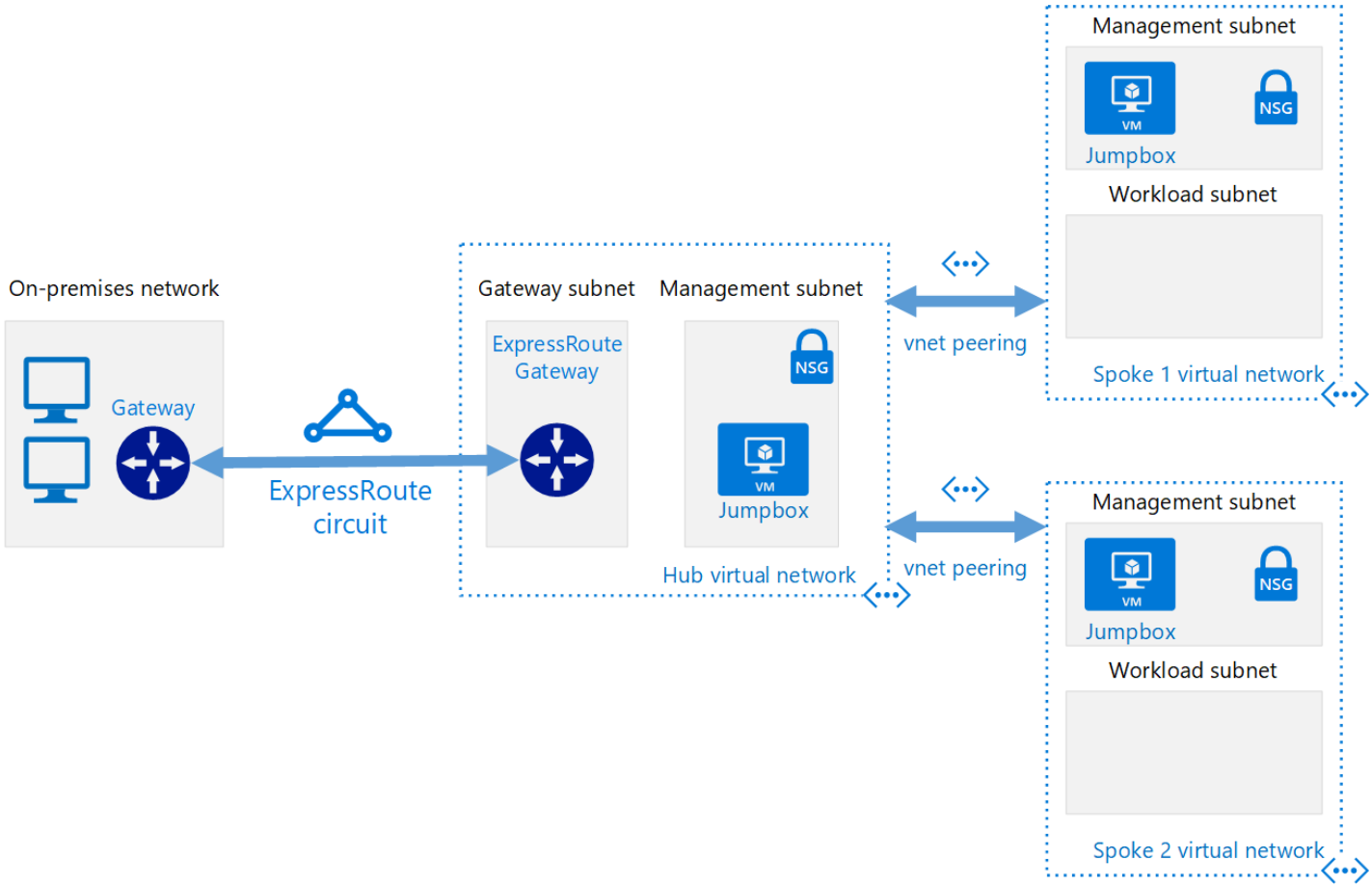
Implement a hub-spoke network topology in Azure

