Table of Contents

- Application Gateway
- Load Balancer
- Azure DNS

Resources

- Pricing
- SLA
- Service Limits
- Blog
- MSDN Forum
- Service updates
- Videos

Overview of Traffic Manager

1/17/2017 • 3 min to read • Edit on GitHub

Microsoft Azure Traffic Manager allows you to control the distribution of user traffic for service endpoints in different datacenters. 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. Traffic Manager provides a range of traffic-routing methods to suit different application needs, endpoint health monitoring, and automatic failover. Traffic Manager is resilient to failure, including the failure of an entire Azure region.

Traffic Manager benefits

Traffic Manager can help you:

- **Improve availability of critical applications**

  Traffic Manager delivers high availability for your applications by monitoring your endpoints and providing automatic failover when an endpoint goes down.

- **Improve responsiveness for high-performance applications**

  Azure allows you to run cloud services or websites in datacenters located around the world. Traffic Manager improves application responsiveness by directing traffic to the endpoint with the lowest network latency for the client.

- **Perform service maintenance without downtime**

  You can perform planned maintenance operations on your applications without downtime. Traffic Manager directs traffic to alternative endpoints while the maintenance is in progress.

- **Combine on-premises and Cloud-based applications**

  Traffic Manager supports external, non-Azure endpoints enabling it to be used with hybrid cloud and on-premises deployments, including the "burst-to-cloud," "migrate-to-cloud," and "failover-to-cloud" scenarios.

- **Distribute traffic for large, complex deployments**

  Using nested Traffic Manager profiles, traffic-routing methods can be combined to create sophisticated and flexible rules to support the needs of larger, more complex deployments.

Load Balancer differences

There are different options to distribute network traffic using Microsoft Azure. These options work differently from each other, having a different feature set and support different scenarios. They can each be used in isolation, or combining them.

- **Azure Load Balancer** works at the transport layer (Layer 4 in the OSI network reference stack). It provides network-level distribution of traffic across instances of an application running in the same Azure data center.
- **Application Gateway** works at the application layer (Layer 7 in the OSI network reference stack). It acts as a reverse-proxy service, terminating the client connection and forwarding requests to back-end endpoints.
- **Traffic Manager** works at the DNS level. It uses DNS responses to direct end-user traffic to globally distributed endpoints. Clients then connect to those endpoints directly.

The following table summarizes the features offered by each service:

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 GET

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
Protocols	UDP/TCP	HTTP/ HTTPS
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
Load balancing mode (source IP /sticky sessions)	2-tuple (source IP and destination IP), 3-tuple (source IP, destination IP, and port). Can scale up or down based on the number of virtual machines	Cookie-based affinity Routing based on URL
Health probes	Default: probe interval - 15 secs. Taken out of rotation: 2 Continuous failures. Supports user-defined probes	Idle probe interval 30 secs. Taken out after 5 consecutive live traffic failures or a single probe failure in idle mode. Supports user-defined probes
SSL offloading	Not supported	Supported

Next Steps

- Learn more about how Traffic Manager works.
- Learn how to develop high-availability applications using Traffic Manager endpoint monitoring.
- Learn more about the traffic-routing methods supported by Traffic Manager.
- Create a Traffic Manager profile.

How Traffic Manager works

1/17/2017 • 8 min to read • Edit on GitHub

Azure Traffic Manager enables you to control the distribution of traffic across your application endpoints. An endpoint is any Internet-facing service hosted inside or outside of Azure.

Traffic Manager provides two key benefits:

1. Distribution of traffic according to one of several traffic-routing methods
2. Continuous monitoring of endpoint health and automatic failover when endpoints fail

When a client attempts to connect to a service, it must first resolve the DNS name of the service to an IP address. The client then connects to that IP address to access the service.

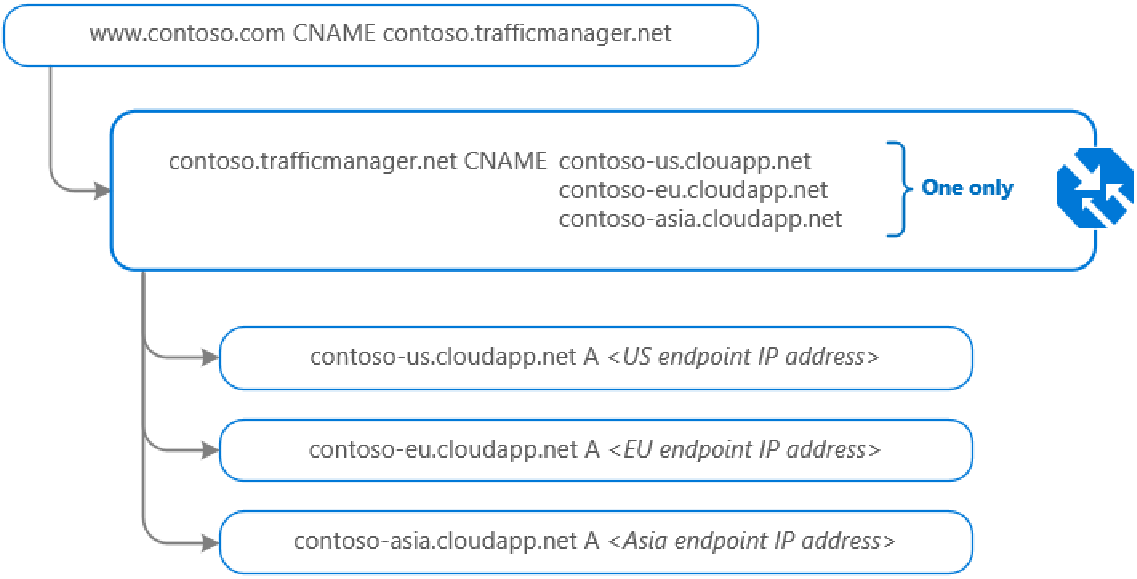
The most important point to understand is that Traffic Manager works at the DNS level. Traffic Manager uses DNS to direct clients to specific service endpoints based on the rules of the traffic-routing method. Clients connect to the selected endpoint **directly**. Traffic Manager is not a proxy or a gateway. Traffic Manager does not see the traffic passing between the client and the service.

Traffic Manager example

Contoso Corp have developed a new partner portal. The URL for this portal is https://partners.contoso.com/login.aspx. The application is hosted in three regions of Azure. To improve availability and maximize global performance, they use Traffic Manager to distribute client traffic to the closest available endpoint.

To achieve this configuration:

- They deploy three instances of their service. The DNS names of these deployments are 'contoso-us.cloudapp.net', 'contoso-eu.cloudapp.net', and 'contoso-asia.cloudapp.net'.
- They then create a Traffic Manager profile, named 'contoso.trafficmanager.net', and configure it to use the 'Performance' traffic-routing method across the three endpoints.
- Finally, they configure their vanity domain name, 'partners.contoso.com', to point to 'contoso.trafficmanager.net', using a DNS CNAME record.

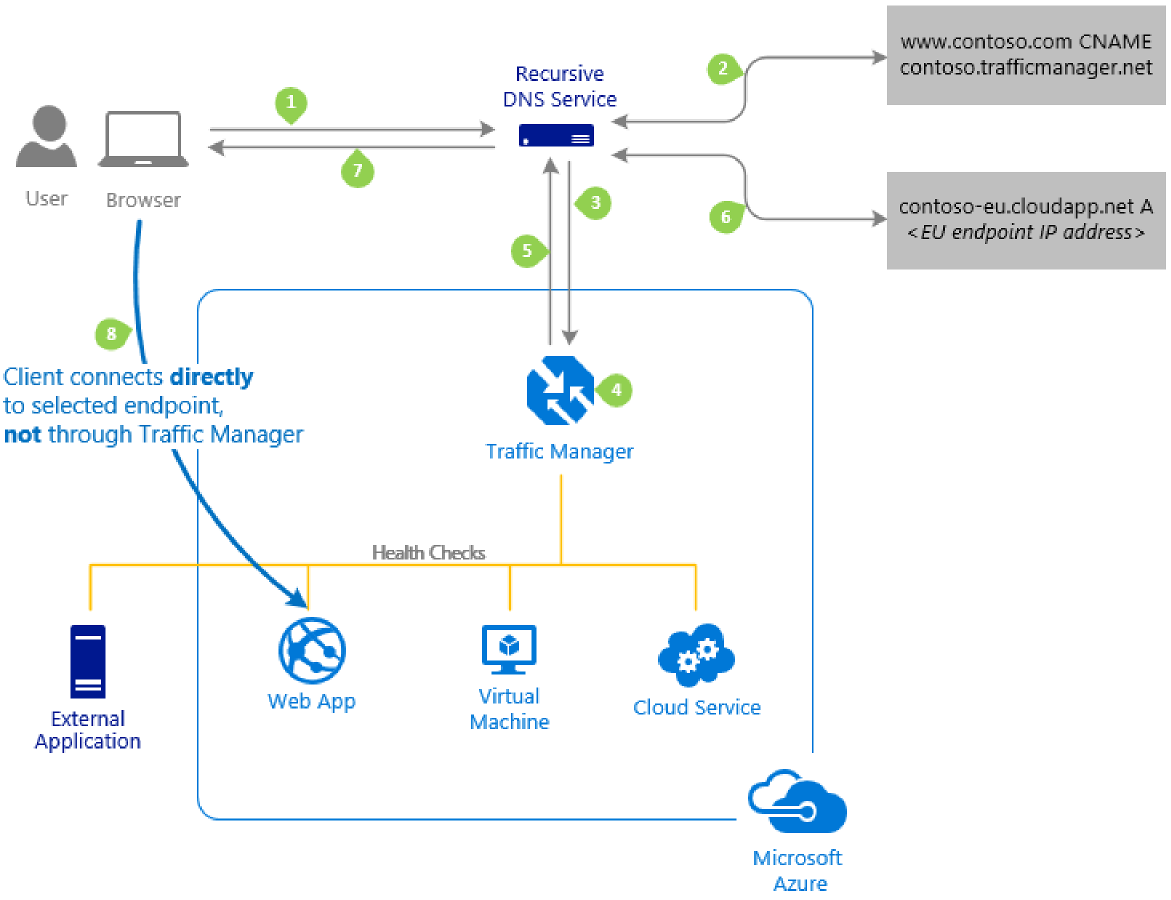

www.contoso.com CNAME contoso.trafficmanager.net
contoso.trafficmanager.net CNAME contoso-us.clouapp.net
contoso-eu.cloudapp.net
contoso-asia.cloudapp.net
One only
contoso-us.cloudapp.net A <US endpoint IP address>
contoso-eu.cloudapp.net A <EU endpoint IP address>
contoso-asia.cloudapp.net A <Asia endpoint IP address>

NOTE

When using a vanity domain with Azure Traffic Manager, you must use a CNAME to point your vanity domain name to your Traffic Manager domain name. DNS standards do not allow you to create a CNAME at the 'apex' (or root) of a domain. Thus you cannot create a CNAME for 'contoso.com' (sometimes called a 'naked' domain). You can only create a CNAME for a domain under 'contoso.com', such as 'www.contoso.com'. To work around this limitation, we recommend using a simple HTTP redirect to direct requests for 'contoso.com' to an alternative name such as 'www.contoso.com'.

How clients connect using Traffic Manager

Continuing from the previous example, when a client requests the page https://partners.contoso.com/login.aspx, the client performs the following steps to resolve the DNS name and establish a connection:



1. The client sends a DNS query to its configured recursive DNS service to resolve the name 'partners.contoso.com'. A recursive DNS service, sometimes called a 'local DNS' service, does not host DNS domains directly. Rather, the client off-loads the work of contacting the various authoritative DNS services across the Internet needed to resolve a DNS name.
2. To resolve the DNS name, the recursive DNS service finds the name servers for the 'contoso.com' domain. It then contacts those name servers to request the 'partners.contoso.com' DNS record. The contoso.com DNS servers return the CNAME record that points to contoso.trafficmanager.net.
3. Next, the recursive DNS service finds the name servers for the 'trafficmanager.net' domain, which are provided by the Azure Traffic Manager service. It then sends a request for the 'contoso.trafficmanager.net' DNS record to those DNS servers.
4. The Traffic Manager name servers receive the request. They choose an endpoint based on:
   - The configured state of each endpoint (disabled endpoints are not returned)
   - The current health of each endpoint, as determined by the Traffic Manager health checks. For more information, see Traffic Manager Endpoint Monitoring.
   - The chosen traffic-routing method. For more information, see Traffic Manager Routing Methods.
5. The chosen endpoint is returned as another DNS CNAME record. In this case, let us suppose contoso-us.cloudapp.net is returned.
6. Next, the recursive DNS service finds the name servers for the 'cloudapp.net' domain. It contacts those name servers to request the 'contoso-us.cloudapp.net' DNS record. A DNS 'A' record containing the IP address of the US-based service endpoint is returned.
7. The recursive DNS service consolidates the results and returns a single DNS response to the client.
8. The client receives the DNS results and connects to the given IP address. The client connects to the application service endpoint directly, not through Traffic Manager. Since it is an HTTPS endpoint, the client performs the necessary SSL/TLS handshake, and then makes an HTTP GET request for the '/login.aspx' page.

The recursive DNS service caches the DNS responses it receives. The DNS resolver on the client device also caches the result. Caching enables subsequent DNS queries to be answered more quickly by using data from the cache rather than querying other name servers. The duration of the cache is determined by the 'time-to-live' (TTL) property of each DNS record. Shorter values result in faster cache expiry and thus more round-trips to the Traffic Manager name servers. Longer values mean that it can take longer to direct traffic away from a failed endpoint. Traffic Manager allows you to configure the TTL used in Traffic Manager DNS responses, enabling you to choose the value that best balances the needs of your application.

FAQ

What IP address does Traffic Manager use?

As explained in How Traffic Manager Works, Traffic Manager works at the DNS level. It sends DNS responses to direct clients to the appropriate service endpoint. Clients then connect to the service endpoint directly, not through Traffic Manager.

Therefore, Traffic Manager does not provide an endpoint or IP address for clients to connect to. Therefore, if you want static IP address for your service, that must be configured at the service, not in Traffic Manager.

Does Traffic Manager support 'sticky' sessions?

As explained previously, Traffic Manager works at the DNS level. It uses DNS responses to direct clients to the appropriate service endpoint. Clients connect to the service endpoint directly, not through Traffic Manager. Therefore, Traffic Manager does not see the HTTP traffic between the client and the server.

Additionally, the source IP address of the DNS query received by Traffic Manager belongs to the recursive DNS service, not the client. Therefore, Traffic Manager has no way to track individual clients and cannot implement 'sticky' sessions. This limitation is common to all DNS-based traffic management systems and is not specific to Traffic Manager.

Why am I seeing an HTTP error when using Traffic Manager?

As explained previously, Traffic Manager works at the DNS level. It uses DNS responses to direct clients to the appropriate service endpoint. Clients then connect to the service endpoint directly, not through Traffic Manager. Traffic Manager does not see HTTP traffic between client and server. Therefore, any HTTP error you see must be coming from your application. For the client to connect to the application, all DNS resolution steps are complete. That includes any interaction that Traffic Manager has on the application traffic flow.

Further investigation should therefore focus on the application.

The HTTP host header sent from the client's browser is the most common source of problems. Make sure that the application is configured to accept the correct host header for the domain name you are using. For endpoints using the Azure App Service, see configuring a custom domain name for a web app in Azure App Service using Traffic Manager.

What is the performance impact of using Traffic Manager?

As explained previously, Traffic Manager works at the DNS level. Since clients connect to your service endpoints directly, there is no performance impact incurred when using Traffic Manager once the connection is established.

Since Traffic Manager integrates with applications at the DNS level, it does require an additional DNS lookup to be inserted into the DNS resolution chain (see Traffic Manager examples). The impact of Traffic Manager on DNS resolution time is minimal. Traffic Manager uses a global network of name servers, and uses anycast networking to ensure DNS queries are always routed to the closest available name server. In addition, caching of DNS responses means that the additional DNS latency incurred by using Traffic Manager applies only to a fraction of sessions.

The Performance method routes traffic to the closest available endpoint. The net result is that the overall performance impact associated with this method should be minimal. Any increase in DNS latency should be offset by lower network latency to the endpoint.

What application protocols can I use with Traffic Manager?

As explained previously, Traffic Manager works at the DNS level. Once the DNS lookup is complete, clients connect to the application endpoint directly, not through Traffic Manager. Therefore the connection can use any application protocol. However, Traffic Manager's endpoint health checks require either an HTTP or HTTPS endpoint. The endpoint for a health check can be different than the application endpoint that clients connect to.

Can I use Traffic Manager with a 'naked' domain name?

No. The DNS standards do not permit CNAMEs to co-exist with other DNS records of the same name. The apex (or root) of a DNS zone always contains two pre-existing DNS records; the SOA and the authoritative NS records. This means a CNAME record cannot be created at the zone apex without violating the DNS standards.

As explained in the Traffic Manager example, Traffic Manager requires a DNS CNAME record to map the vanity DNS name. For example, you map www.contoso.com to the Traffic Manager profile DNS name contoso.trafficmanager.net. Additionally, the Traffic Manager profile returns a second DNS CNAME to indicate which endpoint the client should connect to.

To work around this issue, we recommend using an HTTP redirect to direct traffic from the naked domain name to a different URL, which can then use Traffic Manager. For example, the naked domain 'contoso.com' can redirect users to the CNAME 'www.contoso.com' that points to the Traffic Manager DNS name.

Full support for naked domains in Traffic Manager is tracked in our feature backlog. You can register your support for this feature request by voting for it on our community feedback site.

Next steps

Learn more about Traffic Manager endpoint monitoring and automatic failover.

Learn more about Traffic Manager traffic routing methods.

Traffic Manager endpoints

1/17/2017 • 8 min to read • Edit on GitHub

Microsoft Azure Traffic Manager allows you to control how network traffic is distributed to application deployments running in different datacenters. You configure each application deployment as an 'endpoint' in Traffic Manager. When Traffic Manager receives a DNS request, it chooses an available endpoint to return in the DNS response. Traffic manager bases the choice on the current endpoint status and the traffic-routing method. For more information, see How Traffic Manager Works.

There are three types of endpoint supported by Traffic Manager:

- **Azure endpoints** are used for services hosted in Azure.
- **External endpoints** are used for services hosted outside Azure, either on-premises or with a different hosting provider.
- **Nested endpoints** are used to combine Traffic Manager profiles to create more flexible traffic-routing schemes to support the needs of larger, more complex deployments.

There is no restriction on how endpoints of different types are combined in a single Traffic Manager profile. Each profile can contain any mix of endpoint types.

The following sections describe each endpoint type in greater depth.

Azure endpoints

Azure endpoints are used for Azure-based services in Traffic Manager. The following Azure resource types are supported:

- 'Classic' IaaS VMs and PaaS cloud services.
- Web Apps
- PublicIPAddress resources (which can be connected to VMs either directly or via an Azure Load Balancer). The publicIpAddress must have a DNS name assigned to be used in a Traffic Manager profile.

PublicIPAddress resources are Azure Resource Manager resources. They do not exist in the classic deployment model. Thus they are only supported in Traffic Manager's Azure Resource Manager experiences. The other endpoint types are supported via both Resource Manager and the classic deployment model.

When using Azure endpoints, Traffic Manager detects when a 'Classic' IaaS VM, cloud service, or a Web App is stopped and started. This status is reflected in the endpoint status. See Traffic Manager endpoint monitoring for details. When the underlying service is stopped, Traffic Manager does not perform endpoint health checks or direct traffic to the endpoint. No Traffic Manager billing events occur for the stopped instance. When the service is restarted, billing resumes and the endpoint is eligible to receive traffic. This detection does not apply to PublicIpAddress endpoints.

External endpoints

External endpoints are used for services outside of Azure. For example, a service hosted on-premises or with a different provider. External endpoints can be used individually or combined with Azure Endpoints in the same Traffic Manager profile. Combining Azure endpoints with External endpoints enables various scenarios:

- In either an active-active or active-passive failover model, use Azure to provide increased redundancy for an existing on-premises application.
- To reduce application latency for users around the world, extend an existing on-premises application to

additional geographic locations in Azure. For more information, see Traffic Manager 'Performance' traffic routing.
- Use Azure to provide additional capacity for an existing on-premises application, either continuously or as a 'burst-to-cloud' solution to meet a spike in demand.

In certain cases, it is useful to use External endpoints to reference Azure services (for examples, see the FAQ). In this case, health checks are billed at the Azure endpoints rate, not the External endpoints rate. However, unlike Azure endpoints, if you stop or delete the underlying service, health check billing continues until you disable or delete the endpoint in Traffic Manager.

Nested endpoints

Nested endpoints combine multiple Traffic Manager profiles to create flexible traffic-routing schemes and support the needs of larger, complex deployments. With Nested endpoints, a 'child' profile is added as an endpoint to a 'parent' profile. Both the child and parent profiles can contain other endpoints of any type, including other nested profiles. For more information, see nested Traffic Manager profiles.

Web Apps as endpoints

Some additional considerations apply when configuring Web Apps as endpoints in Traffic Manager:

1. Only Web Apps at the 'Standard' SKU or above are eligible for use with Traffic Manager. Attempts to add a Web App of a lower SKU fail. Downgrading the SKU of an existing Web App results in Traffic Manager no longer sending traffic to that Web App.
2. When an endpoint receives an HTTP request, it uses the 'host' header in the request to determine which Web App should service the request. The host header contains the DNS name used to initiate the request, for example 'contosoapp.azurewebsites.net'. To use a different DNS name with your Web App, the DNS name must be registered as a custom domain name for the App. When adding a Web App endpoint as an Azure endpoint, the Traffic Manager profile DNS name is automatically registered for the App. This registration is automatically removed when the endpoint is deleted.
3. Each Traffic Manager profile can have at most one Web App endpoint from each Azure region. To work around for this constraint, you can configure a Web App as an External endpoint. For more information, see the FAQ.

Enabling and disabling endpoints

Disabling an endpoint in Traffic Manager can be useful to temporarily remove traffic from an endpoint that is in maintenance mode or being redeployed. Once the endpoint is running again, it can be re-enabled.

Endpoints can be enabled and disabled via the Traffic Manager portal, PowerShell, CLI or REST API, all of which are supported in both Resource Manager and the classic deployment model.

> **NOTE**
>
> Disabling an Azure endpoint has nothing to do with its deployment state in Azure. An Azure service (such as a VM or Web App remains running and able to receive traffic even when disabled in Traffic Manager. Traffic can be addressed directly to the service instance rather than via the Traffic Manager profile DNS name. For more information, see how Traffic Manager works.

The current eligibility of each endpoint to receive traffic depends on the following factors:

- The profile status (enabled/disabled)
- The endpoint status (enabled/disabled)
- The results of the health checks for that endpoint

For details, see Traffic Manager endpoint monitoring.

> **NOTE**
>
> Since Traffic Manager works at the DNS level, it is unable to influence existing connections to any endpoint. When an endpoint is unavailable, Traffic Manager directs new connections to another available endpoint. However, the host behind the disabled or unhealthy endpoint may continue to receive traffic via existing connections until those sessions are terminated. Applications should limit the session duration to allow traffic to drain from existing connections.

If all endpoints in a profile are disabled, or if the profile itself is disabled, then Traffic Manager sends an 'NXDOMAIN' response to a new DNS query.

FAQ

Can I use Traffic Manager with endpoints from multiple subscriptions?

Using endpoints from multiple subscriptions is not possible with Azure Web Apps. Azure Web Apps requires that any custom domain name used with Web Apps is only used within a single subscription. It is not possible to use Web Apps from multiple subscriptions with the same domain name.

For other endpoint types, it is possible to use Traffic Manager with endpoints from more than one subscription. How you configure Traffic Manager depends on whether you are using the classic deployment model or the Resource Manager experience.

- In Resource Manager, endpoints from any subscription can be added to Traffic Manager, so long as the person configuring the Traffic Manager profile has read access to the endpoint. These permissions can be granted using Azure Resource Manager role-based access control (RBAC).
- In the classic deployment model interface, Traffic Manager requires that Cloud Services or Web Apps configured as Azure endpoints reside in the same subscription as the Traffic Manager profile. Cloud Service endpoints in other subscriptions can be added to Traffic Manager as 'external' endpoints. These external endpoints are billed as Azure endpoints, rather than the external rate.

Can I use Traffic Manager with Cloud Service 'Staging' slots?

Yes. Cloud Service 'staging' slots can be configured in Traffic Manager as External endpoints. Health checks are still be charged at the Azure Endpoints rate. Because the External endpoint type is in use, changes to the underlying service are not picked up automatically. With external endpoints, Traffic Manager cannot detect when the Cloud Service is stopped or deleted. Therefore, the Traffic Manager continues billing for health checks until the endpoint is disabled or deleted.

Does Traffic Manager support IPv6 endpoints?

Traffic Manager does not currently provide IPv6-addressible name servers. However, Traffic Manager can still be used by IPv6 clients connecting to IPv6 endpoints. A client does not make DNS requests directly to Traffic Manager. Instead, the client uses a recursive DNS service. An IPv6-only client sends requests to the recursive DNS service via IPv6. Then the recursive service should be able to contact the Traffic Manager name servers using IPv4.

Traffic Manager responds with the DNS name of the endpoint. To support an IPv6 endpoint, a DNS AAAA record pointing the endpoint DNS name to the IPv6 address must exist. Traffic Manager health checks only support IPv4 addresses. The service needs to expose an IPv4 endpoint on the same DNS name.

Can I use Traffic Manager with more than one Web App in the same region?

Typically, Traffic Manager is used to direct traffic to applications deployed in different regions. However, it can also be used where an application has more than one deployment in the same region. The Traffic Manager Azure endpoints do not permit more than one Web App endpoint from the same Azure region to be added to the same Traffic Manager profile.

The following steps provide a workaround to this constraint:

1. Check that your endpoints are in different web app 'scale units'. A domain name must map to a single site in a given scale unit. Therefore, two Web Apps in the same scale unit cannot share a Traffic Manager profile.
2. Add your vanity domain name as a custom hostname to each Web App. Each Web App must be in a different scale unit. All Web Apps must belong to the same subscription.
3. Add one (and only one) Web App endpoint to your Traffic Manager profile, as an Azure endpoint.
4. Add each additional Web App endpoint to your Traffic Manager profile as an External endpoint. External endpoints can only be added using the Resource Manager deployment model.
5. Create a DNS CNAME record in your vanity domain that points to your Traffic Manager profile DNS name (<...>.trafficmanager.net).
6. Access your site via the vanity domain name, not the Traffic Manager profile DNS name.

Next steps

- Learn how Traffic Manager works.
- Learn about Traffic Manager endpoint monitoring and automatic failover.
- Learn about Traffic Manager traffic routing methods.

