

Contents

[Azure Firewall Manager Documentation](#)

[Overview](#)

[What is Azure Firewall Manager?](#)

[Tutorials](#)

[Secure your virtual WAN - portal](#)

[Secure your hub virtual network - portal](#)

[Concepts](#)

[Policy overview](#)

[What is a secured virtual hub?](#)

[General deployment process](#)

[Trusted security partners](#)

[Rule processing logic](#)

[Architecture options](#)

[How-to guides](#)

[Deploy trusted security partners](#)

[Migrate to Firewall policy - PowerShell](#)

[Reference](#)

[Azure CLI](#)

[Azure PowerShell](#)

[.NET](#)

[Java](#)

[Node.js](#)

[Python](#)

[REST](#)

[Azure Resource Manager](#)

[Resources](#)

[Author templates](#)

[Azure Roadmap](#)

[Community templates](#)

Pricing

Regional availability

What is Azure Firewall Manager Preview?

2/18/2020 • 3 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](#).

Azure Firewall Manager Preview is a security management service that provides central security policy and route management for cloud-based security perimeters.

Firewall Manager can provide security management for two network architecture types:

- **secured virtual hub**

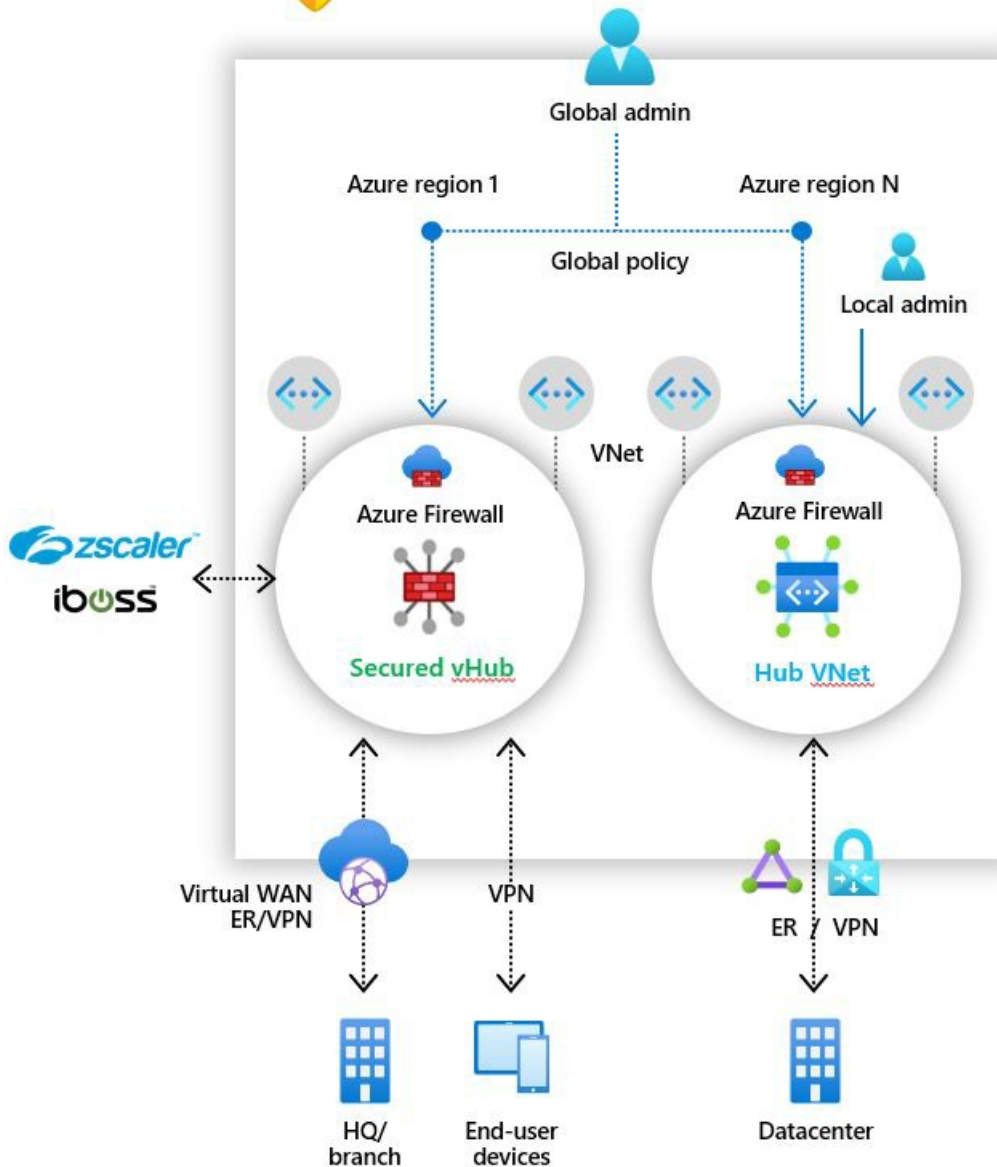
An [Azure Virtual WAN Hub](#) is a Microsoft-managed resource that lets you easily create hub and spoke architectures. When security and routing policies are associated with such a hub, it is referred to as a *secured virtual hub*.

- **hub virtual network**

This is a standard Azure virtual network that you create and manage yourself. When security policies are associated with such a hub, it is referred to as a *hub virtual network*. At this time, only Azure Firewall Policy is supported. You can peer spoke virtual networks that contain your workload servers and services. You can also manage firewalls in standalone virtual networks that are not peered to any spoke.

For a detailed comparison of *secured virtual hub* and *hub virtual network* architectures, see [What are the Azure Firewall Manager architecture options?](#).

Azure Firewall Manager



Azure Firewall Manager Preview features

Azure Firewall Manager Preview offers the following features:

Central Azure Firewall deployment and configuration

You can centrally deploy and configure multiple Azure Firewall instances that span different Azure regions and subscriptions.

Hierarchical policies (global and local)

You can use Azure Firewall Manager Preview to centrally manage Azure Firewall policies across multiple secured virtual hubs. Your central IT teams can author global firewall policies to enforce organization wide firewall policy across teams. Locally authored firewall policies allow a DevOps self-service model for better agility.

Integrated with third-party security-as-a-service for advanced security

In addition to Azure Firewall, you can integrate third-party security as a service (SECaaS) providers to provide additional network protection for your VNet and branch Internet connections.

This feature is available only with secured virtual hub deployments.

- VNet to Internet (V2I) traffic filtering

- Filter outbound virtual network traffic with your preferred third-party security provider.
- Leverage advanced user-aware Internet protection for your cloud workloads running on Azure.
- Branch to Internet (B2I) traffic filtering

Leverage your Azure connectivity and global distribution to easily add third-party filtering for branch to Internet scenarios.

For more information about trusted security providers, see [What are Azure Firewall Manager trusted security partners \(preview\)?](#)

Centralized route management

Easily route traffic to your secured hub for filtering and logging without the need to manually set up User Defined Routes (UDR) on spoke virtual networks.

This feature is available only with secured virtual hub deployments.

You can use third-party providers for Branch to Internet (B2I) traffic filtering, side by side with Azure Firewall for Branch to VNet (B2V), VNet to VNet (V2V) and VNet to Internet (V2I). You can also use third-party providers for V2I traffic filtering as long as Azure Firewall is not required for B2V or V2V.

Region availability

Azure Firewall Policies can be used across regions. For example, you can create a policy in West US, and use it in East US.

Known issues

Azure Firewall Manager Preview has the following known issues:

ISSUE	DESCRIPTION	MITIGATION
Third-party filtering limitations	V2I traffic filtering with third-party providers is not supported with Azure Firewall B2V and V2V.	Currently investigating.
Traffic splitting not currently supported	Office 365 and Azure Public PaaS traffic splitting is not currently supported. As such, selecting a third-party provider for V2I or B2I also sends all Azure Public PaaS and Office 365 traffic via the partner service.	Currently investigating traffic splitting at the hub.
One secured virtual hub per region	You can't have more than one secured virtual hub per region	Create multiple virtual WANs in a region.
Base policies must be in same region as local policy	Create all your local policies in the same region as the base policy. You can still apply a policy that was created in one region on a secured hub from another region.	Currently investigating.
Inter-hub communication not working with Secured Virtual Hub	Secured Virtual Hub to Secured Virtual Hub communication is not yet supported.	Currently investigating.

ISSUE	DESCRIPTION	MITIGATION
All Secured Virtual Hubs sharing the same virtual WAN must be in the same resource group.	This behavior is aligned with Virtual WAN Hubs today.	Create multiple Virtual WANs to allow Secured Virtual Hubs to be created in different resource groups.
IP Groups are not supported in Firewall Policy	IP Groups are in public preview and currently only supported with traditional firewall rules	Fix in progress

Next steps

- Review [Azure Firewall Manager Preview deployment overview](#)
- Learn about [secured Virtual Hubs](#).

