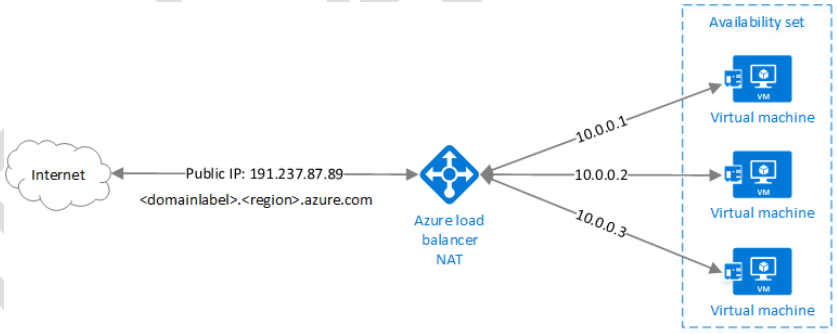
Agenda: Implement Advanced Virtual Networking

- Load Balancing
  - Configure external and internal Azure Load balancer
  - Load Balancing Rules
  - Implement front end IP configuration
  - Azure Application Gateway
  - Azure Traffic Manager
- Integrate on-premises network with Azure virtual network
  - Point-to-Site Network using Azure VPN Gateway
  - Site-to-Site VPN
  - Express Route Solution
- Monitor and Manage Networking
  - Verify, Manage and Monitor on-premises connectivity;
  - Use network resource monitoring and Network Watcher
  - Manage external networking and virtual network connectivity

Azure Load Balancer

The Azure Load Balancer delivers **high availability** and **network performance** to your applications. It is a **Layer 4** (TCP, UDP) load balancer that distributes incoming traffic among healthy service instances in virtual machines defined in a load-balanced set.



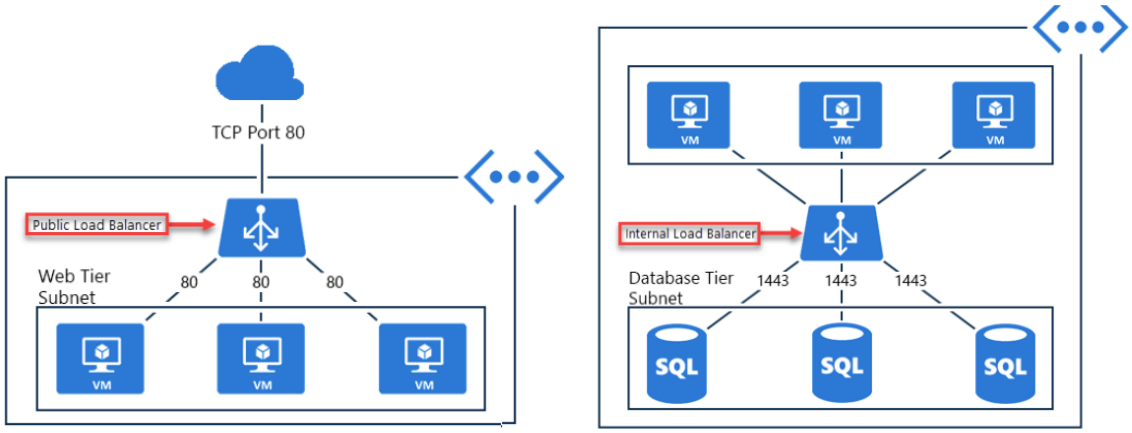
It can be configured to:

- Load balance incoming Internet traffic to virtual machines. This is called **Internet-facing** load balancing.
- Load balance traffic between virtual machines in a virtual network, between virtual machines in cloud services,
- Load balance between on-premises computers and virtual machines in a cross-premises virtual network. This is called **internal load balancing** (ILB).
- Forward external traffic to a specific virtual machine using NAT Rule

All resources in the cloud need a public IP address to be reachable from the Internet. The cloud infrastructure in Microsoft Azure uses non-routable IP addresses for its resources. It uses network address translation (NAT) with public IP addresses to communicate to the Internet.

Public load balancer. You can use an external load balancer to provide high availability for IaaS VMs and PaaS role instances accessed **from the public Internet**.

Internal load balancer. You can use an internal load balancer to provide high availability for IaaS VMs and PaaS role instances accessed **from other services** in your Vnet



Hash-based distribution:

Azure Load Balancer uses a **hash-based distribution algorithm**. By default, it uses a **5-tuple** hash composed of **source IP, source port, destination IP, destination port, and protocol type** to map traffic to available servers. It provides **stickiness** only *within* a transport session. Packets in the same TCP or UDP session will be directed to the same instance behind the load-balanced endpoint. When the client closes and reopens the connection or starts a new session from the same source IP, the source port changes. This may cause the traffic to go to a different endpoint.

Steps to setup load balancer in Azure Portal

1. Azure Portal → More Services → Load Balancer → +Add
2. Type=Public, Public IP address, . . .
3. SKU = Standard

   - A standalone virtual machine resource, availability set resource, or virtual machine scale set resource can reference one SKU, never both.
   - There is no charge for the Basic load balancer. The Standard load balancer is charged based on number of rules and data processed.

4. → Create

5. Create a **back-end address pool** → Name=LB-backend, Select Choose an availability set → Select WebServer-availablityset

	Standard SKU	**Basic SKU**
Backend pool endpoints	Any VM in a single virtual network, including a blend of VMs, availability sets, and VM scale sets.	VMs in a single availability set or VM scale set.
Instances	1000 Instances	100 Instances

Standard SKU LB

Requires IP Address also should be Standard

Individual VM can be added

VM can be in different VNet (but same region)

Basic SKU

Requires IP Address also should be Basic

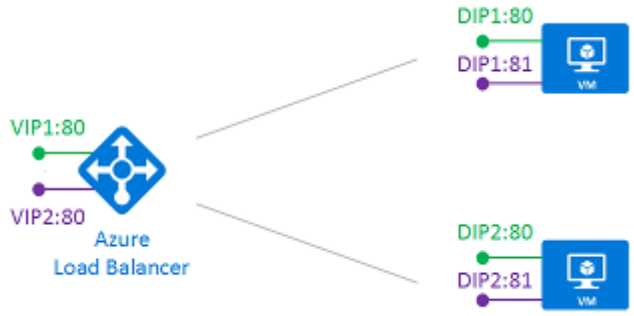
Availability Set

Individual VM cannot be added

6. Choose the Virtual Machines → Select WebVM1 and WebVM2
7. Create a Probe, Name=**HealthProbe**, Protocol=HTTP, Port=80, Path=/ . . . → OK
8. Create Load Balancer **Rule**, Name=HTTP, Protocol=TCP, Port=80, Backend port=80, Backend pool=LB-backend, Probe=HealthProbe, Session persistence=None, . . . → OK

To disable direct traffic to Web Server.