06/05/2019 • 10 minutes to read • Contributors all

In this article

Architecture
Recommendations
Considerations
Deploy the solution

This reference architecture shows how to implement a hub-spoke topology in Azure. The *hub* is a virtual network (VNet) in Azure that acts as a central point of connectivity to your on-premises network. The *spokes* are VNets that peer with the hub, and can be used to isolate workloads. Traffic flows between the on-premises datacenter and the hub through an ExpressRoute or VPN gateway connection. Deploy this solution.



Download a Visio file of this architecture

The benefits of this topology include:

- **Cost savings** by centralizing services that can be shared by multiple workloads, such as network virtual appliances (NVAs) and DNS servers, in a single location.
- **Overcome subscriptions limits** by peering VNets from different subscriptions to the central hub.
- **Separation of concerns** between central IT (SecOps, InfraOps) and workloads (DevOps).

Typical uses for this architecture include:

- Workloads deployed in different environments, such as development, testing, and production, that require shared services such as DNS, IDS, NTP, or AD DS. Shared services are placed in the hub VNet, while each environment is deployed to a spoke to maintain isolation.
- Workloads that do not require connectivity to each other, but require access to shared services.
- Enterprises that require central control over security aspects, such as a firewall in the hub as a DMZ, and segregated management for the workloads in each spoke.

Architecture

The architecture consists of the following components.

- **On-premises network**. A private local-area network running within an organization.
- **VPN device**. A device or service that provides external connectivity to the on-premises network. The VPN device may be a hardware device, or a software solution such as the Routing and Remote Access Service (RRAS) in Windows Server 2012. For a list of supported VPN appliances and information on configuring selected VPN appliances for connecting to Azure, see About VPN devices for Site-to-Site VPN Gateway connections.
- **VPN virtual network gateway or ExpressRoute gateway**. The virtual network gateway enables the VNet to connect to the VPN device, or ExpressRoute circuit, used for connectivity with your on-premises network. For more information, see Connect an on-premises network to a Microsoft Azure virtual network.

> **Note**
>
> The deployment scripts for this reference architecture use a VPN gateway for connectivity, and a VNet in Azure to simulate your on-premises network.

- **Hub VNet**. Azure VNet used as the hub in the hub-spoke topology. The hub is the central point of connectivity to your on-premises network, and a place to host services that can be consumed by the different workloads hosted in the spoke VNets.
- **Gateway subnet**. The virtual network gateways are held in the same subnet.
- **Spoke VNets**. One or more Azure VNets that are used as spokes in the hub-spoke topology. Spokes can be used to isolate workloads in their own VNets, managed separately from other spokes. Each workload might include multiple tiers, with multiple subnets connected through Azure load balancers. For more information about the application infrastructure, see Running Windows VM workloads and Running Linux VM workloads.
- **VNet peering**. Two VNets can be connected using a peering connection. Peering connections are non-transitive, low latency connections between VNets. Once peered, the VNets exchange traffic by using the Azure backbone, without the need for a router. In a hub-spoke network topology, you use VNet peering to connect the hub to each spoke. You can peer virtual networks in the same region, or different regions. For more information, see Requirements and constraints.

> **Note**
>
> This article only covers Resource Manager deployments, but you can also connect a classic VNet to a Resource Manager VNet in the same subscription. That way, your spokes can host classic deployments and still benefit from services shared in the hub.

Recommendations

The following recommendations apply for most scenarios. Follow these recommendations unless you have a specific requirement that overrides them.

Resource groups

The hub VNet, and each spoke VNet, can be implemented in different resource groups, and even different subscriptions. When you peer virtual networks in different subscriptions, both subscriptions can be associated to the same or different Azure Active Directory tenant. This allows for a decentralized management of each workload, while sharing services maintained in the hub VNet.