Tutorial: Secure your virtual WAN using Azure Firewall Manager preview

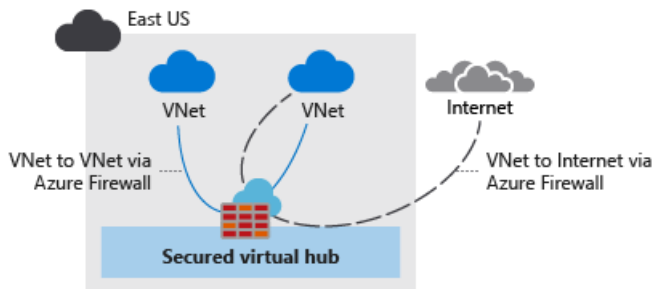
2/18/2020 • 6 minutes to read • [Edit Online](#)

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](#).

Using Azure Firewall Manager Preview, you can create secured virtual hubs to secure your cloud network traffic destined to private IP addresses, Azure PaaS, and the Internet. Traffic routing to the firewall is automated, so there's no need to create user defined routes (UDRs).



Firewall Manager also supports a hub virtual network architecture. For a comparison of the secured virtual hub and hub virtual network architecture types, see [What are the Azure Firewall Manager architecture options?](#)

In this tutorial, you learn how to:

- Create the spoke virtual network
- Create a secured virtual hub
- Connect the hub and spoke VNets
- Create a firewall policy and secure your hub
- Route traffic to your hub
- Test the firewall

Create a hub and spoke architecture

First, create a spoke VNet where you can place your servers.

Create a spoke VNet and subnets

1. From the Azure portal home page, select **Create a resource**.
2. Under **Networking**, select **Virtual network**.
3. For **Name**, type **Spoke-01**.
4. For **Address space**, type **10.0.0.0/16**.
5. For **Subscription**, select your subscription.
6. For **Resource group**, select **Create new**, and type **FW-Manager** for the name and select **OK**.
7. For **Location**, select **(US) East US**.

8. Under **Subnet**, for **Name** type **Workload-SN**.
9. For **Address range**, type **10.0.1.0/24**.
10. Accept the other default settings, and then select **Create**.

Next, create a subnet for a jump server.

1. On the Azure portal home page, select **Resource groups > FW-Manager**.
2. Select the **Spoke-01** virtual network.
3. Select **Subnets > +Subnet**.
4. For **Name**, type **Jump-SN**.
5. For **Address range**, type **10.0.2.0/24**.
6. Select **OK**.

Create the secured virtual hub

Create your secured virtual hub using Firewall Manager.

1. From the Azure portal home page, select **All services**.
2. In the search box, type **Firewall Manager** and select **Firewall Manager**.
3. On the **Firewall Manager** page, select **Create a Secured Virtual Hub**.
4. On the **Create new Secured virtual hub** page, select your subscription and the **FW-Manager** resource group.
5. For the **Secured virtual hub name**, type **Hub-01**.
6. For **Location**, select **East US**.
7. For **Hub address space**, type **10.1.0.0/16**.
8. For the new vWAN name, type **vwan-01**.
9. Leave the **Include VPN gateway to enable Trusted Security Partners** check box cleared.
10. Select **Next: Azure Firewall**.
11. Accept the default **Azure Firewall Enabled** setting and then select **Next: Trusted Security Partner**.
12. Accept the default **Trusted Security Partner Disabled** setting, and select **Next: Review + create**.
13. Select **Create**. It will take about 30 minutes to deploy.

Connect the hub and spoke VNets

Now you can peer the hub and spoke VNets.

1. Select the **FW-Manager** resource group, then select the **vwan-01** virtual WAN.
2. Under **Connectivity**, select **Virtual network connections**.
3. Select **Add connection**.
4. For **Connection name**, type **hub-spoke**.
5. For **Hubs**, select **Hub-01**.
6. For **Virtual network**, select **Spoke-01**.
7. Select **OK**.

Create a firewall policy and secure your hub

A firewall policy defines collections of rules to direct traffic on one or more Secured virtual hubs. You'll create your firewall policy and then secure your hub.

1. From Firewall Manager, select **Create an Azure Firewall Policy**.
2. Select your subscription, and then select the **FW-Manager** resource group.
3. Under **Policy details**, for the **Name** type **Policy-01** and for **Region** select **East US**.
4. Select **Next: Rules**.

5. On the **Rules** tab, select **Add a rule collection**.
6. On the **Add a rule collection** page, type **RC-01** for the **Name**.
7. For **Rule collection type**, select **Application**.
8. For **Priority**, type **100**.
9. Ensure **Rule collection action** is **Allow**.
10. For the rule **Name** type **Allow-msft**.
11. For **Source address**, type *****.
12. For **Protocol**, type **http,https**.
13. Ensure ****Destination** type is **FQDN**.
14. For **Destination**, type ***.microsoft.com**.
15. Select **Add**.
16. Select **Next: Secured virtual hubs**.
17. On the **Secured virtual hubs** tab, select **Hub-01**.
18. Select **Review + create**.
19. Select **Create**.

This can take about five minutes or more to complete.

Route traffic to your hub

Now you must ensure that network traffic gets routed to through your firewall.

1. From Firewall Manager, select **Secured virtual hubs**.
2. Select **Hub-01**.
3. Under **Settings**, select **Route settings**.
4. Under **Internet traffic, Traffic from Virtual Networks**, select **Send via Azure Firewall**.
5. Under **Azure private traffic, Traffic to Virtual Networks**, select **Send via Azure Firewall**.
6. Select **Edit IP address prefix(es)**.
7. Select **Add an IP address prefix**.
8. Type **10.0.1.0/24** as the address of the Workload subnet and select **Save**.
9. Under **Settings**, select **Connections**.
10. Select the **hub-spoke** connection, and then select **Secure internet traffic** and then select **OK**.

Test your firewall

To test your firewall rules, you'll need to deploy a couple servers. You'll deploy Workload-Srv in the Workload-SN subnet to test the firewall rules, and Jump-Srv so you can use Remote Desktop to connect from the Internet and then connect to Workload-Srv.

Deploy the servers

1. On the Azure portal, select **Create a resource**.
2. Select **Windows Server 2016 Datacenter** in the **Popular** list.
3. Enter these values for the virtual machine:

SETTING	VALUE
Resource group	FW-Manager
Virtual machine name	Jump-Srv

SETTING	VALUE
Region	(US) East US
Administrator user name	azureuser
Password	type your password

- Under **Inbound port rules**, for **Public inbound ports**, select **Allow selected ports**.
- For **Select inbound ports**, select **RDP (3389)**.
- Accept the other defaults and select **Next: Disks**.
- Accept the disk defaults and select **Next: Networking**.
- Make sure that **Spoke-01** is selected for the virtual network and the subnet is **Jump-SN**.
- For **Public IP**, accept the default new public ip address name (Jump-Srv-ip).
- Accept the other defaults and select **Next: Management**.
- Select **Off** to disable boot diagnostics. Accept the other defaults and select **Review + create**.
- Review the settings on the summary page, and then select **Create**.

Use the information in the following table to configure another virtual machine named **Workload-Srv**. 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

Add a route table and default route

To allow an Internet connection to Jump-Srv, you must create a route table and a default gateway route to the Internet from the **Jump-SN** subnet.

- On the Azure portal, select **Create a resource**.
- Type **route table** in the search box, and then select **Route table**.
- Select **Create**.
- Type **RT-01** for **Name**.
- Select your subscription, **FW-Manager** for the resource group and **(US) East US** for the region.
- Select **Create**.
- When the deployment completes, select the **RT-01** route table.
- Select **Routes** and then select **Add**.
- Type **jump-to-inet** for the **Route name**.
- Type **0.0.0.0/0** for the **Address prefix**.
- Select **Internet** for the **Next hop type**.
- Select **OK**.
- When the deployment completes, select **Subnets**, then select **Associate**.
- Select **Spoke-01** for **Virtual network**.

15. Select **Jump-SN** for **Subnet**.

16. Select **OK**.

Test the rules

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

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

You should see the Microsoft home page.

5. Browse to <https://www.google.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.

Next steps

[Learn about trusted security partners](#)

Tutorial: Secure your hub virtual network using Azure Firewall Manager preview

2/18/2020 • 14 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](#).

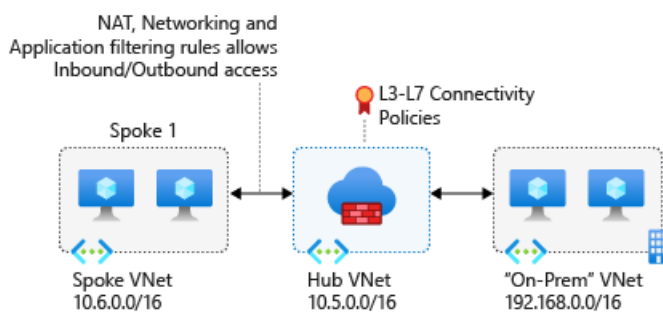
When you connect your on-premises network to an Azure virtual network to create a hybrid network, the ability to control access to your Azure network resources is an important part of an overall security plan.

Using Azure Firewall Manager Preview, you can create a hub virtual network to secure your hybrid network traffic destined to private IP addresses, Azure PaaS, and the Internet. You can use Azure Firewall Manager to control network access in a hybrid network using policies that define allowed and denied network traffic.