9. Detach Public IP from the NIC of the all the Web Servers
10. For Load Balancer, Add Frontend IP (IP detached in previous step)
11. Create NAT rules for remote desktop access/SSH for virtual machines behind the load balancer.



*DIP: Destination IP address

Create an Internet facing load balancer in RM by using PowerShell

```
https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-get-started-internet-arm-ps
$NRPLB = New-AzureRmLoadBalancer -ResourceGroupName NRP-RG -Name NRP-LB -Location 'West US' -
FrontendIpConfiguration $frontendIP -InboundNatRule $inboundNATRule1,$inboundNatRule2 -LoadBalancingRule
$lbrule -BackendAddressPool $beAddressPool -Probe $healthProbe
```

Understanding and Creating Availability Sets.

There are two types of Microsoft Azure platform events that can affect the availability of your virtual machines:

1. **Planned maintenance events** are periodic updates made by Microsoft to the underlying Azure platform to improve overall reliability, performance, and security of the platform infrastructure that your virtual machines run on.
2. **Unplanned maintenance events** occur when the hardware or physical infrastructure underlying your virtual machine has faulted in some way. This may include local network failures, local disk failures, or other rack level failures.

Fault Domains:

- A fault domain is a **physical point** of failure. Think of a computer (or a rack of servers) that is physically plugged in to a power outlet in one location (Unplanned update). If a power outage happens, that computer goes offline.
- When creating a new virtual machine instance, Azure will automatically place that instance in a new Fault Domain. This ensures that if you have 2 instances of a service, they cannot be in the same fault domain.

Update Domains:

- Whereas Fault Domains are a physical separation, Update Domains are a **logical separation**. Update domains exist so when Microsoft rolls out a new software feature or bug fix (Planned update), each update domain is upgraded at different times. This ensures that if you have at least 1 instances, your service will never go down as the result of an upgrade.
- Azure services can have up to 5 upgrade domains by default (max of 20). When you create a new service instance, Azure automatically places it in the next update domain. If you have more than 5 instances, 7 for example, upgrade domains 0-1 will have 2 instances and upgrade domains 2-4 will have 1 instance.

Follow best practices when you design your application for high availability.

1. Configure multiple virtual machines in an availability set for redundancy.
2. Configure each application tier into separate availability sets.
3. Combine a load balancer with availability sets.

Important Note:

- It is critical to understand that it is not possible to add an existing Azure virtual machine to an availability set. You need to specify that a virtual machine will be part of an availability set when you provision it.
- **There is also a limit of 100 virtual machines in each Azure availability set when you use Azure Resource Manager.**
- If the VM you wish to **resize** is part of an availability set, then you must **stop all VMs** in the availability set before changing the size of any VM in the availability set. The reason all VMs in the availability set must be stopped before performing the resize operation is that all running VMs in the availability set must be using the **same physical hardware cluster**. Therefore, if a change of physical hardware cluster is required to change the VM size then all VMs must be first stopped and then restarted one-by-one to a different physical hardware clusters.

Both FDs and UDs are assigned in the order that Azure discovers them as they are provisioned. So if you provision machines in the order Srv0, Srv1, Srv2, Srv3, Srv4, Srv5, Srv6, Srv7, Srv8, Srv9, Srv10, Srv11 you'll end up with a table that looks like this:

VM	Fault Domain	Update Domain
Srv0	0	0
Srv1	1	1
Srv2	2	2
Srv3	0	3
Srv4	1	4
Srv5	2	0
Srv6	0	1
Srv7	1	2
Srv8	2	3
Srv9	0	4
Srv10	1	0
Srv11	2	1
Srv12	0	2

Availablity Set can be either created using Portal or PowerShell Commands

```
New-AzureRmAvailabilitySet –ResourceGroupName –DemoAvailSet –Location "East US" –
PlatformUpdateDomainCount 5 –PlatformFaultDomainCount 3
```

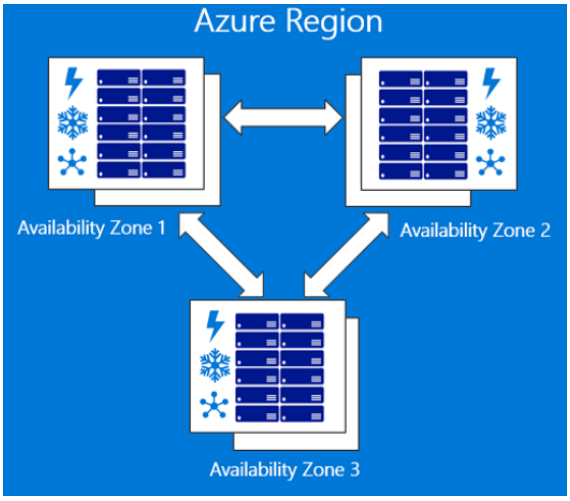
Note: However, because Availability Sets cannot span regions, if this region were to fail we would lose access to the whole application.

Availability Zones

- Availability Zones are unique physical locations within an Azure region.
- Each zone is made up of one or more datacenters equipped with independent power, cooling, and networking.
- To ensure resiliency, there's a minimum of **three separate zones** in all **enabled regions**.
- The **physical separation** of Availability Zones within a region protects applications and data from **datacenter failures**.
- Zone-redundant services replicate your applications and data across Availability Zones to protect from single-points-of-failure.
- With Availability Zones, Azure offers industry best 99.99% VM uptime SLA.
- An Availability Zone in an Azure region is a **combination of a fault domain and an update domai**n. So, if you're deploying a web tier consisting of 2 VMs in Ireland, you can now make sure that VM1 is placed in Availability Zone 1 and VM2 is placed in Availability Zone 2. If zone 1 was to fail, you (and your customers) would still be able to access VM2 in AZ2.
- This means your service **won't** have to run from a **separate Azure region** and will be faster as a result. This is especially useful if your customers are concentrated in a single region.
- Availability Zones are also ideal if you must obey regulatory requirements and laws that require your data/services to be highly available inside a single Azure Region.

Build high-availability into your application architecture by co-locating your compute, storage, networking, and data resources within a zone and replicating in other zones. Azure services that support Availability Zones fall into two categories:

- **Zonal services –** you pin the resource to a specific zone (for example, virtual machines, managed disks, IP addresses), or
- **Zone-redundant services** – platform replicates automatically across zones (for example, zone-redundant storage, SQL Database).