Traffic Manager endpoint monitoring and failover

1/17/2017 • 11 min to read • Edit on GitHub

Azure Traffic Manager includes built-in endpoint monitoring and automatic endpoint failover. This feature helps you deliver high-availability applications that are resilient to endpoint failure, including Azure region failures.

Configure endpoint monitoring

To configure endpoint monitoring, you must specify the following settings on your Traffic Manager profile:

- **Protocol**. Choose HTTP or HTTPS. It's important to note that HTTPS monitoring does not verify whether your SSL certificate is valid--it only checks that the certificate is present.
- **Port**. Choose the port used for the request.
- **Path**. Give the relative path and the name of the webpage or file that the monitoring accesses. A forward slash (/) is a valid entry for the relative path. This value implies that the file is in the root directory (default).

To check the health of each endpoint, Traffic Manager makes a GET request to the endpoint using the protocol, port, and relative path given.

A common practice is to implement a custom page within your application, for example, /health.aspx. Using this path for monitoring, you can perform application-specific checks, such as checking performance counters or verifying database availability. Based on these custom checks, the page returns an appropriate HTTP status code.

All endpoints in a Traffic Manager profile share monitoring settings. If you need to use different monitoring settings for different endpoints, you can create nested Traffic Manager profiles.

Endpoint and profile status

You can enable and disable Traffic Manager profiles and endpoints. However, a change in endpoint status also might occur as a result of Traffic Manager automated settings and processes.

Endpoint status

You can enable or disable a specific endpoint. The underlying service, which might still be healthy, is unaffected. Changing the endpoint status controls the availability of the endpoint in the Traffic Manager profile. When an endpoint status is disabled, Traffic Manager does not check its health and the endpoint is not included in a DNS response.

Profile status

Using the profile status setting, you can enable or disable a specific profile. While endpoint status affects a single endpoint, profile status affects the entire profile, including all endpoints. When you disable a profile, the endpoints are not checked for health and no endpoints are included in a DNS response. An NXDOMAIN response code is returned for the DNS query.

Endpoint monitor status

Endpoint monitor status is a Traffic Manager-generated value that shows the status of the endpoint. You cannot change this setting manually. The endpoint monitor status is a combination of the results of endpoint monitoring and the configured endpoint status. The possible values of endpoint monitor status are shown in the following table:

PROFILE STATUS	ENDPOINT STATUS	ENDPOINT MONITOR STATUS	NOTES
Disabled	Enabled	Inactive	The profile has been disabled. Although the endpoint status is Enabled, the profile status (Disabled) takes precedence. Endpoints in disabled profiles are not monitored. An NXDOMAIN response code is returned for the DNS query.
<any>	Disabled	Disabled	The endpoint has been disabled. Disabled endpoints are not monitored. The endpoint is not included in DNS responses, therefore, it does not receive traffic.
Enabled	Enabled	Online	The endpoint is monitored and is healthy. It is included in DNS responses and can receive traffic.
Enabled	Enabled	Degraded	Endpoint monitoring health checks are failing. The endpoint is not included in DNS responses and does not receive traffic.
Enabled	Enabled	CheckingEndpoint	The endpoint is monitored, but the results of the first probe have not been received yet. CheckingEndpoint is a temporary state that usually occurs immediately after adding or enabling an endpoint in the profile. An endpoint in this state is included in DNS responses and can receive traffic.
Enabled	Enabled	Stopped	The cloud service or web app that the endpoint points to is not running. Check the cloud service or web app settings. An endpoint with a Stopped status is not monitored. It is not included in DNS responses and does not receive traffic.

For details about how endpoint monitor status is calculated for nested endpoints, see nested Traffic Manager profiles.

Profile monitor status

The profile monitor status is a combination of the configured profile status and the endpoint monitor status values for all endpoints. The possible values are described in the following table:

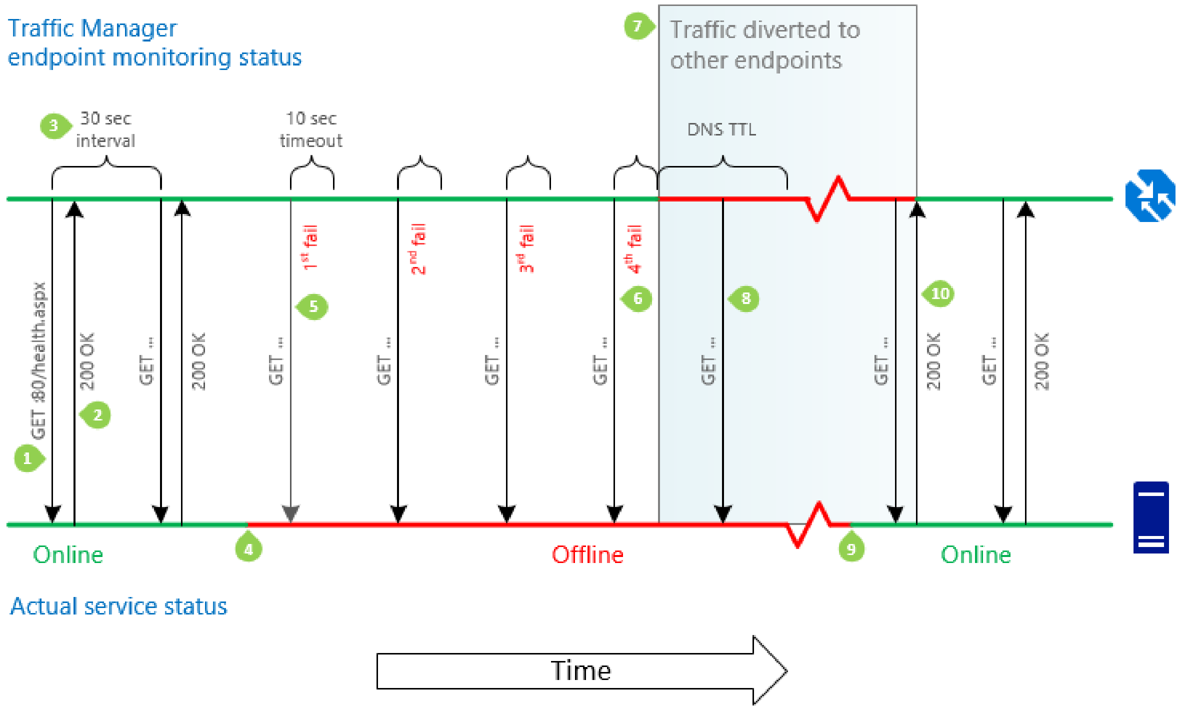
PROFILE STATUS (AS CONFIGURED)	ENDPOINT MONITOR STATUS	PROFILE MONITOR STATUS	NOTES
Disabled	<any> or a profile with no defined endpoints.	Disabled	The profile has been disabled.
Enabled	The status of at least one endpoint is Degraded.	Degraded	Review the individual endpoint status values to determine which endpoints require further attention.
Enabled	The status of at least one endpoint is Online. No endpoints have a Degraded status.	Online	The service is accepting traffic. No further action is required.
Enabled	The status of at least one endpoint is CheckingEndpoint. No endpoints are in Online or Degraded status.	CheckingEndpoints	This transition state occurs when a profile if created or enabled. The endpoint health is being checked for the first time.
Enabled	The statuses of all endpoints in the profile are either Disabled or Stopped, or the profile has no defined endpoints.	Inactive	No endpoints are active, but the profile is still Enabled.

Endpoint failover and recovery

Traffic Manager periodically checks the health of every endpoint, including unhealthy endpoints. Traffic Manager detects when an endpoint becomes healthy and brings it back into rotation.

> **NOTE**
>
> Traffic Manager only considers an endpoint to be online if the return message is 200 OK. An endpoint is unhealthy when any of the following events occur:
>
> - A non-200 response is received (including a different 2xx code, or a 301/302 redirect)
> - Request for client authentication
> - Timeout (the timeout threshold is 10 seconds)
> - Unable to connect
>
> For more information about troubleshooting failed checks, see Troubleshooting Degraded status on Azure Traffic Manager.

The following timeline is a detailed description of the monitoring process.



1. **GET**. For each endpoint, the Traffic Manager monitoring system performs a GET request on the path and file specified in the monitoring settings.
2. **200 OK**. The monitoring system expects an HTTP 200 OK message to be returned within 10 seconds. When it receives this response, it recognizes that the service is available.
3. **30 seconds between checks**. The endpoint health check is repeated every 30 seconds.
4. **Service unavailable**. The service becomes unavailable. Traffic Manager will not know until the next health check.
5. **Attempts to access monitoring file (four tries)**. The monitoring system performs a GET request, but does not receive a response within the timeout period of 10 seconds (alternatively, a non-200 response may be received). It then tries three more times, at 30-second intervals. If one of the tries is successful, then the number of tries is reset.
6. **Status set to Degraded**. After a fourth consecutive failure, the monitoring system marks the unavailable endpoint status as Degraded.
7. **Traffic is diverted to other endpoints**. The Traffic Manager DNS name servers are updated and Traffic Manager no longer returns the endpoint in response to DNS queries. New connections are directed to other, available endpoints. However, previous DNS responses that include this endpoint may still be cached by recursive DNS servers and DNS clients. Clients continue to use the endpoint until the DNS cache expires. As the DNS cache expires, clients make new DNS queries and are directed to different endpoints. The cache duration is controlled by the TTL setting in the Traffic Manager profile, for example, 30 seconds.
8. **Health checks continue**. Traffic Manager continues to check the health of the endpoint while it has a Degraded status. Traffic Manager detects when the endpoint returns to health.
9. **Service comes back online**. The service becomes available. The endpoint retains its Degraded status in Traffic Manager until the monitoring system performs its next health check.
10. **Traffic to service resumes**. Traffic Manager sends a GET request and receives a 200 OK status response. The service has returned to a healthy state. The Traffic Manager name servers are updated, and they begin to hand out the service's DNS name in DNS responses. Traffic returns to the endpoint as cached DNS responses that return other endpoints expire, and as existing connections to other endpoints are terminated.

NOTE

Because Traffic Manager works at the DNS level, it cannot influence existing connections to any endpoint. When it directs traffic between endpoints (either by changed profile settings, or during failover or failback), Traffic Manager directs new connections to available endpoints. However, other endpoints might continue to receive traffic via existing connections until those sessions are terminated. To enable traffic to drain from existing connections, applications should limit the session duration used with each endpoint.

Traffic-routing methods

When an endpoint has a Degraded status, it is no longer returned in response to DNS queries. Instead, an alternative endpoint is chosen and returned. The traffic-routing method configured in the profile determines how the alternative endpoint is chosen.

- **Priority**. Endpoints form a prioritized list. The first available endpoint on the list is always returned. If an endpoint status is Degraded, then the next available endpoint is returned.
- **Weighted**. Any available endpoint is chosen at random based on their assigned weights and the weights of the other available endpoints.
- **Performance**. The endpoint closest to the end user is returned. If that endpoint is unavailable, an endpoint is randomly chosen from all the other available endpoints. Choosing a random endpoint avoids a cascading failure that can occur when the next-closest endpoint becomes overloaded. You can configure alternative failover plans for performance traffic-routing by using nested Traffic Manager profiles.

For more information, see Traffic Manager traffic-routing methods.

NOTE

One exception to normal traffic-routing behavior occurs when all eligible endpoints have a degraded status. Traffic Manager makes a "best effort" attempt and *responds as if all the Degraded status endpoints actually are in an online state*. This behavior is preferable to the alternative, which would be to not return any endpoint in the DNS response. Disabled or Stopped endpoints are not monitored, therefore, they are not considered eligible for traffic.

This condition is commonly caused by improper configuration of the service, such as:

- An access control list [ACL] blocking the Traffic Manager health checks
- An improper configuration of the monitoring path in the Traffic manager profile

The consequence of this behavior is that if Traffic Manager health checks are not configured correctly, it might appear from the traffic routing as though Traffic Manager *is* working properly. However, in this case, endpoint failover cannot happen which affects overall application availability. It is important to check that the profile shows an Online status, not a Degraded status. An Online status indicates that the Traffic Manager health checks are working as expected.

For more information about troubleshooting failed health checks, see Troubleshooting Degraded status on Azure Traffic Manager.

FAQ

Is Traffic Manager resilient to Azure region failures?

Traffic Manager is a key component of the delivery of highly available applications in Azure. To deliver high availability, Traffic Manager must have an exceptionally high level of availability and be resilient to regional failure.

By design, Traffic Manager components are resilient to a complete failure of any Azure region. This resilience applies to all Traffic Manager components: the DNS name servers, the API, the storage layer, and the endpoint monitoring service.

In the unlikely event of an outage of an entire Azure region, Traffic Manager is expected to continue to function normally. Applications deployed in multiple Azure regions can rely on Traffic Manager to direct traffic to an available instance of their application.

How does the choice of resource group location affect Traffic Manager?

Traffic Manager is a single, global service. It is not regional. The choice of resource group location makes no difference to Traffic Manager profiles deployed in that resource group.

Azure Resource Manager requires all resource groups to specify a location, which determines the default location for resources deployed in that resource group. When you create a Traffic Manager profile, it is created in a resource group. All Traffic Manager profiles use **global** as their location, overriding the resource group default.

How do I determine the current health of each endpoint?

The current monitoring status of each endpoint, in addition to the overall profile, is displayed in the Azure portal. This information also is available via the Traffic Monitor REST API, PowerShell cmdlets, and cross-platform Azure CLI.

Azure does not provide historical information about past endpoint health or the ability to raise alerts about changes to endpoint health.

Can I monitor HTTPS endpoints?

Yes. Traffic Manager supports probing over HTTPS. Configure **HTTPS** as the protocol in the monitoring configuration.

Traffic manager cannot provide any certificate validation, including:

- Server-side certificates are not validated
- SNI server-side certificates are not supported
- Client certificates are not supported

What host header do endpoint health checks use?

Traffic Manager uses host headers in HTTP and HTTPS health checks. The host header used by Traffic Manager is the name of the endpoint target configured in the profile. The value used in the host header cannot be specified separately from the target property.

What are the IP addresses from which the health checks originate?

The following list contains the IP addresses from which Traffic Manager health checks can originate. You may use this list to ensure that incoming connections from these IP addresses are allowed at the endpoints to check its health status.

- 40.68.30.66
- 40.68.31.178
- 137.135.80.149
- 137.135.82.249
- 23.96.236.252
- 65.52.217.19
- 40.87.147.10
- 40.87.151.34
- 13.75.124.254
- 13.75.127.63
- 52.172.155.168
- 52.172.158.37

- 104.215.91.84
- 13.75.153.124
- 13.84.222.37
- 23.101.191.199
- 23.96.213.12
- 137.135.46.163
- 137.135.47.215
- 191.232.208.52
- 191.232.214.62
- 13.75.152.253
- 104.41.187.209
- 104.41.190.203

Next steps

Learn how Traffic Manager works

Learn more about the traffic-routing methods supported by Traffic Manager

Learn how to create a Traffic Manager profile

Troubleshoot Degraded status on a Traffic Manager endpoint

Traffic Manager traffic-routing methods

1/17/2017 • 7 min to read • Edit on GitHub

Azure Traffic Manager supports three traffic-routing methods to determine how to route network traffic to the various service endpoints. Traffic Manager applies the traffic-routing method to each DNS query it receives. The traffic-routing method determines which endpoint returned in the DNS response.

The Azure Resource Manager support for Traffic Manager uses different terminology than the classic deployment model. The following table shows the differences between the Resource Manager and Classic terms:

RESOURCE MANAGER TERM	CLASSIC TERM
Traffic-routing method	Load-balancing method
Priority method	Failover method
Weighted method	Round-robin method
Performance method	Performance method

Based on customer feedback, we changed the terminology to improve clarity and reduce common misunderstandings. There is no difference in functionality.

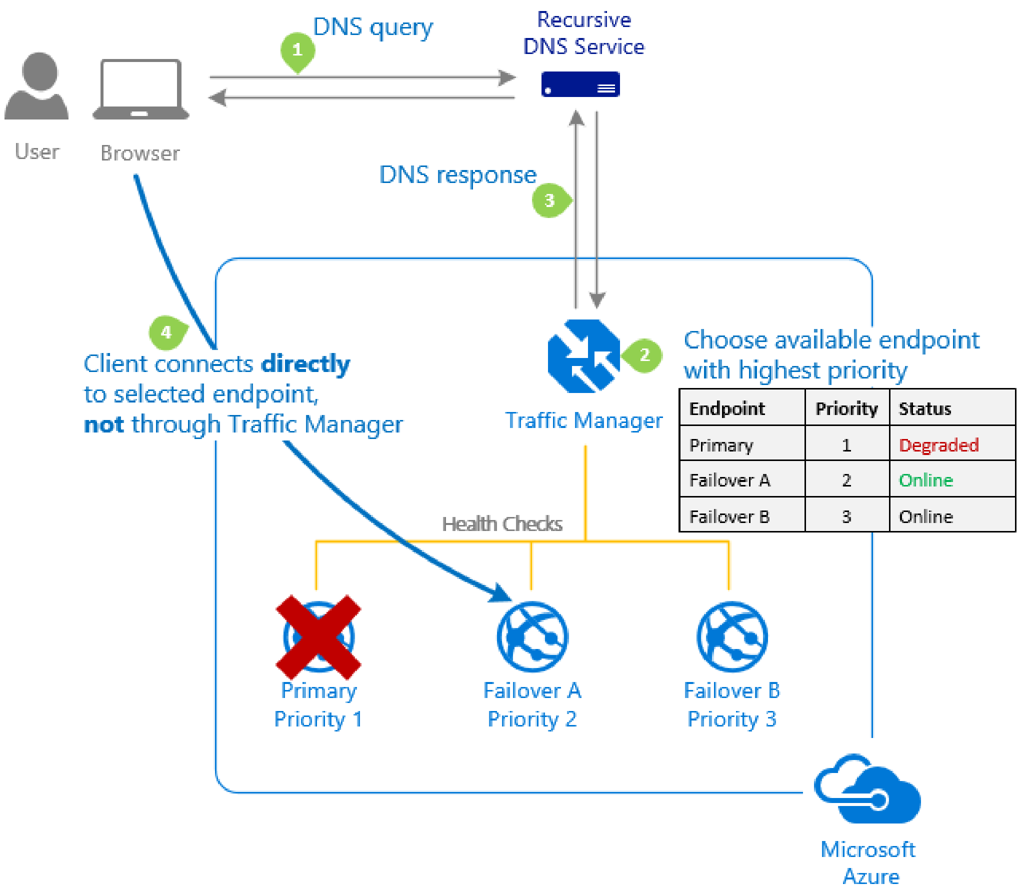
There are three traffic routing methods available in Traffic Manager:

- **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.
- **Weighted:** Select 'Weighted' when you want to distribute traffic across a set of endpoints, either evenly or according to weights, which you define.
- **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.

All Traffic Manager profiles include monitoring of endpoint health and automatic endpoint failover. For more information, see Traffic Manager Endpoint Monitoring. A single Traffic Manager profile can use only one traffic routing method. You can select a different traffic routing method for your profile at any time. Changes are applied within one minute, and no downtime is incurred. Traffic-routing methods can be combined by using nested Traffic Manager profiles. Nesting enables sophisticated and flexible traffic-routing configurations that meet the needs of larger, complex applications. For more information, see nested Traffic Manager profiles.

Priority traffic-routing method

Often an organization wants to provide reliability for its services by deploying one or more backup services in case their primary service goes down. The 'Priority' traffic-routing method allows Azure customers to easily implement this failover pattern.



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

The Traffic Manager profile contains a prioritized list of service endpoints. By default, Traffic Manager sends all traffic to the primary (highest-priority) endpoint. If the primary endpoint is not available, Traffic Manager routes the traffic to the second endpoint. If both the primary and secondary endpoints are not available, the traffic goes to the third, and so on. Availability of the endpoint is based on the configured status (enabled or disabled) and the ongoing endpoint monitoring.

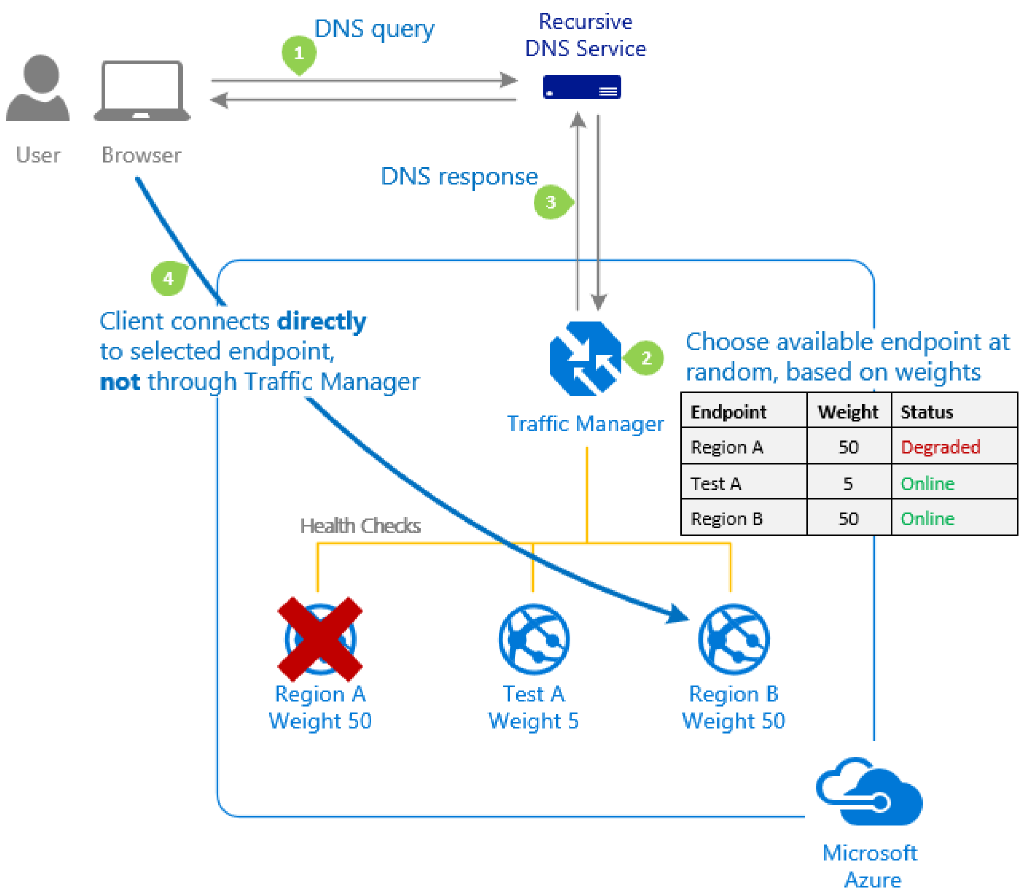
Configuring endpoints

With Azure Resource Manager, you configure the endpoint priority explicitly using the 'priority' property for each endpoint. This property is a value between 1 and 1000. Lower values represent a higher priority. Endpoints cannot share priority values. Setting the property is optional. When omitted, a default priority based on the endpoint order is used.

With the Classic interface, the endpoint priority is configured implicitly. The priority is based on the order in which the endpoints are listed in the profile definition.

Weighted traffic-routing method

The 'Weighted' traffic-routing method allows you to distribute traffic evenly or to use a pre-defined weighting.



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

In the Weighted traffic-routing method, you assign a weight to each endpoint in the Traffic Manager profile configuration. The weight is an integer from 1 to 1000. This parameter is optional. If omitted, Traffic Managers uses a default weight of '1'.

For each DNS query received, Traffic Manager randomly chooses an available endpoint. The probability of choosing an endpoint is based on the weights assigned to all available endpoints. Using the same weight across all endpoints results in an even traffic distribution. Using higher or lower weights on specific endpoints causes those endpoints to be returned more or less frequently in the DNS responses.

The weighted method enables some useful scenarios:

- Gradual application upgrade: Allocate a percentage of traffic to route to a new endpoint, and gradually increase the traffic over time to 100%.
- Application migration to Azure: Create a profile with both Azure and external endpoints. Adjust the weight of the endpoints to prefer the new endpoints.
- Cloud-bursting for additional capacity: Quickly expand an on-premises deployment into the cloud by putting it behind a Traffic Manager profile. When you need extra capacity in the cloud, you can add or enable more endpoints and specify what portion of traffic goes to each endpoint.

The new Azure portal supports the configuration of weighted traffic routing. Weights cannot be configured in the Classic portal. You can also configure weights using the Resource Manager and classic versions of Azure PowerShell, CLI, and the REST APIs.

It is important to understand that DNS responses are cached by clients and by the recursive DNS servers that the clients use to resolve DNS names. This caching can have an impact on weighted traffic distributions. When the number of clients and recursive DNS servers is large, traffic distribution works as expected. However, when the number of clients or recursive DNS servers is small, caching can significantly skew the traffic distribution.

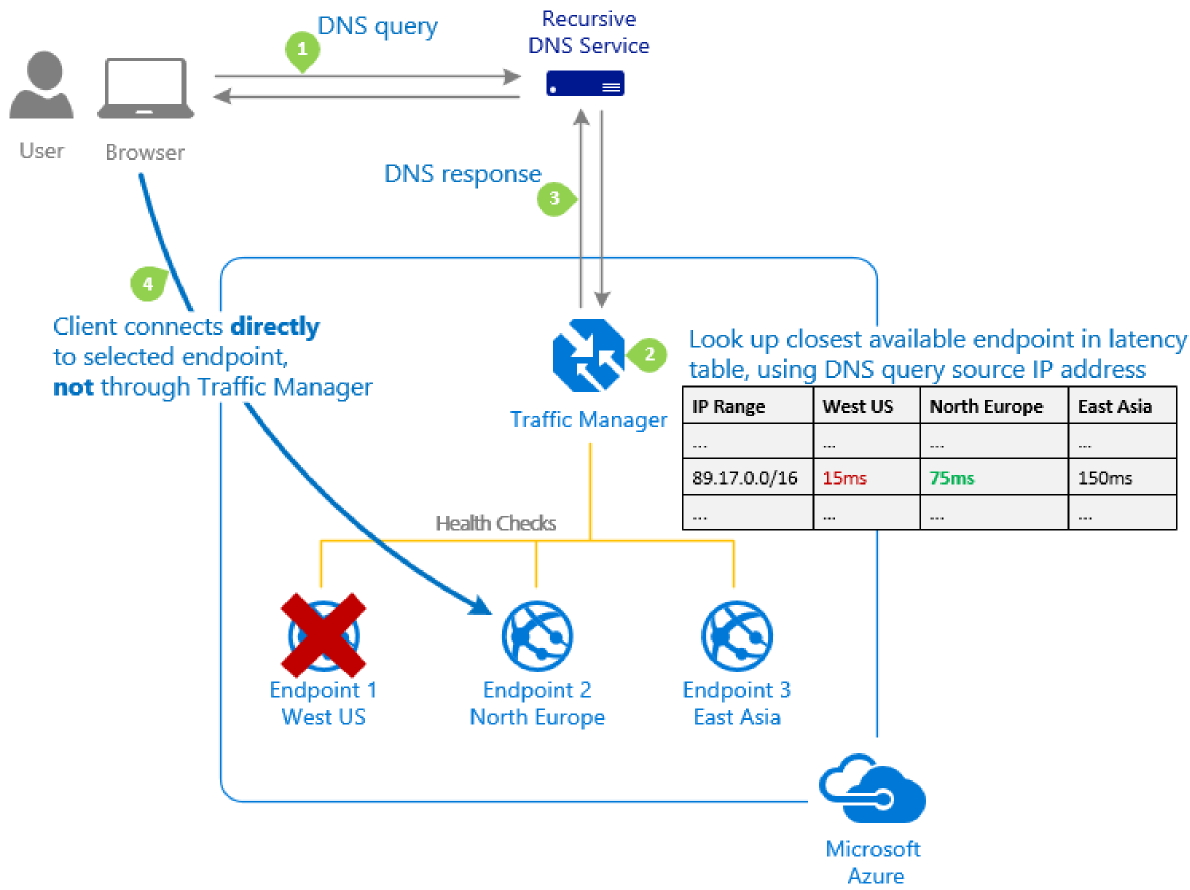
Common use cases include:

- Development and testing environments
- Application-to-application communications
- Applications aimed at a narrow user-base that share a common recursive DNS infrastructure (for example, employees of company connecting through a proxy)

These DNS caching effects are common to all DNS-based traffic routing systems, not just Azure Traffic Manager. In some cases, explicitly clearing the DNS cache may provide a workaround. In other cases, an alternative traffic-routing method may be more appropriate.