Firewall Manager also supports a secured virtual hub architecture. For a comparison of the secured virtual hub and hub virtual network architecture types, see [What are the Azure Firewall Manager architecture options?](#)

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:

- Create a firewall policy
- Create the virtual networks
- 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

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 **Virtual network gateway route propagation** setting disabled. This 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](#).

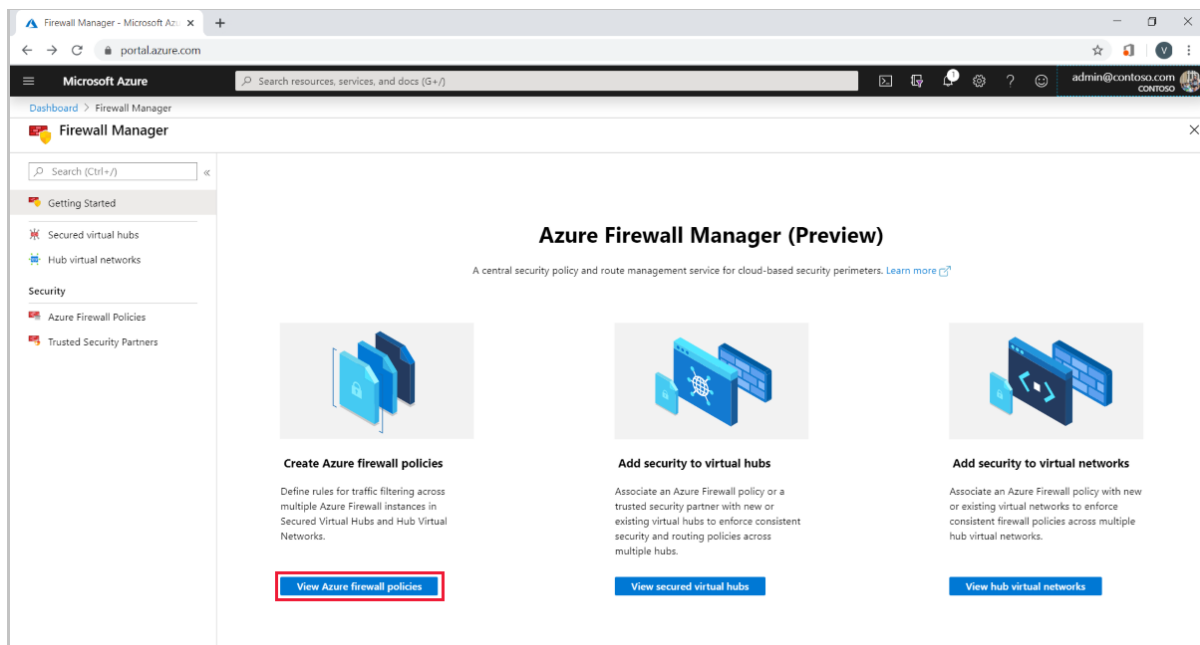
NOTE

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 a Firewall Policy

1. Sign in to the Azure portal at <https://portal.azure.com>.
2. In the Azure portal search bar, type **Firewall Manager** and press **Enter**.
3. On the Azure Firewall Manager page, select **View Azure firewall policies**.



4. Select **Create Azure Firewall Policy**.
5. Select your subscription, and for Resource group, select **Create new** and create a resource group named **FW-Hybrid-Test**.
6. For the policy name, type **Pol-Net01**.
7. For Region, select **East US**.
8. Select **Next:Rules**.
9. Select **Add a rule collection**.
10. For **Name**, type **RCNet01**.
11. For **Rule collection type**, select **Network**.
12. For **Priority**, type **100**.
13. For **Action**, select **Allow**.
14. Under **Rules**, for **Name**, type **AllowWeb**.
15. For **Source Addresses**, type **192.168.1.0/24**.
16. For **Protocol**, select **TCP**.
17. For **Destination Ports**, type **80**.
18. For **Destination Type**, select **IP Address**.
19. For **Destination**, type **10.6.0.0/16**.
20. On the next rule row, enter the following information:
 - Name: type **AllowRDP**
 - Source IP address: type **192.168.1.0/24**
 - Protocol, select **TCP**
 - Destination Ports, type **3389**
 - Destination Type, select **IP Address**
 - For Destination, type **10.6.0.0/16**
21. Select **Add**.

22. Select **Review + Create**.
23. Review the details and then select **Create**.

Create the firewall hub virtual network

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**.

After the virtual network is deployed, 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 **East US**.
7. Accept the other defaults, and then select **Create**.

Configure and deploy the firewall

When security policies are associated with a hub, it is referred to as a *hub virtual network*.

Convert the **VNet-Hub** virtual network into a *hub virtual network* and secure it with Azure Firewall.

1. In the Azure portal search bar, type **Firewall Manager** and press **Enter**.
2. On the Azure Firewall Manager page, under **Add security to virtual networks**, select **View hub virtual networks**.
3. Select **Convert virtual networks**.
4. Select **VNet-hub** and then select **Next : Azure Firewall**.
5. For the **Firewall Policy**, select **Pol-Net01**.
6. Select **Next " Review + confirm**
7. Review the details and then select **Confirm**.

This takes a few minutes to deploy.

8. After deployment completes, go to the **FW-Hybrid-Test** resource group, and select the firewall.
9. Note the **Firewall private IP** address on the **Overview** page. You'll use it later when you create the default route.

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 **(US) East US**.
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 **(US) East US**.
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 **Use existing*, and select **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 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**.

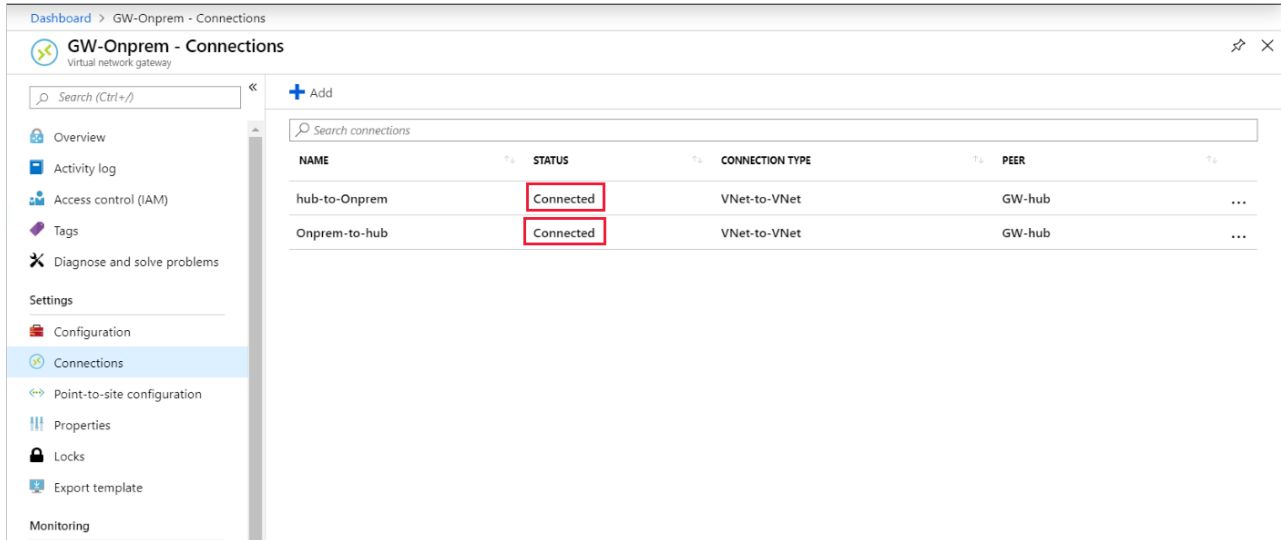
Create the on-premises to hub virtual network connection. This step is similar to the previous one, except you create the connection from VNet-Onprem to VNet-hub. Make sure the shared keys match. The connection will be established after a few minutes.

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**.



NAME	STATUS	CONNECTION TYPE	PEER
hub-to-Onprem	Connected	VNet-to-VNet	GW-hub
Onprem-to-hub	Connected	VNet-to-VNet	GW-hub

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 **(US) East US**.
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. Under **Virtual network**, select **VNet-hub**.
4. Under **Subnet**, select **GatewaySubnet**.
5. 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 **(US) East US**.
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. Under **Virtual network**, select **VNet-spoke**.

- Under **Subnet**, select **SN-Workload**.
- 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.

- From the Azure portal home page, select **Create a resource**.
- Under **Popular**, select **Windows Server 2016 Datacenter**.
- Enter these values for the virtual machine:
 - **Resource group** - Select **FW-Hybrid-Test**.
 - **Virtual machine name**: *VM-Spoke-01*.
 - **Region** - *(US) East US*.
 - **User name**: *azureuser*.
 - **Password**: type your password
- Select **Next:Disks**.
- Accept the defaults and select **Next: Networking**.
- Select **VNet-Spoke** for the virtual network and the subnet is **SN-Workload**.
- For **Public IP**, select **None**.
- For **Public inbound ports**, select **Allow selected ports**, and then select **HTTP (80)**, and **RDP (3389)**
- Select **Next:Management**.
- For **Boot diagnostics**, Select **Off**.
- Select **Review+Create**, review the settings on the summary page, and then select **Create**.