Regions that support Availability Zones

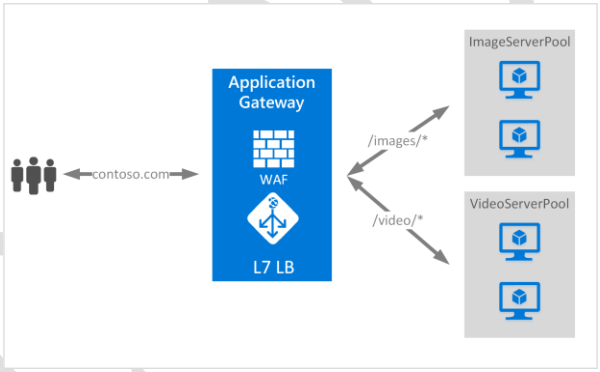
- Central US
- East US
- East US 2
- France Central
- North Europe
- Southeast Asia
- UK South *
- West Europe
- West US 2

Services that support Availability Zones

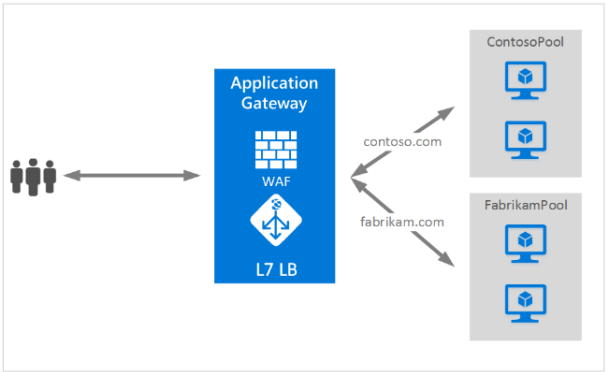
- Linux Virtual Machines
- Windows Virtual Machines
- Virtual Machine Scale Sets
- Managed Disks
- Standard Load Balancer *
- Standard public IP address *
- Zone-redundant storage
- SQL Database
- Event Hubs
- Service Bus (Premium Tier Only)
- VPN Gateway
- ExpressRoute
- Application Gateway (preview)

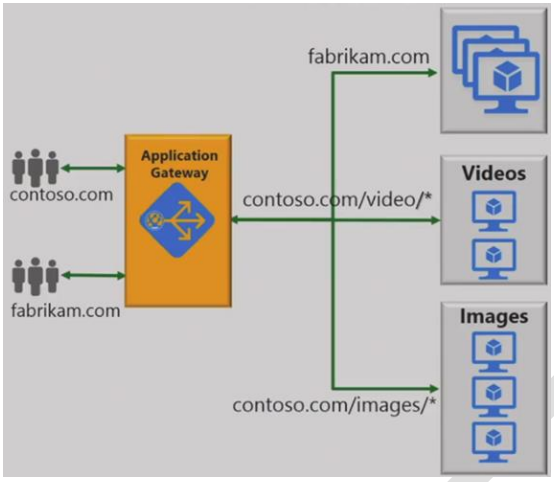
Azure Application Gateway

- Azure Application Gateway is a **web traffic load balancer** that enables you to manage traffic to your **web appli cations**.
- Azure Application Gateway is a **OSI layer-7 load balancer**. It provides failover, performance-routing HTTP requests between different servers, whether they are on the cloud or on-premises.
- Application Gateway provides many **Application Delivery Controller** (ADC) features including
  1. **HTTP load balancing**: Application Gateway provides **round robin load balancing**. Load balancing is done at Layer 7 and is used for HTTP(S) traffic only.
  2. **Cookie-based Session Affinity**: This feature is useful when you want to keep a user session on the same back-end.
  3. **Health Monitoring**: Application gateway provides default health monitoring of backend resources and custom probes to monitor for more specific scenarios.
  4. **Secure Sockets Layer (SSL) offload**. This feature takes the costly task of decrypting HTTPS traffic off your web servers
  5. **URL Path based Routing**: This feature provides the capability to use different back-end servers for different traffic.

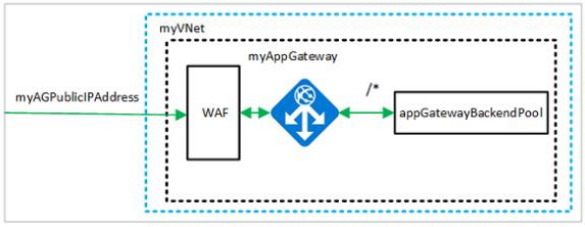


  6. **Support for Multiple Site Hosting**: Application gateway allows for you to consolidate up to **20 websites** on a single application gateway.





7. **Web Application Firewall**: The web application firewall (WAF) in Azure Application Gateway protects web applications from common **web-based attacks** like SQL injection, cross-site scripting attacks, and session hijacks.

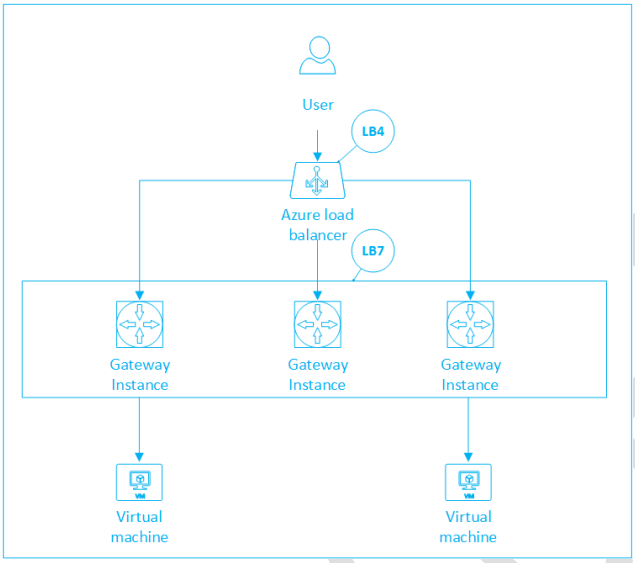


Some of the **vulnerabilities that WAF currently protects you against** are:

1. SQL injection
2. Cross-site scripting
3. Common web attacks, including command injection, remote file inclusion attack, and more.
4. HTTP protocol violations
5. HTTP protocol anomalies
6. Denial of service (DoS), including HTTP flooding and slow HTTP DoS
7. Bots, crawlers, and scanners
8. Common IIS and Apache misconfigurations

- Application Gateway can be configured as **internet** facing gateway, **internal** only gateway, or a combination of both.
- You can associate a **public IP** address with an Azure Application Gateway, by assigning it to the gateway's frontend configuration. This public IP address serves as a load-balanced IP. Currently, you can only assign a ***dynamic*** **public IP** address to an application gateway frontend configuration.
- An endpoint (public IP or internal IP) is associated and used for ingress network traffic. This VIP or ILB IP is provided by Azure Load Balancer working at the transport level (TCP/UDP) and having all incoming

network traffic being load balanced to the application gateway worker instances. The application gateway then routes the HTTP/HTTPS traffic based on its configuration whether it's a virtual machine, cloud service, internal or an external IP address.