VNet and GatewaySubnet

Create a subnet named *GatewaySubnet*, with an address range of /27. This subnet is required by the virtual network gateway. Allocating 32 addresses to this subnet will help to prevent reaching gateway size limitations in the future.

For more information about setting up the gateway, see the following reference architectures, depending on your connection type:

- Hybrid network using ExpressRoute
- Hybrid network using a VPN gateway

For higher availability, you can use ExpressRoute plus a VPN for failover. See Connect an on-premises network to Azure using ExpressRoute with VPN failover.

A hub-spoke topology can also be used without a gateway, if you don't need connectivity with your on-premises network.

VNet peering

VNet peering is a non-transitive relationship between two VNets. If you require spokes to connect to each other, consider adding a separate peering connection between those spokes.

However, if you have several spokes that need to connect with each other, you will run out of possible peering connections very quickly due to the limitation on number of VNets peerings per VNet. In this scenario, consider using user defined routes (UDRs) to force traffic destined to a spoke to be sent to Azure Firewall or an NVA acting as a router at the hub VNet. This will allow the spokes to connect to each other.

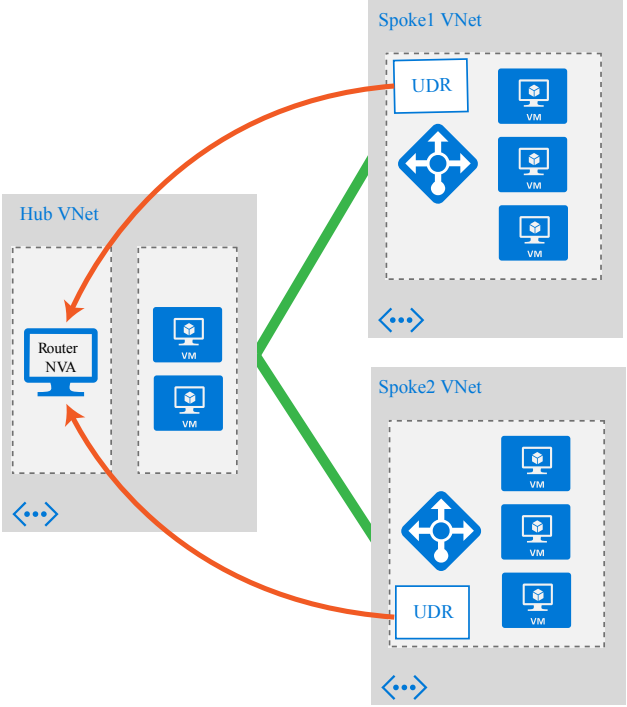
You can also configure spokes to use the hub VNet gateway to communicate with remote networks. To allow gateway traffic to flow from spoke to hub, and connect to remote networks, you must:

- Configure the VNet peering connection in the hub to **allow gateway transit**.
- Configure the VNet peering connection in each spoke to **use remote gateways**.
- Configure all VNet peering connections to **allow forwarded traffic**.

Considerations

Spoke connectivity

If you require connectivity between spokes, consider deploying Azure Firewall or an NVA for routing in the hub, and using UDRs in the spoke to forward traffic to the hub. The deployment steps below include an optional step that sets up this configuration.



In this scenario, you must configure the peering connections to **allow forwarded traffic**.

Also consider what services are shared in the hub, to ensure the hub scales for a larger number of spokes. For instance, if your hub provides firewall services, consider the bandwidth limits of your firewall solution when adding multiple spokes. You might want to move some of these shared services to a second level of hubs.

Deploy the solution

A deployment for this architecture is available on GitHub. It uses VMs in each VNet to test connectivity. There are no actual services hosted in the **shared-services** subnet in the **hub VNet**.

The deployment creates the following resource groups in your subscription:

- hub-vnet-rg
- onprem-jb-rg
- onprem-vnet-rg
- spoke1-vnet-rg
- spoke2-vent-rg

The template parameter files refer to these names, so if you change them, update the parameter files to match.