Install IIS

- From the Azure portal, open the Cloud Shell and make sure that it's set to **PowerShell**.
- 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.

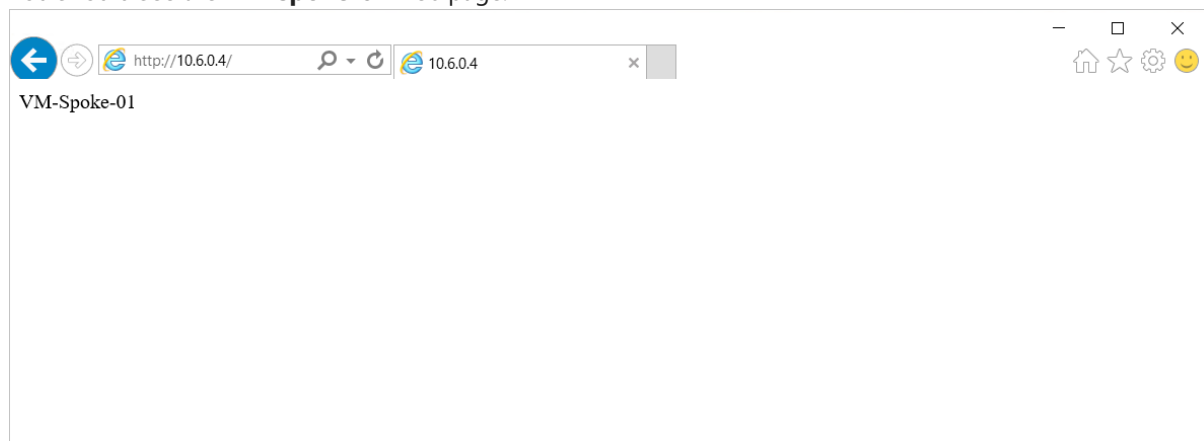
- From the Azure portal home page, select **Create a resource**.

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** - *(US) East US*.
 - **User name**: *azureuser*.
 - **Password**: type your password.
4. Select **Next:Disks**.
5. Accept the defaults and select **Next:Networking**.
6. Select **VNet-Onprem** for virtual network and verify 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 on the VM-Spoke-01 Overview page.
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. Open the **FW-Hybrid-Test** resource group and select the **Pol-Net01** firewall policy.
2. Under **Settings**, select **Rules**.
3. Under **Network rules**, select the **RCNet01** rule collection, select the ellipses (...), and select **Edit**.

4. For **Rule collection action**, select **Deny**.
5. Select **Save**.

Close any existing remote desktops and browsers on **VM-Onprem** before testing the changed rules. After the rule collection update is complete, 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

[Tutorial: Secure your virtual WAN using Azure Firewall Manager preview](#)

Azure Firewall Manager Preview policy overview

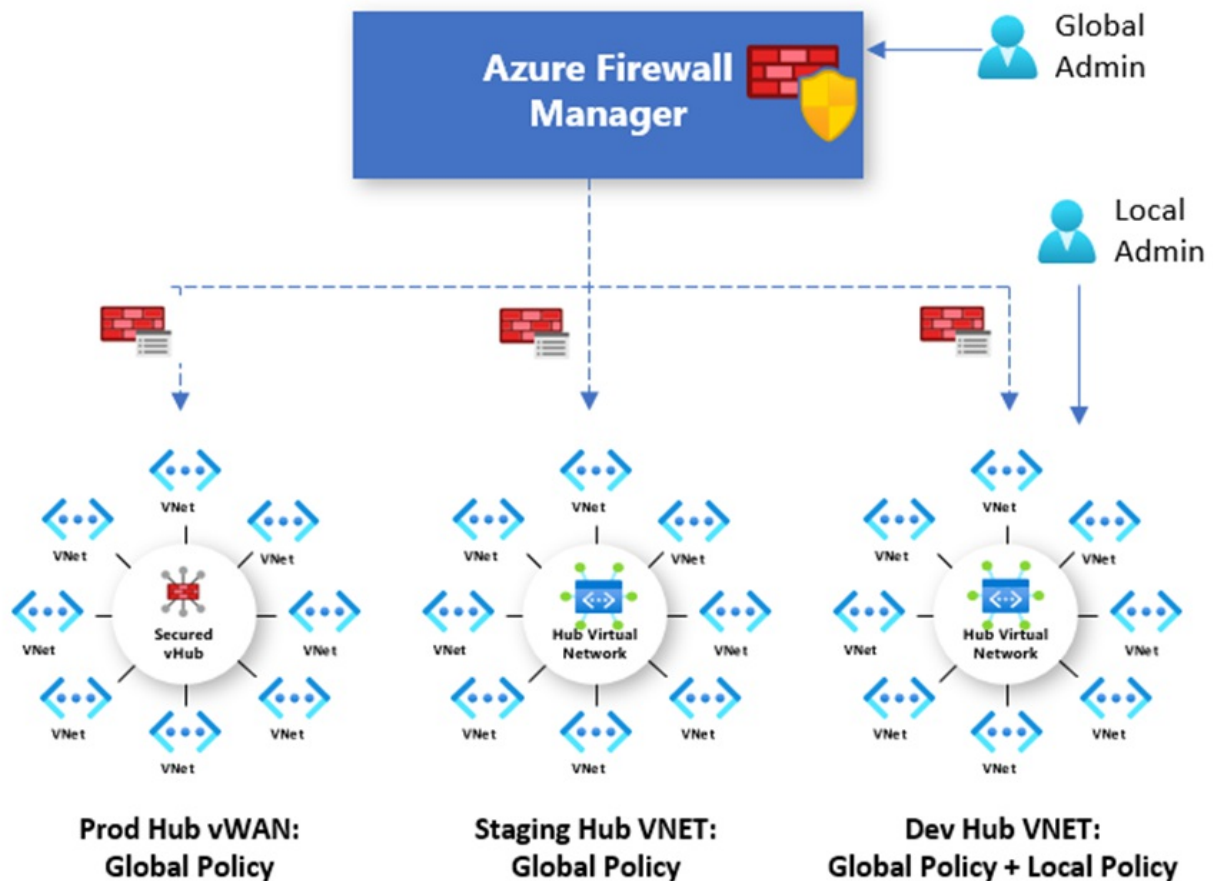
2/18/2020 • 2 minutes to read • [Edit Online](#)

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](#).

Firewall policy is an Azure resource that contains NAT, network, and application rule collections as well as 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. Policies work across regions and subscriptions.



Policy creation and association

A policy can be created and managed in multiple ways, including the Azure portal, REST API, templates, Azure PowerShell, and CLI.

You can also migrate existing rules from Azure Firewall using the portal or Azure PowerShell to create policies. For more information, see [How to migrate Azure Firewall configurations to Azure Firewall policy \(preview\)](#).

Policies can be associated with one or more virtual hubs or VNets. The firewall can be in any subscription associated with your account and in any region.

Hierarchical policies

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 created with non-empty parent policies inherit all rule collections from the parent policy. Network rule collections inherited from a parent policy are always prioritized above network rule collections defined as part of a new policy. The same logic also applies to application rule collections. However, network rule collections are always processed before application rule collections regardless of inheritance.

Threat Intelligence mode is also inherited from the parent policy. You can set your threat Intelligence mode to a different value to override this behavior, but you can't turn it off. It's only possible to override with a stricter value. For example, if your parent policy is set to **Alert only**, you can configure this local policy to **Alert and deny**.

NAT rule collections aren't inherited because they're specific to a given firewall.

With inheritance, any changes to the parent policy are automatically applied down to associated firewall child policies.

Traditional rules and policies

Azure Firewall supports both traditional rules and policies. The following table compares policies and rules:

	POLICY	RULES
Contains	NAT, Network, Application rules, and Threat Intelligence settings	NAT, Network, and Application rules
Protects	Virtual hubs and Virtual Networks	Virtual Networks only
Portal experience	Central management using Firewall Manager	Standalone firewall experience
Multiple firewall support	Firewall Policy is a separate resource that can be used across firewalls	Manually export and import rules, or using third-party management solutions
Pricing	Billed based on firewall association. See Pricing .	Free
Supported deployment mechanisms	Portal, REST API, templates, Azure PowerShell, and CLI	Portal, REST API, templates, PowerShell, and CLI.
Release Status	Public Preview	General Availability

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](#).

Next steps

To learn how to deploy an Azure Firewall, see [Tutorial: Secure your cloud network with Azure Firewall Manager Preview using the Azure portal](#).

What is a secured virtual hub?

11/4/2019 • 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](#).

A virtual hub is a Microsoft-managed virtual network that enables connectivity from other resources. When a virtual hub is created from a Virtual WAN in the Azure portal, a virtual hub VNet and gateways (optional) are created as its components.

A *secured* virtual hub is an [Azure Virtual WAN Hub](#) with associated security and routing policies configured by Azure Firewall Manager. Use secured virtual hubs to easily create hub-and-spoke and transitive architectures with native security services for traffic governance and protection.