Gateway Sizes and Instances

Back-end page response size	Small	Medium	Large
6KB	7.5 Mbps	13 Mbps	50 Mbps
100KB	35 Mbps	100 Mbps	200 Mbps

Note: Small instance sizes are intended for development and testing scenarios.

Limits:

You can create up to **50 application gateways** per subscription, and each application gateway can have up to 10 instances each. Each application gateway can consist of 20 http listeners, 20 frontend ports, and 20 backend pools with 100 backend servers per pool. Note that all such metrics can change over time, so be sure to consult the documentation.

Steps: (These steps will not work with trail account)

1. Create two VM – WebVM1 and WebVM2 and install IIS in both of them.
2. Create an Appication Gateway (You should have a VNet having Subnet (Gateway dedicated) without any resources)

   Note: Creating an Application Gateway would take around 20 mins.
3. Add WebVM1 / Add WebVM2 to BackendPool of the Gatewat
4. Find the Public IP address of Gateway from the Overview blade.
5. **Observation:** In browser open http://<PublicIPOfGateway>/

   Note: Load is divided between WebVM1 and WebVM2 in round robin order

6. Go to HTTPSettings → Select appGatewayBackendHttpSettings → Cookie Based Affinity = Enabled
7. **Observation:**In browser open http://<PublicIPOfGateway>/

   Note: From a give client load is **always** forwarded to either WebVM1 or WebVM2.

NOTE: ENSURE THAT THE SUBNET WITH APPLICATION GATEWAY IS NOT ASSOCIATED WITH ANY NSG OR NSG MUST BE COFIGURED WITH PROPER RULES.

Explanation: How a request landing on listerner is binded to any one server in the backendpool

HTTP Settings

appGatewayBackendHttpSettings = HTTP / 80 - Cookie = false

appGatewayBackendHttpsSettings = HTTPS / 443 - Cookie = true

Frontend IP cofiguration

appGatewayFrontendIP = 104.211.219.165 (Public)

BackendPool

appGatewayBackendPool, Associated Rule=rule1

Web1-vm & Web2-vm

SiteAPool

Web3-VM & Web3-VM

Listeners

appGatewayHttpListener (Basic Listener)

Frontend IP configuration = appGatewayFrontendIP

appGatewayFrontendPort = 80

Associated rule = rule1

AppGatewayHttpListener80 (Basic Listener)

Frontend IP configuration = appGatewayFrontendIP

appGatewayFrontendPort = 8080

Associated rule = rule1

SiteADomainBasedListener (Multisite Listener)

Frontend IP configuration = appGatewayFrontendIP

appGatewayFrontendPort = 80,

Hostname: www.SiteA.com

Associated rule = rule2

Rules

rule1

Listener = appGatewayHttpListener

Backend Pool = appGatewayBackendPool

HTTP settings = appGatewayBackendHttpSettings

rule2

Listener = SiteADomainBasedListener (Hostname: www.SiteA.com)

Backend Pool = SiteAPool

HTTP settings = appGatewayBackendHttpSettings

Lifecycle of request in browser:

http://104.211.219.165:80/index.html (Web1-VM or Web2-VM)

1. Because of IP (appGatewayFrontendIP) and Port (appGatewayFrontendPort), Listener is selected (appGatewayHttpListener)
2. Based on Associated rule, the rule is selected (rule1)
3. As per rule1, Listerner (appGatewayHttpListener) => BackendPool (appGatewayBackendPool) and HTTP Settings should be appGatewayBackendHttpSettings
4. Now the request will be forwarded to any one machine in the backendpool (Either Web-vm or Web2-vm) and Affinity will be managed as in appGatewayBackendHttpSettings

http://www.SizeA.com:80/ (Web3-vm or Web4-vm)

1. Because of Host Name (appGatewayFrontendIP) and Port (appGatewayFrontendPort), Listener is selected (SiteADomainBasedListener)
2. Based on Associated rule, the rule is selected (rule1)
3. As per rule1, Listerner (SiteADomainBasedListener) => BackendPool (SiteAPool) and HTTP Settings should be appGatewayBackendHttpSettings
4. Now the request will be forwarded to any one machine in the backendpool (Either Web3-vm or Web4-vm) and Affinity will be managed as in appGatewayBackendHttpSettings

Path Based Rules:

RDP to WebVM3 and create c:\inetput\wwwroot\images\smiley.jpg

RDP to WebVM4 and create c:\inetput\wwwroot\images\smiley.jpg

RDP to WebVM5 and create c:\inetput\wwwroot\videos\introduction.mp4

RDP to WebVM6 and create c:\inetput\wwwroot\videos\introduction.mp4

Create Two BackendPools

1. ImagesBackendPool with WebVM3 & WebVM4

2. VideosBackendPool with WebVM5 & WebVM6

Create Path Based Rule

with two entries mapping

1. /images/* to ImagesBackendPool

2. /videos/* to VideosBackendPool

Azure Load Balancer and Application Gateway route network traffic to endpoints but they have different usage scenarios to which traffic to handle. The following table helps understanding the difference between the two load balancers:+

Type	Azure Load Balancer	Application Gateway
Technology	Transport Layer (level 4)	Application Layer (level 7)
Protocols	Any (TCP / UDP)	HTTP, HTTPS, and WebSockets
IP reservation	Supported	Not supported
Load balancing mode	5-tuple(source IP, source port, destination IP, destination port, protocol type)	Round Robin Routing based on URL (Path and Domain name)
Health probes	Default: probe interval - 15 secs. Taken out of rotation: 2 Continuous failures.	Idle probe interval 30 secs. Taken out after 5 consecutive live traffic failures or a single probe failure in idle mode.
SSL offloading	Not supported	Supported
Url-based routing	Not supported	Supported

Azure Traffic Manager

Microsoft Azure Traffic Manager allows you to control the distribution of user traffic for service endpoints in **different datacenters around the world.**

Service endpoints supported by Traffic Manager include Azure VMs, Web Apps, and cloud services. You can also use Traffic Manager with external, non-Azure endpoints.

