespace of the resource provider and the resource type (such as Microsoft.Storage/storageAccounts). To determine available values, see template reference . For a child resource, the format of the type depends on whether it's nested within the parent resource or defined outside of the parent resource. See Set name and type for child resources .
apiVersion	Yes	Version of the REST API to use for creating the resource. To determine available values, see template reference .
name	Yes	Name of the resource. The name must follow URI component restrictions defined in RFC3986. Azure services that expose the resource name to outside parties validate the name to make sure it isn't an attempt to spoof another identity. For a child resource, the format of the name depends on whether it's nested within the parent resource or defined outside of the parent resource. See Set name and type for child resources .
comments	No	Your notes for documenting the resources in your template. For more information, see Comments in templates .
location	Varies	Supported geo-locations of the provided resource. You can select any of the available locations, but typically it makes sense to pick one that is close to your users. Usually, it also makes sense to place resources that interact with each other in the same region. Most resource types require a location, but some types (such as a role assignment) don't require a location. See Set resource location .
dependsOn	No	Resources that must be deployed before this resource is deployed. Resource Manager evaluates the dependencies between resources and deploys them in the correct order. When resources aren't dependent on each other, they're deployed in parallel. The value can be a comma-separated list of a resource names or resource unique identifiers. Only list resources that are deployed in this template. Resources that aren't defined in this template must already exist. Avoid adding unnecessary dependencies as they can slow your deployment and create circular dependencies. For guidance on setting dependencies, see Defining dependencies in Azure Resource Manager templates .

ELEMENT NAME	REQUIRED	DESCRIPTION
tags	No	Tags that are associated with the resource. Apply tags to logically organize resources across your subscription.
sku	No	Some resources allow values that define the SKU to deploy. For example, you can specify the type of redundancy for a storage account.
kind	No	Some resources allow a value that defines the type of resource you deploy. For example, you can specify the type of Cosmos DB to create.
copy	No	If more than one instance is needed, the number of resources to create. The default mode is parallel. Specify serial mode when you don't want all or the resources to deploy at the same time. For more information, see Create several instances of resources in Azure Resource Manager .
plan	No	Some resources allow values that define the plan to deploy. For example, you can specify the marketplace image for a virtual machine.
properties	No	Resource-specific configuration settings. The values for the properties are the same as the values you provide in the request body for the REST API operation (PUT method) to create the resource. You can also specify a copy array to create several instances of a property. To determine available values, see template reference .
resources	No	Child resources that depend on the resource being defined. Only provide resource types that are permitted by the schema of the parent resource. Dependency on the parent resource isn't implied. You must explicitly define that dependency. See Set name and type for child resources .

Outputs

In the Outputs section, you specify values that are returned from deployment. Typically, you return values from resources that were deployed.

The following example shows the structure of an output definition:

```

"outputs": {
  "<output-name>": {
    "condition": "<boolean-value-whether-to-output-value>",
    "type": "<type-of-output-value>",
    "value": "<output-value-expression>",
    "copy": {
      "count": <number-of-iterations>,
      "input": <values-for-the-variable>
    }
  }
}

```

ELEMENT NAME	REQUIRED	DESCRIPTION
output-name	Yes	Name of the output value. Must be a valid JavaScript identifier.
condition	No	Boolean value that indicates whether this output value is returned. When <code>true</code> , the value is included in the output for the deployment. When <code>false</code> , the output value is skipped for this deployment. When not specified, the default value is <code>true</code> .
type	Yes	Type of the output value. Output values support the same types as template input parameters. If you specify securestring for the output type, the value isn't displayed in the deployment history and can't be retrieved from another template. To use a secret value in more than one template, store the secret in a Key Vault and reference the secret in the parameter file. For more information, see Use Azure Key Vault to pass secure parameter value during deployment .
value	No	Template language expression that is evaluated and returned as output value. Specify either value or copy .
copy	No	Used to return more than one value for an output. Specify value or copy . For more information, see Output iteration in Azure Resource Manager templates .

For examples of how to use outputs, see [Outputs in Azure Resource Manager template](#).

Comments and metadata

You have a few options for adding comments and metadata to your template.