Performance traffic-routing method

Deploying endpoints in two or more locations across the globe can improve the responsiveness of many applications by routing traffic to the location that is 'closest' to you. The 'Performance' traffic-routing method provides this capability.



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

The 'closest' endpoint is not necessarily closest as measured by geographic distance. Instead, the 'Performance' traffic-routing method determines the closest endpoint 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.

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.

As explained in How Traffic Manager Works, Traffic Manager does not receive DNS queries directly from clients. Rather, DNS queries come from the recursive DNS service that the clients are configured to use. Therefore, the IP address used to determine the 'closest' endpoint is not the client's IP address, but it is the IP address of the recursive DNS service. In practice, this IP address is a good proxy for the client.

Traffic Manager regularly updates the Internet Latency Table to account for changes in the global Internet and new Azure regions. However, application performance varies based on real-time variations in load across the Internet. Performance traffic-routing does not monitor load on a given service endpoint. However, if an endpoint becomes unavailable, Traffic Manager does not return it in DNS query responses.

Points to note:

- If your profile contains multiple endpoints in the same Azure region, then Traffic Manager distributes traffic evenly across the available endpoints in that region. If you prefer a different traffic distribution within a region, you can use nested Traffic Manager profiles.
- If all enabled endpoints in a given Azure region are degraded, Traffic Manager distributes traffic across all other available endpoints instead of the next-closest endpoint. This logic prevents a cascading failure from occurring by not overloading the next-closest endpoint. If you want to define a preferred failover sequence, use nested Traffic Manager profiles.
- When using the Performance traffic routing method with external endpoints or nested endpoints, you need to specify the location of those endpoints. Choose the Azure region closest to your deployment. Those locations are the values supported by the Internet Latency Table.
- The algorithm that chooses the endpoint is deterministic. Repeated DNS queries from the same client are directed to the same endpoint. Typically, clients use different recursive DNS servers when traveling. The client may be routed to a different endpoint. Routing can also be affected by updates to the Internet Latency Table. Therefore, the Performance traffic-routing method does not guarantee that a client is always routed to the same endpoint.
- When the Internet Latency Table changes, you may notice that some clients are directed to a different endpoint. This routing change is more accurate based on current latency data. These updates are essential to maintain the accuracy of Performance traffic-routing as the Internet continually evolves.

Next steps

Learn how to develop high-availability applications using Traffic Manager endpoint monitoring

Learn how to create a Traffic Manager profile

Nested Traffic Manager profiles

1/17/2017 • 8 min to read • Edit on GitHub

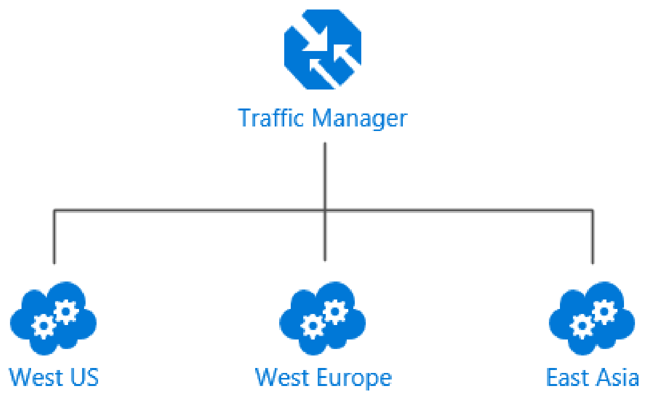
Traffic Manager includes a range of traffic-routing methods that allow you to control how Traffic Manager chooses which endpoint should receive traffic from each end user. For more information, see Traffic Manager traffic-routing methods.

Each Traffic Manager profile specifies a single traffic-routing method. However, there are scenarios that require more sophisticated traffic routing than the routing provided by a single Traffic Manager profile. You can nest Traffic Manager profiles to combine the benefits of more than one traffic-routing method. Nested profiles allow you to override the default Traffic Manager behavior to support larger and more complex application deployments.

The following examples illustrate how to use nested Traffic Manager profiles in various scenarios.

Example 1: Combining 'Performance' and 'Weighted' traffic routing

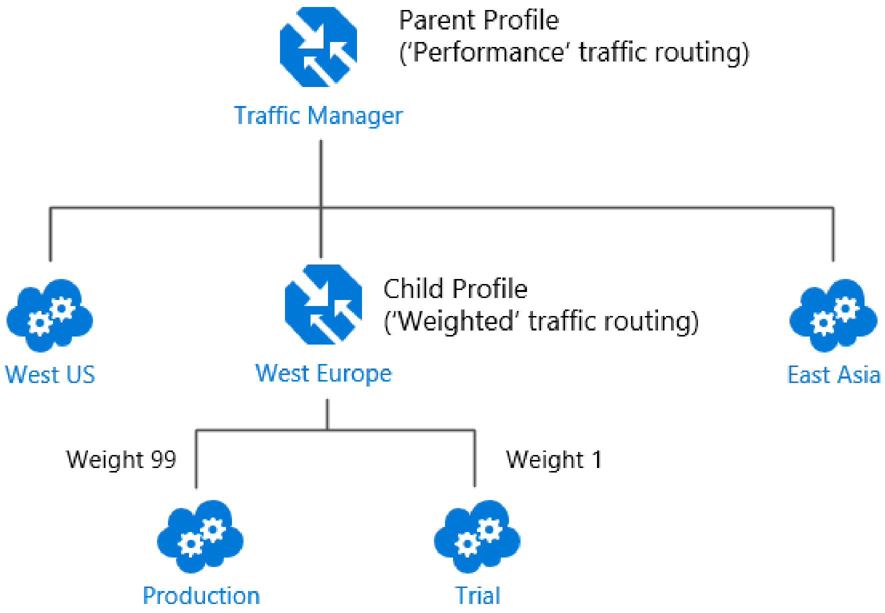
Suppose that you deployed an application in the following Azure regions: West US, West Europe, and East Asia. You use Traffic Manager's 'Performance' traffic-routing method to distribute traffic to the region closest to the user.



Now, suppose you wish to test an update to your service before rolling it out more widely. You want to use the 'weighted' traffic-routing method to direct a small percentage of traffic to your test deployment. You set up the test deployment alongside the existing production deployment in West Europe.

You cannot combine both 'Weighted' and 'Performance traffic-routing in a single profile. To support this scenario, you create a Traffic Manager profile using the two West Europe endpoints and the 'Weighted' traffic-routing method. Next, you add this 'child' profile as an endpoint to the 'parent' profile. The parent profile still uses the Performance traffic-routing method and contains the other global deployments as endpoints.

The following diagram illustrates this example:



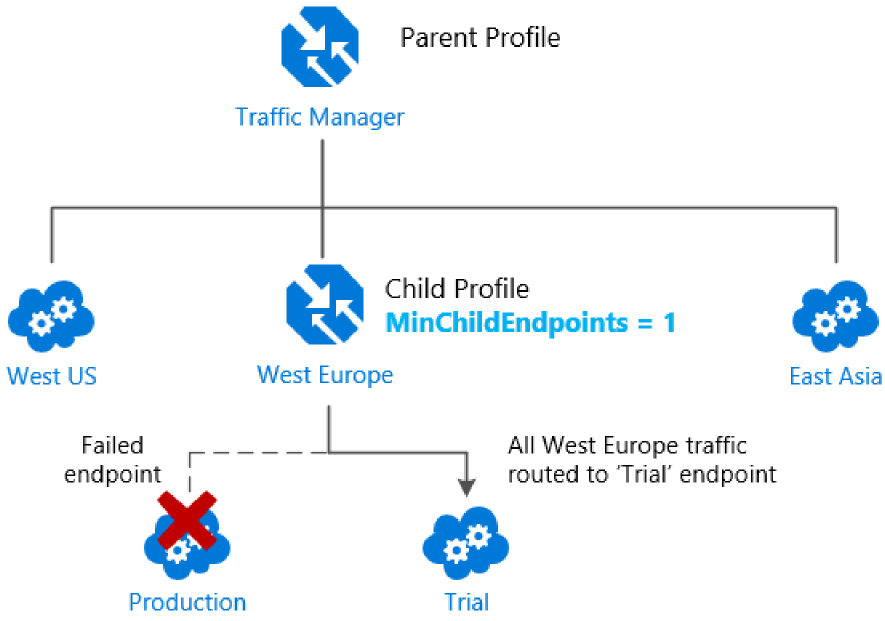
In this configuration, traffic directed via the parent profile distributes traffic across regions normally. Within West Europe, the nested profile distributes traffic to the production and test endpoints according to the weights assigned.

When the parent profile uses the 'Performance' traffic-routing method, each endpoint must be assigned a location. The location is assigned when you configure the endpoint. Choose the Azure region closest to your deployment. The Azure regions are the location values supported by the Internet Latency Table. For more information, see Traffic Manager 'Performance' traffic-routing method.

Example 2: Endpoint monitoring in Nested Profiles

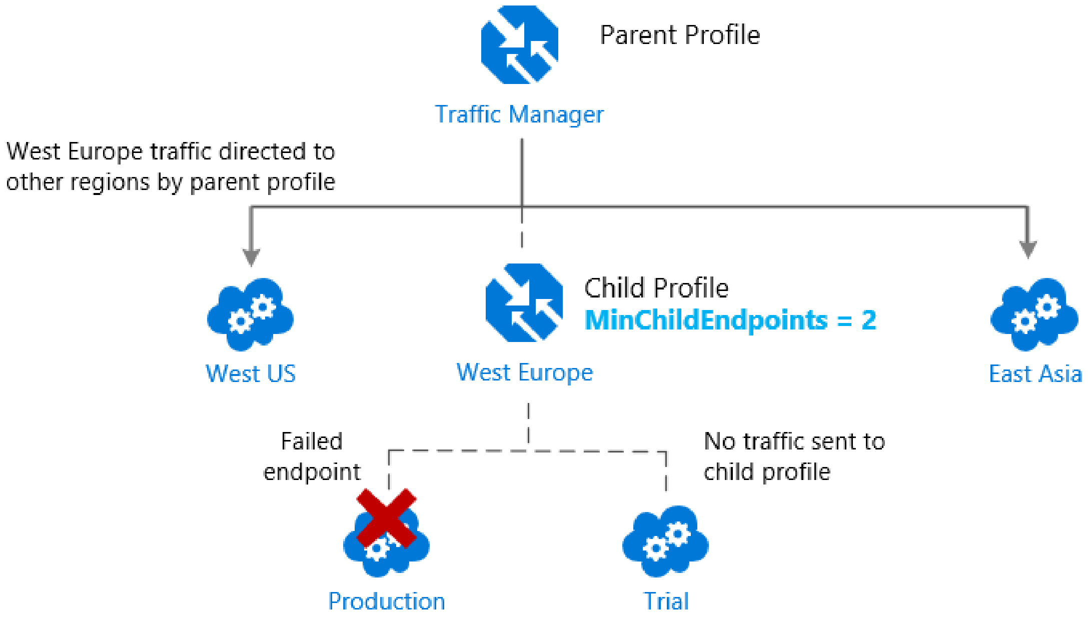
Traffic Manager actively monitors the health of each service endpoint. If an endpoint is unhealthy, Traffic Manager directs users to alternative endpoints to preserve the availability of your service. This endpoint monitoring and failover behavior applies to all traffic-routing methods. For more information, see Traffic Manager Endpoint Monitoring. Endpoint monitoring works differently for nested profiles. With nested profiles, the parent profile doesn't perform health checks on the child directly. Instead, the health of the child profile's endpoints is used to calculate the overall health of the child profile. This health information is propagated up the nested profile hierarchy. The parent profile uses this aggregated health to determine whether to direct traffic to the child profile. See the FAQ section of this article for full details on health monitoring of nested profiles.

Returning to the previous example, suppose the production deployment in West Europe fails. By default, the 'child' profile directs all traffic to the test deployment. If the test deployment also fails, the parent profile determines that the child profile should not receive traffic since all child endpoints are unhealthy. Then, the parent profile distributes traffic to the other regions.



You might be happy with this arrangement. Or you might be concerned that all traffic for West Europe is now going to the test deployment instead of a limited subset traffic. Regardless of the health of the test deployment, you want to fail over to the other regions when the production deployment in West Europe fails. To enable this failover, you can specify the 'MinChildEndpoints' parameter when configuring the child profile as an endpoint in the parent profile. The parameter determines the minimum number of available endpoints in the child profile. The default value is '1'. For this scenario, you set the MinChildEndpoints value to 2. Below this threshold, the parent profile considers the entire child profile to be unavailable and directs traffic to the other endpoints.

The following figure illustrates this configuration:

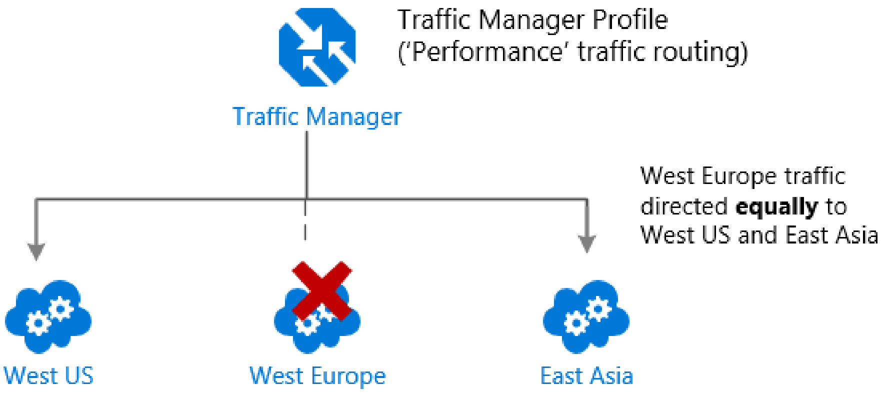


NOTE

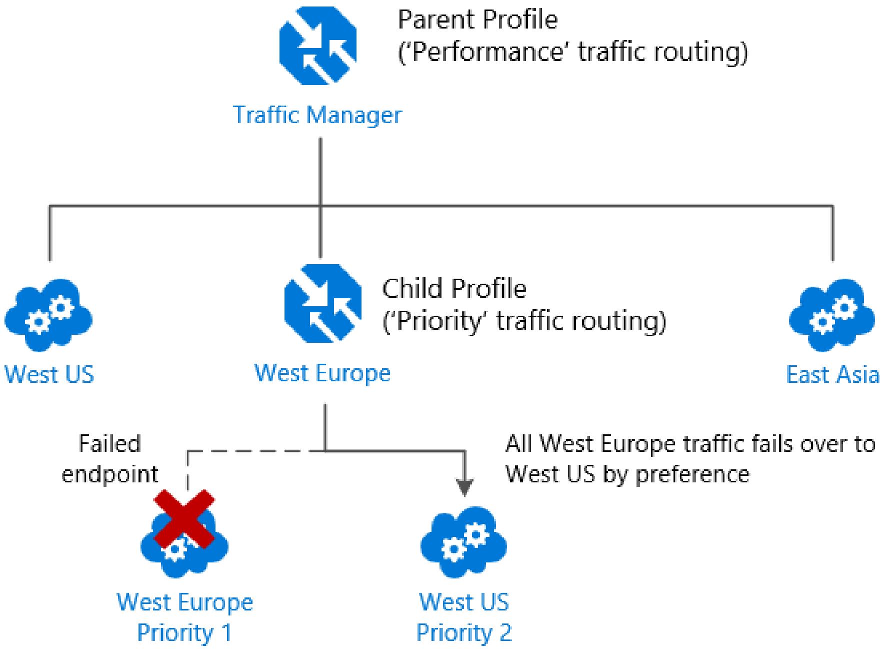
The 'Priority' traffic-routing method distributes all traffic to a single endpoint. Thus there is little purpose in a MinChildEndpoints setting other than '1' for a child profile.

Example 3: Prioritized failover regions in 'Performance' traffic routing

The default behavior for the 'Performance' traffic-routing method is designed to avoid over-loading the next nearest endpoint and causing a cascading series of failures. When an endpoint fails, all traffic that would have been directed to that endpoint is evenly distributed to the other endpoints across all regions.



However, suppose you prefer the West Europe traffic failover to West US, and only direct traffic to other regions when both endpoints are unavailable. You can create this solution using a child profile with the 'Priority' traffic-routing method.

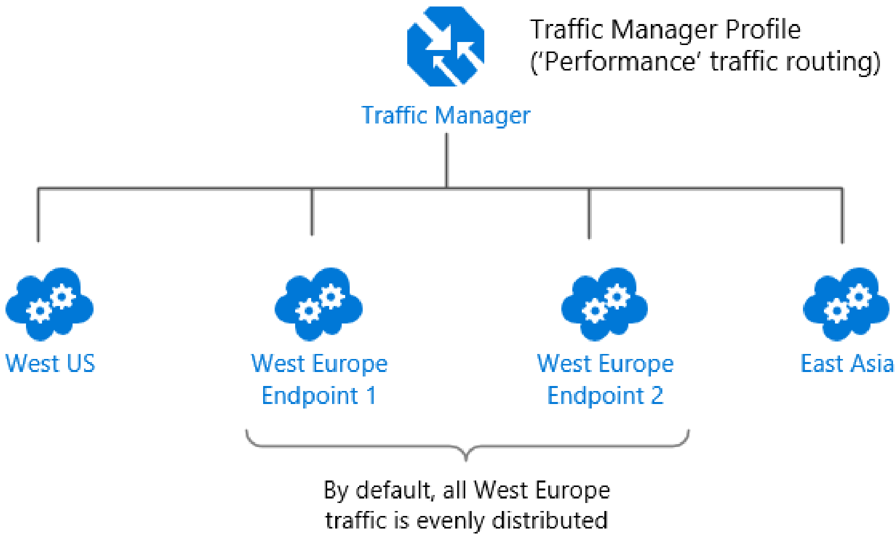


Since the West Europe endpoint has higher priority than the West US endpoint, all traffic is sent to the West Europe endpoint when both endpoints are online. If West Europe fails, its traffic is directed to West US. With the nested profile, traffic is directed to East Asia only when both West Europe and West US fail.

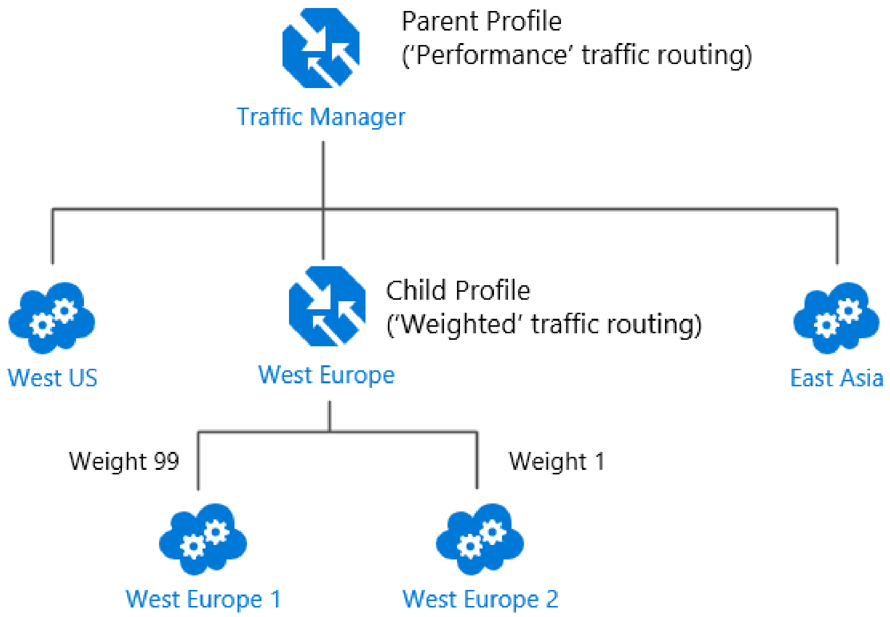
You can repeat this pattern for all regions. Replace all three endpoints in the parent profile with three child profiles, each providing a prioritized failover sequence.

Example 4: Controlling 'Performance' traffic routing between multiple endpoints in the same region

Suppose the 'Performance' traffic-routing method is used in a profile that has more than one endpoint in a particular region. By default, traffic directed to that region is distributed evenly across all available endpoints in that region.

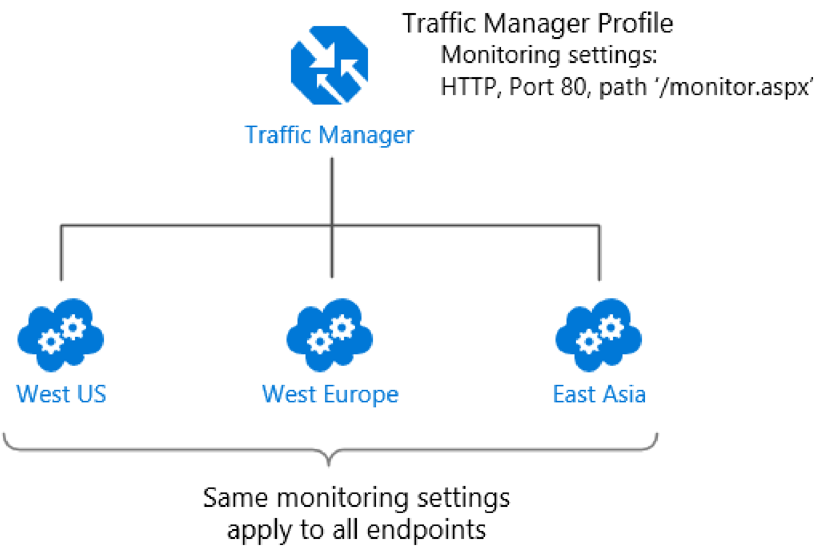


Instead of adding multiple endpoints in West Europe, those endpoints are enclosed in a separate child profile. The child profile is added to the parent as the only endpoint in West Europe. The settings on the child profile can control the traffic distribution with West Europe by enabling priority-based or weighted traffic routing within that region.

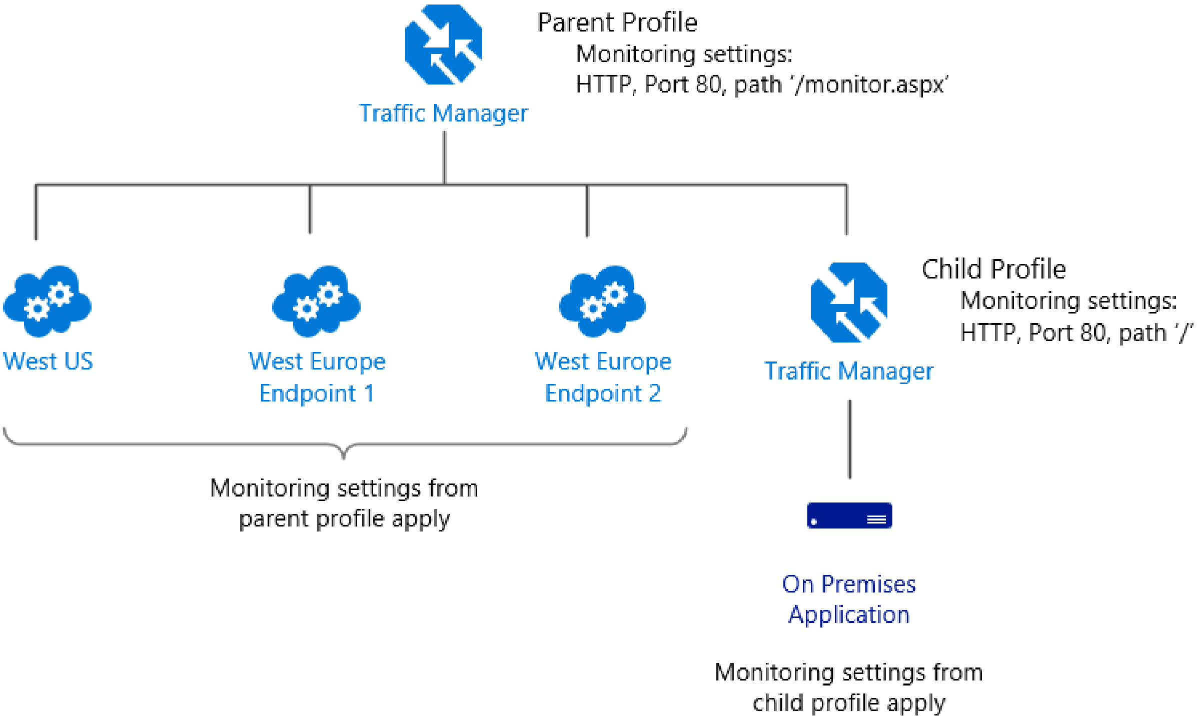


Example 5: Per-endpoint monitoring settings

Suppose you are using Traffic Manager to smoothly migrate traffic from a legacy on-premises web site to a new Cloud-based version hosted in Azure. For the legacy site, you want to use the home page URI to monitor site health. But for the new Cloud-based version, you are implementing a custom monitoring page (path '/monitor.aspx') that includes additional checks.



The monitoring settings in a Traffic Manager profile apply to all endpoints within a single profile. With nested profiles, you use a different child profile per site to define different monitoring settings.



FAQ

How do I configure nested profiles?