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Install Azure CLI 2.0.
3. Install Node and NPM
4. Install the Azure building blocks npm package.

```bash
npm install -g @mspnp/azure-building-blocks
```

5. From a command prompt, bash prompt, or PowerShell prompt, sign into your Azure account as follows:

```bash
```

```
az login
```

Deploy the simulated on-premises datacenter

To deploy the simulated on-premises datacenter as an Azure VNet, follow these steps:

1. Navigate to the `hybrid-networking/hub-spoke` folder of the reference architectures repository.
2. Open the `onprem.json` file. Replace the values for `adminUsername` and `adminPassword`.

```JSON
"adminUsername": "<user name>",
"adminPassword": "<password>",
```

3. (Optional) For a Linux deployment, set `osType` to `Linux`.
4. Run the following command:

```bash
azbb -s <subscription_id> -g onprem-vnet-rg -l <location> -p onprem.json --deploy
```

5. Wait for the deployment to finish. This deployment creates a virtual network, a virtual machine, and a VPN gateway. It can take about 40 minutes to create the VPN gateway.

Deploy the hub VNet

To deploy the hub VNet, perform the following steps.

1. Open the `hub-vnet.json` file. Replace the values for `adminUsername` and `adminPassword`.

```JSON
"adminUsername": "<user name>",
"adminPassword": "<password>",
```

2. (Optional) For a Linux deployment, set `osType` to `Linux`.
3. Find both instances of `sharedKey` and enter a shared key for the VPN connection. The values must match.

```JSON
"sharedKey": "",
```

4. Run the following command:

```bash
azbb -s <subscription_id> -g hub-vnet-rg -l <location> -p hub-vnet.json --deploy
```

5. Wait for the deployment to finish. This deployment creates a virtual network, a virtual machine, a VPN gateway, and a connection to the gateway. It can take about 40 minutes to create the VPN gateway.

Test connectivity to the hub VNet — Windows deployment

To test connectivity from the simulated on-premises environment to the hub VNet using Windows VMs, follow these steps:

1. Use the Azure portal to find the VM named `jb-vm1` in the `onprem-jb-rg` resource group.
2. Click `Connect` to open a remote desktop session to the VM. Use the password that you specified in the `onprem.json` parameter file.
3. Open a PowerShell console in the VM, and use the `Test-NetConnection` cmdlet to verify that you can connect to the jumpbox VM in the hub VNet.

```PowerShell
Test-NetConnection 10.0.0.68 -CommonTCPPort RDP
```

The output should look similar to the following:

```PowerShell
ComputerName     : 10.0.0.68
RemoteAddress    : 10.0.0.68
RemotePort       : 3389
InterfaceAlias   : Ethernet 2
SourceAddress    : 192.168.1.000
TcpTestSucceeded : True
```

> **Note**
>
> By default, Windows Server VMs do not allow ICMP responses in Azure. If you want to use `ping` to test connectivity, you need to enable ICMP traffic in the Windows Advanced Firewall for each VM.

Test connectivity to the hub VNet — Linux deployment

To test connectivity from the simulated on-premises environment to the hub VNet using Linux VMs, follow these steps:

1. Use the Azure portal to find the VM named `jb-vm1` in the `onprem-jb-rg` resource group.
2. Click `Connect` and copy the `ssh` command shown in the portal.
3. From a Linux prompt, run `ssh` to connect to the simulated on-premises environment. Use the password that you specified in the `onprem.json` parameter file.
4. Use the `ping` command to test connectivity to the jumpbox VM in the hub VNet:

```shell
ping 10.0.0.68
```

Deploy the spoke VNets

To deploy the spoke VNets, perform the following steps.

1. Open the `spoke1.json` file. Replace the values for `adminUsername` and `adminPassword`.

```JSON
```

```
"adminUsername": "<user name>",
"adminPassword": "<password>",
```

2. (Optional) For a Linux deployment, set `osType` to `Linux`.

