Implement Advanced Virtual Networking

- Understanding and Creating Availability Set
- Understanding Availability Zones
- Load Balancing
    - Configure external and internal Azure Load balancer
    - Load Balancing Rules
    - Implement front end IP configuration
- Azure Application Gateway
- Azure Traffic Manager
- Azure Front Door
- Integrate on-premises network with Azure virtual network
    - 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

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.

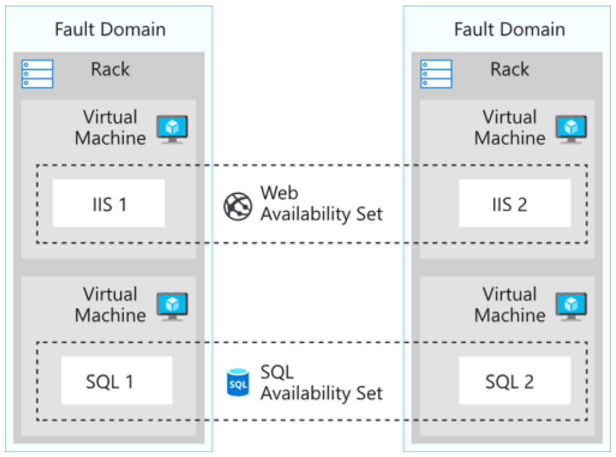
Fault Domains:

- 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.



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 200 virtual machines in each Azure availability set.**
- 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

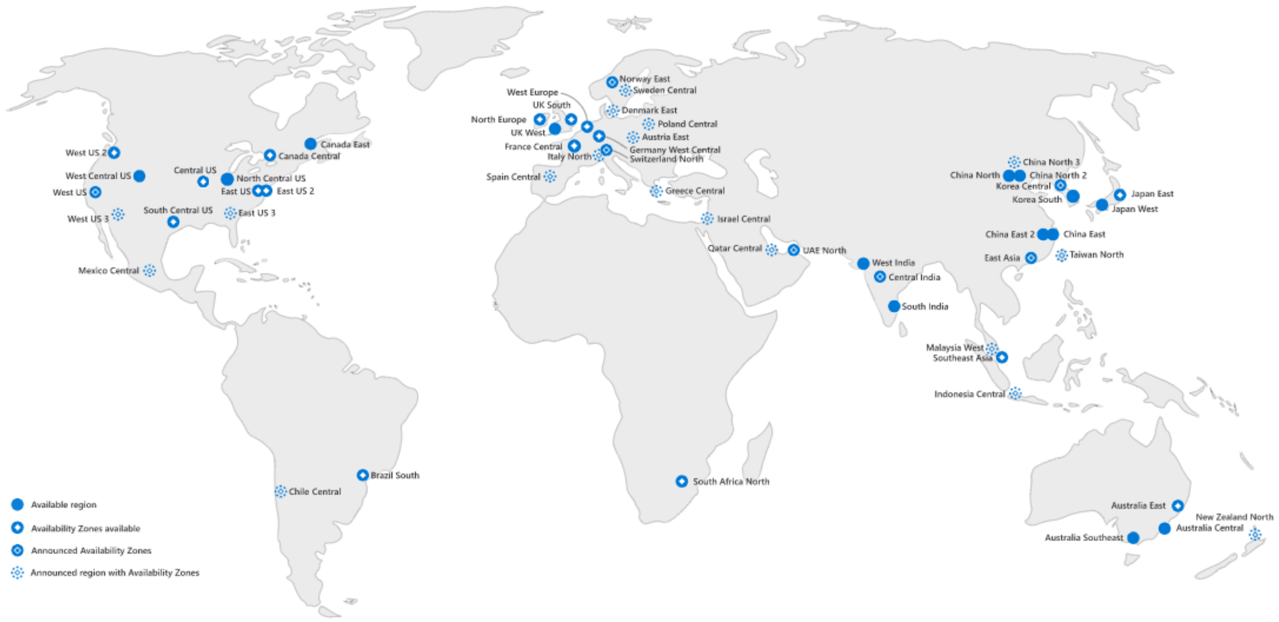
```
New-AzureRmAvailabilitySet –ResourceGroupName –DemoAvailSet –Location "East US" –
PlatformUpdateDomainCount 5 –PlatformFaultDomainCount 3
```

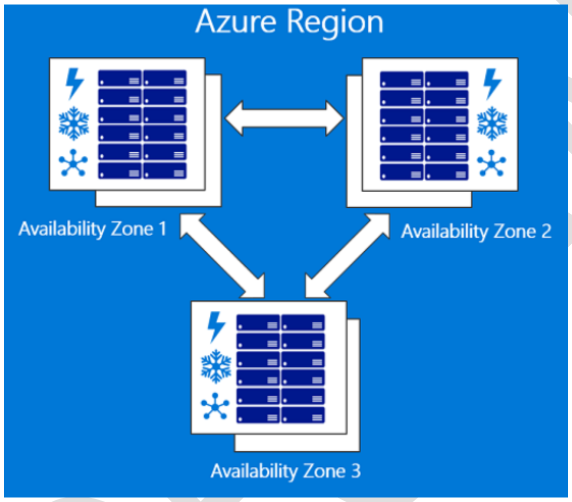
Follow best practices when you design your application for high availability.

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.

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

Understanding 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 domain**. 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:

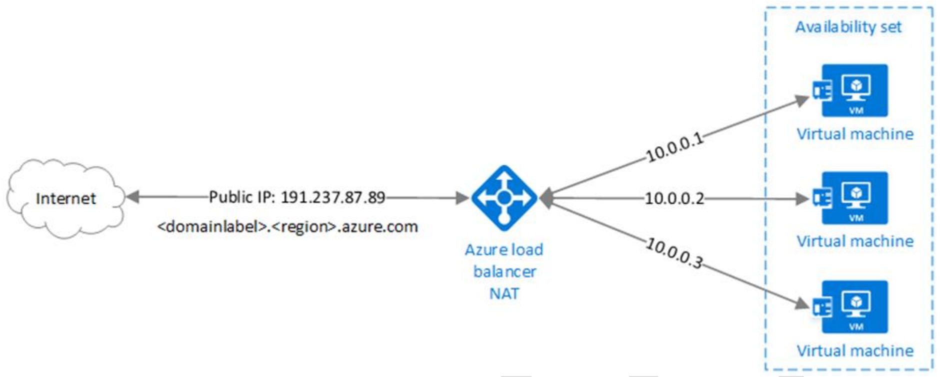
- **Zonal services** – you pin the resource to a specific zone (for example, virtual machines, managed disks, Standard SKU IP addresses), or
- **Zone-redundant services (ZRS)** – platform replicates automatically across zones (for example, zone-redundant storage, SQL Database).

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

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.