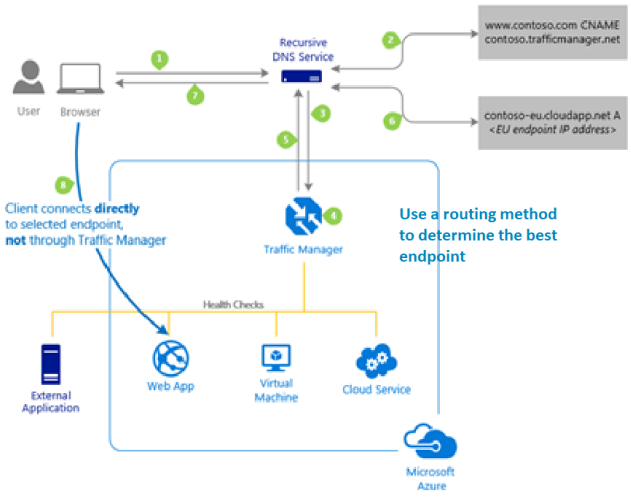
Traffic Manager uses the **Domain Name System (DNS)** to direct client requests to the most appropriate endpoint based on a **traffic-routing method** and the health of the endpoints.

There are four traffic routing methods available in Traffic Manager:

1. **Priority:** Select 'Priority' when you want to use a primary service endpoint for all traffic, and provide backups in case the primary or the backup endpoints are unavailable.

2. **Weighted:** Select 'Weighted' when you want to distribute traffic across a set of endpoints, either evenly or according to weights, which you define. The weight is an integer from 1 to 1000. The higher weignt, the higher the priority.
3. **Performance:** Select 'Performance' when you have endpoints in different geographic locations and you want end users to use the **"closest**" endpoint in terms of the **lowest network latency**.
   The closest endpoint is not necessarily measured by geographic distance. Instead Traffic Manager determines closeness by **measuring network latency**. Traffic Manager maintains an Internet Latency Table to track the round-trip time between IP address ranges and each Azure datacenter.
   With this method Traffic Manager looks up the source IP address of the incoming DNS request in the Internet Latency Table. Traffic Manager chooses an available endpoint in the Azure datacenter that has the lowest latency for that IP address range, then returns that endpoint in the DNS response
4. **Geographic:** Select 'Geographic' so that users are directed to specific endpoints (Azure, External or Nested) based on which geographic location their **DNS query originates** from. This empowers Traffic Manager customers to enable scenarios where knowing a user's geographic region and routing them based on that is important. Examples include complying with data sovereignty mandates, localization of content & user experience and measuring traffic from different regions.

Following steps are used to resolve the DNS name and establish a connection:



1. **User traffic to company domain name**: The client requests information using the company domain name. The goal is to resolve a DNS name to an IP address. Company domains must be reserved through normal Internet domain name registrations that are maintained outside of Traffic Manager. In Figure 1, the example company domain is *www.contoso.com*.
2. **Company domain name to Traffic Manager domain name**: The DNS resource record for the company domain points to a Traffic Manager domain name maintained in Azure Traffic Manager. This is achieved by using a CNAME resource record that maps the company domain name to the Traffic Manager domain name. In the example, the Traffic Manager domain name is ***contoso.trafficmanager.net***.

3. **Traffic Manager Domain name and profile**: The Traffic Manager domain name is part of the Traffic Manager profile. The user's DNS server sends a new DNS query for the Traffic Manager domain name (in our example***, contoso.trafficmanager.net***), which is received by the Traffic Manager DNS name servers.
4. **Traffic Manager Profile rules processed**: Traffic Manager uses the specified traffic routing method and monitoring status to determine which Azure or other endpoint should service the request.
5. **Endpoint domain name sent to user**: Traffic Manager returns a CNAME record that maps the Traffic Manager domain name to the domain name of the endpoint. The user's DNS server resolves the endpoint domain name to its IP address and sends it to the user.
6. **User calls the endpoint**: The user calls the returned endpoint directly using its IP address.

Benefits of Traffic Manager

1. Improve availability of critical applications.
2. Improve responsiveness for high performance applications.
3. Upgrade and perform service maintenance without downtime.
4. Combine on-premises and Cloud-based applications.
5. Distribute traffic for large, complex deployments.

To implement Traffic Manager

1. Deploy the Web Apps in different Geographical locations
2. Browse → Traffic Manager profiles → Add
3. Set Name=Demo, Routing Method = Weighted → Create
4. Go to Traffic Manger → Settings → End Points → Add
5. Type = Azure EndPoint, Name=WebApp1EP, Target Resource Type = App Service, Choose an App Service, Weight = 1 → OK
6. Repeat step 5 for every Web App deployment.

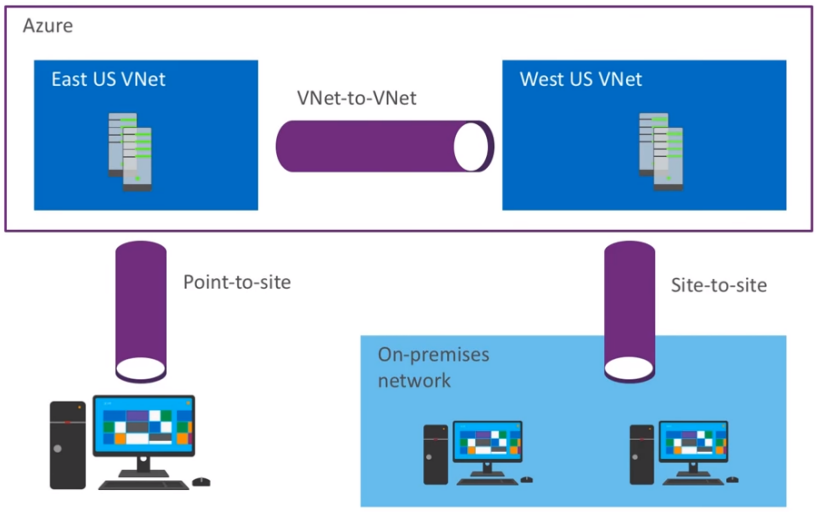
Comparison

Service	Azure Load Balancer	Application Gateway	Traffic Manager
Technology	Transport level (Layer 4)	Application level (Layer 7)	DNS level
Application protocols supported	Any	HTTP and HTTPS	Any (An HTTP endpoint is required for endpoint monitoring)
Endpoints	Azure VMs and Cloud Services role instances	Any Azure Internal IP address or public internet IP address	Azure VMs, Cloud Services, Azure Web Apps, and external endpoints

Vnet support	Can be used for both Internet facing and internal (Vnet) applications	Can be used for both Internet facing and internal (Vnet) applications	Only supports Internet-facing applications
Endpoint Monitoring	Supported via probes	Supported via probes	Supported via HTTP/HTTPS

Create connectivity between virtual networks

There are multiple ways to connect VNets. The sections below describe different ways to connect virtual networks.