Nested Traffic Manager profiles can be configured using both the Azure Resource Manager and the classic Azure REST APIs, Azure PowerShell cmdlets and cross-platform Azure CLI commands. They are also supported via the new Azure portal. They are not supported in the classic portal.

How many layers of nesting does Traffic Manger support?

You can nest profiles up to 10 levels deep. 'Loops' are not permitted.

Can I mix other endpoint types with nested child profiles, in the same Traffic Manager profile?

Yes. There are no restrictions on how you combine endpoints of different types within a profile.

How does the billing model apply for Nested profiles?

There is no negative pricing impact of using nested profiles.

Traffic Manager billing has two components: endpoint health checks and millions of DNS queries

- Endpoint health checks: There is no charge for a child profile when configured as an endpoint in a parent profile. Monitoring of the endpoints in the child profile are billed in the usual way.
- DNS queries: Each query is only counted once. A query against a parent profile that returns an endpoint from a child profile is counted against the parent profile only.

For full details, see the Traffic Manager pricing page.

Is there a performance impact for nested profiles?

No. There is no performance impact incurred when using nested profiles.

The Traffic Manager name servers traverse the profile hierarchy internally when processing each DNS query. A DNS query to a parent profile can receive a DNS response with an endpoint from a child profile. A single CNAME record is used whether you are using a single profile or nested profiles. There is no need to create a CNAME record for each profile in the hierarchy.

How does Traffic Manager compute the health of a nested endpoint in a parent profile?

The parent profile doesn't perform health checks on the child directly. Instead, the health of the child profile's endpoints are used to calculate the overall health of the child profile. This information is propagated up the nested profile hierarchy to determine the health of the nested endpoint. The parent profile uses this aggregated health to determine whether the traffic can be directed to the child.

The following table describes the behavior of Traffic Manager health checks for a nested endpoint.

CHILD PROFILE MONITOR STATUS	PARENT ENDPOINT MONITOR STATUS	NOTES
Disabled. The child profile has been disabled.	Stopped	The parent endpoint state is Stopped, not Disabled. The Disabled state is reserved for indicating that you have disabled the endpoint in the parent profile.
Degraded. At least one child profile endpoint is in a Degraded state.	Online: the number of Online endpoints in the child profile is at least the value of MinChildEndpoints. CheckingEndpoint: the number of Online plus CheckingEndpoint endpoints in the child profile is at least the value of MinChildEndpoints. Degraded: otherwise.	Traffic is routed to an endpoint of status CheckingEndpoint. If MinChildEndpoints is set too high, the endpoint is always degraded.
Online. At least one child profile endpoint is an Online state. No endpoint is in the Degraded state.	See above.	
CheckingEndpoints. At least one child profile endpoint is 'CheckingEndpoint'. No endpoints are 'Online' or 'Degraded'	Same as above.	
Inactive. All child profile endpoints are either Disabled or Stopped, or this profile has no endpoints.	Stopped	

Next steps

Learn more about how Traffic Manager works

Learn how to create a Traffic Manager profile

Performance considerations for Traffic Manager

1/17/2017 • 3 min to read • Edit on GitHub

This page explains performance considerations using Traffic Manager. Consider the following scenario:

You have instances of your website in the WestUS and EastAsia regions. One of the instances is failing the health check for the traffic manager probe. Application traffic is directed to the healthy region. This failover is expected but performance can be a problem based on the latency of the traffic now traveling to a distant region.

How Traffic Manager works

The only performance impact that Traffic Manager can have on your website is the initial DNS lookup. A DNS request for the name of your Traffic Manager profile is handled by the Microsoft DNS root server that hosts the trafficmanager.net zone. Traffic Manager populates, and regularly updates, the Microsoft's DNS root servers based on the Traffic Manager policy and the probe results. So even during the initial DNS lookup, no DNS queries are sent to Traffic Manager.

Traffic Manager is made up of several components: DNS name servers, an API service, the storage layer, and an endpoint monitoring service. If a Traffic Manager service component fails, there is no effect on the DNS name associated with your Traffic Manager profile. The records in the Microsoft DNS servers remain unchanged. However, endpoint monitoring and DNS updating do not happen. Therefore, Traffic Manager is not able to update DNS to point to your failover site when your primary site goes down.

DNS name resolution is fast and results are cached. The speed of the initial DNS lookup depends on the DNS servers the client uses for name resolution. Typically, a client can complete a DNS lookup within ~50 ms. The results of the lookup are cached for the duration of the DNS Time-to-live (TTL). The default TTL for Traffic Manager is 300 seconds.

Traffic does NOT flow through Traffic Manager. Once the DNS lookup completes, the client has an IP address for an instance of your web site. The client connects directly to that address and does not pass through Traffic Manager. The Traffic Manager policy you choose has no influence on the DNS performance. However, a Performance routing-method can negatively impact the application experience. For example, if your policy redirects traffic from North America to an instance hosted in Asia, the network latency for those sessions may be a performance issue.

Measuring Traffic Manager Performance

There are several websites you can use to understand the performance and behavior of a Traffic Manager profile. Many of these sites are free but may have limitations. Some sites offer enhanced monitoring and reporting for a fee.

The tools on these sites measure DNS latencies and display the resolved IP addresses for client locations around the world. Most of these tools do not cache the DNS results. Therefore, the tools show the full DNS lookup each time a test is run. When you test from your own client, you only experience the full DNS lookup performance once during the TTL duration.

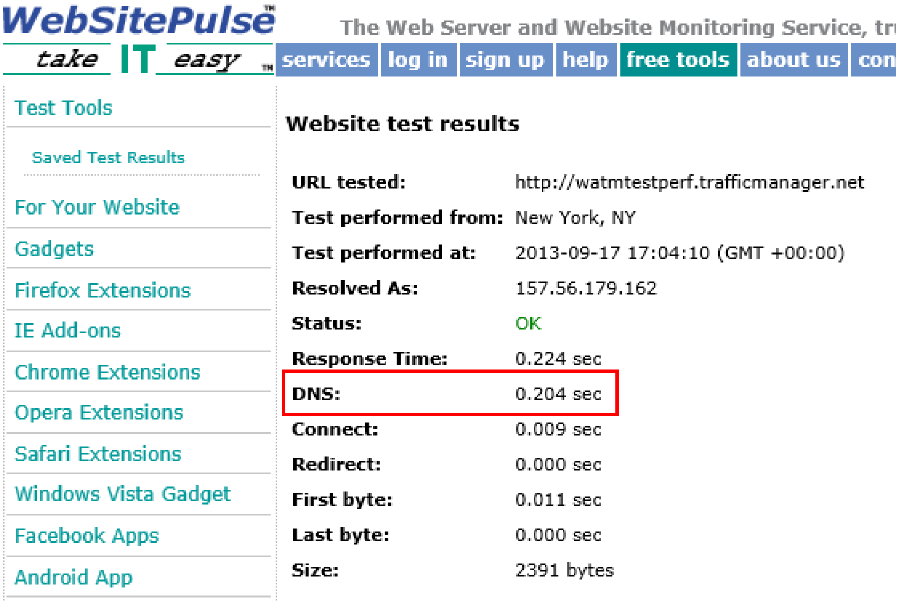
Sample tools to measure DNS performance

- SolveDNS

  SolveDNS offers many performance tools. The DNS Comparison tool can show you how long it takes to resolve your DNS name and how that compares to other DNS service providers.

- WebSitePulse

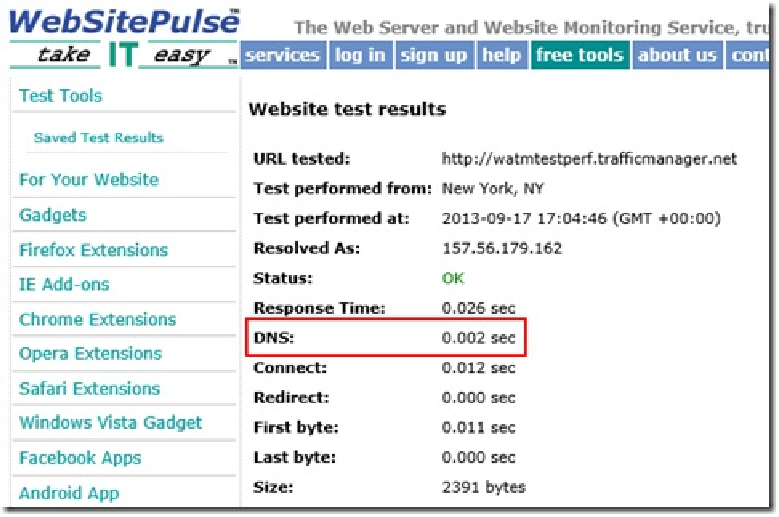
One of the simplest tools is WebSitePulse. Enter the URL to see DNS resolution time, First Byte, Last Byte, and other performance statistics. You can choose from three different test locations. In this example, you see that the first execution shows that DNS lookup takes 0.204 sec.



Website test results

URL tested: http://watmtestperf.trafficmanager.net
Test performed from: New York, NY
Test performed at: 2013-09-17 17:04:10 (GMT +00:00)
Resolved As: 157.56.179.162
Status: OK
Response Time: 0.224 sec
DNS: 0.204 sec
Connect: 0.009 sec
Redirect: 0.000 sec
First byte: 0.011 sec
Last byte: 0.000 sec
Size: 2391 bytes

Because the results are cached, the second test for the same Traffic Manager endpoint the DNS lookup takes 0.002 sec.

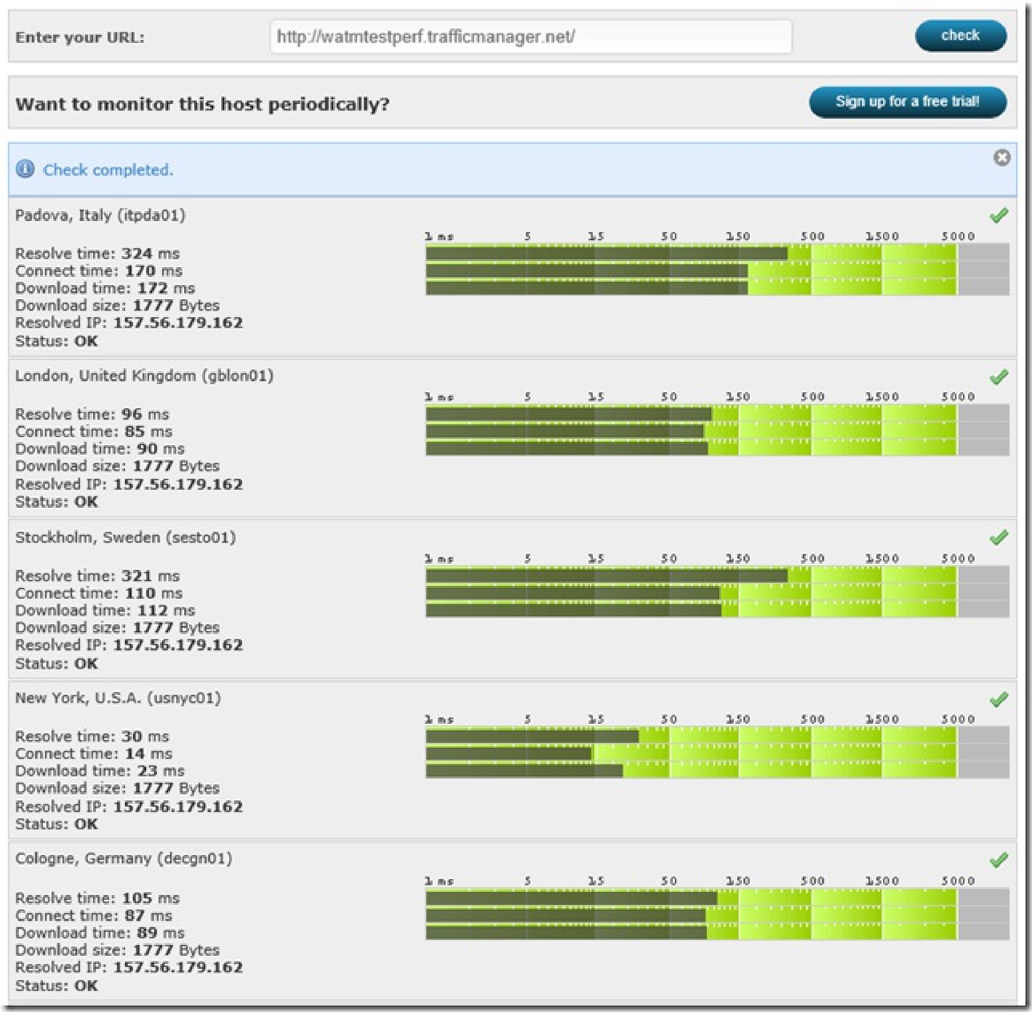


Website test results

URL tested: http://watmtestperf.trafficmanager.net
Test performed from: New York, NY
Test performed at: 2013-09-17 17:04:46 (GMT +00:00)
Resolved As: 157.56.179.162
Status: OK
Response Time: 0.026 sec
DNS: 0.002 sec
Connect: 0.012 sec
Redirect: 0.000 sec
First byte: 0.011 sec
Last byte: 0.000 sec
Size: 2391 bytes

- CA App Synthetic Monitor

Formerly known as the Watchmouse Check Website tool, this site show you the DNS resolution time from multiple geographic regions simultaneously. Enter the URL to see DNS resolution time, connection time, and speed from several geographic locations. Use this test to see which hosted service is returned for different locations around the world.



Enter your URL: http://watmtestperf.trafficmanager.net/ check

Want to monitor this host periodically? Sign up for a free trial!

Check completed.

Padova, Italy (itpda01)

Resolve time: **324** ms
Connect time: **170** ms
Download time: **172** ms
Download size: **1777** Bytes
Resolved IP: **157.56.179.162**
Status: **OK**

London, United Kingdom (gblon01)

Resolve time: **96** ms
Connect time: **85** ms
Download time: **90** ms
Download size: **1777** Bytes
Resolved IP: **157.56.179.162**
Status: **OK**

Stockholm, Sweden (sesto01)

Resolve time: **321** ms
Connect time: **110** ms
Download time: **112** ms
Download size: **1777** Bytes
Resolved IP: **157.56.179.162**
Status: **OK**

New York, U.S.A. (usnyc01)

Resolve time: **30** ms
Connect time: **14** ms
Download time: **23** ms
Download size: **1777** Bytes
Resolved IP: **157.56.179.162**
Status: **OK**

Cologne, Germany (decgn01)

Resolve time: **105** ms
Connect time: **87** ms
Download time: **89** ms
Download size: **1777** Bytes
Resolved IP: **157.56.179.162**
Status: **OK**

- Pingdom

  This tool provides performance statistics for each element of a web page. The Page Analysis tab shows the percentage of time spent on DNS lookup.

- What's My DNS?

  This site does a DNS lookup from 20 different locations and displays the results on a map.

- Dig Web Interface

  This site shows more detailed DNS information including CNAMEs and A records. Make sure you check the 'Colorize output' and 'Stats' under options, and select 'All' under Nameservers.

Next Steps

About Traffic Manager traffic routing methods

Test your Traffic Manager settings

Operations on Traffic Manager (REST API Reference)

Azure Traffic Manager Cmdlets

Azure Resource Manager support for Azure Traffic Manager

1/17/2017 • 11 min to read • Edit on GitHub

Azure Resource Manager is the preferred management interface for services in Azure. Azure Traffic Manager profiles can be managed using Azure Resource Manager-based APIs and tools.

Resource model

Azure Traffic Manager is configured using a collection of settings called a Traffic Manager profile. This profile contains DNS settings, traffic routing settings, endpoint monitoring settings, and a list of service endpoints to which traffic is routed.

Each Traffic Manager profile is represented by a resource of type 'TrafficManagerProfiles'. At the REST API level, the URI for each profile is as follows:

```
https://management.azure.com/subscriptions/{subscription-id}/resourceGroups/{resource-group-
name}/providers/Microsoft.Network/trafficManagerProfiles/{profile-name}?api-version={api-version}
```

Comparison with the Azure Traffic Manager classic API

The Azure Resource Manager support for Traffic Manager uses different terminology than the classic deployment model. The following table shows the differences between the Resource Manager and Classic terms:

RESOURCE MANAGER TERM	CLASSIC TERM
Traffic-routing method	Load-balancing method
Priority method	Failover method
Weighted method	Round-robin method
Performance method	Performance method

Based on customer feedback, we changed the terminology to improve clarity and reduce common misunderstandings. There is no difference in functionality.

Limitations

When referencing an endpoint of type 'AzureEndpoints' for a Web App, Traffic Manager endpoints can only reference the default (production) Web App slot. Custom slots are not supported. As a workaround, custom slots can be configured using the 'ExternalEndpoints' type.

Setting up Azure PowerShell

These instructions use Microsoft Azure PowerShell. The following article explains how to install and configure Azure PowerShell.

- How to install and configure Azure PowerShell

The examples in this article assume that you have an existing resource group. You can create a resource group using the following command:

```
New-AzureRmResourceGroup -Name MyRG -Location "West US"
```

> **NOTE**
>
> Azure Resource Manager requires that all resource groups have a location. This location is used as the default for resources created in that resource group. However, since Traffic Manager profile resources are global, not regional, the choice of resource group location has no impact on Azure Traffic Manager.

Create a Traffic Manager Profile

To create a Traffic Manager profile, use the `New-AzureRmTrafficManagerProfile` cmdlet:

```
$profile = New-AzureRmTrafficManagerProfile -Name MyProfile -ResourceGroupName MyRG -TrafficRoutingMethod Performance -
RelativeDnsName contoso -Ttl 30 -MonitorProtocol HTTP -MonitorPort 80 -MonitorPath "/"
```

The following table describes the parameters:

PARAMETER	DESCRIPTION
Name	The resource name for the Traffic Manager profile resource. Profiles in the same resource group must have unique names. This name is separate from the DNS name used for DNS queries.
ResourceGroupName	The name of the resource group containing the profile resource.
TrafficRoutingMethod	Specifies the traffic-routing method used to determine which endpoint is returned in response a DNS query. Possible values are 'Performance', 'Weighted' or 'Priority'.
RelativeDnsName	Specifies the hostname portion of the DNS name provided by this Traffic Manager profile. This value is combined with the DNS domain name used by Azure Traffic Manager to form the fully qualified domain name (FQDN) of the profile. For example, setting the value of 'contoso' becomes 'contoso.trafficmanager.net.'
TTL	Specifies the DNS Time-to-Live (TTL), in seconds. This TTL informs the Local DNS resolvers and DNS clients how long to cache DNS responses for this Traffic Manager profile.
MonitorProtocol	Specifies the protocol to use to monitor endpoint health. Possible values are 'HTTP' and 'HTTPS'.
MonitorPort	Specifies the TCP port used to monitor endpoint health.
MonitorPath	Specifies the path relative to the endpoint domain name used to probe for endpoint health.

The cmdlet creates a Traffic Manager profile in Azure and returns a corresponding profile object to PowerShell. At this point, the profile does not contain any endpoints. For more information about adding endpoints to a Traffic

Manager profile, see Adding Traffic Manager Endpoints.

Get a Traffic Manager Profile

To retrieve an existing Traffic Manager profile object, use the `Get-AzureRmTrafficManagerProfle` cmdlet:

```
$profile = Get-AzureRmTrafficManagerProfile -Name MyProfile -ResourceGroupName MyRG
```

This cmdlet returns a Traffic Manager profile object.

Update a Traffic Manager Profile

Modifying Traffic Manager profiles follows a 3-step process:

1. Retrieve the profile using `Get-AzureRmTrafficManagerProfile` or use the profile returned by `New-AzureRmTrafficManagerProfile`.
2. Modify the profile. You can add and remove endpoints or change endpoint or profile parameters. These changes are off-line operations. You are only changing the local object in memory that represents the profile.
3. Commit your changes using the `Set-AzureRmTrafficManagerProfile` cmdlet.

All profile properties can be changed except the profile's RelativeDnsName. To change the RelativeDnsName, you must delete profile and a new profile with a new name.

The following example demonstrates how to change the profile's TTL:

```
$profile = Get-AzureRmTrafficManagerProfile -Name MyProfile -ResourceGroupName MyRG
$profile.Ttl = 300
Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile
```

There are three types of Traffic Manager endpoints:

1. **Azure endpoints** are services hosted in Azure
2. **External endpoints** are services hosted outside of Azure
3. **Nested endpoints** are used to construct nested hierarchies of Traffic Manager profiles. Nested endpoints enable advanced traffic-routing configurations for complex applications.

In all three cases, endpoints can be added in two ways:

1. Using a 3-step process described previously. The advantage of this method is that several endpoint changes can be made in a single update.
2. Using the New-AzureRmTrafficManagerEndpoint cmdlet. This cmdlet adds an endpoint to an existing Traffic Manager profile in a single operation.

Adding Azure Endpoints

Azure endpoints reference services hosted in Azure. Three types of Azure endpoints are supported:

1. Azure Web Apps
2. 'Classic' cloud services (which can contain either a PaaS service or IaaS virtual machines)
3. Azure PublicIpAddress resources (which can be attached to a load-balancer or a virtual machine NIC). The PublicIpAddress must have a DNS name assigned to be used in Traffic Manager.

In each case:

- The service is specified using the 'targetResourceId' parameter of `Add-AzureRmTrafficManagerEndpointConfig` or `New-AzureRmTrafficManagerEndpoint`.

- The 'Target' and 'EndpointLocation' are implied by the TargetResourceId.
- Specifying the 'Weight' is optional. Weights are only used if the profile is configured to use the 'Weighted' traffic-routing method. Otherwise, they are ignored. If specified, the value must be a number between 1 and 1000. The default value is '1'.
- Specifying the 'Priority' is optional. Priorities are only used if the profile is configured to use the 'Priority' traffic-routing method. Otherwise, they are ignored. Valid values are from 1 to 1000 with lower values indicating a higher priority. If specified for one endpoint, they must be specified for all endpoints. If omitted, default values starting from '1' are applied in the order that the endpoints are listed.

Example 1: Adding Web App endpoints using `Add-AzureRmTrafficManagerEndpointConfig`

In this example, we create a Traffic Manager profile and add two Web App endpoints using the `Add-AzureRmTrafficManagerEndpointConfig` cmdlet.

```
$profile = New-AzureRmTrafficManagerProfile -Name myprofile -ResourceGroupName MyRG -TrafficRoutingMethod Performance -
RelativeDnsName myapp -Ttl 30 -MonitorProtocol HTTP -MonitorPort 80 -MonitorPath "/"
$webapp1 = Get-AzureRMWebApp -Name webapp1
Add-AzureRmTrafficManagerEndpointConfig -EndpointName webapp1ep -TrafficManagerProfile $profile -Type AzureEndpoints -TargetResourceId
$webapp1.Id -EndpointStatus Enabled
$webapp2 = Get-AzureRMWebApp -Name webapp2
Add-AzureRmTrafficManagerEndpointConfig -EndpointName webapp2ep -TrafficManagerProfile $profile -Type AzureEndpoints -TargetResourceId
$webapp2.Id -EndpointStatus Enabled
Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile
```

Example 2: Adding a 'classic' cloud service endpoint using `New-AzureRmTrafficManagerEndpoint`

In this example, a 'classic' Cloud Service endpoint is added to a Traffic Manager profile. In this example, we specified the profile using the profile and resource group names, rather than passing a profile object. Both approaches are supported.

```
$cloudService = Get-AzureRmResource -ResourceName MyCloudService -ResourceType "Microsoft.ClassicCompute/domainNames" -
ResourceGroupName MyCloudService
New-AzureRmTrafficManagerEndpoint -Name MyCloudServiceEndpoint -ProfileName MyProfile -ResourceGroupName MyRG -Type
AzureEndpoints -TargetResourceId $cloudService.Id -EndpointStatus Enabled
```

Example 3: Adding a publicIpAddress endpoint using `New-AzureRmTrafficManagerEndpoint`

In this example, a public IP address resource is added to the Traffic Manager profile. The public IP address must have a DNS name configured, and can be bound either to the NIC of a VM or to a load balancer.

```
$ip = Get-AzureRmPublicIpAddress -Name MyPublicIP -ResourceGroupName MyRG
New-AzureRmTrafficManagerEndpoint -Name MyIpEndpoint -ProfileName MyProfile -ResourceGroupName MyRG -Type AzureEndpoints -
TargetResourceId $ip.Id -EndpointStatus Enabled
```

Adding External Endpoints

Traffic Manager uses external endpoints to direct traffic to services hosted outside of Azure. As with Azure endpoints, external endpoints can be added either using `Add-AzureRmTrafficManagerEndpointConfig` followed by `Set-AzureRmTrafficManagerProfile`, or `New-AzureRMTrafficManagerEndpoint`.

When specifying external endpoints:

- The endpoint domain name must be specified using the 'Target' parameter
- If the 'Performance' traffic-routing method is used, the 'EndpointLocation' is required. Otherwise it is optional. The value must be a valid Azure region name.
- The 'Weight' and 'Priority' are optional.

Example 1: Adding external endpoints using `Add-AzureRmTrafficManagerEndpointConfig` and `Set-AzureRmTrafficManagerProfile`

In this example, we create a Traffic Manager profile, add two external endpoints, and commit the changes.

```
$profile = New-AzureRmTrafficManagerProfile -Name myprofile -ResourceGroupName MyRG -TrafficRoutingMethod Performance -RelativeDnsName myapp -Ttl 30 -MonitorProtocol HTTP -MonitorPort 80 -MonitorPath "/"
Add-AzureRmTrafficManagerEndpointConfig -EndpointName eu-endpoint -TrafficManagerProfile $profile -Type ExternalEndpoints -Target app-eu.contoso.com -EndpointStatus Enabled
Add-AzureRmTrafficManagerEndpointConfig -EndpointName us-endpoint -TrafficManagerProfile $profile -Type ExternalEndpoints -Target app-us.contoso.com -EndpointStatus Enabled
Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile
```

Example 2: Adding external endpoints using `New-AzureRmTrafficManagerEndpoint`

In this example, we add an external endpoint to an existing profile. The profile is specified using the profile and resource group names.

```
New-AzureRmTrafficManagerEndpoint -Name eu-endpoint -ProfileName MyProfile -ResourceGroupName MyRG -Type ExternalEndpoints -Target app-eu.contoso.com -EndpointStatus Enabled
```

Adding 'Nested' endpoints

Each Traffic Manager profile specifies a single traffic-routing method. However, there are scenarios that require more sophisticated traffic routing than the routing provided by a single Traffic Manager profile. You can nest Traffic Manager profiles to combine the benefits of more than one traffic-routing method. Nested profiles allow you to override the default Traffic Manager behavior to support larger and more complex application deployments. For more detailed examples, see Nested Traffic Manager profiles.