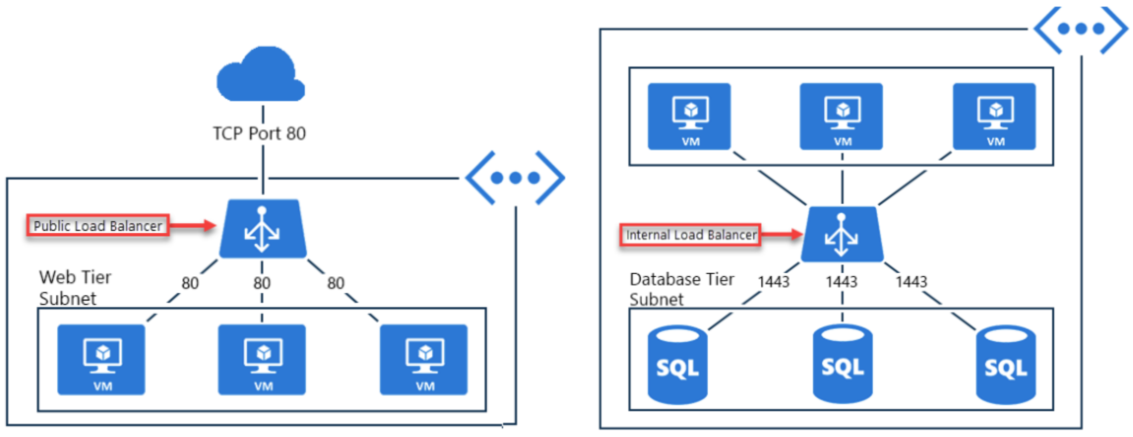


It can be configured to:

- 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

Basic SKU LB

- VMs in a **single availability set** or **VM scale set** can be added to Backend Pool.

- Requires IP Address also should be **Basic** SKU
- Only one VM not in availability set can be added to Backend Pool.
- Upto 100 Instances are allowed.
- Basic Load Balancer is free of charge.
- NSG Rules at Nic or Subnet of the VM is honored but **not** mandatory.
- Diagnostics through Azure Log Analytics for public-facing load balancers.
- Only HTTP and TCP Health Probes are supported.

Standard SKU LB

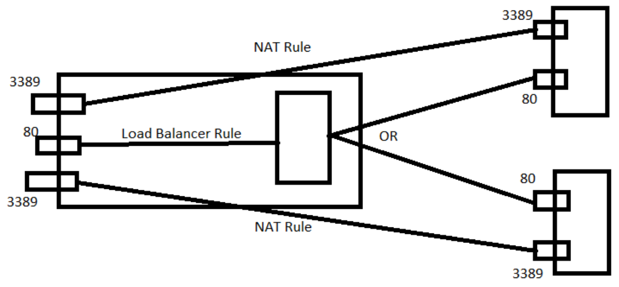
- Any VM in a **single** virtual network, including a blend of VMs, availability sets, and VM scale sets.
- Any VM etiher having **Standard** SKU Public IP address or **not** at all having Public IP address. VM may or may not be in Availability Set.
- VM can be in different VNet (but same region). Though one backend pool can contain only VM from one Virtual Network. This can help in reusability of LB.
- Upto 1000 VM instances are allowed.
- Its secured by default. This means Network Security Groups (NSGs) are used to explicitly permit and whitelist allowed traffic. **If you do not have an NSG on a subnet or NIC of your virtual machine resource, traffic is not allowed to reach this resource**.
- Charged on the basis of Number of Rules and volume of Data processed.
- Supports Availability Zones. A guaranteed SLA (99.99% for two or more virtual machines in different zones)
- Diagnostics through Azure Log Analytics for public-facing load balancers. Diagnostics through Azure Monitor, for multidimensional metrics.
- HTTP, HTTPS and TCP health probes are supported.

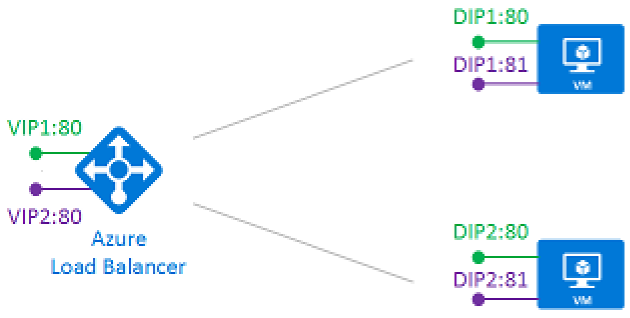
6. Choose the Virtual Machines → Select WebVM1 and WebVM2
7. Create a Probe, Name=**HealthProbe**, Protocol=HTTP, Port=80, Path=/ . . . → OK
8. Create Load Balancer **Rule**, Name=HTTP, Protocol=TCP, Port=80, Backend port=80, Backend pool=LB-backend, Probe=HealthProbe, Session persistence=None, . . .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

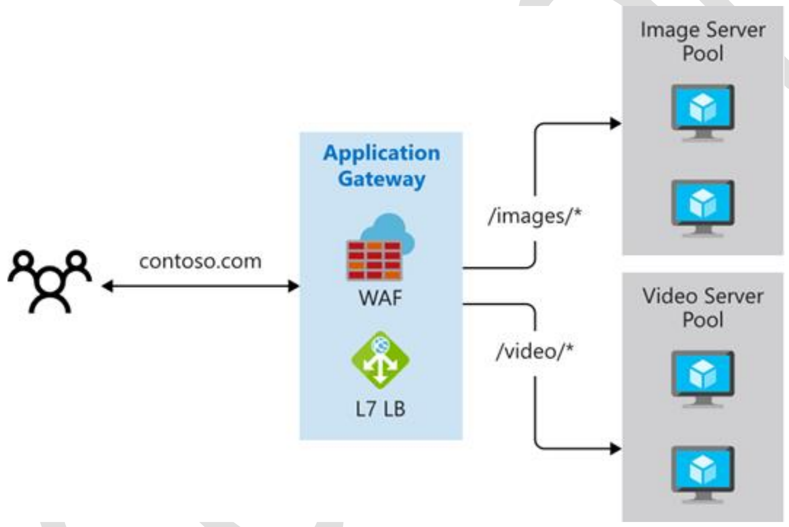
Cross Region Load Balancer:

https://docs.microsoft.com/en-us/azure/load-balancer/cross-region-overview

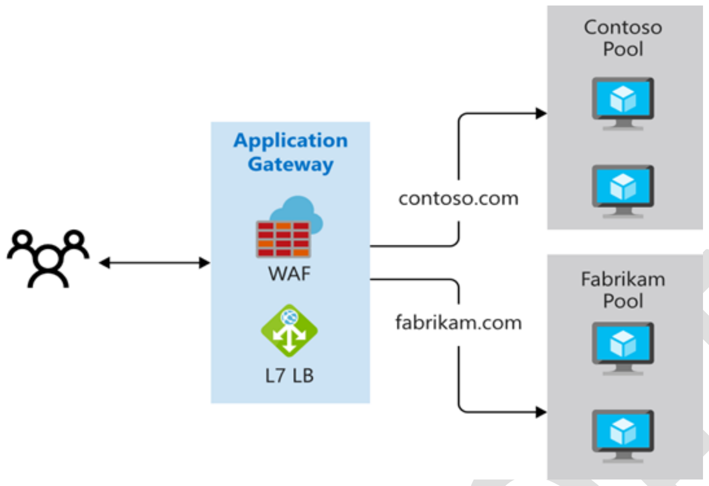
Azure Application Gateway

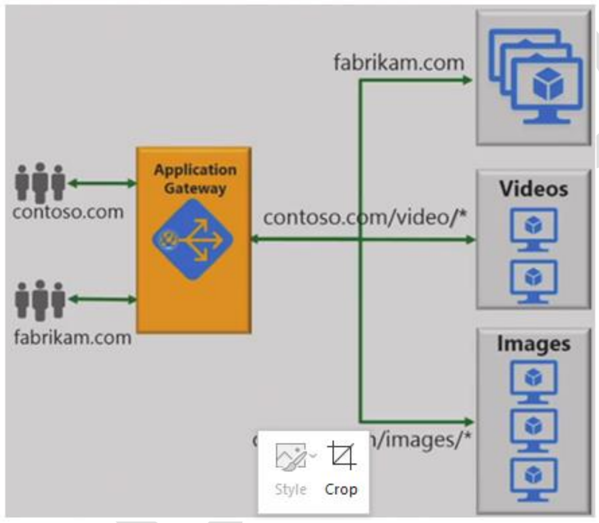
- Azure Application Gateway is a **web traffic load balancer** that enables you to manage traffic to your **web appli cations**.
- Azure Application Gateway is a **OSI layer-7 load balancer**. It provides failover, performance-routing HTTP requests between different servers, whether they are on the cloud 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 and HTTPS 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.

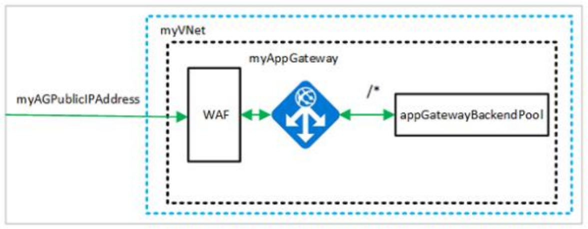


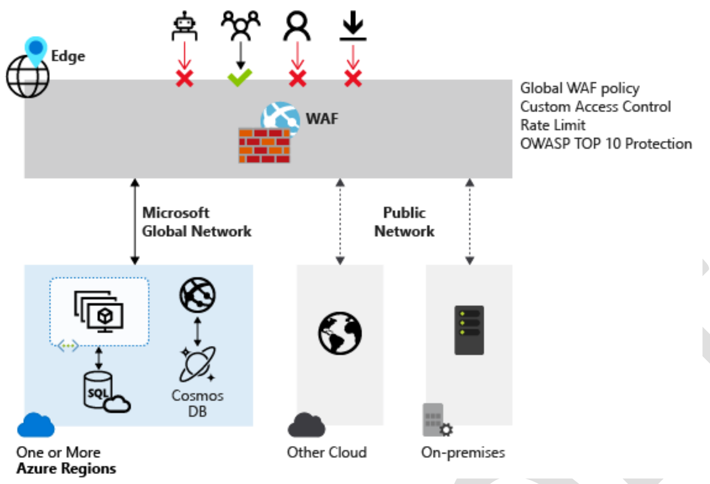
6. **Support for Multiple Site Hosting**: Application gateway allows for you to consolidate up to **20 websites** on a single application gateway.





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 ip configuration. This public IP address serves as a load-balanced IP. Currently, you can only assign a ***static* public IP** address to an application gateway frontend configuration.

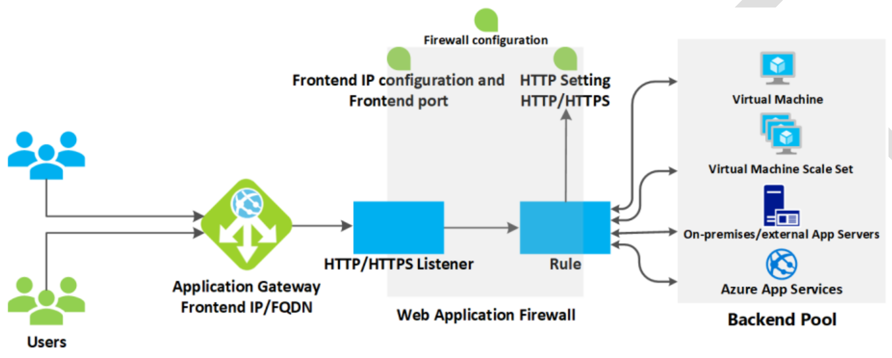
Gateway Sizes and Instances

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 125 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 free 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 (Application Gateway dedicated) without any resources)

   Note: Creating an Application Gateway would take around 10 mins.
3. Add WebVM1 / Add WebVM2 to BackendPool of the Gateway
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

Frontend IP cofiguration

DemoFrontendIP = 104.211.219.165 (Public)

HTTP Settings

DemoHttp80Settings = HTTP / 80 – Cookie Affinity = false, ….

DemoHttp8080Settings = HTTP / 8080 – Cookie Affinity = false

DemoHttpsSettings = HTTPS / 443 - Cookie Affinity = true

BackendPool (Selecting one of the VM in the Pool to serve the request)

BackendPoolA

Web1-vm & Web2-vm

BackendPoolB

Web3-VM & Web4-VM

Listeners (Its selected based on Protocol/Port/IPAddress in the Request URL)

BasicHttp80Listener (Basic Listener)

Frontend IP configuration = DemoFrontendIP (104.211.219.165)

FrontendPort = 80

BasicHttp8080Listener (Basic Listener)

Frontend IP configuration = DemoFrontendIP (104.211.219.165)

FrontendPort = 8080

BasicHttps443Listener (Basic Listener)

Frontend IP configuration = DemoFrontendIP (104.211.219.165)

FrontendPort = 443

DomainBasedSiteAListener (Multisite Listener)

Frontend IP configuration = DemoFrontendIP

FrontendPort = 80,

Hostname: www.SiteA.com

Rules (Mapping a Listener to Backend Pool)

DemoRule1

HTTP settings = DemoHttp80Settings

Listener = BasicHttp80Listener

Backend Pool = BackendPoolA

DemoRule2

HTTP settings = DemoHttp80Settings

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

Backend Pool = BackendPoolB

Lifecycle of request in browser:

Case1:

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

1. Because of IP (DemoFrontendIP) and Port (HTTPFrontendPort80), Listener is selected (BasicHttp80Listener)

2. Based on Listener (BasicHttp80Listener) and Http Settings (HTTPFrontendPort80), the rule is selected (DemoRule1)

3. As per DemoRul1, Listerner (BasicHttp80Listener) => BackendPool (BackendPoolA) and HTTP Settings should be DemoHttp80Settings

4. Now the request will be forwarded to any one machine in the backendpool (Either Web1-vm or Web2-vm) and Affinity will be managed as in DemoHttp80Settings.