You can use a secured virtual hub as a managed central VNet with no on-prem connectivity. It replaces the central VNet that was previously required for an Azure Firewall deployment. Since the secured virtual hub provides automated routing, there's no need to configure your own UDRs (user defined routes) to route traffic through your firewall.

It's also possible to use secured virtual hubs as part of a full Virtual WAN architecture. This architecture provides secured, optimized, and automated branch connectivity to and through Azure. You can choose the services to protect and govern your network traffic, including Azure Firewall and other third-party security as a service (SECaaS) providers.

Create a secured virtual hub

Using Firewall Manager in the Azure portal, you can either create a new secured virtual hub, or convert an existing virtual hub that you previously created using Azure Virtual WAN.

Next steps

To create a secured virtual hub and use it to secure and govern a hub and spoke network, see [Tutorial: Secure your cloud network with Azure Firewall Manager using the Azure portal](#).

Azure Firewall Manager Preview deployment overview

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](#).

There's more than one way to deploy Azure Firewall Manager Preview, but the following general process is recommended.

General deployment process

Hub virtual networks

1. Create a firewall policy
 - Create a new policy
or
 - Derive a base policy and customize a local policy
or
 - Import rules from an existing Azure Firewall. Make sure to remove NAT rules from policies that should be applied across multiple firewalls
2. Create your hub and spoke architecture
 - Create a Hub Virtual Network using Azure Firewall Manager and peer spoke virtual networks to it using virtual network peering
or
 - Create a virtual network and add virtual network connections and peer spoke virtual networks to it using virtual network peering
3. Select security providers and associate firewall policy. Currently, only Azure Firewall is a supported provider.
 - This is done while you create a Hub Virtual Network
or
 - Convert an existing virtual network to a Hub Virtual Network. It is also possible to convert multiple virtual networks.
4. Configure User Define Routes to route traffic to your Hub Virtual Network firewall.

Secured virtual hubs

1. Create your hub and spoke architecture
 - Create a Secured Virtual Hub using Azure Firewall Manager and add virtual network connections.
or
 - Create a Virtual WAN Hub and add virtual network connections.

2. Select security providers

- Done while creating a Secured Virtual Hub.
or
- Convert an existing Virtual WAN Hub to Secure Virtual Hub.

3. Create a firewall policy and associate it with your hub

- Applicable only if using Azure Firewall.
- Third-party security as a service (SECaaS) policies are configured via partners management experience.

4. Configure route settings to route traffic to your secured hub

- Easily route traffic to your secured hub for filtering and logging without User Defined Routes (UDR) on spoke Virtual Networks using the Secured Virtual Hub Route Setting page.

NOTE

- You can't have more than one hub per virtual wan per region. But you can add multiple virtual WANs in the region to achieve this.
- You can't have overlapping IP spaces for hubs in a vWAN.
- Your hub VNet connections must be in the same region as the hub.

Next steps

- [Tutorial: Secure your cloud network with Azure Firewall Manager Preview using the Azure portal](#)

What are trusted security partners (preview)?

12/11/2019 • 3 minutes to read • [Edit Online](#)

IMPORTANT

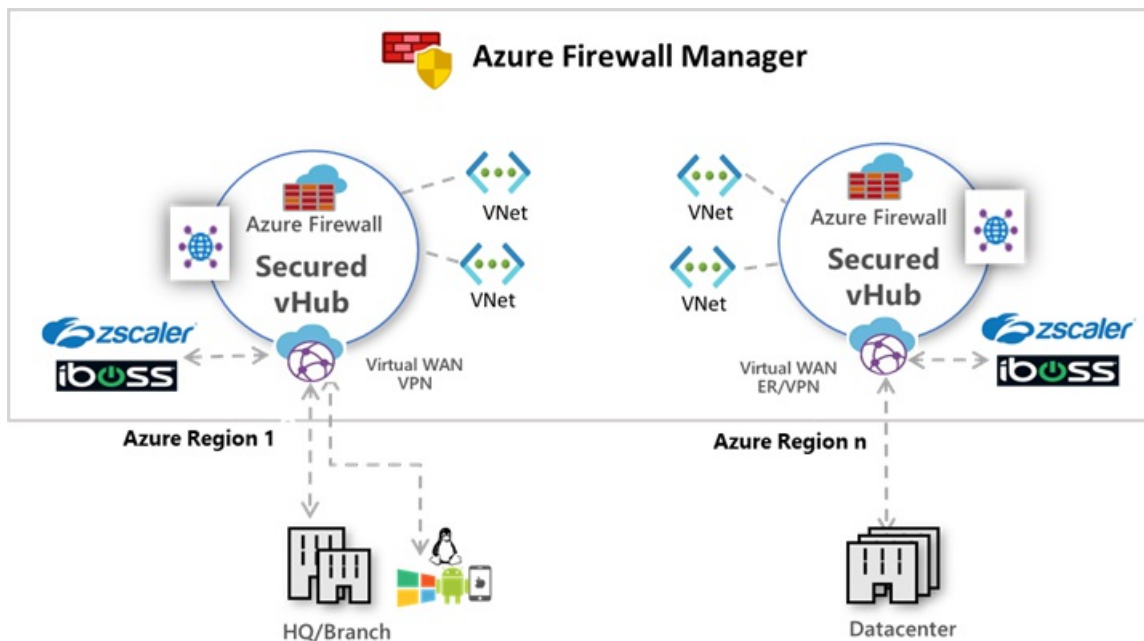
This public preview is provided without a service level agreement and should not be used for production workloads. Certain features may not be supported, may have constrained capabilities, or may not be available in all Azure locations. See the [Supplemental Terms of Use for Microsoft Azure Previews](#)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

Trusted security partners (preview) in Azure Firewall Manager allows you to use your familiar, best-in-breed, third-party security as a service (SECaaS) offerings to protect Internet access for your users.

With a quick configuration, you can secure a hub with a supported security partner, and route and filter Internet traffic from your Virtual Networks (VNETs) or branch locations within a region. This is done using automated route management, without setting up and managing User Defined Routes (UDRs).

You can deploy secured hubs configured with the security partner of your choice in multiple Azure regions to get connectivity and security for your users anywhere across the globe in those regions. With the ability to use the security partner's offering for Internet/SaaS application traffic, and Azure Firewall for private traffic in the secured hubs, you can now start building your security edge on Azure that is close to your globally distributed users and applications.

For this preview, the supported security partners are **ZScaler** and **iboss**. Supported regions are WestCentralUS, NorthCentralUS, WestUS, WestUS2, and EastUS.



Key scenarios

You can use the security partners to filter Internet traffic in following scenarios:

- Virtual Network (VNet) to Internet

Leverage advanced user-aware Internet protection for your cloud workloads running on Azure.

- Branch to Internet

Leverage your Azure connectivity and global distribution to easily add third-party NSaaS filtering for branch to Internet scenarios. You can build your global transit network and security edge using Azure Virtual WAN.

The following scenarios are supported:

- VNet to Internet via a third-party partner offering.
- Branch to Internet via a third-party partner offering.
- Branch to Internet via a third-party partner offering, the rest of the private traffic (Spoke-to-Spoke, Spoke-to-Branches, Branch-to-Spokes) via Azure Firewall.

The following scenario isn't supported:

- VNet to Internet via a partner offering can't be combined with Azure Firewall for private traffic. See the following limitations.

Current limitations

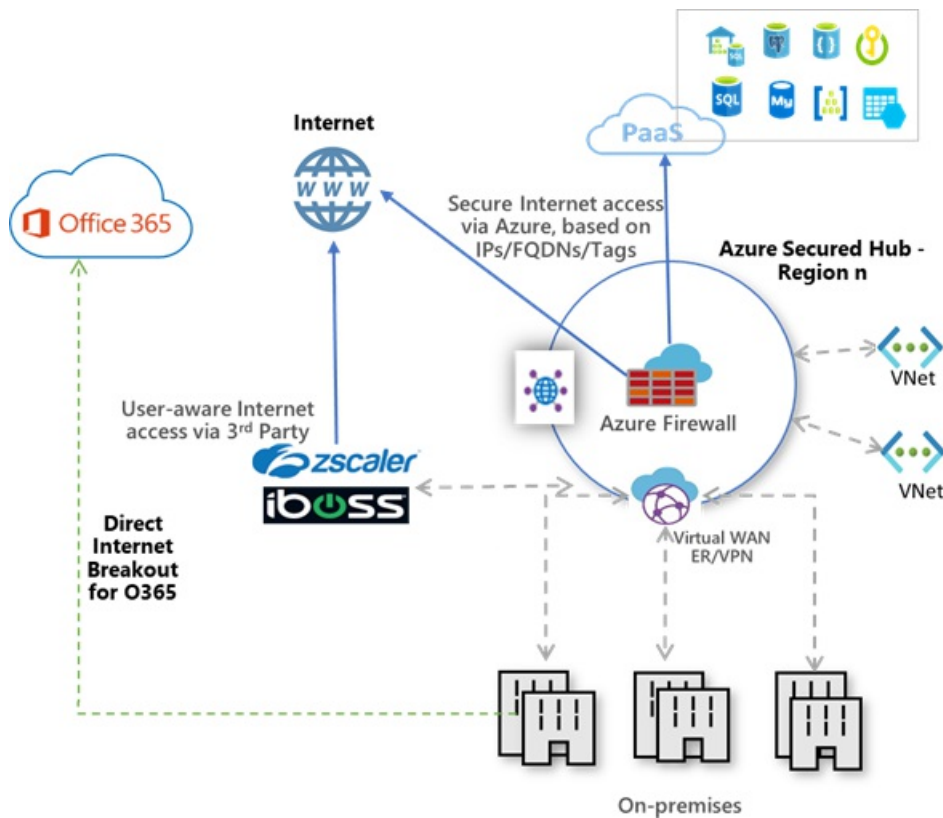
- For VNet to Internet, you can't mix adding Azure Firewall for private traffic and a partner offering for Internet traffic. You can either send Internet traffic to Azure Firewall or a third-party security partner offering in the secured virtual hub, but not both.
- You can deploy at most one security partner per virtual hub. If you need to change the provider, you must remove the existing partner and add a new one.