Nested endpoints are configured at the parent profile, using a specific endpoint type, 'NestedEndpoints'. When specifying nested endpoints:

- The endpoint must be specified using the 'targetResourceId' parameter
- If the 'Performance' traffic-routing method is used, the 'EndpointLocation' is required. Otherwise it is optional. The value must be a valid Azure region name.
- The 'Weight' and 'Priority' are optional, as for Azure endpoints.
- The 'MinChildEndpoints' parameter is optional. The default value is '1'. If the number of available endpoints falls below this threshold, the parent profile considers the child profile 'degraded' and diverts traffic to the other endpoints in the parent profile.

Example 1: Adding nested endpoints using `Add-AzureRmTrafficManagerEndpointConfig` and `Set-AzureRmTrafficManagerProfile`

In this example, we create new Traffic Manager child and parent profiles, add the child as a nested endpoint to the parent, and commit the changes.

```
$child = New-AzureRmTrafficManagerProfile -Name child -ResourceGroupName MyRG -TrafficRoutingMethod Priority -RelativeDnsName child -Ttl 30 -MonitorProtocol HTTP -MonitorPort 80 -MonitorPath "/"
$parent = New-AzureRmTrafficManagerProfile -Name parent -ResourceGroupName MyRG -TrafficRoutingMethod Performance -RelativeDnsName parent -Ttl 30 -MonitorProtocol HTTP -MonitorPort 80 -MonitorPath "/"
Add-AzureRmTrafficManagerEndpointConfig -EndpointName child-endpoint -TrafficManagerProfile $parent -Type NestedEndpoints -TargetResourceId $child.Id -EndpointStatus Enabled -EndpointLocation "North Europe" -MinChildEndpoints 2
Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile
```

For brevity in this example, we did not add any other endpoints to the child or parent profiles.

Example 2: Adding nested endpoints using `New-AzureRmTrafficManagerEndpoint`

In this example, we add an existing child profile as a nested endpoint to an existing parent profile. The profile is specified using the profile and resource group names.

```
$child = Get-AzureRmTrafficManagerEndpoint -Name child -ResourceGroupName MyRG
New-AzureRmTrafficManagerEndpoint -Name child-endpoint -ProfileName parent -ResourceGroupName MyRG -Type NestedEndpoints -
TargetResourceId $child.Id -EndpointStatus Enabled -EndpointLocation "North Europe" -MinChildEndpoints 2
```

Update a Traffic Manager Endpoint

There are two ways to update an existing Traffic Manager endpoint:

1. Get the Traffic Manager profile using `Get-AzureRmTrafficManagerProfile`, update the endpoint properties within the profile, and commit the changes using `Set-AzureRmTrafficManagerProfile`. This method has the advantage of being able to update more than one endpoint in a single operation.
2. Get the Traffic Manager endpoint using `Get-AzureRmTrafficManagerEndpoint`, update the endpoint properties, and commit the changes using `Set-AzureRmTrafficManagerEndpoint`. This method is simpler, since it does not require indexing into the Endpoints array in the profile.

Example 1: Updating endpoints using `Get-AzureRmTrafficManagerProfile` and `Set-AzureRmTrafficManagerProfile`

In this example, we modify the priority on two endpoints within an existing profile.

```
$profile = Get-AzureRmTrafficManagerProfile -Name myprofile -ResourceGroupName MyRG
$profile.Endpoints[0].Priority = 2
$profile.Endpoints[1].Priority = 1
Set-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile
```

Example 2: Updating an endpoint using `Get-AzureRmTrafficManagerEndpoint` and `Set-AzureRmTrafficManagerEndpoint`

In this example, we modify the weight of a single endpoint in an existing profile.

```
$endpoint = Get-AzureRmTrafficManagerEndpoint -Name myendpoint -ProfileName myprofile -ResourceGroupName MyRG -Type ExternalEndpoints
$endpoint.Weight = 20
Set-AzureRmTrafficManagerEndpoint -TrafficManagerEndpoint $endpoint
```

Enabling and Disabling Endpoints and Profiles

Traffic Manager allows individual endpoints to be enabled and disabled, as well as allowing enabling and disabling of entire profiles. These changes can be made by getting/updating/setting the endpoint or profile resources. To streamline these common operations, they are also supported via dedicated cmdlets.

Example 1: Enabling and disabling a Traffic Manager profile

To enable a Traffic Manager profile, use `Enable-AzureRmTrafficManagerProfile`. The profile can be specified using a profile object. The profile object can be passed via the pipeline or by using the '-TrafficManagerProfile' parameter. In this example, we specify the profile by the profile and resource group name.

```
Enable-AzureRmTrafficManagerProfile -Name MyProfile -ResourceGroupName MyResourceGroup
```

To disable a Traffic Manager profile:

```
Disable-AzureRmTrafficManagerProfile -Name MyProfile -ResourceGroupName MyResourceGroup
```

The Disable-AzureRmTrafficManagerProfile cmdlet prompts for confirmation. This prompt can be suppressed using the '-Force' parameter.

Example 2: Enabling and disabling a Traffic Manager endpoint

To enable a Traffic Manager endpoint, use `Enable-AzureRmTrafficManagerEndpoint`. There are two ways to specify the endpoint

1. Using a TrafficManagerEndpoint object passed via the pipeline or using the '-TrafficManagerEndpoint' parameter
2. Using the endpoint name, endpoint type, profile name, and resource group name:

```
Enable-AzureRmTrafficManagerEndpoint -Name MyEndpoint -Type AzureEndpoints -ProfileName MyProfile -ResourceGroupName MyRG
```

Similarly, to disable a Traffic Manager endpoint:

```
Disable-AzureRmTrafficManagerEndpoint -Name MyEndpoint -Type AzureEndpoints -ProfileName MyProfile -ResourceGroupName MyRG -Force
```

As with `Disable-AzureRmTrafficManagerProfile`, the `Disable-AzureRmTrafficManagerEndpoint` cmdlet prompts for confirmation. This prompt can be suppressed using the '-Force' parameter.

Delete a Traffic Manager Endpoint

To remove individual endpoints, use the `Remove-AzureRmTrafficManagerEndpoint` cmdlet:

```
Remove-AzureRmTrafficManagerEndpoint -Name MyEndpoint -Type AzureEndpoints -ProfileName MyProfile -ResourceGroupName MyRG
```

This cmdlet prompts for confirmation. This prompt can be suppressed using the '-Force' parameter.

Delete a Traffic Manager Profile

To delete a Traffic Manager profile, use the `Remove-AzureRmTrafficManagerProfile` cmdlet, specifying the profile and resource group names:

```
Remove-AzureRmTrafficManagerProfile -Name MyProfile -ResourceGroupName MyRG [-Force]
```

This cmdlet prompts for confirmation. This prompt can be suppressed using the '-Force' parameter.

The profile to be deleted can also be specified using a profile object:

```
$profile = Get-AzureRmTrafficManagerProfile -Name MyProfile -ResourceGroupName MyRG
Remove-AzureRmTrafficManagerProfile -TrafficManagerProfile $profile [-Force]
```

This sequence can also be piped:

```
Get-AzureRmTrafficManagerProfile -Name MyProfile -ResourceGroupName MyRG | Remove-AzureRmTrafficManagerProfile [-Force]
```

Next steps

Traffic Manager monitoring

Traffic Manager performance considerations

Add, disable, enable, or delete endpoints

1/17/2017 • 4 min to read • Edit on GitHub

The Web Apps feature in Azure App Service already provides failover and round-robin traffic routing functionality for websites within a datacenter, regardless of the website mode. Azure Traffic Manager allows you to specify failover and round-robin traffic routing for websites and cloud services in different datacenters. The first step necessary to provide that functionality is to add the cloud service or website endpoint to Traffic Manager.

> **NOTE**
>
> This article explains how to use the classic portal. The Azure classic portal only supports the creation and assignment of cloud services and Web apps as endpoints. The new Azure portal is the preferred interface.

You can also disable individual endpoints that are part of a Traffic Manager profile. Disabling an endpoint leaves it as part of the profile, but the profile acts as if the endpoint is not included in it. This action is useful for temporarily removing an endpoint that is in maintenance mode or being redeployed. Once the endpoint is up and running again, it can be enabled.

> **NOTE**
>
> Disabling an endpoint has nothing to do with its deployment state in Azure. A healthy endpoint remains up and able to receive traffic even when disabled in Traffic Manager. Additionally, disabling an endpoint in one profile does not affect its status in another profile.

To add a cloud service or website endpoint

1. On the Traffic Manager pane in the Azure classic portal, locate the Traffic Manager profile that contains the endpoint settings that you want to modify. To open the settings page, click the arrow to the right of the profile name.
2. At the top of the page, click **Endpoints** to view the endpoints that are already part of your configuration.
3. At the bottom of the page, click **Add** to access the **Add Service Endpoints** page. By default, the page lists the cloud services under **Service Endpoints**.
4. For cloud services, select the cloud services in the list to add them as endpoints for this profile. Clearing the cloud service name removes it from the list of endpoints.
5. For websites, click the **Service Type** drop-down list, and then select **Web app**.
6. Select the websites in the list to add them as endpoints for this profile. Clearing the website name removes it from the list of endpoints. You can select only one website per Azure datacenter (also known as a region). When you select the first website, the other websites in the same datacenter become unavailable for selection. Also note that only Standard websites are listed.
7. After you select the endpoints for this profile, click the checkmark on the lower right to save your changes.

> **NOTE**
>
> After you add or remove an endpoint from a profile using the *Failover* traffic routing method, the failover priority list may not be ordered they way you want. You can adjust the order of the Failover Priority List on the Configuration page. For more information, see Configure Failover traffic routing.

To disable an endpoint

1. On the Traffic Manager pane in the Azure classic portal, locate the Traffic Manager profile that contains the endpoint settings that you want to modify. To open the settings page, click the arrow to the right of the profile name.
2. At the top of the page, click **Endpoints** to view the endpoints that are included in your configuration.
3. Click the endpoint that you want to disable, and then click **Disable** at the bottom of the page.
4. Clients continue to send traffic to the endpoint for the duration of Time-to-Live (TTL). You can change the TTL on the Configuration page of the Traffic Manager profile.

To enable an endpoint

1. On the Traffic Manager pane in the Azure classic portal, locate the Traffic Manager profile that contains the endpoint settings that you want to modify. To open the settings page, click the arrow to the right of the profile name.
2. At the top of the page, click **Endpoints** to view the endpoints that are included in your configuration.
3. Click the endpoint that you want to enable, and then click **Enable** at the bottom of the page.
4. Clients are directed to the enabled endpoint as dictated by the profile.

To delete a cloud service or website endpoint

1. On the Traffic Manager pane in the Azure classic portal, locate the Traffic Manager profile that contains the endpoint settings that you want to modify. To open the settings page, click the arrow to the right of the profile name.
2. At the top of the page, click **Endpoints** to view the endpoints that are already part of your configuration.
3. On the Endpoints page, click the name of the endpoint that you want to delete from the profile.
4. At the bottom of the page, click **Delete**.

Next steps

- Manage Traffic Manager profiles
- Configure routing methods
- Troubleshooting Traffic Manager degraded state
- Traffic Manager performance considerations
- Operations on Traffic Manager (REST API Reference)

Manage an Azure Traffic Manager profile

1/17/2017 • 4 min to read • Edit on GitHub

Traffic Manager profiles use traffic-routing methods to control the distribution of traffic to your cloud services or website endpoints. This article explains how to create and manage these profiles.

Create a Traffic Manager profile using Quick Create

You can quickly create a Traffic Manager profile by using Quick Create in the Azure classic portal. Quick Create allows you to create profiles with basic configuration settings. However, you cannot use Quick Create for settings such as the set of endpoints (cloud services and websites), the failover order for the failover traffic routing method, or monitoring settings. After creating your profile, you can configure these settings in the Azure classic portal. Traffic Manager supports up to 200 endpoints per profile. However, most usage scenarios require only a few of endpoints.

To create a Traffic Manager profile

1. **Deploy your cloud services and websites to your production environment.** For more information about cloud services, see Cloud Services. For more information about websites, see Websites.
2. **Log in to the Azure classic portal.** Click **New** on the lower left of the portal, click **Network Services > Traffic Manager**, and then click **Quick Create** to begin configuring your profile.
3. **Configure the DNS prefix.** Give your traffic manager profile a unique DNS prefix name. You can specify only the prefix for a Traffic Manager domain name.
4. **Select the subscription.** Select the appropriate Azure subscription. Each profile is associated with a single subscription. If you only have one subscription, this option does not appear.
5. **Select the traffic routing method.** Select the traffic routing method in **traffic routing Policy**. For more information about traffic routing methods, see About Traffic Manager traffic routing methods.
6. **Click "Create" to create the profile**. When the profile configuration is completed, you can locate your profile in the Traffic Manager pane in the Azure classic portal.
7. **Configure endpoints, monitoring, and additional settings in the Azure classic portal.** Using Quick Create only configures basic settings. It is necessary to configure additional settings such as the list of endpoints and the endpoint failover order.

Disable, enable, or delete a profile

You can disable an existing profile so that Traffic Manager does not refer user requests to the configured endpoints. When you disable a Traffic Manager profile, the profile and the information contained in the profile remain intact and can be edited in the Traffic Manager interface. Referrals resume when you re-enable the profile. When you create a Traffic Manager profile in the Azure classic portal, it's automatically enabled. If you decide a profile is no longer necessary, you can delete it.

To disable a profile

1. If you are using a custom domain name, change the CNAME record on your Internet DNS server so that it no longer points to your Traffic Manager profile.
2. Traffic stops being directed to the endpoints through the Traffic Manager profile settings.
3. Select the profile that you want to disable. On the Traffic Manager page, highlight the profile by clicking the column next to the profile name. Note, clicking the name of the profile or the arrow next to the name opens the settings page for the profile.
4. After selecting the profile, click **Disable** at the bottom of the page.

To enable a profile

1. Select the profile that you want to disable. On the Traffic Manager page, highlight the profile by clicking the column next to the profile name. Note, clicking the name of the profile or the arrow next to the name opens the settings page for the profile.
2. After selecting the profile, click **Enable** at the bottom of the page.
3. If you are using a custom domain name, create a CNAME resource record on your Internet DNS server to point to the domain name of your Traffic Manager profile.
4. Traffic is directed to the endpoints again.

To delete a profile

1. Ensure that the DNS resource record on your Internet DNS server no longer uses a CNAME resource record that points to the domain name of your Traffic Manager profile.
2. Select the profile that you want to disable. On the Traffic Manager page, highlight the profile by clicking the column next to the profile name. Note, clicking the name of the profile or the arrow next to the name opens the settings page for the profile.
3. After selecting the profile, click **Delete** at the bottom of the page.

View Traffic Manager profile change history

You can view the change history for your Traffic Manager profile in the Azure classic portal in Management Services.

To view your Traffic Manager change history

1. In the left pane of the Azure classic portal, click **Management Services**.
2. On the Management Services page, click **Operation Logs**.
3. On the Operation Logs page, you can filter to view the change history for your Traffic Manager profile. After selecting your filtering options, click the checkmark to view the results.

   - To view the changes for all your profiles, select your subscription and time range and then select **Traffic Manager** from the **Type** shortcut menu.
   - To filter by profile name, type the name of the profile in the **Service Name** field or select it from the shortcut menu.
   - To view details for each individual change, select the row with the change that you want to view, and then click **Details** at the bottom of the page. In the **Operation Details** window, you can view the XML representation of the API object that was created or updated as part of the operation.

Next steps

- Add an endpoint
- Configure failover routing method
- Configure round robin routing method
- Configure performance routing method
- Point a company Internet domain to a Traffic Manager domain name
- Troubleshooting Traffic Manager degraded state

Point a company Internet domain to an Azure Traffic Manager domain

1/17/2017 • 1 min to read • Edit on GitHub

When you create a Traffic Manager profile, Azure automatically assigns a DNS name for that profile. To use a name from your DNS zone, create a CNAME DNS record that maps to the domain name of your Traffic Manager profile. You can find the Traffic Manager domain name in the **General** section on the Configuration page of the Traffic Manager profile.

For example, to point name www.contoso.com to the Traffic Manager DNS name contoso.trafficmanager.net, you would create the following DNS resource record:

```
www.contoso.com IN CNAME contoso.trafficmanager.net
```

All traffic requests to *www.contoso.com* get directed to *contoso.trafficmanager.net*.

IMPORTANT

You cannot point a second-level domain, such as *contoso.com*, to the Traffic Manager domain. DNS protocol standards do not allow CNAME records for second-level domain names.

Next steps

- Traffic Manager routing methods
- Traffic Manager - Disable, enable or delete a profile
- Traffic Manager - Disable or enable an endpoint

Configure Traffic Manager routing methods

1/17/2017 • 5 min to read • Edit on GitHub

Azure Traffic Manager provides three routing methods that control how traffic is routed to available service endpoints. The traffic-routing method is applied to each DNS query received to determine which endpoint should be returned in the DNS response.

There are three traffic routing methods available in Traffic Manager:

- **Priority:** Select 'Priority' when you want to use a primary service endpoint and provide backups in case the primary is unavailable.
- **Weighted:** Select 'Weighted' when you want to distribute traffic across a set of endpoints, either evenly or according to weights, which you define.
- **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.

Configure Priority routing method

Regardless of the website mode, Azure Websites already provide failover functionality for websites within a datacenter (also known as a region). Traffic Manager provides failover for websites in different datacenters.

A common pattern for service failover is to send traffic to a primary service and provide a set of identical backup services for failover. The following steps explain how to configure this prioritized failover with Azure cloud services and websites:

1. In the Azure classic portal, in the left pane, click the **Traffic Manager** icon to open the Traffic Manager pane.
2. On the Traffic Manager pane in the Azure classic portal, locate the Traffic Manager profile that contains the settings that you want to modify. To open the profile settings page, click the arrow to the right of the profile name.
3. On your profile page, click **Endpoints** at the top of the page. Verify that both the cloud services and websites that you want to include in your configuration are present.
4. Click **Configure** at the top to open the configuration page.
5. For **traffic routing method settings**, verify that the traffic routing method is **Failover**. If it is not, click **Failover** from the dropdown list.
6. For **Failover Priority List**, adjust the failover order for your endpoints. When you select the **Failover** traffic routing method, the order of the selected endpoints matters. The primary endpoint is on top. Use the up and down arrows to change the order as needed. For information about how to set the failover priority by using Windows PowerShell, see Set-AzureTrafficManagerProfile.
7. Verify that the **Monitoring Settings** are configured appropriately. Monitoring ensures that endpoints that are offline are not sent traffic. To monitor endpoints, you must specify a path and filename. A forward slash "/" is a valid entry for the relative path and implies that the file is in the root directory (default).
8. After you complete your configuration changes, click **Save** at the bottom of the page.
9. Test the changes in your configuration.
10. Once your Traffic Manager profile is working, edit the DNS record on your authoritative DNS server to point your company domain name to the Traffic Manager domain name.

Configure weighted routing method

A common traffic routing method pattern is to provide a set of identical endpoints, which include cloud services and websites, and send traffic to each in a round-robin fashion. The following steps outline how to configure this type of traffic routing method.

> **NOTE**
>
> Azure Websites already provide round-robin load balancing functionality for websites within a datacenter (also known as a region). Traffic Manager allows you to specify round-robin traffic routing method for websites in different datacenters.

1. In the Azure classic portal, in the left pane, click the **Traffic Manager** icon to open the Traffic Manager pane.
2. In the Traffic Manager pane, locate the Traffic Manager profile that contains the settings that you want to modify. To open the profile settings page, click the arrow to the right of the profile name.
3. On the page for your profile, click **Endpoints** at the top of the page and verify that the service endpoints that you want to include in your configuration are present.
4. On your profile page, click **Configure** at the top to open the configuration page.
5. For **traffic routing method Settings**, verify that the traffic routing method is **Round Robin**. If it is not, click **Round Robin** in the dropdown list.
6. Verify that the **Monitoring Settings** are configured appropriately. Monitoring ensures that endpoints that are offline are not sent traffic. To monitor endpoints, you must specify a path and filename. A forward slash "/" is a valid entry for the relative path and implies that the file is in the root directory (default).
7. After you complete your configuration changes, click **Save** at the bottom of the page.
8. Test the changes in your configuration.
9. Once your Traffic Manager profile is working, edit the DNS record on your authoritative DNS server to point your company domain name to the Traffic Manager domain name.

Configure Performance traffic routing method

The Performance traffic routing method allows you to direct traffic to the endpoint with the lowest latency from the client's network. Typically, the datacenter with the lowest latency is the closest in geographic distance. This traffic routing method cannot account for real-time changes in network configuration or load.

1. In the Azure classic portal, in the left pane, click the **Traffic Manager** icon to open the Traffic Manager pane.
2. In the Traffic Manager pane, locate the Traffic Manager profile that contains the settings that you want to modify. To open the profile settings page, click the arrow to the right of the profile name.
3. On the page for your profile, click **Endpoints** at the top of the page and verify that the service endpoints that you want to include in your configuration are present.
4. On the page for your profile, click **Configure** at the top to open the configuration page.
5. For **traffic routing method settings**, verify that the traffic routing method is **Performance**. If it's not, click **Performance** in the dropdown list.
6. Verify that the **Monitoring Settings** are configured appropriately. Monitoring ensures that endpoints that are offline are not sent traffic. To monitor endpoints, you must specify a path and filename. A forward slash "/" is a valid entry for the relative path and implies that the file is in the root directory (default).
7. After you complete your configuration changes, click **Save** at the bottom of the page.
8. Test the changes in your configuration.
9. Once your Traffic Manager profile is working, edit the DNS record on your authoritative DNS server to point your company domain name to the Traffic Manager domain name.

Next steps

- Manage Traffic Manager Profiles
- Traffic Manager routing methods
- Testing Traffic Manager Settings

- Point a company Internet domain to a Traffic Manager domain
- Manage Traffic Manager endpoints
- Troubleshooting Traffic Manager degraded state

Test your Traffic Manager settings

1/17/2017 • 3 min to read • Edit on GitHub

To test your Traffic Manager settings, you need to have multiple clients, in various locations, from which you can run your tests. Then, bring the endpoints in your Traffic Manager profile down one at a time.

- Set the DNS TTL value low so that changes propagate quickly (for example, 30 seconds).
- Know the IP addresses of your Azure cloud services and websites in the profile you are testing.
- Use tools that let you resolve a DNS name to an IP address and display that address.

You are checking to see that the DNS names resolve to IP addresses of the endpoints in your profile. The names should resolve in a manner consistent with the traffic routing method defined in the Traffic Manager profile. You can use the tools like **nslookup** or **dig** to resolve DNS names.

The following examples help you test your Traffic Manager profile.

Check Traffic Manager profile using nslookup and ipconfig in Windows

1. Open a command or Windows PowerShell prompt as an administrator.
2. Type `ipconfig /flushdns` to flush the DNS resolver cache.
3. Type `nslookup <your Traffic Manager domain name>`. For example, the following command checks the domain name with the prefix *myapp.contoso*

   ```
   nslookup myapp.contoso.trafficmanager.net
   ```

   A typical result shows the following information:

   - The DNS name and IP address of the DNS server being accessed to resolve this Traffic Manager domain name.
   - The Traffic Manager domain name you typed on the command line after "nslookup" and the IP address to which the Traffic Manager domain resolves. The second IP address is the important one to check. It should match a public virtual IP (VIP) address for one of the cloud services or websites in the Traffic Manager profile you are testing.

How to test the failover traffic routing method

1. Leave all endpoints up.
2. Using a single client, request DNS resolution for your company domain name using nslookup or a similar utility.
3. Ensure that the resolved IP address matches the primary endpoint.
4. Bring down your primary endpoint or remove the monitoring file so that Traffic Manager thinks that the application is down.
5. Wait for the DNS Time-to-Live (TTL) of the Traffic Manager profile plus an additional two minutes. For example, if your DNS TTL is 300 seconds (5 minutes), you must wait for seven minutes.
6. Flush your DNS client cache and request DNS resolution using nslookup. In Windows, you can flush your DNS cache with the ipconfig /flushdns command.
7. Ensure that the resolved IP address matches your secondary endpoint.
8. Repeat the process, bringing down each endpoint in turn. Verify that the DNS returns the IP address of the next endpoint in the list. When all endpoints are down, you should obtain the IP address of the primary endpoint again.

How to test the weighted traffic routing method

1. Leave all endpoints up.
2. Using a single client, request DNS resolution for your company domain name using nslookup or a similar utility.
3. Ensure that the resolved IP address matches one of your endpoints.
4. Flush your DNS client cache and repeat steps 2 and 3 for each endpoint. You should see different IP addresses returned for each of your endpoints.

How to test the performance traffic routing method

To effectively test a performance traffic routing method, you must have clients located in different parts of the world. You can create clients in different Azure regions that can be used to test your services. If you have a global network, you can remotely sign in to clients in other parts of the world and run your tests from there.

Alternatively, there are free web-based DNS lookup and dig services available. Some of these tools give you the ability to check DNS name resolution from various locations around the world. Do a search on "DNS lookup" for examples. Third-party services like Gomez or Keynote can be used to confirm that your profiles are distributing traffic as expected.