Case2:

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

1. Because of Host Name (DemoFrontendIP) and Port (HTTPFrontendPort80), Listener is selected (DomainBasedSiteAListener)

2. Based on Associated rule, the rule is selected (DemoRule2)

3. As per DemoRule2, Listerner (DomainBasedSiteAListener) => BackendPool (BackendPoolB) and HTTP Settings should be DemoHttp80Settings

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 DemoHttp80Settings

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	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

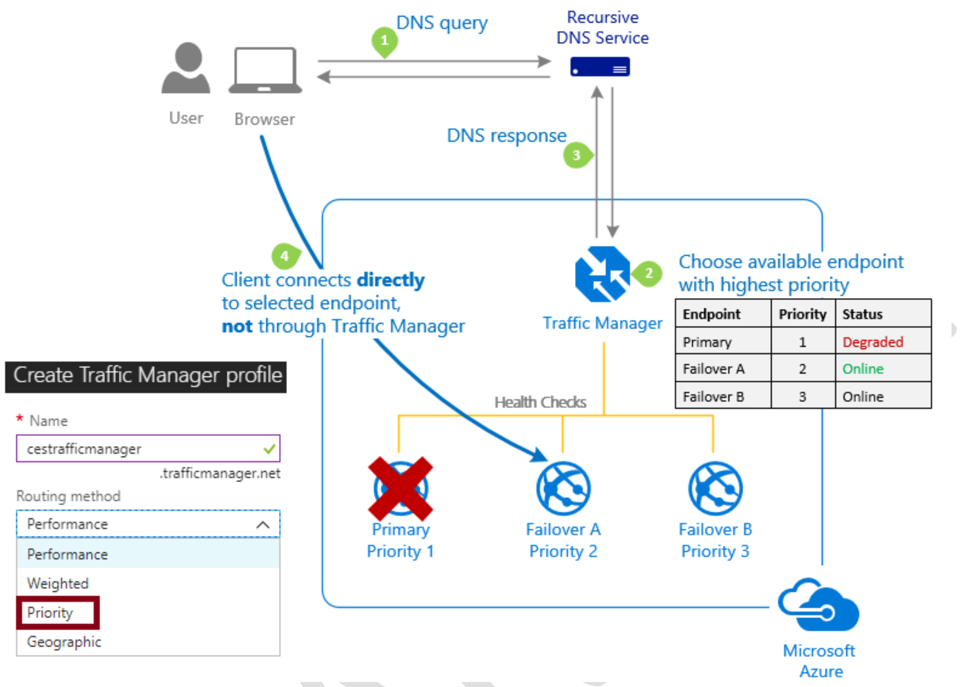
Microsoft Azure Traffic Manager allows you to control the distribution of user traffic for service endpoints in **different datacenters around the world.**

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.

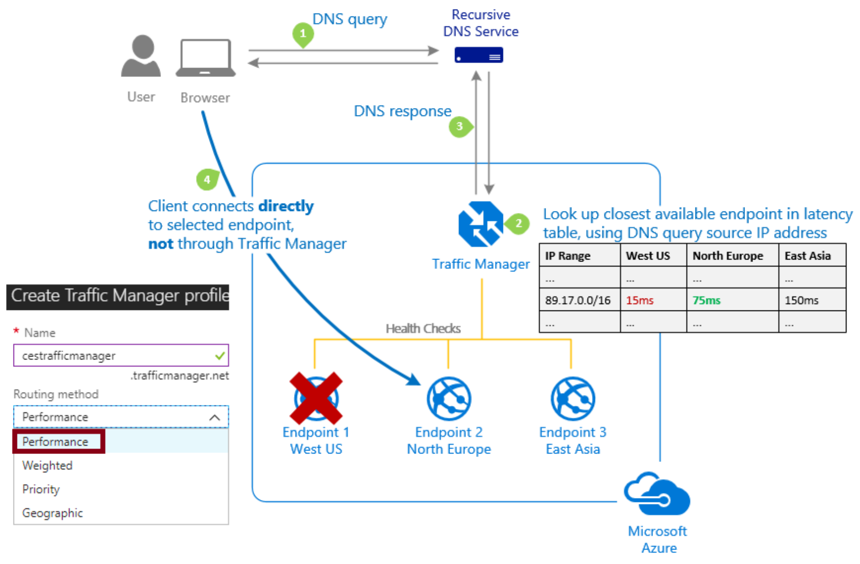


Endpoint	Priority	Status
Primary	1	Degraded
Failover A	2	Online
Failover B	3	Online

* Name: cestrafficmanager .trafficmanager.net

Routing method: Priority

2. **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.


DNS query
1
Recursive DNS Service
User
Browser
DNS response
3
4
Client connects **directly** to selected endpoint, **not** through Traffic Manager
2
Traffic Manager
Look up closest available endpoint in latency table, using DNS query source IP address
Health Checks
Endpoint 1 West US
Endpoint 2 North Europe
Endpoint 3 East Asia
Microsoft Azure
Create Traffic Manager profile
Performance
Weighted
Priority
Geographic