Best practices for Internet traffic filtering in secured virtual hubs

Internet traffic typically includes web traffic. But it also includes traffic destined to SaaS applications like Office 365 (O365) and Azure public PaaS services like Azure Storage, Azure Sql, and so on. The following are best practice recommendations for handling traffic to these services:

Handling Azure PaaS traffic

- Use Azure Firewall for protection if your traffic consists mostly of Azure PaaS, and the resource access for your applications can be filtered using IP addresses, FQDNs, Service tags, or FQDN tags.
- Use a third-party partner solution in your hubs if your traffic consists of SaaS application access, or you need user-aware filtering (for example, for your virtual desktop infrastructure (VDI) workloads) or you need advanced Internet filtering capabilities.



Handling Office 365 (O365) traffic

In globally distributed branch location scenarios, you should redirect Office 365 traffic directly at the branch before sending the remaining Internet traffic your Azure secured hub.

For Office 365, network latency and performance are critical for successful user experience. To achieve these goals around optimal performance and user experience, customers must implement Office 365 direct and local escape before considering routing the rest of Internet traffic through Azure.

[Office 365 network connectivity principles](#) call for key Office 365 network connections to be routed locally from the user branch or mobile device and directly over the Internet into nearest Microsoft network point of presence.

Furthermore Office 365 connections are strongly encrypted for privacy and use efficient, proprietary protocols for performance reasons. This makes it impractical and impactful to subject those connections to traditional network level security solutions. For these reasons we strongly recommend that customers send Office 365 traffic directly from branches, before sending rest of the traffic through Azure. Microsoft has partnered with several SD-WAN solution providers, who integrate with Azure and Office 365 and make it easy for customers to enable Office 365 direct and local Internet breakout. For details see [How do I set my O365 policies via Virtual WAN?](#)

Next steps

[Deploy a trusted security offering in a secured hub, using Azure Firewall Manager.](#)

Azure Firewall rule processing logic

11/4/2019 • 2 minutes to read • [Edit Online](#)

Azure Firewall has NAT rules, network rules, and applications rules. The rules are processed according to the rule type.

Network rules and applications rules

Network rules are applied first, then application rules. The rules are terminating. So if a match is found in network rules, then application rules are not processed. If there is no network rule match, and if the packet protocol is HTTP/HTTPS, the packet is then evaluated by the application rules. 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.

NAT rules

Inbound 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](#). DNAT rules are applied first. 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. No application rules are applied for these connections.

Inherited rules

Network rule collections inherited from a parent policy are always prioritized above network rule collections that are defined as part of your new policy. The same logic also apply to application rule collections. However, network rule collections are always processed before application rule collections regardless of inheritance.

By default, your policy inherits its parent policy threat intelligence mode. You can override this by setting your threat Intelligence mode to a different value in the policy settings page. It is only possible to override with a stricter value. For example, if your parent policy is set to *Alert only*, you can configure this local policy to *Alert and deny*, but you can't turn it off.

Next steps

- [Learn more about Azure Firewall Manager Preview](#)

What are the Azure Firewall Manager architecture options?

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

Azure Firewall Manager can provide security management for two network architecture types:

- **secured virtual hub**

An [Azure Virtual WAN Hub](#) is a Microsoft-managed resource that lets you easily create hub and spoke architectures. When security and routing policies are associated with such a hub, it's referred to as a *secured virtual hub*.

- **hub virtual network**

This is a standard Azure virtual network that you create and manage yourself. When security policies are associated with such a hub, it is referred to as a *hub virtual network*. At this time, only Azure Firewall Policy is supported. You can peer spoke virtual networks that contain your workload servers and services. You can also manage firewalls in standalone virtual networks that are not peered to any spoke.

Comparison

The following table compares these two architecture options and can help you decide which one is right for your organization's security requirements:

	HUB VIRTUAL NETWORK	SECURED VIRTUAL HUB
Underlying resource	Virtual network	Virtual WAN Hub
Hub & Spoke	Uses Virtual network peering	Automated using hub virtual network connection
On-prem connectivity	VPN Gateway up to 10 Gbps and 30 S2S connections; ExpressRoute	More scalable VPN Gateway up 20 Gbps and 1000 S2S connections; Express Route
Automated branch connectivity using SDWAN	Not supported	Supported
Hubs per region	Multiple Virtual Networks per region	Single Virtual Hub per region. Multiple hubs possible with multiple Virtual WANs
Azure Firewall – multiple public IP addresses	Customer provided	Auto generated. To be available by GA.
Azure Firewall Availability Zones	Supported	Not available in preview. To be available by GA
Advanced Internet security with third-party Security as a Service partners	Customer established and managed VPN connectivity to partner service of choice	Automated via Trusted Security Partner flow and partner management experience

	HUB VIRTUAL NETWORK	SECURED VIRTUAL HUB
Centralized route management to route traffic to the hub	Customer-managed User Defined Route	Supported using BGP
Web Application Firewall on Application Gateway	Supported in Virtual Network	Currently supported in spoke network
Network Virtual Appliance	Supported in Virtual Network	Currently supported in spoke network

Next steps

- Review [Azure Firewall Manager Preview deployment overview](#)
- Learn about [secured Virtual Hubs](#).

Deploy a trusted security partner (preview)

11/12/2019 • 5 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](#).

Trusted security partners in Azure Firewall Manager allows you to use your familiar, best-in-breed third-party security-as-a-service (SECaaS) offerings to protect Internet access for your users.

To learn more about supported scenarios and best practice guidelines, see [What are trusted security partners \(preview\)?](#).

The supported security partners are **ZScaler** and **iboss** for this preview. Supported regions are WestCentralUS, NorthCentralUS, WestUS, WestUS2, and EastUS.

Prerequisites

IMPORTANT

Azure Firewall Manager Preview must be explicitly enabled using the `Register-AzProviderFeature` PowerShell command.

From a PowerShell command prompt, run the following commands:

```
connect-azaccount  
Register-AzProviderFeature -FeatureName AllowCortexSecurity -ProviderNamespace Microsoft.Network
```

It takes up to 30 minutes for the feature registration to complete. Run the following command to check your registration status:

```
Get-AzProviderFeature -FeatureName AllowCortexSecurity -ProviderNamespace Microsoft.Network
```

Deploy a third-party security provider in a new hub

1. Sign in to the Azure portal at <https://portal.azure.com>.
2. In **Search**, type **Firewall Manager** and select it under **Services**.
3. Navigate to **Getting Started**. Select **Create a Secured Virtual Hub**.
4. Enter your subscription and resource group, select a supported region, and add your hub and virtual WAN information.
5. **Deploy VPN gateway** is enabled by default. A VPN Gateway is required to deploy a Trusted security partner in the hub.
6. Select **Next: Azure Firewall**

NOTE

Trusted security partners connect to your hub using VPN Gateway tunnels. If you delete the VPN Gateway, the connections to your Trusted security partners are lost.

7. If you want to deploy Azure Firewall to filter private traffic along with third-party service provider to filter Internet traffic, select a policy for Azure Firewall. See the [supported scenarios](#).
8. If you want to only deploy a third-party security provider in the hub, select **Azure Firewall: Enabled/Disabled** to set it to **Disabled**.
9. Select **Next: Trusted Security Partners**.
10. Select **Trusted Security Partner** to set it to **Enabled**. Select a partner.
11. Select **Next**.
12. Review the content and then select **Create**.

The VPN gateway deployment can take more than 30 minutes.

To verify that the hub has been created, navigate to Azure Firewall Manager->Secured Hubs. Select the hub->Overview page to show the partner name and the status as **Security Connection Pending**.

Once the hub is created and the security partner is set up, continue on to connect the security provider to the hub.

Deploy a third-party security provider in an existing hub

You can also select an existing hub in a Virtual WAN and convert that to a *secured virtual hub*.

1. In **Getting Started**, select **Convert Existing Hubs**.
2. Select a subscription and an existing hub. Follow rest of the steps to deploy a third-party provider in a new hub.

Remember that a VPN gateway must be deployed to convert an existing hub to secured hub with third-party providers.