Cloud-Only Virtual Networks

You can choose not to make any kind of virtual private network (VPN) connection to a VNet. Instead, when you create a VM or cloud service, you can specify endpoints that external clients can connect to. An endpoint is a VIP and a port number. Therefore an endpoint can be used only for a specific protocol, such as connecting a Remote Desktop Protocol (RDP) client or browsing a website. These VNets are known as cloud-only virtual networks. A dynamic routing gateway is not required in the VNet. Endpoints are published to the Internet, so they can be used by anyone with an Internet connection, including your on-premises computers.

Point-to-Site VPNs

A simple way to connect a VPN to an Azure VNet is to use a Point-to-Site VPN. In these VPNs, you configure the connection on individual on-premises computers. No extra hardware is required but you must complete the configuration procedure on every computer that you want to connect to the VNet. Point-to-site VPNs can be used by the client computer to connect to a VNet from any location with an Internet connection. Once the VPN is connected, the client computer can access all VMs and cloud services in the VNet as if they were running on the local network.

Site-to-Site VPNs

To connect **all the computers** in a physical site to an Azure VNet, you can create a Site-to-Site VPN. In this configuration, you do not need to configure individual computers to connect to the VNet, **instead you configure a VPN device**, which acts as a gateway to the VNet.

When you use the Site-to-Site IPsec steps, you create and configure the local network gateways manually. The local network gateway for each VNet treats the other VNet as a local site. This lets you specify additional address space for the local network gateway in order to route traffic. If the address space for a VNet changes, you need to update the corresponding local network gateway to reflect that. It does not automatically update.

VNet-to-VNet

Connecting a virtual network to another virtual network (VNet-to-VNet) is similar to connecting a virtual network to an on-premises site location. Both connectivity types use a VPN gateway to provide a secure tunnel using IPsec/IKE. The VNets you connect can be in **different subscriptions** and different regions.

The difference between the S2S AND V2V connection types is the way the local network gateway is configured. When you create a VNet-to-VNet connection, you do not see the local network gateway address space. It is automatically created and populated. If you update the address space for one VNet, the other VNet automatically knows to route to the updated address space. Creating a VNet-to-VNet connection is typically faster and easier than creating a Site-to-Site connection between VNets.

You can combine VNet to VNet communication with multi-site configurations. This lets you establish network topologies that combine cross-premises connectivity with inter-virtual network connectivity.

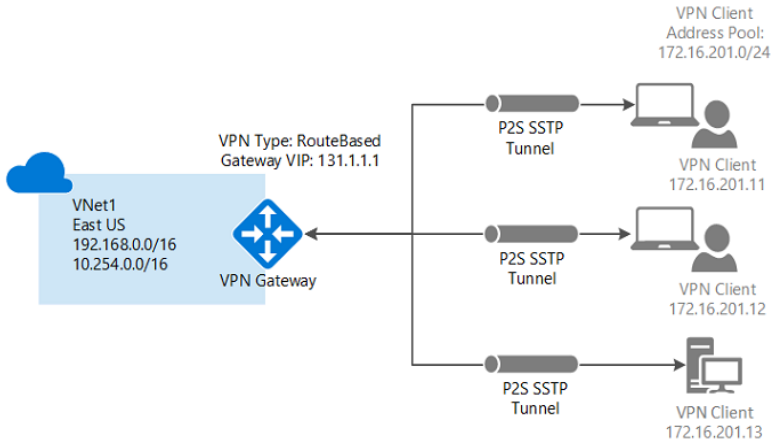
VNet peering

You may want to consider connecting your VNets using VNet Peering. VNet peering does not use a VPN gateway and has different constraints. Additionally, VNet peering pricing is calculated differently than VNet-to-VNet VPN Gateway pricing

ExpressRoute

ExpressRoute is a service that enables Azure customers to create a dedicated connection to Azure, which does not connect through the public Internet. This contrasts with VPNs, which use encryption to tunnel securely through the public Internet. Because ExpressRoute connections are dedicated, they can offer faster speeds, higher security, lower latencies, and higher reliability than VPNs.

Create a Point-to-Site VPN



Ensure that a Gateway Subnet is already created in a VNet

1. Select VNet → Subnets → + Gateway subnet → OK

Generate Certificates – Self signed root certificate for P2S connection

2. Open the Powershell command window and execute the following (Do not close the window)

```
$cert = New-SelfSignedCertificate -Type Custom -KeySpec Signature `
-Subject "CN=P2SRootCert" -KeyExportPolicy Exportable `
-HashAlgorithm sha256 -KeyLength 2048 `
-CertStoreLocation "Cert:\CurrentUser\My" -KeyUsageProperty Sign -KeyUsage CertSign
```

3. **To obtain the public key (.cer file)**
    1. Search → **Manage User Certificates** → Personal → Certificates

       or
    2. **MMC → File → Add/Remove Snap-In → Certificates → Add → OK**

       open **certmgr.msc**., typically in **'Certificates - Current User\Personal\Certificates'**.
    3. Locate the self-signed root certificate **(P2SRootCert)** → Right Click → **All Tasks**, and then click **Export**. This opens the **Certificate Export Wizard**.
    4. In the Wizard, click **Next**. Select **No, do not export the private key**, and then click **Next**.
    5. On the **Export File Format** page, select **Base-64 encoded X.509 (.CER).**, and then click **Next**.
    6. **File to Export**, **Browse** = d:\P2SRootCert.cer → **Next**.
    7. Click **Finish** to export the certificate. You will see **The export was successful**. Click **OK** to close the wizard.

A client certificate that is present on the device is used to authenticate the connecting user. Client certificates are generated from a trusted root certificate and then installed on each client computer. You can use a root certificate that was generated using an Enterprise solution, or you can generate a self-signed certificate.

The validation of the client certificate is performed by the VPN gateway and happens during establishment of the P2S VPN connection. The root certificate is required for the validation and must be uploaded to Azure

4. **Generate a client certificate: Execute the following command in the same PowerShell window opened earlier.**

```
New-SelfSignedCertificate -Type Custom -KeySpec Signature `
-Subject "CN=P2SChildCert" -KeyExportPolicy Exportable `
-HashAlgorithm sha256 -KeyLength 2048 `
-CertStoreLocation "Cert:\CurrentUser\My" `
-Signer $cert -TextExtension @("2.5.29.37={text}1.3.6.1.5.5.7.3.2")
```

More about Certficates:

https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-certificates-point-to-site

Create a Virtual Network Gateway

5. All Services → Virtual Network Gateway → +Add
    a) Name=TestVNetGateway, . . . ,
    b) Gateway type = VPN
    c) VPN type = Route-based
    d) SKU = Basic (table below)
    e) Choose a virtual network,
    f) Create a New IP
    g) Create.