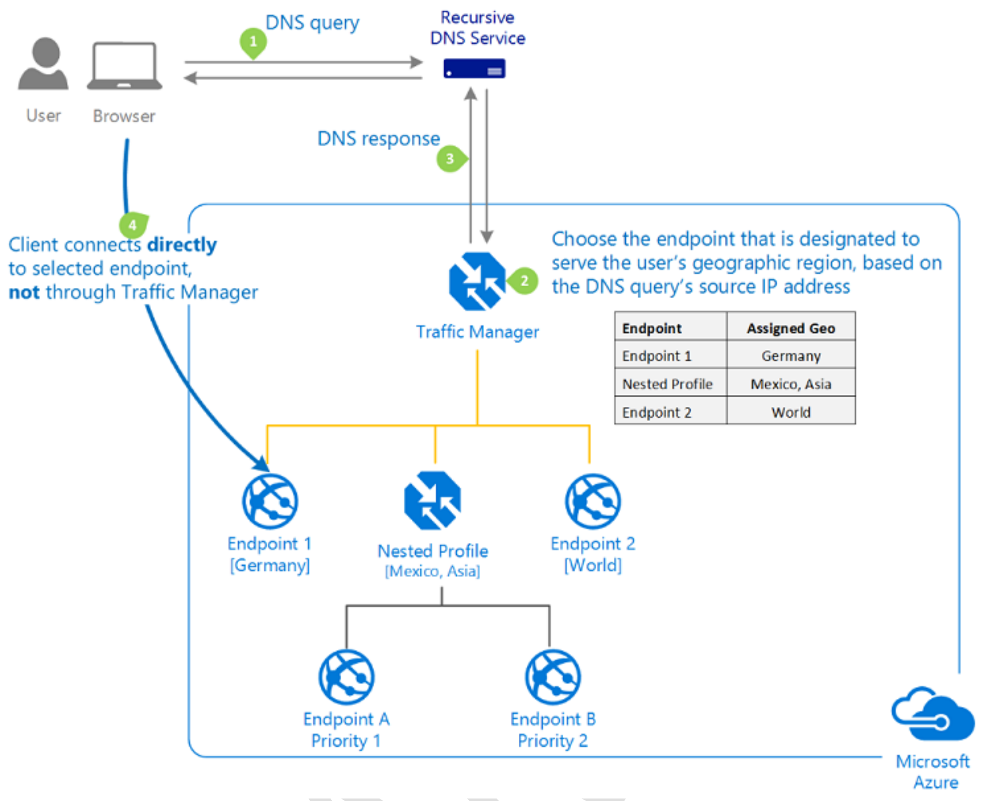
Create Traffic Manager profile

* Name: cestrafficmanager .trafficmanager.net

Routing method: Performance

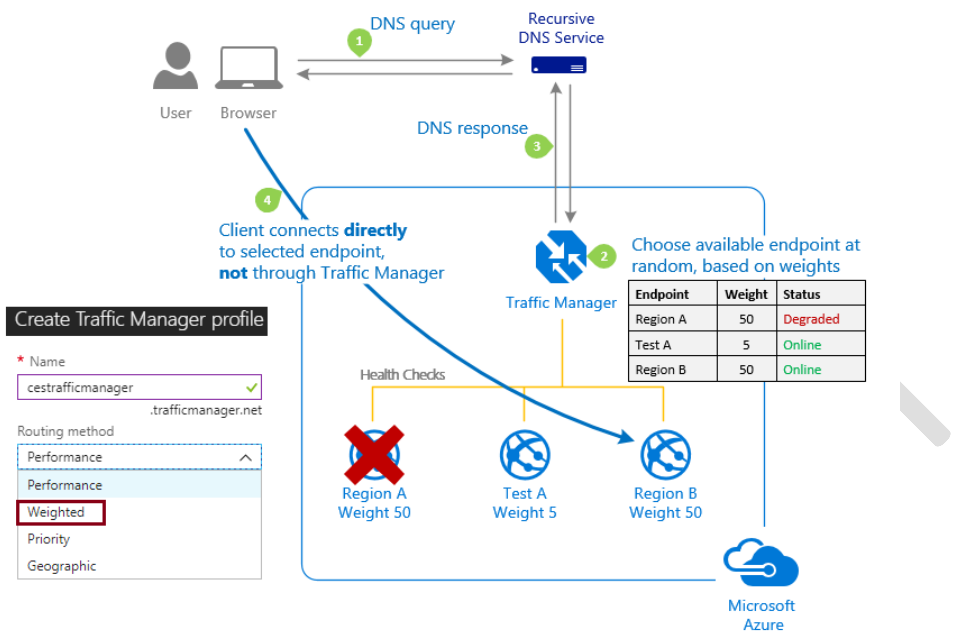
IP Range	West US	North Europe	East Asia
...	...	...	...
89.17.0.0/16	15ms	75ms	150ms
...	...	...	...

3. **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.


DNS query
1
Recursive DNS Service
User
Browser
DNS response
3
4
Client connects **directly** to selected endpoint, **not** through Traffic Manager
2
Choose the endpoint that is designated to serve the user's geographic region, based on the DNS query's source IP address
Traffic Manager
Endpoint 1 [Germany]
Nested Profile [Mexico, Asia]
Endpoint 2 [World]
Endpoint A Priority 1
Endpoint B Priority 2
Microsoft Azure

Endpoint	Assigned Geo
Endpoint 1	Germany
Nested Profile	Mexico, Asia
Endpoint 2	World

4. **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 weight, the higher the priority.



Endpoint	Weight	Status
Region A	50	Degraded
Test A	5	Online
Region B	50	Online

* Name: cestrafficmanager .trafficmanager.net

Routing method: Performance

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.

Azure Front Door

- Azure Front Door Service is Microsoft's highly available and scalable web application acceleration platform and **global HTTP(s) load balancer**.
- Front Door works at **Layer 7 or HTTP/HTTPS layer** and uses split TCP-based anycast protocol.
- Based on **routing method** you can ensure that Front Door will route your client requests to the fastest and most available application backend.
- We can use Front Door with Azure services including **Web/Mobile Apps, Cloud Services and Virtual Machines** – or combine it with **on-premises** services.

Key features included with Front Door:

- Accelerated application **performance** by using split TCP-based anycast protocol.
- Intelligent **health probe** monitoring for backend resources.
- **URL-path based** routing for requests.
- Enables hosting of **multiple websites** for efficient application infrastructure.
- Cookie-based **session affinity**.
- **SSL offloading** and certificate management.
- Define your own **custom domain**.
- Application security with integrated **Web Application Firewall** (WAF).
- Redirect HTTP traffic to HTTPS with **URL redirect**.
- Custom forwarding path with **URL rewrite**.
- Native support of end-to-end IPv6 connectivity and HTTP/2 protocol.

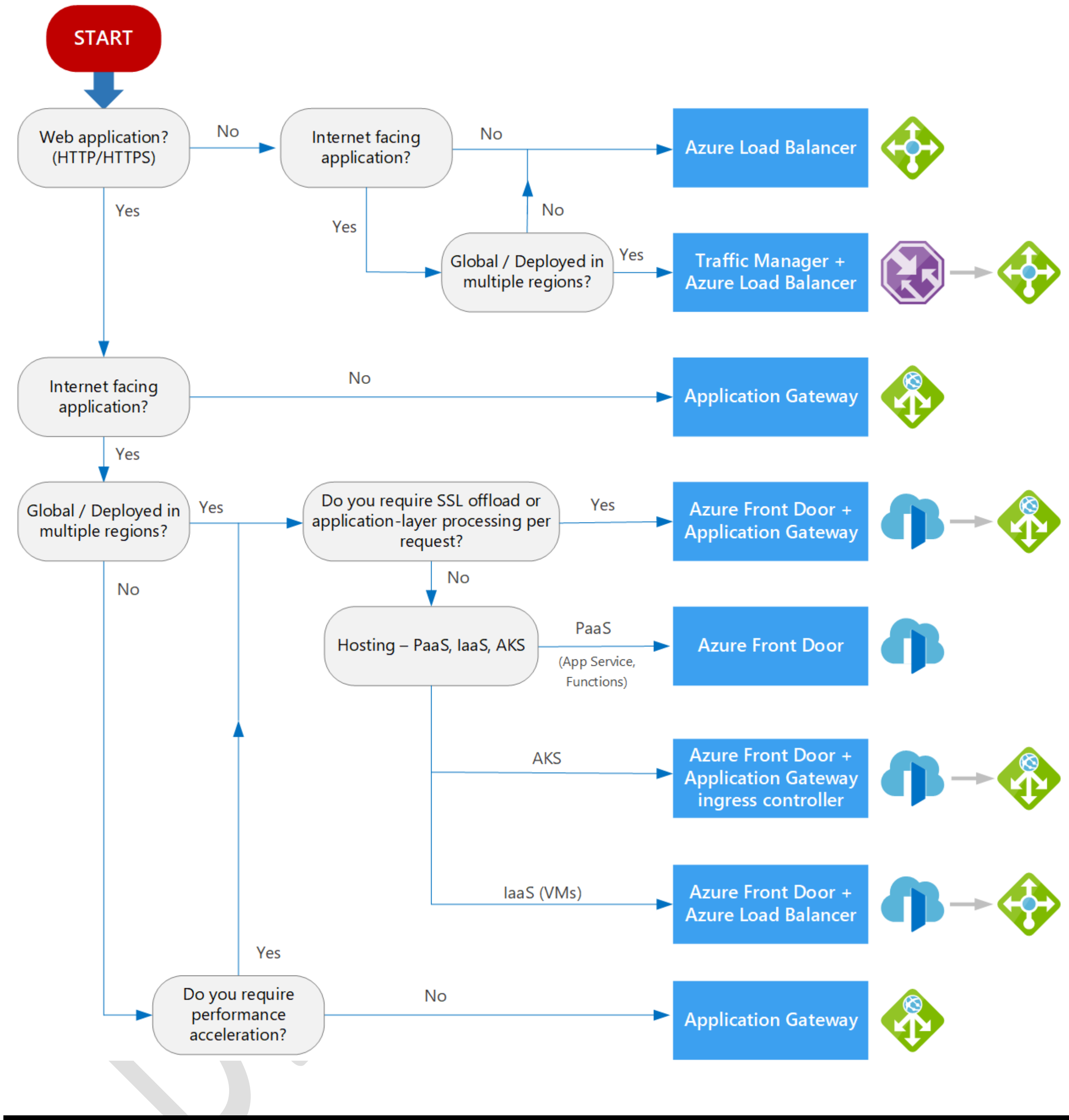
Azure Load Balancing Options Comparison

Service	Azure Load Balancer	Application Gateway	Traffic Manager	Azure Front Door
Technology	Transport level (Layer 4)	Application level (Layer 7)	DNS Resolver	HTTP / HTTPS (Layer 7)
Application protocols supported	Any (TCP or UDP)	HTTP and HTTPS HTTP/2, WebSockets	Any (An HTTP endpoint is required for endpoint monitoring)	Split TCP based anycast protocol