Configure third-party security providers to connect to a secured hub

To set up tunnels to your virtual hub's VPN Gateway, third-party providers need access rights to your hub. To do this, associate a service principal with your subscription or resource group, and grant access rights. You then must give these credentials to the third-party using their portal.

1. Create Azure Active Directory (AD) service principal: You can skip the redirect URL.

[How to: Use the portal to create an Azure AD application and service principal that can access resources](#)

2. Add access rights and scope for the service principal. [How to: Use the portal to create an Azure AD application and service principal that can access resources](#)

NOTE

You can limit access to only your resource group for more granular control.

3. Follow the [ZScaler: Configuring a Microsoft Azure Virtual WAN Integration](#) instructions to:
 - Sign in to the partner portal and add your credentials to give the trusted partner access to your secured hub.
 - Sync the virtual hubs in the partner portal, and set up the tunnel to the virtual hub. You can do so once your Azure AD authentication credentials are validated.

4. You can look at the tunnel creation status on the Azure Virtual WAN portal in Azure. Once the tunnels show **connected** on both Azure and the partner portal, continue with the next steps to set up routes to select which branches and VNets should send Internet traffic to the partner.

Configure route settings

1. Browse to the Azure Firewall Manager -> Secured Hubs.
2. Select a hub. The Hub status should now show **Provisioned** instead of **Security Connection Pending**.

Ensure the third-party provider can connect to the hub. The tunnels on the VPN gateway should be in a **Connected** state. This state is more reflective of the connection health between the hub and the third-party partner, compared to previous status.

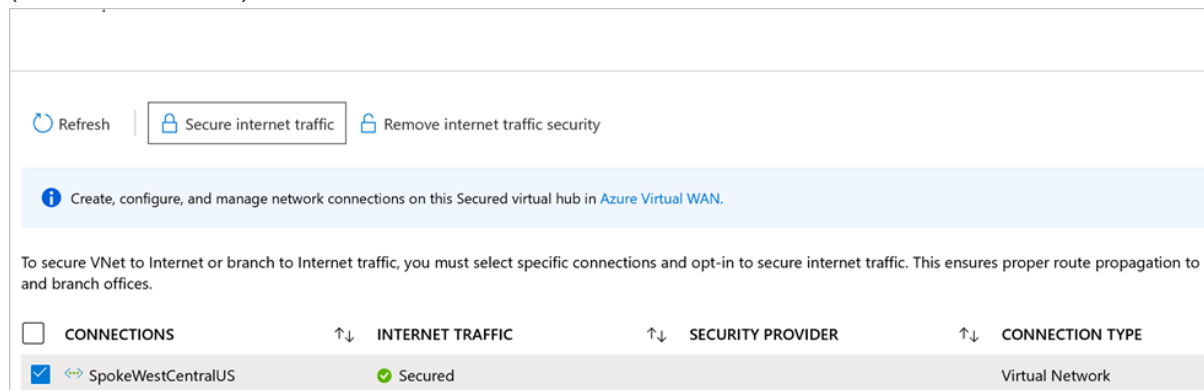
3. Select the hub, and navigate to **Route Settings**.

When you deploy a third-party provider into the hub, it converts the hub into a *secured virtual hub*. This ensures that the third-party provider is advertising a 0.0.0.0/0 (default) route to the hub. However, VNet connections and sites connected to the hub don't get this route unless you opt-in on which connections should get this default route.

4. Under **Internet traffic**, select **VNet-to-Internet** or **Branch-to-Internet** or both so routes are configured send via the third party.

This only indicates which type of traffic should be routed to the hub, but it doesn't affect the routes on VNets or branches yet. These routes are not propagated to all VNets/branches attached to the hub by default.

5. You must select **secure connections** and select the connections on which these routes should be set. This indicates which VNets/branches can start sending Internet traffic to the third-party provider.
6. From **Route settings**, select **Secure connections** under Internet traffic, then select the VNet or branches (*sites* in Virtual WAN) to be secured. Select **Secure Internet traffic**.



7. Navigate back to the hubs page. The hub's **Trusted security partner** status should now be **Secured**.

Branch or VNet Internet traffic via third-party service

Next, you can check if VNet virtual machines or the branch site can access the Internet and validate that the traffic is flowing to the third-party service.

After finishing the route setting steps, the VNet virtual machines as well as the branch sites are sent a 0/0 to third party service route. You can't RDP or SSH into these virtual machines. To sign in, you can deploy the [Azure Bastion](#) service in a peered VNet.

Next steps

- [Tutorial: Secure your cloud network with Azure Firewall Manager Preview using the Azure portal](#)

Migrate Azure Firewall configurations to Azure Firewall policy (preview) using Powershell

2/18/2020 • 3 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](#).

You can use an Azure PowerShell script to migrate existing Azure Firewall configurations to an Azure Firewall policy resource. You can then use Azure Firewall Manager to deploy the policy.

The `AZFWMigrationScript.ps1` script creates a FirewallPolicy with three RuleCollectionGroup objects for ApplicationRuleCollections, NetworkRuleCollections, and NatRuleCollections respectively.

A RuleCollectionGroup is a new top-level grouping for rule collections for future extensibility. Using the above defaults is recommended and is done automatically from the Portal.

The beginning of the script defines the source firewall name and resource group and the target policy name and location. Change these values as appropriate for your organization.

Migration script

Modify the following script to migrate your firewall configuration.

```
#Input params to be modified as needed
$FirewallName = "AZFW"
$ResourceGroupName = "AzFWMigrateRG"
$PolicyName = "fwp9"
$Location = "WestUS"

$DefaultAppRuleCollectionGroupName = "ApplicationRuleCollectionGroup"
$DefaultNetRuleCollectionGroupName = "NetworkRuleCollectionGroup"
$DefaultNatRuleCollectionGroupName = "NatRuleCollectionGroup"
$ApplicationRuleGroupPriority = 300
$NetworkRuleGroupPriority = 200
$NatRuleGroupPriority = 100

#Helper functions for translating ApplicationProtocol and ApplicationRule
Function GetApplicationProtocolsString
{
    Param([Object[]] $Protocols)
    $output = ""
    ForEach ($protocol in $Protocols) {
        $output += $protocol.ProtocolType + ":" + $protocol.Port + ","
    }
    return $output.Substring(0, $output.Length - 1)
}

Function GetApplicationRuleCmd
{
    Param([Object] $ApplicationRule)
```

```

$cmd = "New-AzFirewallPolicyApplicationRule"
$cmd = $cmd + " -Name " + $ApplicationRule.Name
$cmd = $cmd + " -SourceAddress " + $ApplicationRule.SourceAddresses

if ($ApplicationRule.Description) {
    $cmd = $cmd + " -Description " + $ApplicationRule.Description
}
if ($ApplicationRule.TargetFqdns) {
    $protocols = GetApplicationProtocolsString($ApplicationRule.Protocols)
    $cmd = $cmd + " -Protocol " + $protocols
    $cmd = $cmd + " -TargetFqdn " + $ApplicationRule.TargetFqdns
}
if ($ApplicationRule.FqdnTags) {
    $cmd = $cmd + " -FqdnTag " + $ApplicationRule.FqdnTags
}

return $cmd
}

$azfw = Get-AzFirewall -Name $FirewallName -ResourceGroupName $ResourceGroupName
Write-Host "creating empty firewall policy"
$fwp = New-AzFirewallPolicy -Name $PolicyName -ResourceGroupName $ResourceGroupName -Location $Location -
ThreatIntelMode $azfw.ThreatIntelMode
Write-Host $fwp.Name "created"
Write-Host "creating " $azfw.ApplicationRuleCollections.Count " application rule collections"

#Translate ApplicationRuleCollection
If ($azfw.ApplicationRuleCollections.Count -gt 0) {
    $firewallPolicyAppRuleCollections = @()
    ForEach ($appRc in $azfw.ApplicationRuleCollections) {
        If ($appRc.Rules.Count -gt 0) {
            Write-Host "creating " $appRc.Rules.Count " application rules for collection " $appRc.Name
            $firewallPolicyAppRules = @()
            ForEach ($appRule in $appRc.Rules) {
                $cmd = GetApplicationRuleCmd($appRule)
                $firewallPolicyAppRule = Invoke-Expression $cmd
                Write-Host "Created appRule " $firewallPolicyAppRule.Name
                $firewallPolicyAppRules += $firewallPolicyAppRule
            }
            $fwpAppRuleCollection = New-AzFirewallPolicyFilterRuleCollection -Name $appRC.Name -Priority
$appRC.Priority -ActionType $appRC.Action.Type -Rule $firewallPolicyAppRules
            Write-Host "Created appRuleCollection " $fwpAppRuleCollection.Name
        }
        $firewallPolicyAppRuleCollections += $fwpAppRuleCollection
    }
    $appRuleGroup = New-AzFirewallPolicyRuleCollectionGroup -Name $DefaultAppRuleCollectionGroupName -Priority
$ApplicationRuleGroupPriority -RuleCollection $firewallPolicyAppRuleCollections -FirewallPolicyObject $fwp
    Write-Host "Created ApplicationRuleCollectionGroup " $appRuleGroup.Name
}

#Translate NetworkRuleCollection
Write-Host "creating " $azfw.NetworkRuleCollections.Count " network rule collections"
If ($azfw.NetworkRuleCollections.Count -gt 0) {
    $firewallPolicyNetRuleCollections = @()
    ForEach ($src in $azfw.NetworkRuleCollections) {
        If ($src.Rules.Count -gt 0) {
            Write-Host "creating " $src.Rules.Count " network rules for collection " $src.Name
            $firewallPolicyNetRules = @()
            ForEach ($rule in $src.Rules) {
                $firewallPolicyNetRule = New-AzFirewallPolicyNetworkRule -Name $rule.Name -SourceAddress
$rule.SourceAddresses -DestinationAddress $rule.DestinationAddresses -DestinationPort $rule.DestinationPorts -
Protocol $rule.Protocols
                Write-Host "Created network rule " $firewallPolicyNetRule.Name
                $firewallPolicyNetRules += $firewallPolicyNetRule
            }
            $fwpNetRuleCollection = New-AzFirewallPolicyFilterRuleCollection -Name $src.Name -Priority $src.Priority -
ActionType $src.Action.Type -Rule $firewallPolicyNetRules