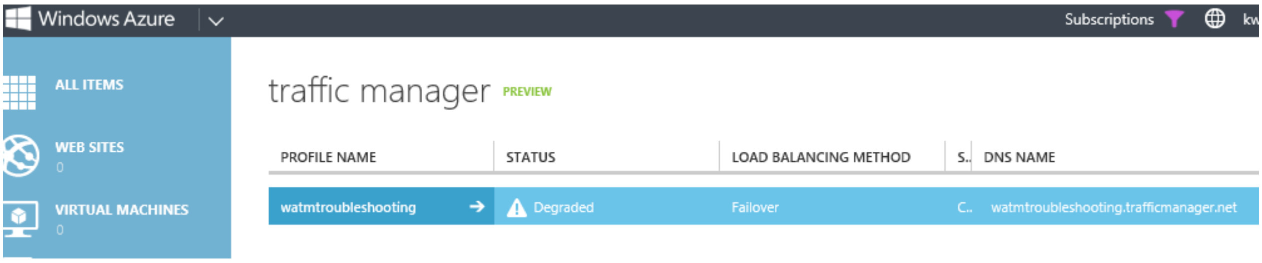
Next steps

- About Traffic Manager traffic routing methods
- Traffic Manager performance considerations
- Troubleshooting Traffic Manager degraded state

Troubleshooting degraded state on Azure Traffic Manager

1/17/2017 • 2 min to read • Edit on GitHub

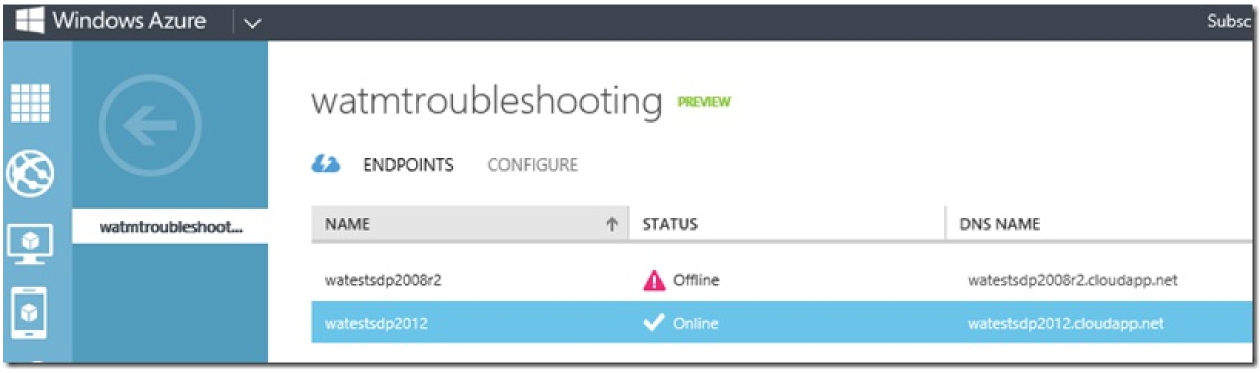
This article describes how to troubleshoot an Azure Traffic Manager profile that is showing a degraded status. For this scenario, consider that you have configured a Traffic Manager profile pointing to some of your cloudapp.net hosted services. When you check the health of your traffic manager, you see that the Status is Degraded.



traffic manager PREVIEW

PROFILE NAME	STATUS	LOAD BALANCING METHOD	S..	DNS NAME
watmtroubleshooting	Degraded	Failover	C..	watmtroubleshooting.trafficmanager.net

If you go into the Endpoints tab of that profile, you see one or more of the endpoints with an Offline status:



watmtroubleshooting PREVIEW

NAME	STATUS	DNS NAME
watestsdp2008r2	Offline	watestsdp2008r2.cloudapp.net
watestsdp2012	Online	watestsdp2012.cloudapp.net

Understanding Traffic Manager probes

- Traffic Manager considers an endpoint to be ONLINE only when the probe receives an HTTP 200 response back from the probe path. Any other non-200 response is a failure.
- A 30x redirect fails, even if the redirected URL returns a 200.
- For HTTPs probes, certificate errors are ignored.
- The actual content of the probe path doesn't matter, as long as a 200 is returned. Probing a URL to some static content like "/favicon.ico" is a common technique. Dynamic content, like the ASP pages, may not always return 200, even when the application is healthy.
- A best practice is to set the Probe path to something that has enough logic to determine that the site is up or down. In the previous example, by setting the path to "/favicon.ico", you are only testing that w3wp.exe is responding. This probe may not indicate that your web application is healthy. A better option would be to set a path to a something such as "/Probe.aspx" that has logic to determine the health of the site. For example, you could use performance counters to CPU utilization or measure the number of failed requests. Or you could attempt to access database resources or session state to make sure that the web application is working.
- If all endpoints in a profile are degraded, then Traffic Manager treats all endpoints as healthy and routes traffic to all endpoints. This behavior ensures that problems with the probing mechanism do not result in a complete outage of your service.

Troubleshooting

To troubleshoot a probe failure, you need a tool that shows the HTTP status code return from the probe URL. There are many tools available that show you the raw HTTP response.

- Fiddler
- curl
- wget

Also, you can use the Network tab of the F12 Debugging Tools in Internet Explorer to view the HTTP responses.

For this example we want to see the response from our probe URL: http://watestsdp2008r2.cloudapp.net:80/Probe. The following PowerShell example illustrates the problem.

```
Invoke-WebRequest 'http://watestsdp2008r2.cloudapp.net/Probe' -MaximumRedirection 0 -ErrorAction SilentlyContinue | Select-Object
StatusCode,StatusDescription
```

Example output:

```
StatusCode StatusDescription
---------- -----------------
       301 Moved Permanently
```

Notice that we received a redirect response. As stated previously, any StatusCode other than 200 is considered a failure. Traffic Manager changes the endpoint status to Offline. To resolve the problem, check the website configuration to ensure that the proper StatusCode can be returned from the probe path. Reconfigure the Traffic Manager probe to point to a path that returns a 200.

If your probe is using the HTTPS protocol, you may need to disable certificate checking to avoid SSL/TLS errors during your test. The following PowerShell statements disable certificate validation for the current PowerShell session:

```
add-type @"
using System.Net;
using System.Security.Cryptography.X509Certificates;
public class TrustAllCertsPolicy : ICertificatePolicy {
    public bool CheckValidationResult(
    ServicePoint srvPoint, X509Certificate certificate,
    WebRequest request, int certificateProblem) {
    return true;
    }
}
"@
[System.Net.ServicePointManager]::CertificatePolicy = New-Object TrustAllCertsPolicy
```

Next Steps

About Traffic Manager traffic routing methods

What is Traffic Manager

Cloud Services

Azure Web Apps

Operations on Traffic Manager (REST API Reference)

Azure Traffic Manager Cmdlets

Using load-balancing services in Azure

1/17/2017 • 10 min to read • Edit on GitHub

Introduction

Microsoft Azure provides multiple services for managing how network traffic is distributed and load balanced. You can use these services individually or combine their methods, depending on your needs, to build the optimal solution.

In this tutorial, we first define a customer use case and see how it can be made more robust and performant by using the following Azure load-balancing portfolio: Traffic Manager, Application Gateway, and Load Balancer. We then provide step-by-step instructions for creating a deployment that is geographically redundant, distributes traffic to VMs, and helps you manage different types of requests.

At a conceptual level, each of these services plays a distinct role in the load-balancing hierarchy.

- **Traffic Manager** provides global DNS load balancing. It looks at incoming DNS requests and responds with a healthy endpoint, in accordance with the routing policy the customer has selected. Options for routing methods are:

  - Performance routing to send the requestor to the closest endpoint in terms of latency.
  - Priority routing to direct all traffic to an endpoint, with other endpoints as backup.
  - Weighted round-robin routing, which distributes traffic based on the weighting that is assigned to each endpoint.

  The client connects directly to that endpoint. Azure Traffic Manager detects when an endpoint is unhealthy and then redirects the clients to another healthy instance. Refer to Azure Traffic Manager documentation to learn more about the service.

- **Application Gateway** provides application delivery controller (ADC) as a service, offering various Layer 7 load-balancing capabilities for your application. It allows customers to optimize web farm productivity by offloading CPU-intensive SSL termination to the application gateway. Other Layer 7 routing capabilities include round-robin distribution of incoming traffic, cookie-based session affinity, URL path-based routing, and the ability to host multiple websites behind a single application gateway. Application Gateway can be configured as an Internet-facing gateway, an internal-only gateway, or a combination of both. Application Gateway is fully Azure managed, scalable, and highly available. It provides a rich set of diagnostics and logging capabilities for better manageability.
- **Load Balancer** is an integral part of the Azure SDN stack, providing high-performance, low-latency Layer 4 load-balancing services for all UDP and TCP protocols. It manages inbound and outbound connections. You can configure public and internal load-balanced endpoints and define rules to map inbound connections to back-end pool destinations by using TCP and HTTP health-probing options to manage service availability.

Scenario

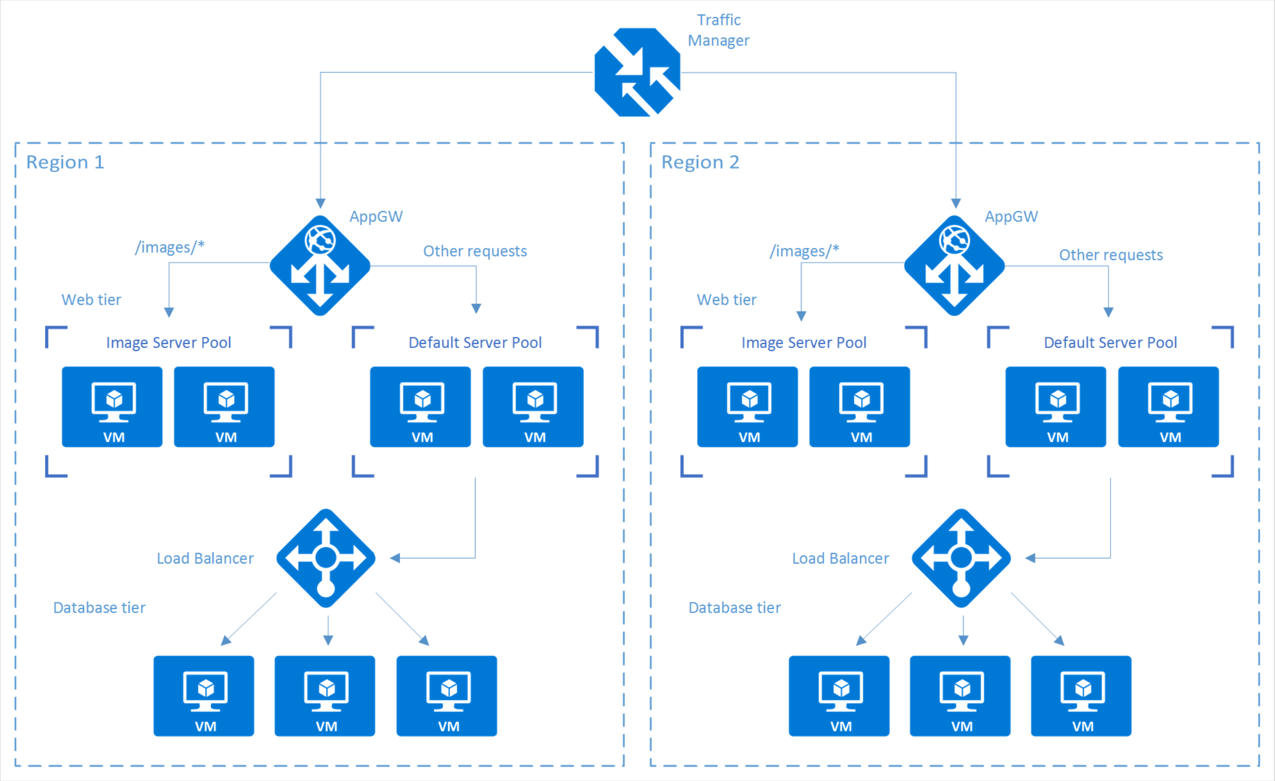
In this example scenario, we use a simple website that serves two types of content: images and dynamically rendered webpages. The website must be geographically redundant, and it should serve its users from the closest (lowest latency) location to them. The application developer has decided that any URLs that match the pattern /images/* are served from a dedicated pool of VMs that are different from the rest of the web farm.

Additionally, the default VM pool serving the dynamic content needs to talk to a back-end database that is hosted on a high-availability cluster. The entire deployment is set up through Azure Resource Manager.

Using Traffic Manager, Application Gateway, and Load Balancer allows this website to achieve these design goals:

- **Multi-geo redundancy**: If one region goes down, Traffic Manager routes traffic seamlessly to the closest region without any intervention from the application owner.
- **Reduced latency**: Because Traffic Manager automatically directs the customer to the closest region, the customer experiences lower latency when requesting the webpage contents.
- **Independent scalability**: Because the web application workload is separated by type of content, the application owner can scale the request workloads independent of each other. Application Gateway ensures that the traffic is routed to the right pools based on the specified rules and the health of the application.
- **Internal load balancing**: Because Load Balancer is in front of the high-availability cluster, only the active and healthy endpoint for a database is exposed to the application. Additionally, a database administrator can optimize the workload by distributing active and passive replicas across the cluster independent of the front-end application. Load Balancer delivers connections to the high-availability cluster and ensures that only healthy databases receive connection requests.

The following diagram shows the architecture of this scenario:



> **NOTE**
>
> This example is only one of many possible configurations of the load-balancing services that Azure offers. Traffic Manager, Application Gateway, and Load Balancer can be mixed and matched to best suit your load-balancing needs. For example, if SSL offload or Layer 7 processing is not necessary, Load Balancer can be used in place of Application Gateway.

Setting up the load-balancing stack

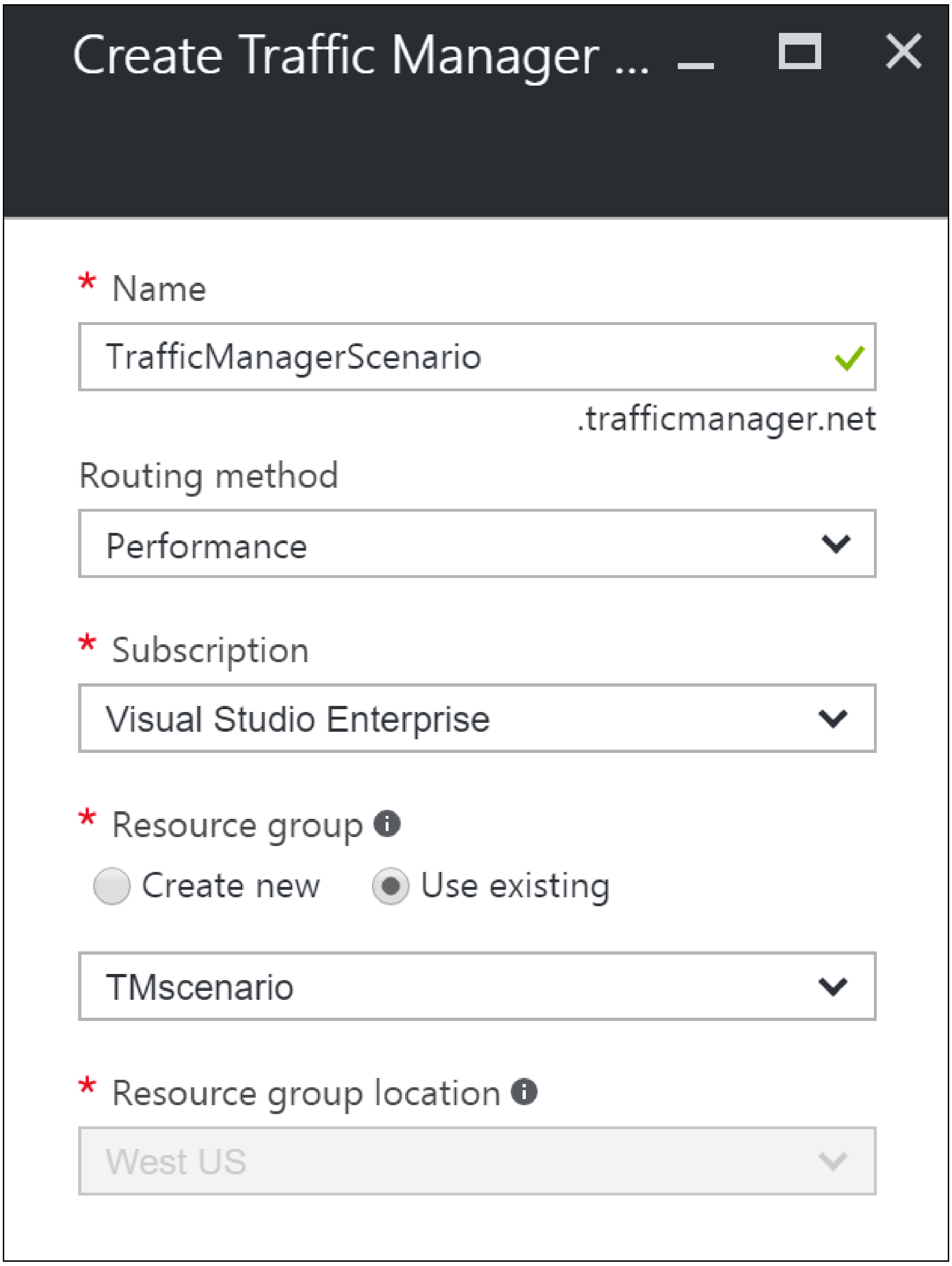
Step 1: Create a Traffic Manager profile

1. In the Azure portal, click **New**, and then search the marketplace for "Traffic Manager profile."
2. On the **Create Traffic Manager profile** blade, enter the following basic information:

   - **Name**: Give your Traffic Manager profile a DNS prefix name.
   - **Routing method**: Select the traffic-routing method policy. For more information about the methods, see About Traffic Manager traffic routing methods.

- **Subscription**: Select the subscription that contains the profile.
- **Resource group**: Select the resource group that contains the profile. It can be a new or existing resource group.
- **Resource group location**: Traffic Manager service is global and not bound to a location. However, you must specify a region for the group where the metadata associated with the Traffic Manager profile resides. This location has no impact on the runtime availability of the profile.

3. Click **Create** to generate the Traffic Manager profile.



Create Traffic Manager ...

* Name

TrafficManagerScenario ✓

.trafficmanager.net

Routing method

Performance

* Subscription

Visual Studio Enterprise

* Resource group

○ Create new ● Use existing

TMscenario

* Resource group location

West US

Step 2: Create the application gateways

1. In the Azure portal, in the left pane, click **New** > **Networking** > **Application Gateway**.
2. Enter the following basic information about the application gateway:

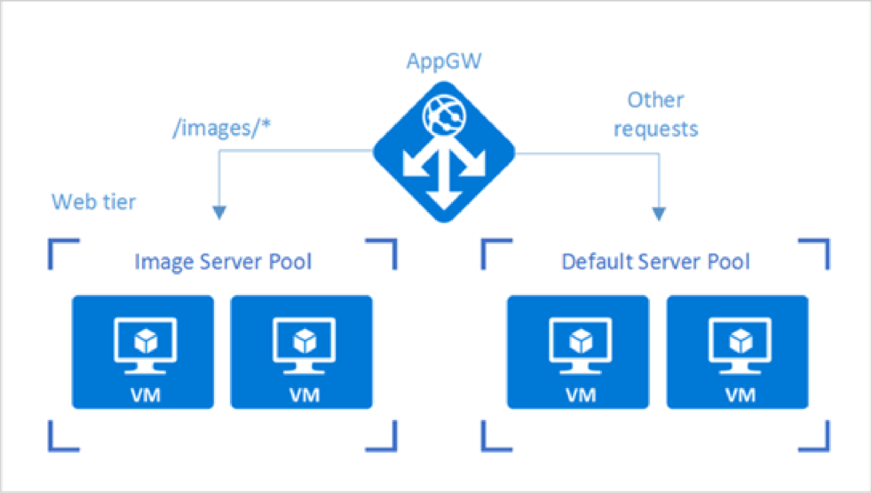
- **Name**: The name of the application gateway.
- **SKU size**: The size of the application gateway, available as Small, Medium, or Large.
- **Instance count**: The number of instances, a value from 2 through 10.
- **Resource group**: The resource group that holds the application gateway. It can be an existing resource group or a new one.

- **Location**: The region for the application gateway, which is the same location as the resource group. The location is important, because the virtual network and public IP must be in the same location as the gateway.

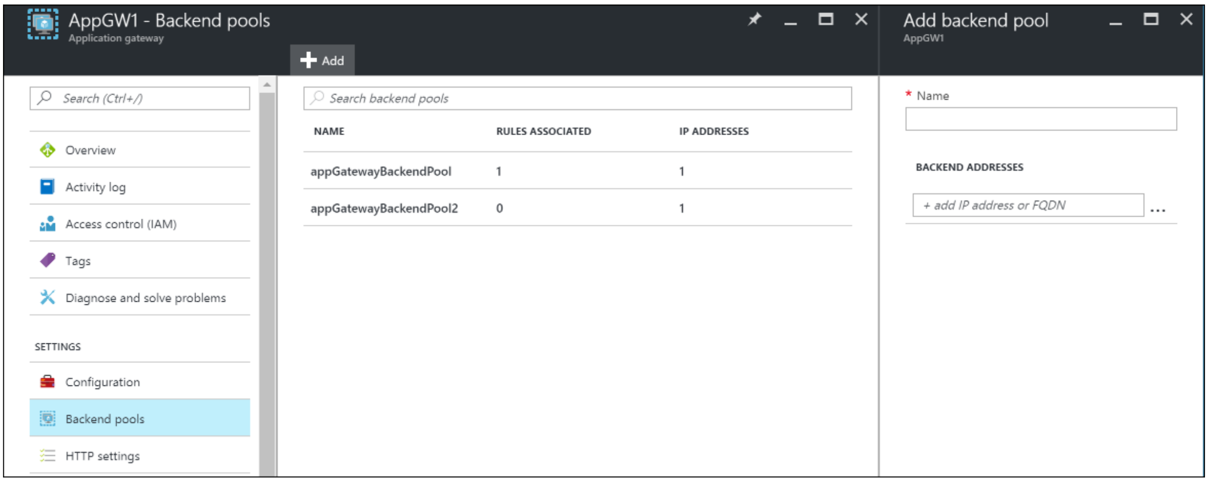
3. Click **OK**.
4. Define the virtual network, subnet, front-end IP, and listener configurations for the application gateway. In this scenario, the front-end IP address is **Public**, which allows it to be added as an endpoint to the Traffic Manager profile later on.
5. Configure the listener with one of the following options:
   - If you use HTTP, there is nothing to configure. Click **OK**.
   - If you use HTTPS, further configuration is required. Refer to Create an application gateway, starting at step 9. When you have completed the configuration, click **OK**.

Configure URL routing for application gateways

When you choose a back-end pool, an application gateway that's configured with a path-based rule takes a path pattern of the request URL in addition to round-robin distribution. In this scenario, we are adding a path-based rule to direct any URL with "/images/*" to the image server pool. For more information about configuring URL path-based routing for an application gateway, refer to Create a path-based rule for an application gateway.

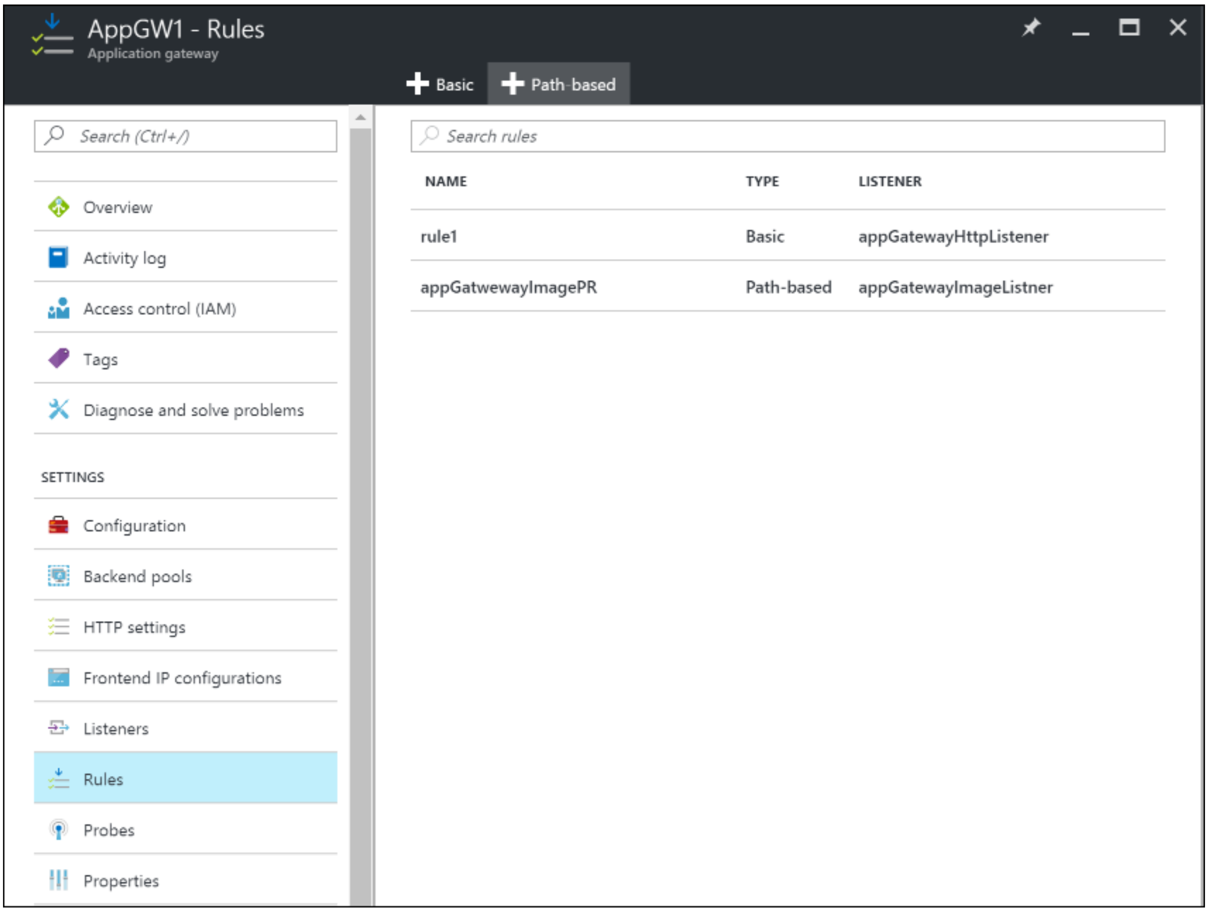


1. From your resource group, go to the instance of the application gateway that you created in the preceding section.
2. Under **Settings**, select **Backend pools**, and then select **Add** to add the VMs that you want to associate with the web-tier back-end pools.
3. On the **Add backend pool** blade, enter the name of the back-end pool and all the IP addresses of the machines that reside in the pool. In this scenario, we are connecting two back-end server pools of virtual machines.



NAME	RULES ASSOCIATED	IP ADDRESSES
appGatewayBackendPool	1	1
appGatewayBackendPool2	0	1

4. Under **Settings** of the application gateway, select **Rules**, and then click the **Path based** button to add a rule.



NAME	TYPE	LISTENER
rule1	Basic	appGatewayHttpListener
appGatwewayImagePR	Path-based	appGatewayImageListner

5. On the **Add path-based rule** blade, configure the rule by providing the following information.

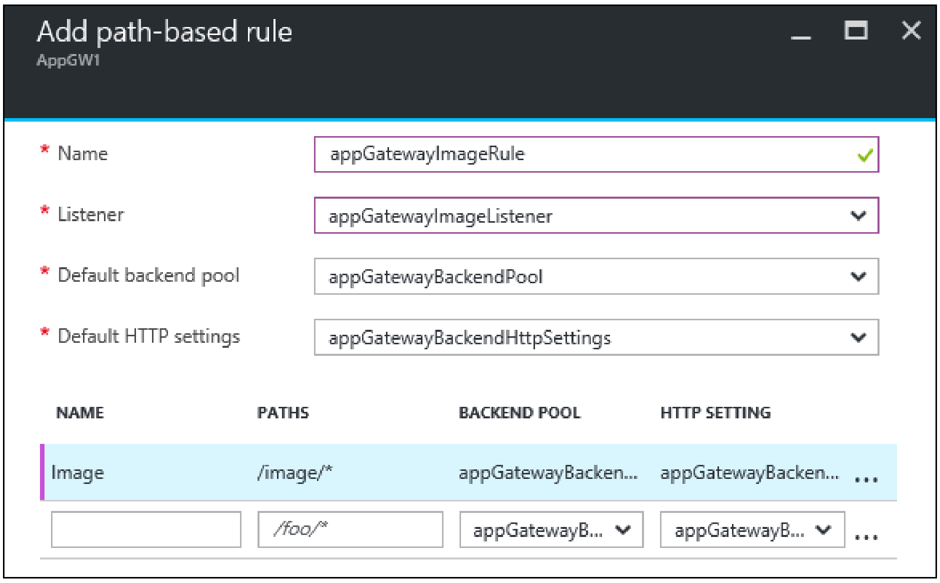
   Basic settings:

   - **Name**: The friendly name of the rule that is accessible in the portal.
   - **Listener**: The listener that is used for the rule.
   - **Default backend pool**: The back-end pool to be used with the default rule.
   - **Default HTTP settings**: The HTTP settings to be used with the default rule.