Backends and Endpoints	Azure VMs and Azure VM Scale Sets	Azure VMs, VM Scale Sets, Azure App Services, IP address and Hostnames	Azure Cloud Services, Azure App Services, Azure App Service Slots and Public IP Addresses	Internet facing services hosted inside or outside of Azure
Network connectivity	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	Can be used for both Internet facing and internal (Vnet) applications
Endpoint Monitoring	Supported via probes	Supported via probes	Supported via HTTP/HTTPS	Supported via HTTP/HTTPS

The following table summarizes the Azure load balancing services by these categories:

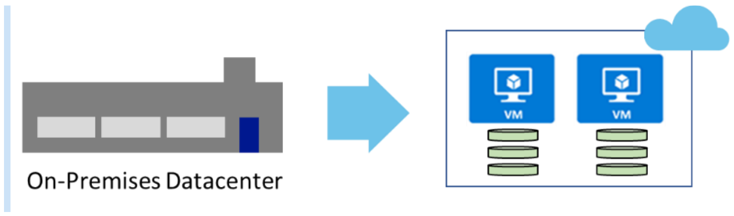
Service	**Global/regional**	**Recommended traffic**
Azure Front Door	Global	HTTP(S)
Traffic Manager	Global	non-HTTP(S)
Application Gateway	Regional	HTTP(S)
Azure Load Balancer	Global	non-HTTP(S)


START
Web application? (HTTP/HTTPS)
No
Internet facing application?
No
Azure Load Balancer
Yes
Yes
No
Global / Deployed in multiple regions?
Yes
Traffic Manager + Azure Load Balancer
Internet facing application?
No
Application Gateway
Yes
Global / Deployed in multiple regions?
Yes
Do you require SSL offload or application-layer processing per request?
Yes
Azure Front Door + Application Gateway
No
No
Hosting – PaaS, IaaS, AKS
PaaS
(App Service, Functions)
Azure Front Door
AKS
Azure Front Door + Application Gateway ingress controller
IaaS (VMs)
Azure Front Door + Azure Load Balancer
Yes
Do you require performance acceleration?
No
Application Gateway

Site-to-Site VPN

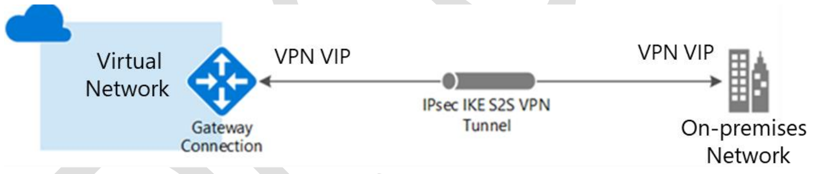
Site-to-Site Scenarios

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



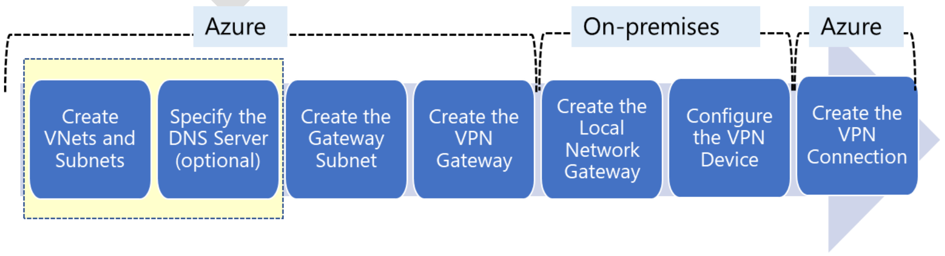
- **Capacity On-Demand:** Azure provides capacity on demand. By creating a connection to Azure, more storage or compute resources can easily be brought online. You can extend your on-premises datacenter without purchasing and installing equipment in the datacenter. This scenario includes spawning remote 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.

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.

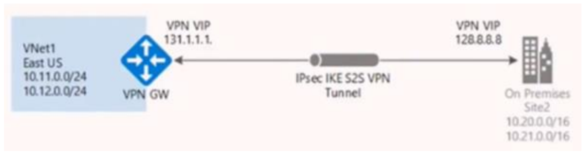


Implementing Site-to-Site VPN

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.


Azure
On-premises
Azure
Create VNets and Subnets
Specify the DNS Server (optional)
Create the Gateway Subnet
Create the VPN Gateway
Create the Local Network Gateway
Configure the VPN Device
Create the VPN Connection

✔ **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.**

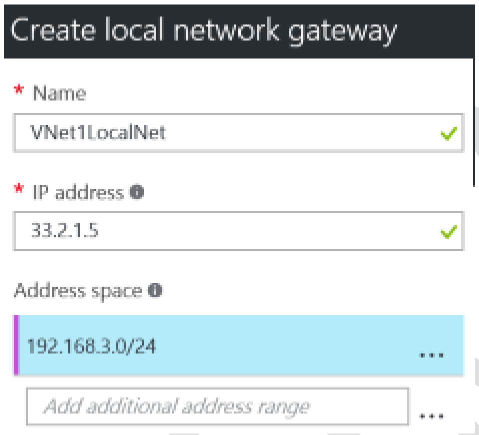


Step 1: Create a VNet, Gateway Subnet, VPN Gateway as described in example of **Point to Site.**

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 (eg: 33.2.1.5)** 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: 192.168.3.0/24 and 10.21.0.0/16.


Create local network gateway

* Name
VNet1LocalNet

* IP address
33.2.1.5

Address space
192.168.3.0/24

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...

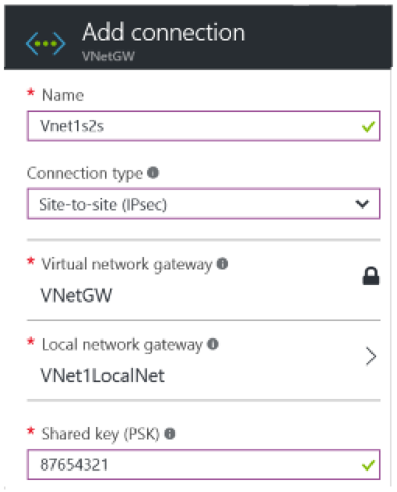
To configure your VPN device, you need the following:

- **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 download a 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.