Comments

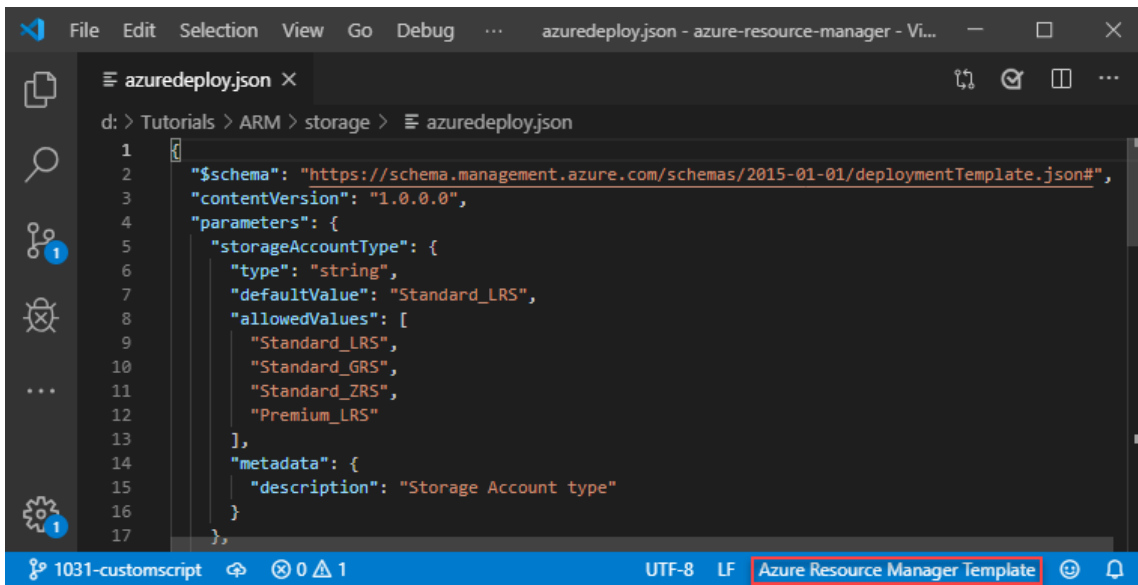
For inline comments, you can use either `//` or `/* ... */` but this syntax doesn't work with all tools. You can't use the portal template editor to work on templates with inline comments. If you add this style of comment, be sure the tools you use support inline JSON comments.

NOTE

To deploy templates with comments by using Azure CLI, you must use the `--handle-extended-json-format` switch.

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]", // to customize name, change it in variables
  "location": "[parameters('location')]", //defaults to resource group location
  "dependsOn": [ /* storage account and network interface must be deployed first */
    "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
  ],
}
```

In Visual Studio Code, the [Azure Resource Manager Tools extension](#) can automatically detect Resource Manager template and change the language mode accordingly. If you see **Azure Resource Manager Template** at the bottom-right corner of VS Code, you can use the inline comments. The inline comments are no longer marked as invalid.



Metadata

You can add a `metadata` object almost anywhere in your template. Resource Manager ignores the object, but your JSON editor may warn you that the property isn't valid. In the object, define the properties you need.

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "metadata": {
    "comments": "This template was developed for demonstration purposes.",
    "author": "Example Name"
  },
}
```

For **parameters**, add a `metadata` object with a `description` property.

```
"parameters": {
  "adminUsername": {
    "type": "string",
    "metadata": {
      "description": "User name for the Virtual Machine."
    }
  }
},
```

When deploying the template through the portal, the text you provide in the description is automatically used as a tip for that parameter.

SETTINGS	
★ Admin Username ⓘ	<input type="text"/>
★ Admin Password ⓘ	<input type="password"/>

User name for the Virtual Machine.

For **resources**, add a `comments` element or a metadata object. The following example shows both a comments element and a metadata object.

```
"resources": [
  {
    "type": "Microsoft.Storage/storageAccounts",
    "apiVersion": "2018-07-01",
    "name": "[concat('storage', uniqueString(resourceGroup().id))]",
    "comments": "Storage account used to store VM disks",
    "location": "[parameters('location')]",
    "metadata": {
      "comments": "These tags are needed for policy compliance."
    },
    "tags": {
      "Dept": "[parameters('deptName')]",
      "Environment": "[parameters('environment')]"
    },
    "sku": {
      "name": "Standard_LRS"
    },
    "kind": "Storage",
    "properties": {}
  }
]
```

For **outputs**, add a metadata object to the output value.

```
"outputs": {
  "hostname": {
    "type": "string",
    "value": "[reference(variables('publicIPAddressName')).dnsSettings.fqdn]",
    "metadata": {
      "comments": "Return the fully qualified domain name"
    }
  }
},
```

You can't add a metadata object to user-defined functions.

Multi-line strings

You can break a string into multiple lines. For example, see the location property and one of the comments in the following JSON example.

```
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]", // to customize name, change it in variables
  "location": "[
    parameters('location')
  ]", //defaults to resource group location
  /*
    storage account and network interface
    must be deployed first
  */
  "dependsOn": [
    "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
  ],
}
```

To deploy templates with multi-line strings by using Azure CLI, you must use the `--handle-extended-json-format` switch.

Next steps

- To view complete templates for many different types of solutions, see the [Azure Quickstart Templates](#).
- For details about the functions you can use from within a template, see [Azure Resource Manager Template Functions](#).
- To combine several templates during deployment, see [Using linked templates with Azure Resource Manager](#).
- For recommendations about creating templates, see [Azure Resource Manager template best practices](#).
- For recommendations on creating Resource Manager templates that you can use across all Azure environments and Azure Stack, see [Develop Azure Resource Manager templates for cloud consistency](#).