   Path-based rules:

   - **Name**: The friendly name of the path-based rule.
   - **Paths**: The path rule that is used for forwarding traffic.
   - **Backend Pool**: The back-end pool to be used with this rule.
   - **HTTP Setting**: The HTTP settings to be used with this rule.

> IMPORTANT
>
> Paths: Valid paths must start with "/". The wildcard "*" is allowed only at the end. Valid examples are /xyz, /xyz*, or /xyz/*.



Add path-based rule
AppGW1

* Name: appGatewayImageRule

* Listener: appGatewayImageListener

* Default backend pool: appGatewayBackendPool

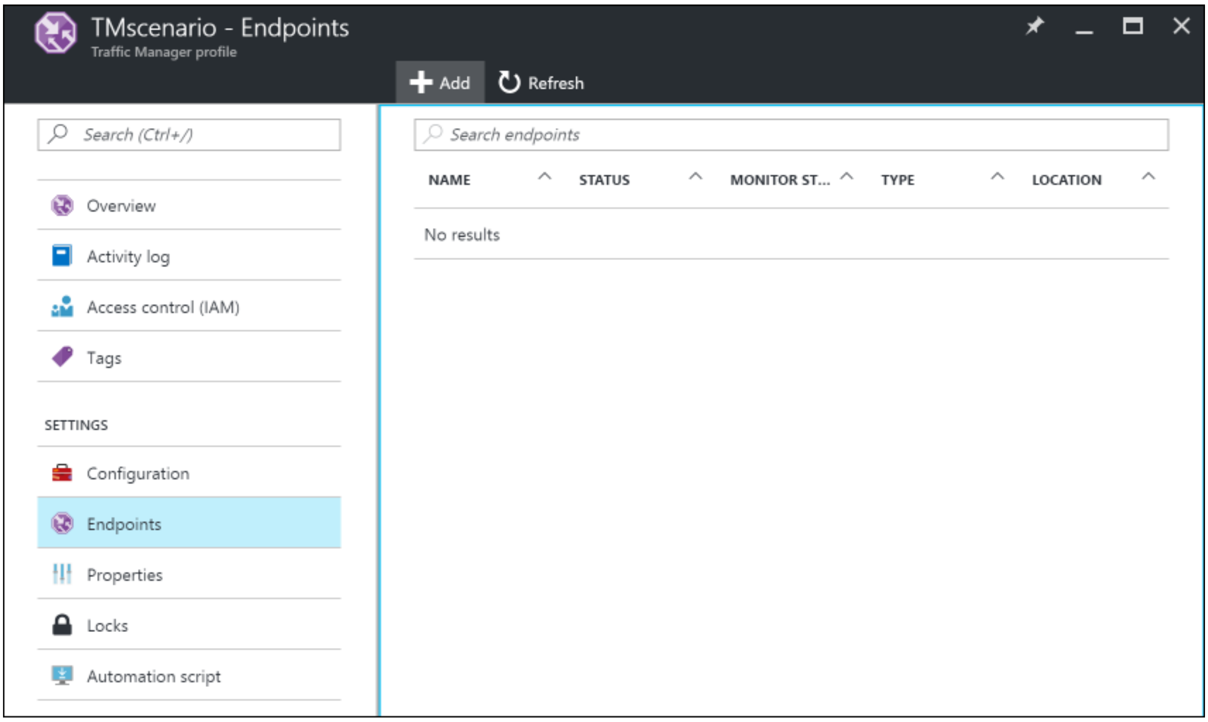
* Default HTTP settings: appGatewayBackendHttpSettings

NAME	PATHS	BACKEND POOL	HTTP SETTING
Image	/image/*	appGatewayBacken...	appGatewayBacken...
	/foo/*	appGatewayB...	appGatewayB...

Step 3: Add application gateways to the Traffic Manager endpoints

In this scenario, Traffic Manager is connected to application gateways (as configured in the preceding steps) that reside in different regions. Now that the application gateways are configured, the next step is to connect them to your Traffic Manager profile.

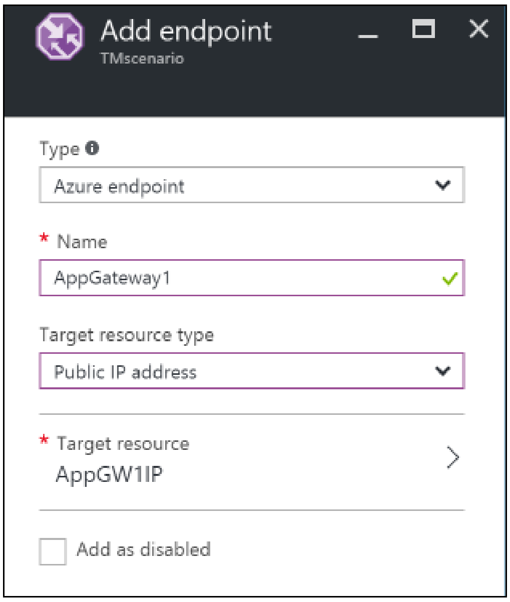
1. Open your Traffic Manager profile. To do so, look in your resource group or search for the name of the Traffic Manager profile from **All Resources**.
2. In the left pane, select **Endpoints**, and then click **Add** to add an endpoint.



NAME	STATUS	MONITOR ST...	TYPE	LOCATION
No results				

3. On the **Add endpoint** blade, create an endpoint by entering the following information:

- **Type**: Select the type of endpoint to load-balance. In this scenario, select **Azure endpoint** because we are connecting it to the application gateway instances that were configured previously.
- **Name**: Enter the name of the endpoint.
- **Target resource type**: Select **Public IP address** and then, under **Target resource**, select the public IP of the application gateway that was configured previously.


Add endpoint
TMscenario

Type
Azure endpoint

* Name
AppGateway1

Target resource type
Public IP address

* Target resource
AppGW1IP

□ Add as disabled

4. Now you can test your setup by accessing it with the DNS of your Traffic Manager profile (in this example: TrafficManagerScenario.trafficmanager.net). You can resend requests, bring up or bring down VMs and web servers that were created in different regions, and change the Traffic Manager profile settings to test your setup.

Step 4: Create a load balancer

In this scenario, Load Balancer distributes connections from the web tier to the databases within a high-availability cluster.

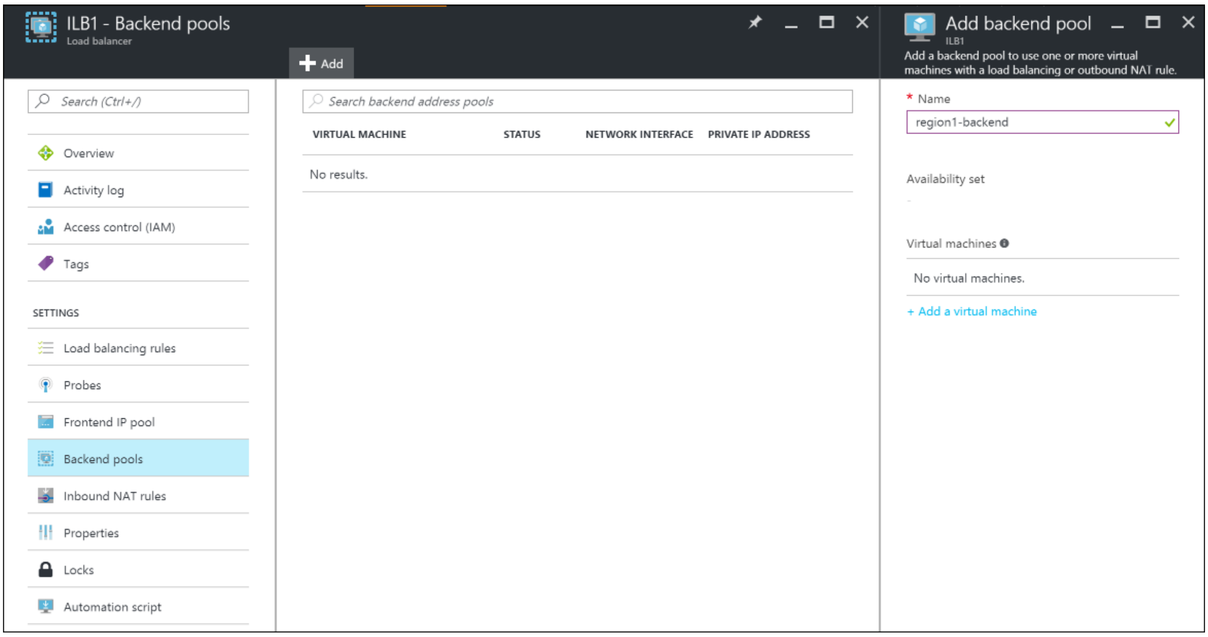
If your high-availability database cluster is using SQL Server AlwaysOn, refer to Configure one or more Always On Availability Group Listeners for step-by-step instructions.

For more information about configuring an internal load balancer, see Create an Internal load balancer in the Azure portal.

1. In the Azure portal, in the left pane, click **New** > **Networking** > **Load balancer**.
2. On the **Create load balancer** blade, choose a name for your load balancer.
3. Set the **Type** to **Internal**, and choose the appropriate virtual network and subnet for the load balancer to reside in.
4. Under **IP address assignment**, select either **Dynamic** or **Static**.
5. Under **Resource group**, choose the resource group for the load balancer.
6. Under **Location**, choose the appropriate region for the load balancer.
7. Click **Create** to generate the load balancer.

Connect a back-end database tier to the load balancer

1. From your resource group, find the load balancer that was created in the previous steps.
2. Under **Settings**, click **Backend pools**, and then click **Add** to add a back-end pool.



No results.

Add a backend pool to use one or more virtual machines with a load balancing or outbound NAT rule.

* Name: region1-backend

Availability set: -

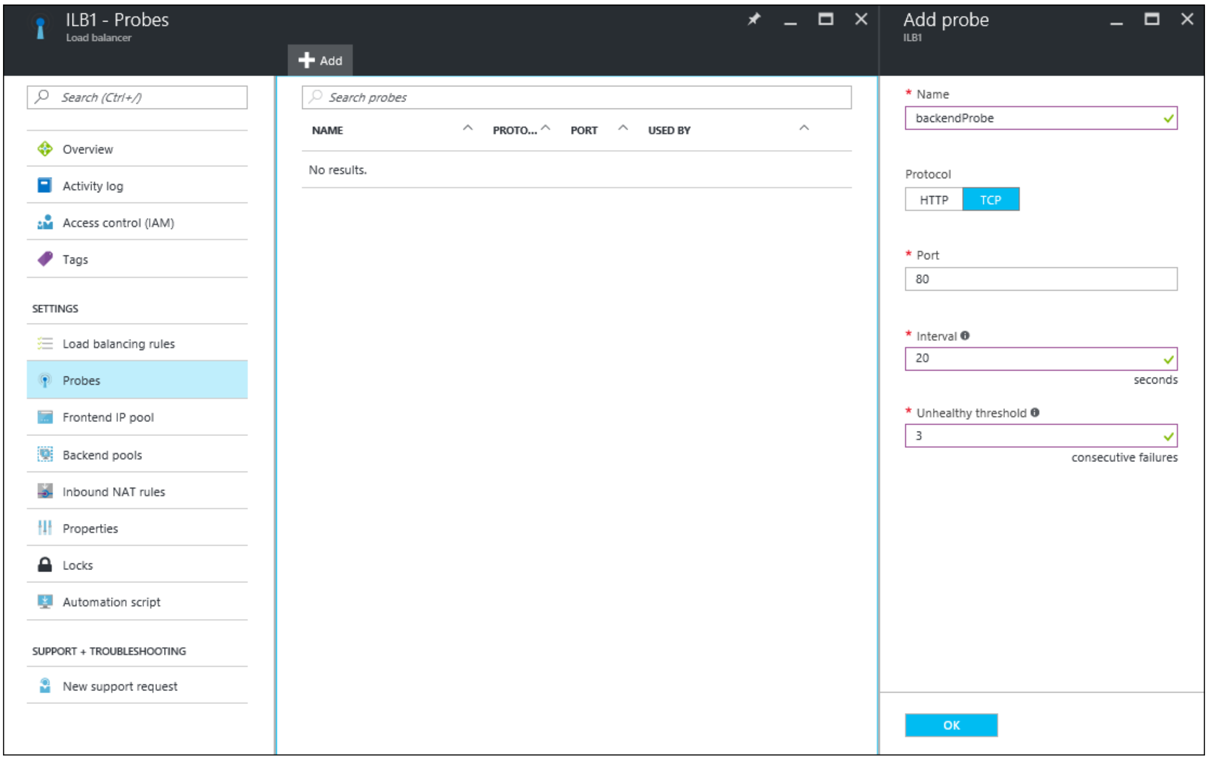
Virtual machines: No virtual machines.

3. On the **Add backend pool** blade, enter the name of the back-end pool.

4. Add either individual machines or an availability set to the back-end pool.

Configure a probe

1. In your load balancer, under **Settings**, select **Probes**, and then click **Add** to add a probe.



No results.

* Name: backendProbe

Protocol: TCP

* Port: 80

* Interval: 20 seconds

* Unhealthy threshold: 3 consecutive failures

2. On the **Add probe** blade, enter the name for the probe.

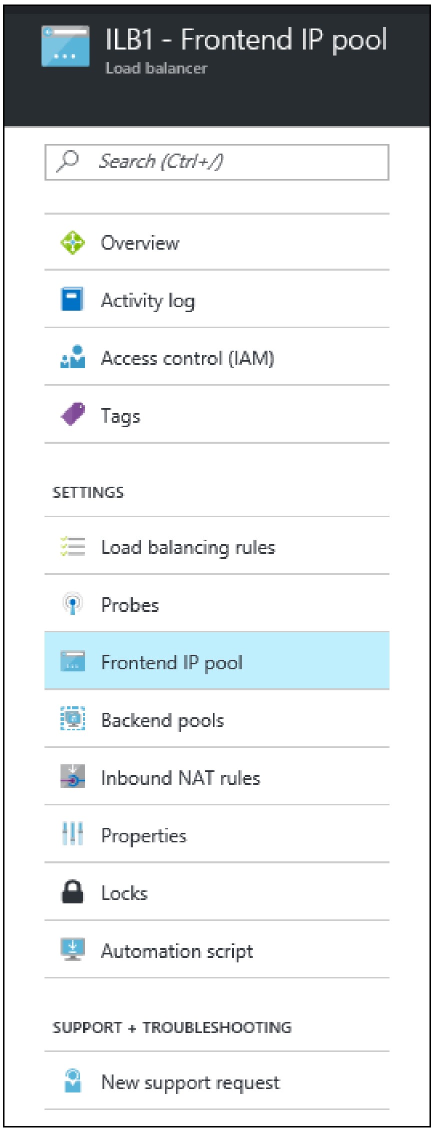
3. Select the **Protocol** for the probe. For a database, you might want a TCP probe rather than an HTTP probe. To learn more about load-balancer probes, refer to Understand load balancer probes.
4. Enter the **Port** of your database to be used for accessing the probe.
5. Under **Interval**, specify how frequently to probe the application.
6. Under **Unhealthy threshold**, specify the number of continuous probe failures that must occur for the back-end VM to be considered unhealthy.
7. Click **OK** to create the probe.

Configure the load-balancing rules

1. Under **Settings** of your load balancer, select **Load balancing rules**, and then click **Add** to create a rule.
2. On the **Add load balancing rule** blade, enter the **Name** for the load-balancing rule.
3. Choose the **Frontend IP Address** of the load balancer, **Protocol**, and **Port**.
4. Under **Backend port**, specify the port to be used in the back-end pool.
5. Select the **Backend pool** and the **Probe** that were created in the previous steps to apply the rule to.
6. Under **Session persistence**, choose how you want the sessions to persist.
7. Under **Idle timeouts**, specify the number of minutes before an idle timeout.
8. Under **Floating IP**, select either **Disabled** or **Enabled**.
9. Click **OK** to create the rule.

Step 5: Connect web-tier VMs to the load balancer

Now we configure the IP address and load-balancer front-end port in the applications that are running on your web-tier VMs for any database connections. This configuration is specific to the applications that run on these VMs. To configure the destination IP address and port, refer to the application documentation. To find the IP address of the front end, in the Azure portal, go to the front-end IP pool on the **Load balancer settings** blade.



Next steps

- Overview of Traffic Manager
- Application Gateway overview
- Azure Load Balancer overview

Network Resource Provider

1/17/2017 • 22 min to read • Edit on GitHub

An underpinning need in today's business success, is the ability to build and manage large scale network aware applications in an agile, flexible, secure and repeatable way. Azure Resource Manager enables you to create such applications, as a single collection of resources in resource groups. Such resources are managed through various resource providers under Resource Manager.

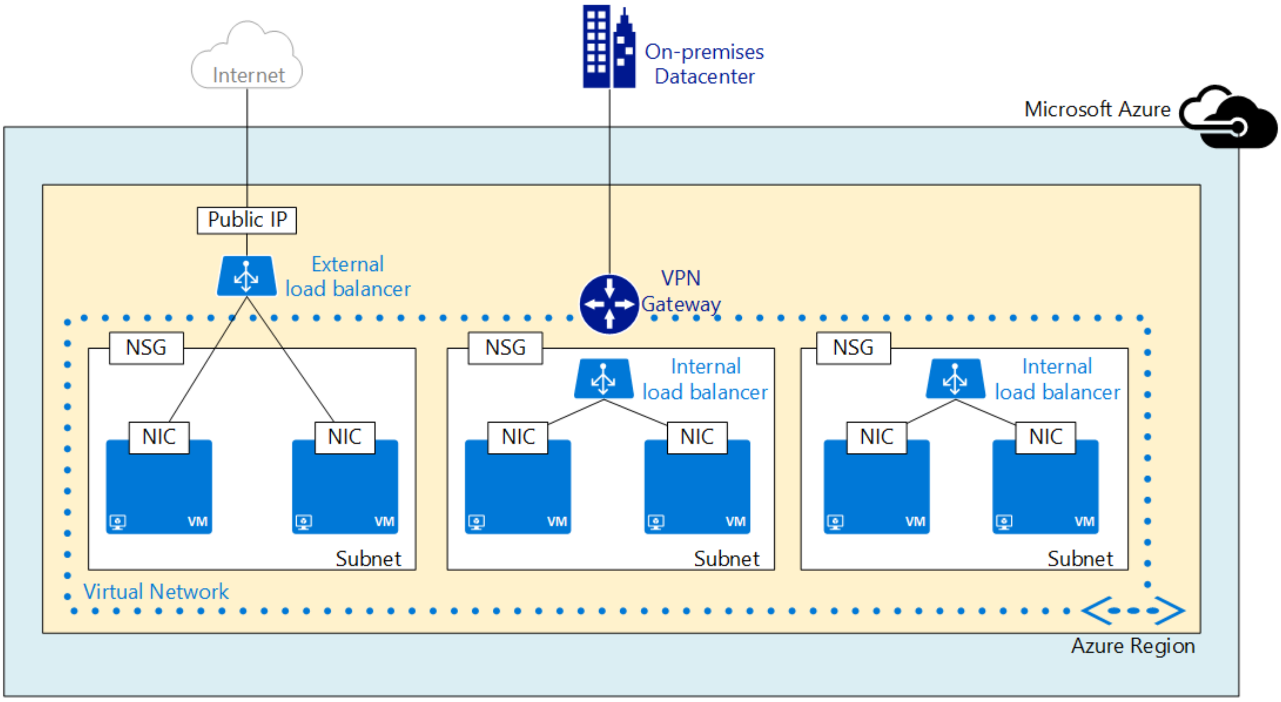
Azure Resource Manager relies on different resource providers to provide access to your resources. There are three main resource providers: Network, Storage and Compute. This document discusses the characteristics and benefits of the Network Resource Provider, including:

- **Metadata** – you can add information to resources using tags. These tags can be used to track resource utilization across resource groups and subscriptions.
- **Greater control of your network** - network resources are loosely coupled and you can control them in a more granular fashion. This means you have more flexibility in managing the networking resources.
- **Faster configuration** - because network resources are loosely coupled, you can create and orchestrate network resources in parallel. This has drastically reduced configuration time.
- **Role Based Access Control** - RBAC provides default roles, with specific security scope, in addition to allowing the creation of custom roles for secure management.
- **Easier management and deployment** - it's easier to deploy and manage applications since you can can create an entire application stack as a single collection of resources in a resource group. And faster to deploy, since you can deploy by simply providing a template JSON payload.
- **Rapid customization** - you can use declarative-style templates to enable repeatable and rapid customization of deployments.
- **Repeatable customization** - you can use declarative-style templates to enable repeatable and rapid customization of deployments.
- **Management interfaces** - you can use any of the following interfaces to manage your resources:
  - REST based API
  - PowerShell
  - .NET SDK
  - Node.JS SDK
  - Java SDK
  - Azure CLI
  - Preview Portal
  - Resource Manager template language

Network resources

You can now manage network resources independently, instead of having them all managed through a single compute resource (a virtual machine). This ensures a higher degree of flexibility and agility in composing a complex and large scale infrastructure in a resource group.

A conceptual view of a sample deployment involving a multi-tiered application is presented below. Each resource you see, such as NICs, public IP addresses, and VMs, can be managed independently.



Every resource contains a common set of properties, and their individual property set. The common properties are:

PROPERTY	DESCRIPTION	SAMPLE VALUES
name	Unique resource name. Each resource type has its own naming restrictions.	PIP01, VM01, NIC01
location	Azure region in which the resource resides	westus, eastus
id	Unique URI based identification	/subscriptions//resourceGroups/TestRG/providers/Microsoft.Network/publicIPAddresses/TestPIP

You can check the individual properties of resources in the sections below.

Public IP address

A public IP address resource provides either a reserved or dynamic Internet facing IP address. Although you can create a public IP address as a stand alone object, you need to associate it to another object to actually use the address. You can associate a public IP address to a load balancer, application gateway, or a NIC to provide Internet access to those resources.

PROPERTY	DESCRIPTION	SAMPLE VALUES
publicIPAllocationMethod	Defines if the IP address is *static* or *dynamic*.	static, dynamic
idleTimeoutInMinutes	Defines the idle time out, with a default value of 4 minutes. If no more packets for a given session is received within this time, the session is terminated.	any value between 4 and 30
ipAddress	IP address assigned to object. This is a read-only property.	104.42.233.77

DNS settings

Public IP addresses have a child object named **dnsSettings** containing the following properties:

PROPERTY	DESCRIPTION	SAMPLE VALUES
domainNameLabel	Host named used for name resolution.	www, ftp, vm1
fqdn	Fully qualified name for the public IP.	www.westus.cloudapp.azure.com
reverseFqdn	Fully qualified domain name that resolves to the IP address and is registered in DNS as a PTR record.	www.contoso.com.

Sample public IP address in JSON format:

```
{
  "name": "PIP01",
  "location": "North US",
  "tags": { "key": "value" },
  "properties": {
    "publicIPAllocationMethod": "Static",
    "idleTimeoutInMinutes": 4,
    "ipAddress": "104.42.233.77",
    "dnsSettings": {
      "domainNameLabel": "mylabel",
      "fqdn": "mylabel.westus.cloudapp.azure.com",
      "reverseFqdn": "contoso.com."
    }
  }
}
```

Additional resources

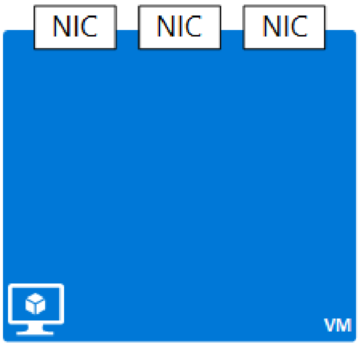
- Get more information about public IP addresses.
- Learn about instance level public IP addresses.
- Read the REST API reference documentation for public IP addresses.

NIC

A network interface card (NIC) resource provides network connectivity to an existing subnet in a VNet resource. Although you can create a NIC as a stand alone object, you need to associate it to another object to actually provide connectivity. A NIC can be used to connect a VM to a subnet, a public IP address, or a load balancer.

PROPERTY	DESCRIPTION	SAMPLE VALUES
virtualMachine	VM the NIC is associated with.	/subscriptions/{guid}/../Microsoft.Compute/virtualMachines/vm1
macAddress	MAC address for the NIC	any value between 4 and 30
networkSecurityGroup	NSG associated to the NIC	/subscriptions/{guid}/../Microsoft.Network/networkSecurityGroups/myNSG1
dnsSettings	DNS settings for the NIC	see PIP

A Network Interface Card, or NIC, represents a network interface that can be associated to a virtual machine (VM). A VM can have one or more NICs.



IP configurations

NICs have a child object named **ipConfigurations** containing the following properties:

PROPERTY	DESCRIPTION	SAMPLE VALUES
subnet	Subnet the NIC is onnected to.	/subscriptions/{guid}/../Microsoft.Network/virtualNetworks/myvnet1/subnets/mysub1
privateIPAddress	IP address for the NIC in the subnet	10.0.0.8
privateIPAllocationMethod	IP allocation method	Dynamic or Static
enableIPForwarding	Whether the NIC can be used for routing	true or false
primary	Whether the NIC is the primary NIC for the VM	true or false
publicIPAddress	PIP associated with the NIC	see DNS Settings
loadBalancerBackendAddressPools	Back end address pools the NIC is associated with	
loadBalancerInboundNatRules	Inbound load balancer NAT rules the NIC is associated with	

Sample public IP address in JSON format:

```
{
  "name": "lb-nic1-be",
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/nrprg/providers/Microsoft.Network/networkInterfaces/lb-nic1-be",
  "etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
  "type": "Microsoft.Network/networkInterfaces",
  "location": "eastus",
  "properties": {
    "provisioningState": "Succeeded",
    "resourceGuid": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "ipConfigurations": [
      {
        "name": "NIC-config",
        "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/nrprg/providers/Microsoft.Network/networkInterfaces/lb-nic1-be/ipConfigurations/NIC-config",
        "etag": "W/\"0027f1a2-3ac8-49de-b5d5-fd46550500b1\"",
        "properties": {
          "provisioningState": "Succeeded",
          "privateIPAddress": "10.0.0.4",
          "privateIPAllocationMethod": "Dynamic",
          "subnet": {
            "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/NRPRG/providers/Microsoft.Network/virtualNetworks/NRPVnet/subnets/NRPVnetSubnet"
          },
          "loadBalancerBackendAddressPools": [
            {
              "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/nrprg/providers/Microsoft.Network/loadBalancers/nrplb/backendAddressPools/NRPbackendpool"
            }
          ],
          "loadBalancerInboundNatRules": [
            {
              "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/nrprg/providers/Microsoft.Network/loadBalancers/nrplb/inboundNatRules/rdp1"
            }
          ]
        }
      }
    ],
    "dnsSettings": { ... },
    "macAddress": "00-0D-3A-10-F1-29",
    "enableIPForwarding": false,
    "primary": true,
    "virtualMachine": {
      "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/nrprg/providers/Microsoft.Compute/virtualMachines/web1"
    }
  }
}
```

Additional resources

- Read the REST API reference documentation for NICs.