Add connection
VNetGW

* Name: Vnet1s2s

Connection type: Site-to-site (IPsec)

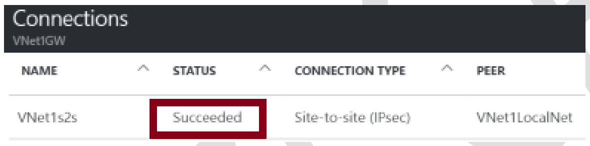
* Virtual network gateway: VNetGW

* Local network gateway: VNet1LocalNet

* Shared key (PSK): 87654321

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.



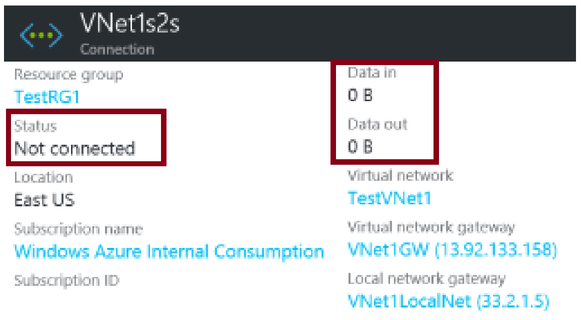
Connections
VNet1GW

NAME	STATUS	CONNECTION TYPE	PEER
VNet1s2s	Succeeded	Site-to-site (IPsec)	VNet1LocalNet

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.


VNet1s2s
Connection

Resource group: TestRG1	Data in: 0 B
Status: Not connected	Data out: 0 B
Location: East US	Virtual network: TestVNet1
Subscription name: Windows Azure Internal Consumption	Virtual network gateway: VNet1GW (13.92.133.158)
Subscription ID	Local network gateway: VNet1LocalNet (33.2.1.5)

PowerShell: To verify your connection with PowerShell

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

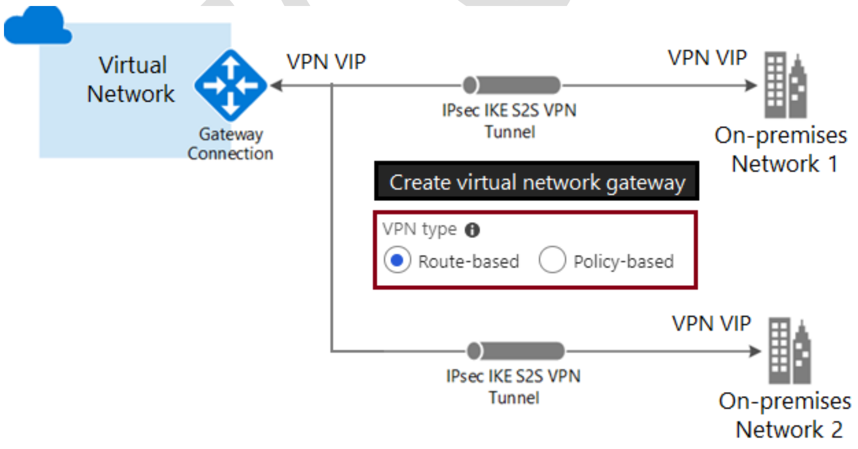
After the cmdlet has finished, view the values. The connection status shouldshow 'Connected' and you can see ingress and egress bytes.

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

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

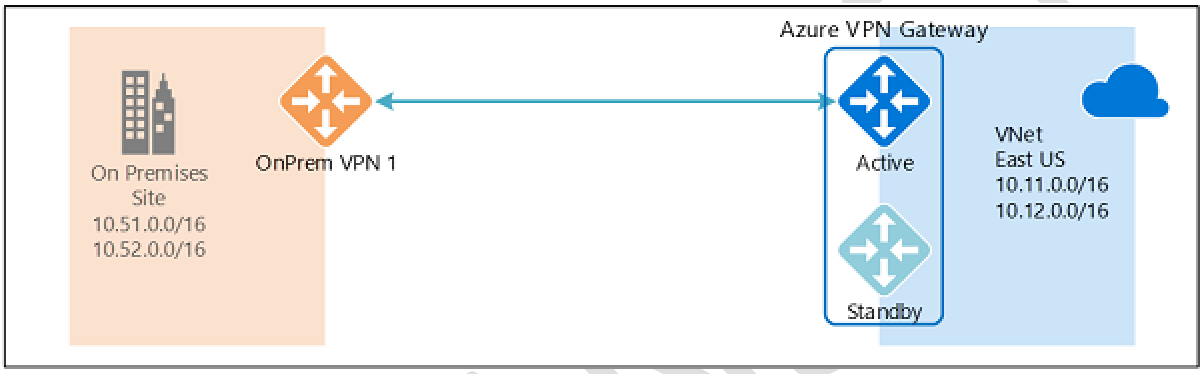
Multiple Sites

A Multi-site connection is a variation of the Site-to-Site connection. You create more than one VPN connection from your virtual network gateway, typically connecting to multiple on-premises sites. When working with multiple connections, you must use a Route-based VPN. Because each virtual network can **only have one VPN gateway**, all connections through the gateway share the available bandwidth.

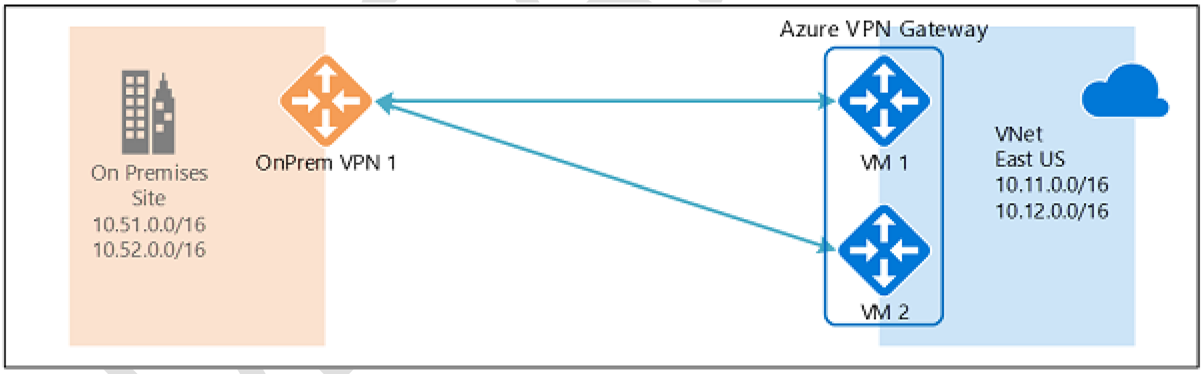


About active-active mode for Availability:

Every Azure VPN gateway consists of **two instances** in an **active-standby** configuration. For any planned maintenance or unplanned disruption that happens to the active instance, the standby instance would take over (failover) **automatically**, and resume the S2S VPN or VNet-to-VNet connections. **The switch over will cause a brief interruption**. For planned maintenance, the connectivity should be restored **within 10 to 15 seconds**. For unplanned issues, the connection recovery will be longer, about 1 minute to 1 and a half minutes in the worst case. For P2S VPN client connections to the gateway, the P2S connections will be disconnected and the users will need to reconnect from the client machines.