Note: Provisioning a virtual network gateway may take up to 45 minutes.

About VPN Types:

1. **RouteBased**: RouteBased VPNs use "routes" in the IP forwarding or routing table to direct packets into their corresponding tunnel interfaces. The tunnel interfaces then encrypt or decrypt the packets in and out of the tunnels.
2. **PolicyBased**: Policy-based VPNs encrypt and direct packets through IPsec tunnels based on the IPsec policies configured with the combinations of address prefixes between your on-premises network and the Azure VNet.
    1. PolicyBased VPNs can **only** be used on the Basic gateway SKU.
    2. You can have only **1 tunnel** when using a PolicyBased VPN.
    3. You can only use PolicyBased VPNs for **S2S connections**.

Which Gateway SKUs Support P2S VPN?

SKU	P2S Connections	S2S/VNet-to-VNet Tunnels	Aggregate Throughput Benchmark

Basic	128	Max. 30	100 Mbps
VpnGw1	128	Max. 30	650 Mbps
VpnGw2	128	Max. 30	1Gbps
VpnGw3	128	Max. 10	1.25 Gbps

Upload the root certificate .cer file

3. Open the Root certificate (not child) with a text editor, such as Notepad. Copy the content between -----BEGIN CERTIFICATE------ and -----END CERTIFICATE------
4. Select VNetGateway created earlier → Point-to-site configuration,
    1. Address Pool=172.16.201.0/24 (is the pool of IP addresses from which clients that connect will receive an IP address.)
    2. Root Certificates: Name=RootCert1, Public Certficate Data <Value copied in prev step>

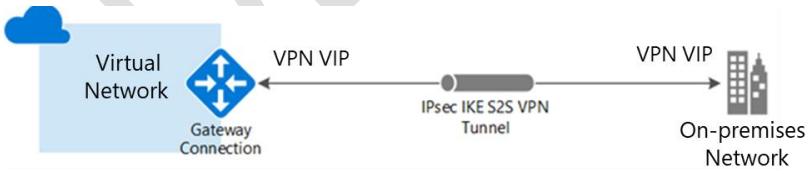
Note: You can add up to **20 trusted** root certificates.

Download and Install VPN client

5. Point-to-site configuration → Download VPN client
6. Select X64 → Download
7. Execute the downloaded EXE file
8. Client Computer → Network Settings → VPN
9. Click on TestVNet and connect to VNet.
10. To verify that your VPN connection is active, open an elevated command prompt, and run *ipconfig/all*.

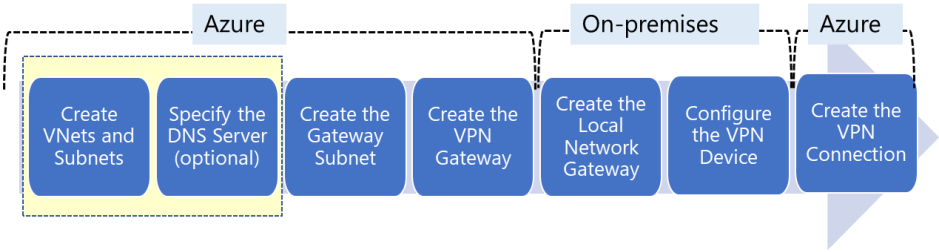
Site-to-Site VPN

A Site-to-Site (S2S) connection is a connection over IPsec/IKE (IKEv1 or IKEv2)VPN tunnel. S2S connections can be used for cross-premises and hybrid configurations. This type of connection requires a VPN device located on-premises that has a public IP address assigned to it.

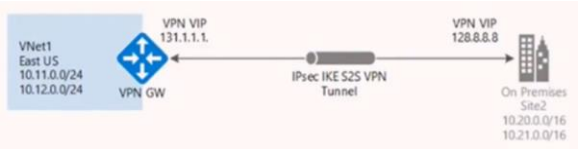


Implementing Site-to-Site VPN

To configure Site-to-Site you must configure both sides of the infrastructure. For example, Azure and on-premises. There are a lot of steps, but we will go through each one. As we do, try to keep the architecture diagrams in mind.



✓ Take time to carefully plan your network configuration. If a duplicate IP address range exists on both sides of the VPN connection, traffic will not route the way you may expect it to.

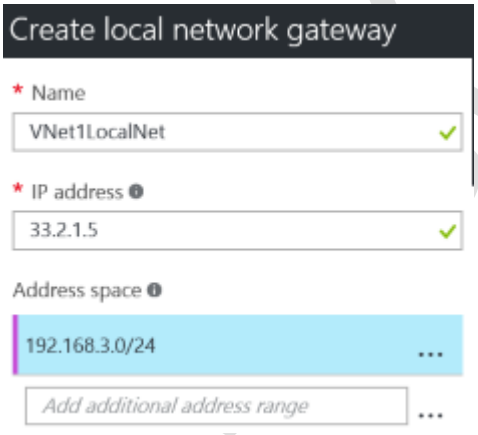


Step 1: Create a VNet, Gateway Subnet, VPN Gateway as described in example of **Point to Site.**

Step 2: Setup Local Network Gateway:

The local network gateway typically refers to the on-premises location.

1. You give the site a **name** by which Azure can refer to it,
2. Specify the **IP address (128.8.8.8)** of the on-premises VPN device for the connection.
3. Specify **Address Space**. One or more IP address ranges (in CIDR notation) that define your local network's address space. For example: 10.20.0.0/16 and 10.21.0.0/16 as in diagram above.



Step 3: Configure the VPN Device

Microsoft has validated a list of standard VPN devices that should work well with the VPN gateway. This list was created in partnership with device manufacturers like Cisco, Juniper, Ubiquiti, and Barracuda Networks. If you don't see your device listed in the validated VPN devices table (reference link), your device may still work with a Site-to-Site connection. Contact your device manufacturer...

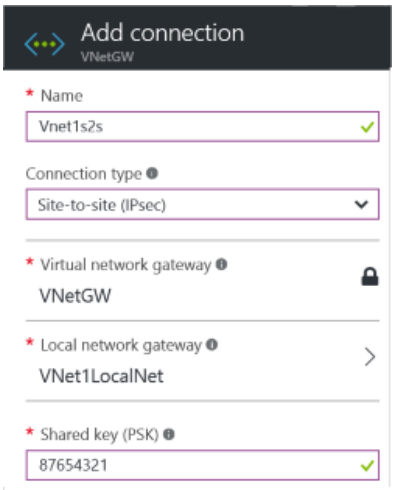
To configure your VPN device, you need the following:

- **A shared key**. This is the same shared key that you will specify when creating the VPN connection (next step).
- The **public IP address** of your VPN gateway.