3. Run the following command:

```bash
azbb -s <subscription_id> -g spoke1-vnet-rg -l <location> -p spoke1.json --deploy
```

4. Repeat steps 1-2 for the `spoke2.json` file.

5. Run the following command:

```bash
azbb -s <subscription_id> -g spoke2-vnet-rg -l <location> -p spoke2.json --deploy
```

6. Run the following command:

```bash
azbb -s <subscription_id> -g hub-vnet-rg -l <location> -p hub-vnet-peering.json --deploy
```

Test connectivity to the spoke VNets — Windows deployment

To test connectivity from the simulated on-premises environment to the spoke VNets using Windows VMs, perform the following steps:

1. Use the Azure portal to find the VM named `jb-vm1` in the `onprem-jb-rg` resource group.

2. Click `Connect` to open a remote desktop session to the VM. Use the password that you specified in the `onprem.json` parameter file.

3. Open a PowerShell console in the VM, and use the `Test-NetConnection` cmdlet to verify that you can connect to the jumpbox VMs in the spoke VNets.

```PowerShell
Test-NetConnection 10.1.0.68 -CommonTCPPort RDP
Test-NetConnection 10.2.0.68 -CommonTCPPort RDP
```

Test connectivity to the spoke VNets — Linux deployment

To test connectivity from the simulated on-premises environment to the spoke VNets using Linux VMs, perform the following steps:

1. Use the Azure portal to find the VM named `jb-vm1` in the `onprem-jb-rg` resource group.

2. Click `Connect` and copy the `ssh` command shown in the portal.

3. From a Linux prompt, run `ssh` to connect to the simulated on-premises environment. Use the password that you specified in the `onprem.json` parameter file.

4. Use the `ping` command to test connectivity to the jumpbox VMs in each spoke:

```bash
ping 10.1.0.68
ping 10.2.0.68
```

Add connectivity between spokes

This step is optional. If you want to allow spokes to connect to each other, use Azure Firewall to force traffic from spokes to the router when trying to connect to another spoke. To deploy Azure Firewall, along with user-defined routes (UDRs) to allow the two spoke VNets to connect, perform the following steps:

1. Add a subnet for Azure Firewall to the hub virtual network.

```bash
az network vnet subnet create -g hub-vnet-rg --vnet-name hub-vnet -n AzureFirewallSubnet --address-prefixes 10.0.0.128/26
```

2. Deploy Azure Firewall:

```bash
az group deployment create -g hub-vnet-rg --template-file hub-firewall.json
```

3. Run the following command to get the privateIPAddress of the firewall created in step 2:

```bash
az resource show -g hub-vnet-rg -n hub-firewall --resource-type Microsoft.Network/azureFirewalls --query properties.ipConfigurations[0].properties.privateIPAddress
```

4. Edit the hub-firewall-routes.json file and replace all occurrences of `<azure_firewall_private_ip>` with the IP Address from the previous command. Save hub-firewall-routes.json and then run the following command.

```bash
azbb -s <subscription_id> -g hub-vnet-rg -l <location> -p hub-firewall-routes.json --deploy
```

5. Run the following command to disable BGP route propagation for the route tables associated with the spoke subnets:

```bash
az network route-table update -g hub-vnet-rg -n spoke1-rt --disable-bgp-route-propagation true
az network route-table update -g hub-vnet-rg -n spoke2-rt --disable-bgp-route-propagation true
```

To verify connectivity, perform the following steps:

1. Log into the VM named `jb-vm1` in the `onprem-jb-rg` resource group.
2. From this login session, log into the jumpbox VM for spoke-1. The private IP address is 10.1.0.68.
3. Use the `Test-NetConnection` cmdlet (Windows) or ping command (Linux) to test connectivity to 10.2.0.68, which is the jumpbox VM for spoke-2.