You can now create an Azure VPN gateway in an **active-active configuration**, where both instances of the gateway VMs will establish S2S VPN tunnels to your on-premises VPN device, as shown the following diagram:

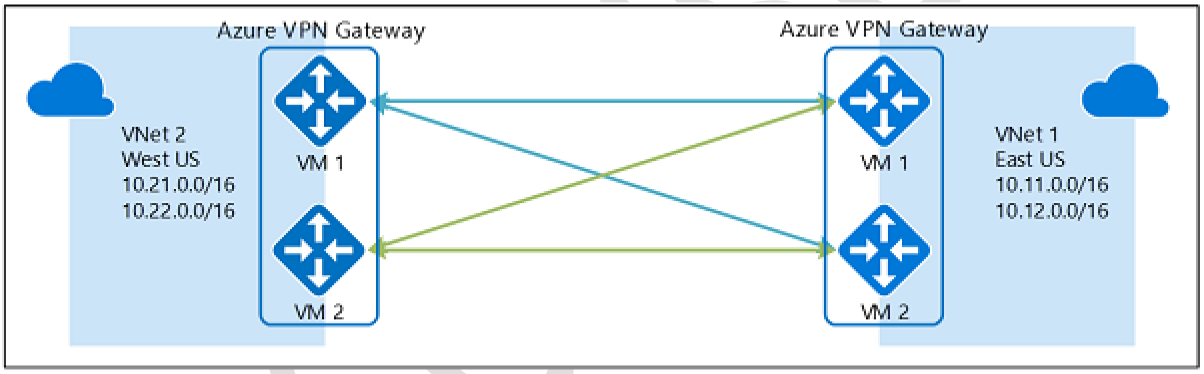


In this configuration, each Azure gateway instance will have a unique public IP address, and each will establish an IPsec/IKE S2S VPN tunnel to your on-premises VPN device specified in your local network gateway and connection. Note that both VPN tunnels are actually part of the same connection. You will still need to configure your on-premises VPN device to accept or establish two S2S VPN tunnels to those two Azure VPN gateway public IP addresses.

Because the Azure gateway instances are in active-active configuration, the traffic from your Azure virtual network to your on-premises network will be routed **through both tunnels simultaneously**, even if your on-premises VPN device may favor one tunnel over the other. Note though the same TCP or UDP flow will always traverse the same tunnel or path, unless a maintenance event happens on one of the instances.

When a planned maintenance or unplanned event happens to one gateway instance, the IPsec tunnel from that instance to your on-premises VPN device will be disconnected. The corresponding routes on your VPN devices should be removed or withdrawn automatically so that the traffic will be switched over to the other active IPsec tunnel. On the Azure side, the switch over will happen automatically from the affected instance to the active instance.

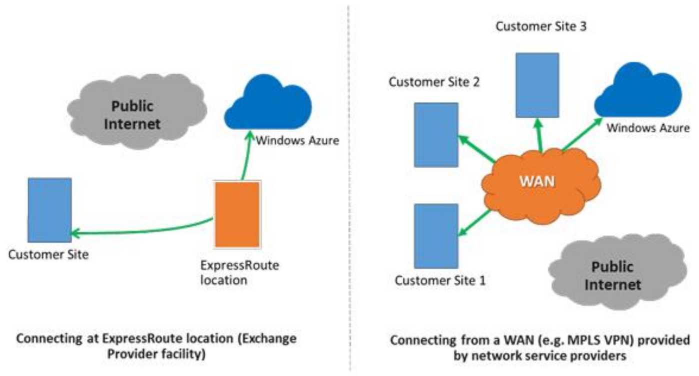
You can create active-active VPN gateways for both virtual networks, and connect them together to form the same full mesh connectivity of 4 tunnels between the two VNets, as shown in the diagram below:



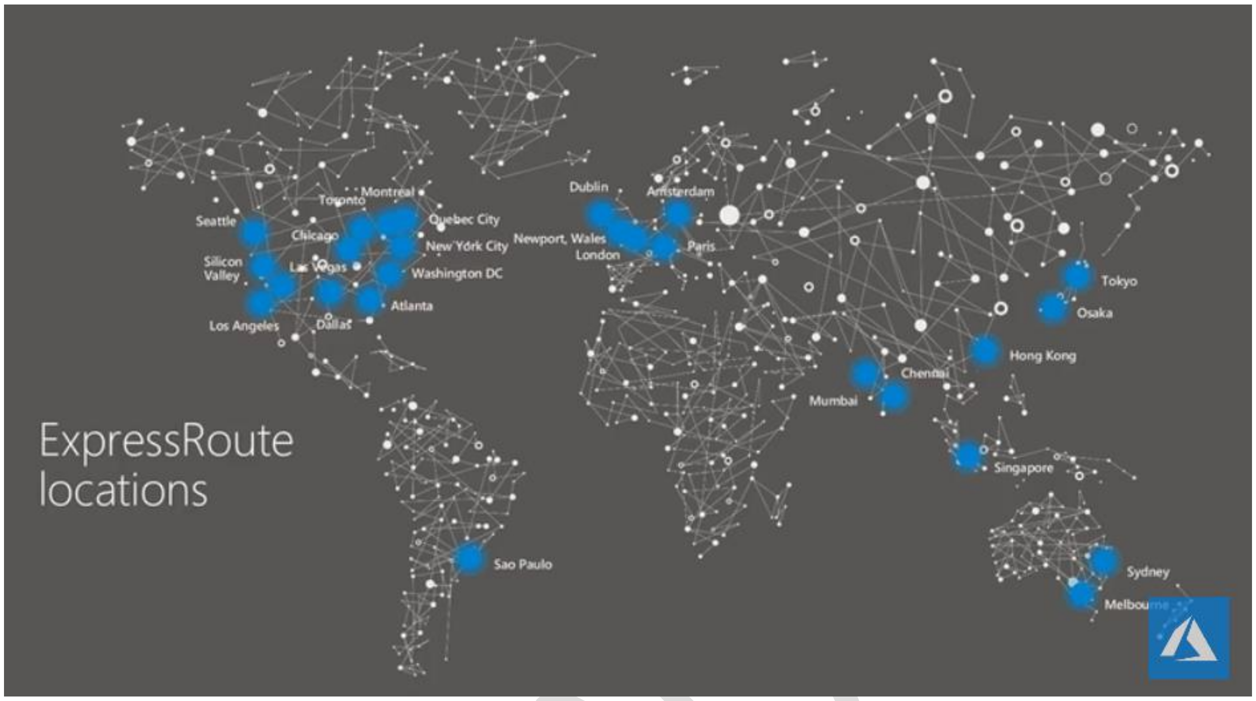
This ensures there are always a pair of tunnels between the two virtual networks for any planned maintenance events, providing even better availability.

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**.


Public Internet
Windows Azure
Customer Site
ExpressRoute location
Connecting at ExpressRoute location (Exchange Provider facility)
Customer Site 3
Customer Site 2
Windows Azure
WAN
Customer Site 1
Public Internet
Connecting from a WAN (e.g. MPLS VPN) provided by network service providers

Express Route Partners to deliver Express Route connectivity



https://youtu.be/HZH6F_gLCQQ

https://youtu.be/wystDyqyRXc

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