✓ Depending on the VPN device that you have, you may be able to downloada VPN device configuration script.

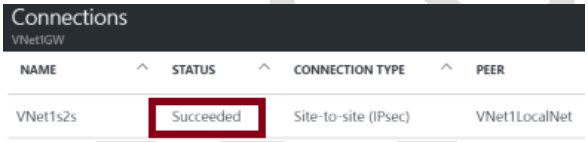
Step 4: Configure the VPN Connection

In this step you will configure the connection between the Azure VPN gatewayand the local network gateway.



For **Shared Key**, the value here must match the value that you are using for your local VPN device. If your VPN device on your local network doesn't provide a shared key, you can make one up and input it here and on your local device. The important thing is that the shared keys match.

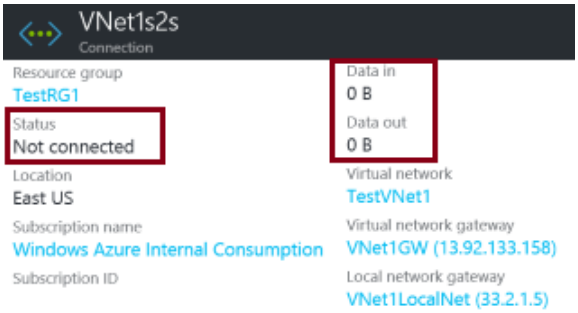
When the connection is complete, you'll see it appear in the Connections bladefor your Gateway.



Step 4: Verify the VPN Connection

After you have configured all the Site-to-Site components it is time to verifythat everything is working. You can verify the connections either in the portal,or by using PowerShell.

Portal: When you view your connection in the Azure portal the Status should beSucceeded or Connected. Also, you should have data flowing in the Data in andData out information.



PowerShell: To verify your connection with PowerShell

```
Get-AzureRmVirtualNetworkGatewayConnection -NameMyGWConnection -ResourceGroupName MyRG
```

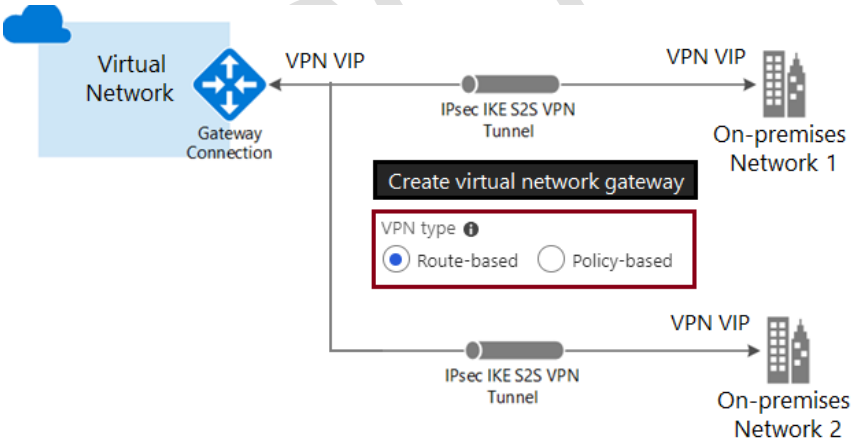
After the cmdlet has finished, view the values. The connection status shouldshow 'Connected' and you can see ingress and egress bytes.

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

YouTube Video: https://youtu.be/5VTdah3VwYU

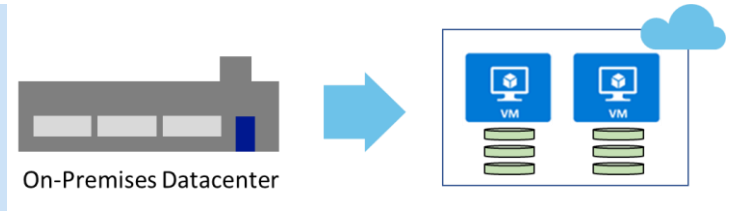
Multiple Sites

A Multi-site connection is a variation of the Site-to-Site connection. You createmore than one VPN connection from your virtual network gateway, typicallyconnecting to multiple on-premises sites. When working with multipleconnections, you must use a Route-based VPN. Because each virtual networkcan only have one VPN gateway, all connections through the gateway share theavailable bandwidth.



Site-to-Site Scenarios

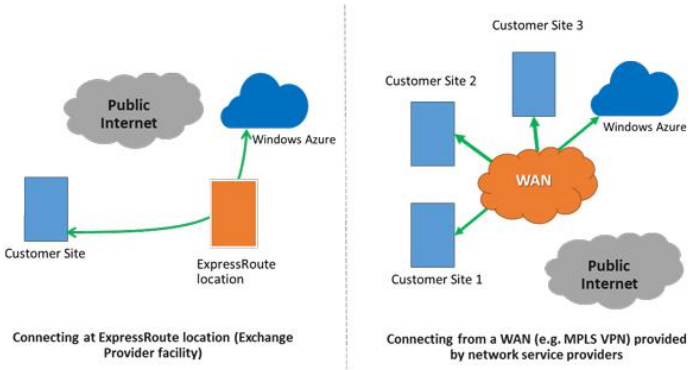
There are many scenarios where Site-to-Site connections can be useful. Here are a few.



- **Capacity On-Demand:** Azure provides capacity on demand. By creating a connection to Azure, more storage or compute resources can easily be brought online. You can extend your on-premises datacenter without purchasing and installing equipment in the datacenter. This scenario includes spawning remote offices.
- **Strategic Migration:** There are many strategic reasons for moving to Azure. Organizations whose core purpose is not related to managing complex datacenter deployments, may want to shed competing interests and focus on improving their core business. They may also want to reduce costs by moving to a pay as you go model. Migrating services is usually faster than responding in-house, especially when you trying to project a global presence.
- **Disaster Recovery:** The cloud offers an efficient, cost effective choice for data backup and recovery. Most cloud platforms let you run third-party software for backup and disaster recovery, but with Microsoft these services are fully integrated and easy to turn on, which means you don't have to install and manage a separate product in the cloud.

Azure Express Route

- Microsoft Azure ExpressRoute lets you extend your on-premises networks into the Microsoft cloud over a dedicated private connection facilitated by a connectivity provider.
- ExpressRoute offers **private, reliable and low-latency** connections between customer's datacenters and Azure.
- Microsoft is positioning the technology as a way for customers to extend their network in Windows Azure as part of a **hybrid IT approach**.



Express Route Partners to deliver Express Route connectivity

https://youtu.be/HZH6F_gLCQQ

https://youtu.be/wystDyqyRXc

Creating Express Circuits: https://youtu.be/_8S3tOwWgc8