```



```

    Write-Host "Created NetworkRuleCollection " $fwpNetRuleCollection.Name
  }
  $firewallPolicyNetRuleCollections += $fwpNetRuleCollection
}
$netRuleGroup = New-AzFirewallPolicyRuleCollectionGroup -Name $DefaultNetRuleCollectionGroupName -Priority
$NetworkRuleGroupPriority -RuleCollection $firewallPolicyNetRuleCollections -FirewallPolicyObject $fwp
Write-Host "Created NetworkRuleCollectionGroup " $netRuleGroup.Name
}

#Translate NatRuleCollection
# Hierarchy for NAT rule collection is different for AZFW and FirewallPolicy. In AZFW you can have a
NatRuleCollection with multiple NatRules
# where each NatRule will have its own set of source , dest, translated IPs and ports.
# In FirewallPolicy a NatRuleCollection has a a set of rules which has one condition (source and dest IPs and
Ports) and the translated IP and ports
# as part of NatRuleCollection.
# So when translating NAT rules we will have to create separate ruleCollection for each rule in AZFW and every
ruleCollection will have only 1 rule.

Write-Host "creating " $azfw.NatRuleCollections.Count " network rule collections"
If ($azfw.NatRuleCollections.Count -gt 0) {
  $firewallPolicyNatRuleCollections = @()
  $priority = 100
  ForEach ($rc in $azfw.NatRuleCollections) {
    If ($rc.Rules.Count -gt 0) {
      Write-Host "creating " $rc.Rules.Count " nat rules for collection " $rc.Name
      ForEach ($rule in $rc.Rules) {
        $firewallPolicyNatRule = New-AzFirewallPolicyNetworkRule -Name $rule.Name -SourceAddress
$rule.SourceAddresses -DestinationAddress $rule.DestinationAddresses -DestinationPort $rule.DestinationPorts -
Protocol $rule.Protocols
        Write-Host "Created nat rule " $firewallPolicyNatRule.Name
        $natRuleCollectionName = $rc.Name+$rule.Name
        $fwpNatRuleCollection = New-AzFirewallPolicyNatRuleCollection -Name $natRuleCollectionName -Priority
$priority -ActionType $rc.Action.Type -Rule $firewallPolicyNatRule -TranslatedAddress $rule.TranslatedAddress
-TranslatedPort $rule.TranslatedPort
        $priority += 1
        Write-Host "Created NatRuleCollection " $fwpNatRuleCollection.Name
        $firewallPolicyNatRuleCollections += $fwpNatRuleCollection
      }
    }
  }
  $natRuleGroup = New-AzFirewallPolicyRuleCollectionGroup -Name $DefaultNatRuleCollectionGroupName -Priority
$NatRuleGroupPriority -RuleCollection $firewallPolicyNatRuleCollections -FirewallPolicyObject $fwp
  Write-Host "Created NatRuleCollectionGroup " $natRuleGroup.Name
}

```

Next steps

Learn more about Azure Firewall Manager deployment: [Azure Firewall Manager Preview deployment overview](#).

Azure Resource Manager overview

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

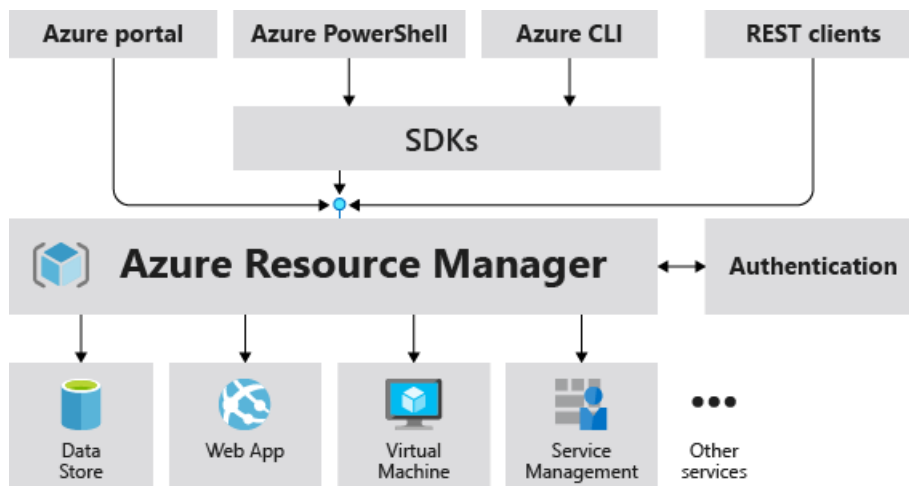
Azure Resource Manager is the deployment and management service for Azure. It provides a management layer that enables you to create, update, and delete resources in your Azure subscription. You use management features, like access control, locks, and tags, to secure and organize your resources after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

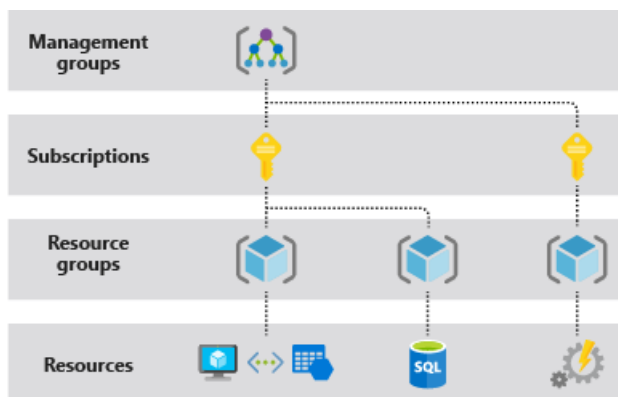
The benefits of using Resource Manager

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](#).

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 of 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

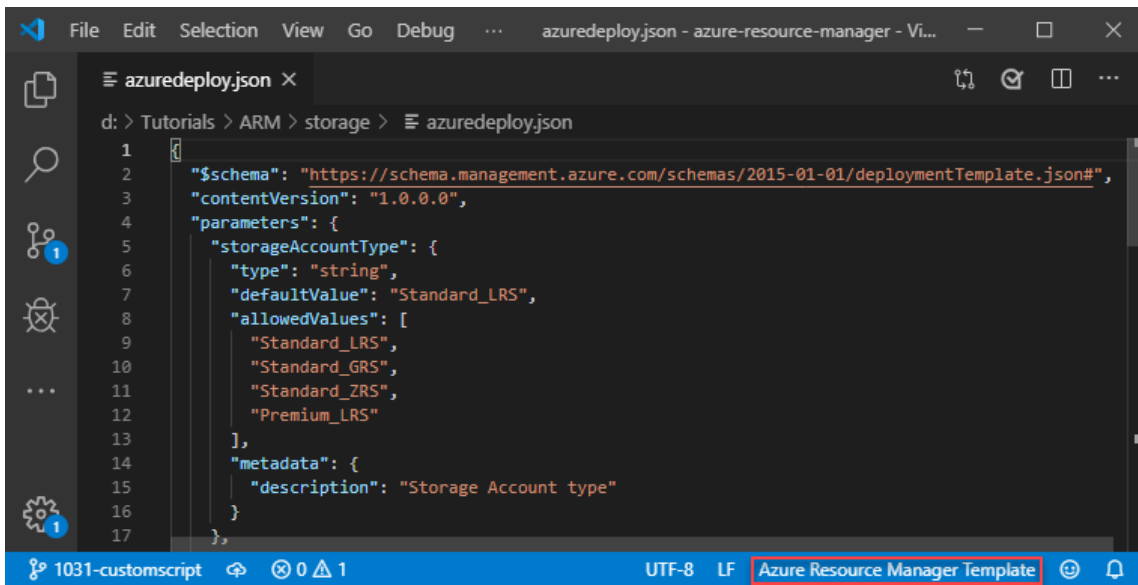
For inline comments, you can use either `//` or `/* ... */` but this syntax doesn't work with all tools. You can't use the portal template editor to work on templates with inline comments. If you add this style of comment, be sure the tools you use support inline JSON comments.

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](#).