Network Security Group

An NSG resource enables the creation of security boundary for workloads, by implementing allow and deny rules. Such rules can be applied to a VM, a NIC, or a subnet.

PROPERTY	DESCRIPTION	SAMPLE VALUES
subnets	List of subnet ids the NSG is applied to.	/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd

PROPERTY	DESCRIPTION	SAMPLE VALUES
securityRules	List of security rules that make up the NSG	See Security rule below
defaultSecurityRules	List of default security rules present in every NSG	See Default security rules below

- **Security rule** - An NSG can have multiple security rules defined. Each rule can allow or deny different types of traffic.

Security rule

A security rule is a child resource of an NSG containing the properties below.

PROPERTY	DESCRIPTION	SAMPLE VALUES
description	Description for the rule	Allow inbound traffic for all VMs in subnet X
protocol	Protocol to match for the rule	TCP, UDP, or *
sourcePortRange	Source port range to match for the rule	80, 100-200, *
destinationPortRange	Destination port range to match for the rule	80, 100-200, *
sourceAddressPrefix	Source address prefix to match for the rule	10.10.10.1, 10.10.10.0/24, VirtualNetwork
destinationAddressPrefix	Destination address prefix to match for the rule	10.10.10.1, 10.10.10.0/24, VirtualNetwork
direction	Direction of traffic to match for the rule	inbound or outbound
priority	Priority for the rule. Rules are checked int he order of priority, once a rule applies, no more rules are tested for matching.	10, 100, 65000
access	Type of access to apply if the rule matches	allow or deny

Sample NSG in JSON format:

```
{
  "name": "NSG-BackEnd",
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-BackEnd",
  "etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
  "type": "Microsoft.Network/networkSecurityGroups",
  "location": "westus",
  "tags": {
    "displayName": "NSG - Front End"
  },
  "properties": {
    "provisioningState": "Succeeded",
    "resourceGuid": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "securityRules": [
      {
        "name": "rdp-rule",
        "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-BackEnd/securityRules/rdp-rule",
        "etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
        "properties": {
          "provisioningState": "Succeeded",
          "description": "Allow RDP",
          "protocol": "Tcp",
          "sourcePortRange": "*",
          "destinationPortRange": "3389",
          "sourceAddressPrefix": "Internet",
          "destinationAddressPrefix": "*",
          "access": "Allow",
          "priority": 100,
          "direction": "Inbound"
        }
      }
    ],
    "defaultSecurityRules": [
      { [...],
    "subnets": [
      {
        "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd"
      }
    ]
  }
}
```

Default security rules

Default security rules have the same properties available in security rules. They exist to provide basic connectivity between resources that have NSGs applied to them. Make sure you know which default security rules exist.

Additional resources

- Get more information about NSGs.
- Read the REST API reference documentation for NSGs.
- Read the REST API reference documentation for security rules.

Route tables

Route table resources contains routes used to define how traffic flows within your Azure infrastructure. You can use user defined routes (UDR) to send all traffic from a given subnet to a virtual appliance, such as a firewall or intrusion detection system (IDS). You can associate a route table to subnets.

Route tables contain the following properties.

PROPERTY	DESCRIPTION	SAMPLE VALUES
routes	Collection of user defined routes in the route table	see user defined routes
subnets	Collection of subnets the route table is applied to	see subnets

User defined routes

You can create UDRs to specify where traffic should be sent to, based on its destination address. You can think of a route as the default gateway definition based on the destination address of a network packet.

UDRs contain the following properties.

PROPERTY	DESCRIPTION	SAMPLE VALUES
addressPrefix	Address prefix, or full IP address for the destination	192.168.1.0/24, 192.168.1.101
nextHopType	Type of device the traffic will be sent to	VirtualAppliance, VPN Gateway, Internet
nextHopIpAddress	IP address for the next hop	192.168.1.4

Sample route table in JSON format:

```
{
  "name": "UDR-BackEnd",
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/routeTables/UDR-BackEnd",
  "etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
  "type": "Microsoft.Network/routeTables",
  "location": "westus",
  "properties": {
    "provisioningState": "Succeeded",
    "routes": [
      {
        "name": "RouteToFrontEnd",
        "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/routeTables/UDR-
BackEnd/routes/RouteToFrontEnd",
        "etag": "W/\"v\"",
        "properties": {
          "provisioningState": "Succeeded",
          "addressPrefix": "192.168.1.0/24",
          "nextHopType": "VirtualAppliance",
          "nextHopIpAddress": "192.168.0.4"
        }
      }
    ],
    "subnets": [
      {
        "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/BackEnd"
      }
    ]
  }
}
```

Additional resources

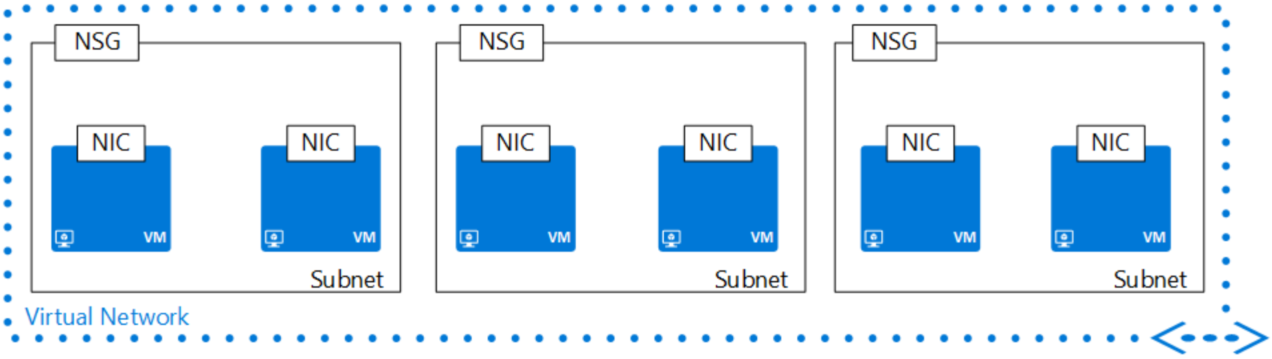
- Get more information about UDRs.
- Read the REST API reference documentation for route tables.

- Read the REST API reference documentation for user defined routes (UDRs).

Virtual Network

Virtual Networks (VNET) and subnets resources help define a security boundary for workloads running in Azure. A VNet is characterized by a collection of address spaces, defined as CIDR blocks.

> **NOTE**
>
> Network administrators are familiar with CIDR notation. If you are not familiar with CIDR, learn more about it.



VNets contain the following properties.

PROPERTY	DESCRIPTION	SAMPLE VALUES
addressSpace	Collection of address prefixes that make up the VNet in CIDR notation	192.168.0.0/16
subnets	Collection of subnets that make up the VNet	see subnets below.
ipAddress	IP address assigned to object. This is a read-only property.	104.42.233.77

Subnets

A subnet is a child resource of a VNet, and helps define segments of address spaces within a CIDR block, using IP address prefixes. NICs can be added to subnets, and connected to VMs, providing connectivity for various workloads.

Subnets contain the following properties.

PROPERTY	DESCRIPTION	SAMPLE VALUES
addressPrefix	Single address prefix that make up the subnet in CIDR notation	192.168.1.0/24
networkSecurityGroup	NSG applied to the subnet	see NSGs
routeTable	Route table applied to the subnet	see UDR
ipConfigurations	Collection of IP configuration objects used by NICs connected to the subnet	see UDR

Sample VNet in JSON format:

```
{
  "name": "TestVNet",
  "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet",
  "etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
  "type": "Microsoft.Network/virtualNetworks",
  "location": "westus",
  "tags": {
    "displayName": "VNet"
  },
  "properties": {
    "provisioningState": "Succeeded",
    "resourceGuid": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "addressSpace": {
      "addressPrefixes": [
        "192.168.0.0/16"
      ]
    },
    "subnets": [
      {
        "name": "FrontEnd",
        "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/virtualNetworks/TestVNet/subnets/FrontEnd",
        "etag": "W/\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"",
        "properties": {
          "provisioningState": "Succeeded",
          "addressPrefix": "192.168.1.0/24",
          "networkSecurityGroup": {
            "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkSecurityGroups/NSG-BackEnd"
          },
          "routeTable": {
            "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/routeTables/UDR-
FrontEnd"
          },
          "ipConfigurations": [
            {
              "id": "/subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/resourceGroups/TestRG/providers/Microsoft.Network/networkInterfaces/NICWEB1/ipConfigurations/ipconfig1"
            },
            ...]
        }
      },
      ...]
  }
}
```

Additional resources

- Get more information about VNet.
- Read the REST API reference documentation for VNets.
- Read the REST API reference documentation for Subnets.

Azure DNS

Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure.

PROPERTY	DESCRIPTION	SAMPLE VALUE
DNSzones	Domain zone information to host DNS records of a particular domain	/subscriptions/{guid}/.../providers/Microsoft.Network/dnszones/contoso.com"

DNS record sets

DNS zones have a child object named record set. Record sets are a collection of host records by type for a DNS zone. Record types are A, AAAA, CNAME, MX, NS, SOA,SRV and TXT.

PROPERTY	DESCRIPTION	SAMPLE VALUE
A	IPv4 record type	/subscriptions/{guid}/.../providers/Microsoft.Network/dnszones/contoso.com/A/www
AAAA	IPv6 record type	/subscriptions/{guid}/.../providers/Microsoft.Network/dnszones/contoso.com/AAAA/hostrecord
CNAME	canonical name record type [1]	/subscriptions/{guid}/.../providers/Microsoft.Network/dnszones/contoso.com/CNAME/www
MX	mail record type	/subscriptions/{guid}/.../providers/Microsoft.Network/dnszones/contoso.com/MX/mail
NS	name server record type	/subscriptions/{guid}/.../providers/Microsoft.Network/dnszones/contoso.com/NS/
SOA	Start of Authority record type [2]	/subscriptions/{guid}/.../providers/Microsoft.Network/dnszones/contoso.com/SOA
SRV	service record type	/subscriptions/{guid}/.../providers/Microsoft.Network/dnszones/contoso.com/SRV

[1] only allows one value per record set.

[2] only allows one record type SOA per DNS zone.

Sample of DNS zone in Json format:

```
{
 "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/deploymentTemplate.json",
 "contentVersion": "1.0.0.0",
 "parameters": {
  "newZoneName": {
   "type": "String",
   "metadata": {
     "description": "The name of the DNS zone to be created."
   }
  },
  "newRecordName": {
   "type": "String",
   "defaultValue": "www",
   "metadata": {
     "description": "The name of the DNS record to be created.  The name is relative to the zone, not the FQDN."
   }
  }
 },
 "resources":
 [
  {
   "type": "microsoft.network/dnszones",
   "name": "[parameters('newZoneName')]",
   "apiVersion": "2015-05-04-preview",
   "location": "global",
   "properties": {
   }
  },
  {
   "type": "microsoft.network/dnszones/a",
   "name": "[concat(parameters('newZoneName'), concat('/', parameters('newRecordName')))]",
   "apiVersion": "2015-05-04-preview",
   "location": "global",
   "properties":
   {
    "TTL": 3600,
    "ARecords":
    [
      {
        "ipv4Address": "1.2.3.4"
      },
      {
        "ipv4Address": "1.2.3.5"
      }
    ]
   },
   "dependsOn": [
    "[concat('Microsoft.Network/dnszones/', parameters('newZoneName'))]"
   ]
  }
   ]
}
```

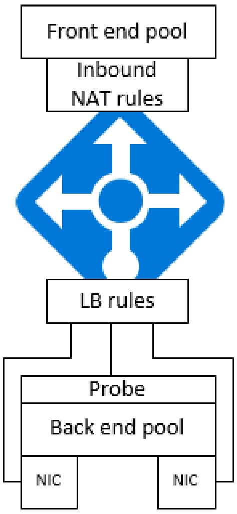
Additional resources

Read the REST API documentation for DNS zones for more information.

Read the REST API documentation for DNS record sets for more information.

Load Balancer

A load balancer is used when you want to scale your applications. Typical deployment scenarios involve applications running on multiple VM instances. The VM instances are fronted by a load balancer that helps to distribute network traffic to the various instances.



PROPERTY	DESCRIPTION
frontendIPConfigurations	a Load balancer can include one or more front end IP addresses, otherwise known as a virtual IPs (VIPs). These IP addresses serve as ingress for the traffic and can be public IP or private IP
backendAddressPools	these are IP addresses associated with the VM NICs to which load will be distributed
loadBalancingRules	a rule property maps a given front end IP and port combination to a set of back end IP addresses and port combination. With a single definition of a load balancer resource, you can define multiple load balancing rules, each rule reflecting a combination of a front end IP and port and back end IP and port associated with virtual machines. The rule is one port in the front end pool to many virtual machines in the back end pool
Probes	probes enable you to keep track of the health of VM instances. If a health probe fails, the virtual machine instance will be taken out of rotation automatically
inboundNatRules	NAT rules defining the inbound traffic flowing through the front end IP and distributed to the back end IP to a specific virtual machine instance. NAT rule is one port in the front end pool to one virtual machine in the back end pool

Example of load balancer template in Json format:

```
{
 "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
 "contentVersion": "1.0.0.0",
 "parameters": {
  "dnsNameforLBIP": {
   "type": "string",
   "metadata": {
    "description": "Unique DNS name"
   }
  },
  "location": {
   "type": "string",
   "allowedValues": [
    "East US",
        "West US",
        "West Europe",
        "East Asia",
        "Southeast Asia"
      ],
      "metadata": {
        "description": "Location to deploy"
      }
    },
    "addressPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Address Prefix"
      }
    },
    "subnetPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/24",
      "metadata": {
        "description": "Subnet Prefix"
      }
    },
    "publicIPAddressType": {
      "type": "string",
      "defaultValue": "Dynamic",
      "allowedValues": [
        "Dynamic",
        "Static"
      ],
      "metadata": {
        "description": "Public IP type"
      }
    }
  },
  "variables": {
    "virtualNetworkName": "virtualNetwork1",
    "publicIPAddressName": "publicIp1",
    "subnetName": "subnet1",
    "loadBalancerName": "loadBalancer1",
    "nicName": "networkInterface1",
    "vnetID": "[resourceId('Microsoft.Network/virtualNetworks',variables('virtualNetworkName'))]",
    "subnetRef": "[concat(variables('vnetID'),'/subnets/',variables('subnetName'))]",
    "publicIPAddressID": "[resourceId('Microsoft.Network/publicIPAddresses',variables('publicIPAddressName'))]",
    "lbID": "[resourceId('Microsoft.Network/loadBalancers',variables('loadBalancerName'))]",
    "nicId": "[resourceId('Microsoft.Network/networkInterfaces',variables('nicName'))]",
    "frontEndIPConfigID": "[concat(variables('lbID'),'/frontendIPConfigurations/loadBalancerFrontEnd')]",
    "backEndIPConfigID": "[concat(variables('nicId'),'/ipConfigurations/ipconfig1')]"
  },
  "resources": [
{
  "apiVersion": "2015-05-01-preview",
  "type": "Microsoft.Network/publicIPAddresses",
  "name": "[variables('publicIPAddressName')]",
  "location": "[parameters('location')]",
  "properties": {
    "publicIPAllocationMethod": "[parameters('publicIPAddressType')]",
    "dnsSettings": {
      "domainNameLabel": "[parameters('dnsNameforLBIP')]"
    }
  }
},
{
  "apiVersion": "2015-05-01-preview",
  "type": "Microsoft.Network/virtualNetworks",
  "name": "[variables('virtualNetworkName')]",
  "location": "[parameters('location')]",
  "properties": {
    "addressSpace": {
      "addressPrefixes": [
        "[parameters('addressPrefix')]"
      ]
    },
    "subnets": [
      {
        "name": "[variables('subnetName')]",
        "properties": {
          "addressPrefix": "[parameters('subnetPrefix')]"
        }
      }
    ]
  }
},
{
  "apiVersion": "2015-05-01-preview",
  "type": "Microsoft.Network/networkInterfaces",
  "name": "[variables('nicName')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[concat('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]",
    "[concat('Microsoft.Network/loadBalancers/', variables('loadBalancerName'))]"
  ],
  "properties": {
    "ipConfigurations": [
      {
        "name": "ipconfig1",
        "properties": {
          "privateIPAllocationMethod": "Dynamic",
          "subnet": {
            "id": "[variables('subnetRef')]"
          },
          "loadBalancerBackendAddressPools": [
            {
              "id": "[concat(variables('lbID'), '/backendAddressPools/LoadBalancerBackend')]"
            }
          ],
          "loadBalancerInboundNatRules": [
            {
              "id": "[concat(variables('lbID'),'/inboundNatRules/RDP')]"
            }
          ]
        }
      }
    ]
  }
},
{
  "apiVersion": "2015-05-01-preview",
  "name": "[variables('loadBalancerName')]",
  "type": "Microsoft.Network/loadBalancers",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[concat('Microsoft.Network/publicIPAddresses/', variables('publicIPAddressName'))]"
  ],
  "properties": {
    "frontendIPConfigurations": [
      {
        "name": "loadBalancerFrontEnd",
        "properties": {
          "publicIPAddress": {
            "id": "[variables('publicIPAddressID')]"
          }
        }
      }
    ],
    "backendAddressPools": [
      {
        "name": "loadBalancerBackEnd"
```

```
   }
  ],
  "inboundNatRules": [
   {
    "name": "RDP",
    "properties": {
     "frontendIPConfiguration": {
      "id": "[variables('frontEndIPConfigID')]"
     },
     "protocol": "tcp",
     "frontendPort": 3389,
     "backendPort": 3389,
     "enableFloatingIP": false
    }
   }
  ]
 }
}
 ]
}
```

Additional resources

Read load balancer REST API for more information.

Application Gateway

Application Gateway provides an Azure-managed HTTP load balancing solution based on layer 7 load balancing. Application load balancing allows the use of routing rules for network traffic based on HTTP.

PROPERTY	DESCRIPTION
backendAddressPools	The list of IP addresses of the back end servers. The IP addresses listed should either belong to the virtual network subnet, or should be a public IP/VIP or private IP
backendHttpSettingsCollection	Every pool has settings like port, protocol, and cookie based affinity. These settings are tied to a pool and are applied to all servers within the pool
frontendPorts	This port is the public port opened on the application gateway. Traffic hits this port, and then gets redirected to one of the back end servers
httpListeners	Listener has a frontend port, a protocol (Http or Https, these are case-sensitive), and the SSL certificate name (if configuring SSL offload)
requestRoutingRules	The rule binds the listener and the back end server pool and defines which back end server pool the traffic should be directed. Currently works only as Round-robin

Example of an application gateway Json template:

```
{
 "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
 "contentVersion": "1.0.0.0",
 "parameters": {
  "location": {
   "type": "string",
      "metadata": {
        "description": "Location to deploy to"
      }
    },
    "addressPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/16",
      "metadata": {
        "description": "Address prefix for the Virtual Network"
      }
    },
    "subnetPrefix": {
      "type": "string",
      "defaultValue": "10.0.0.0/28",
      "metadata": {
        "description": "Subnet prefix"
      }
    },
    "skuName": {
      "type": "string",
      "allowedValues": [
        "Standard_Small",
        "Standard_Medium",
        "Standard_Large"
      ],
      "defaultValue": "Standard_Medium",
      "metadata": {
        "description": "Sku Name"
      }
    },
    "capacity": {
      "type": "int",
      "defaultValue": 2,
      "metadata": {
        "description": "Number of instances"
      }
    },
    "backendIpAddress1": {
      "type": "string",
      "metadata": {
        "description": "IP Address for Backend Server 1"
      }
    },
    "backendIpAddress2": {
      "type": "string",
      "metadata": {
        "description": "IP Address for Backend Server 2"
      }
    }
  },
  "variables": {
    "applicationGatewayName": "applicationGateway1",
    "publicIPAddressName": "publicIp1",
    "virtualNetworkName": "virtualNetwork1",
    "subnetName": "appGatewaySubnet",
    "vnetID": "[resourceId('Microsoft.Network/virtualNetworks',variables('virtualNetworkName'))]",
    "subnetRef": "[concat(variables('vnetID'),'/subnets/',variables('subnetName'))]",
    "publicIPRef": "[resourceId('Microsoft.Network/publicIPAddresses',variables('publicIPAddressName'))]",
    "applicationGatewayID": "[resourceId('Microsoft.Network/applicationGateways',variables('applicationGatewayName'))]",
    "apiVersion": "2015-05-01-preview"
  },
  "resources": [
    {
      "apiVersion": "[variables('apiVersion')]",
      "type": "Microsoft.Network/publicIPAddresses",
      "name": "[variables('publicIPAddressName')]",
      "location": "[parameters('location')]",
      "properties": {
        "publicIPAllocationMethod": "Dynamic"
      }
    },
    {
      "apiVersion": "[variables('apiVersion')]",
      "type": "Microsoft.Network/virtualNetworks",
      "name": "[variables('virtualNetworkName')]",
      "location": "[parameters('location')]",
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[parameters('addressPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[variables('subnetName')]",
            "properties": {
              "addressPrefix": "[parameters('subnetPrefix')]"
            }
          }
        ]
      }
    },
    {
      "apiVersion": "[variables('apiVersion')]",
      "name": "[variables('applicationGatewayName')]",
      "type": "Microsoft.Network/applicationGateways",
      "location": "[parameters('location')]",
      "dependsOn": [
        "[concat('Microsoft.Network/virtualNetworks/', variables    ('virtualNetworkName'))]",
        "[concat('Microsoft.Network/publicIPAddresses/', variables    ('publicIPAddressName'))]"
      ],
      "properties": {
        "sku": {
          "name": "[parameters('skuName')]",
          "tier": "Standard",
          "capacity": "[parameters('capacity')]"
        },
        "gatewayIPConfigurations": [
          {
            "name": "appGatewayIpConfig",
            "properties": {
              "subnet": {
                "id": "[variables('subnetRef')]"
              }
            }
          }
        ],
        "frontendIPConfigurations": [
          {
            "name": "appGatewayFrontendIP",
            "properties": {
              "PublicIPAddress": {
                "id": "[variables('publicIPRef')]"
              }
            }
          }
        ],
        "frontendPorts": [
          {
            "name": "appGatewayFrontendPort",
            "properties": {
              "Port": 80
            }
          }
        ],
        "backendAddressPools": [
          {
            "name": "appGatewayBackendPool",
            "properties": {
              "BackendAddresses": [
                {
                  "IpAddress": "[parameters('backendIpAddress1')]"
                },
                {
                  "IpAddress": "[parameters('backendIpAddress2')]"
                }
              ]
            }
          }
        ],
        "backendHttpSettingsCollection": [
          {
            "name": "appGatewayBackendHttpSettings",
            "properties": {
              "Port": 80,
              "Protocol": "Http",
              "CookieBasedAffinity": "Disabled"
            }
          }
        ],
        "httpListeners": [
          {
            "name": "appGatewayHttpListener",
            "properties": {
              "FrontendIPConfiguration": {
                "Id": "[concat(variables('applicationGatewayID'), '/frontendIPConfigurations/appGatewayFrontendIP')]"
              },
              "FrontendPort": {
                "Id": "[concat(variables('applicationGatewayID'), '/frontendPorts/appGatewayFrontendPort')]"
              },
              "Protocol": "Http",
              "SslCertificate": null
            }
          }
        ],
        "requestRoutingRules": [
          {
            "Name": "rule1",
            "properties": {
              "RuleType": "Basic",
              "httpListener": {
                "id": "[concat(variables('applicationGatewayID'), '/httpListeners/appGatewayHttpListener')]"
              },
              "backendAddressPool": {
                "id": "[concat(variables('applicationGatewayID'), '/backendAddressPools/appGatewayBackendPool')]"
              },
              "backendHttpSettings": {
                "id": "[concat(variables('applicationGatewayID'), '/backendHttpSettingsCollection/appGatewayBackendHttpSettings')]"
              }
            }
          }
        ]
      }
    }
  ]
}
```

Additional resources

Read application gateway REST API for more information.

VPN Gateway

A VPN gateway resource enables you to create a secure connection between their on-premises data center and Azure. A VPN gateway resource can be configured in three different ways:

- **Point to Site** – you can securely access your Azure resources hosted in a VNET by using a VPN client from any computer.
- **Multi-site connection** – you can securely connect from your on-premises data centers to resources running in a VNET.
- **VNET to VNET** – you can securely connect across Azure VNETS within the same region, or across regions to build workloads with geo-redundancy.

Key properties of a VPN gateway include:

- **Gateway type** - dynamically routed or a static routed gateway.
- **VPN Client Address Pool Prefix** – IP addresses to be assigned to clients connecting in a point to site configuration.

Traffic Manager Profile

Traffic manager and its child endpoint resource enable DNS routing to endpoints in Azure and outside of Azure. Such traffic distribution is governed by routing policy methods. Traffic manager also allows endpoint health to be monitored, and traffic diverted appropriately based on the health of an endpoint.

PROPERTY	DESCRIPTION
trafficRoutingMethod	possible values are *Performance*, *Weighted*, and *Priority*
dnsConfig	FQDN for the profile
Protocol	monitoring protocol, possible values are *HTTP* and *HTTPS*
Port	monitoring port
Path	monitoring path
Endpoints	container for endpoint resources

Endpoint

An endpoint is a child resource of a Traffic Manager Profile. It represents a service or web endpoint to which user traffic is distributed based on the configured policy in the Traffic Manager Profile resource.

PROPERTY	DESCRIPTION
Type	the type of the endpoint, possible values are *Azure End point*, *External Endpoint*, and *Nested Endpoint*
targetResourceId	public IP address of a service or web endpoint. This can be an Azure or external endpoint.
Weight	endpoint weight used in traffic management.
Priority	priority of the endpoint, used to define a failover action

Sample of Traffic Manager in Json format:

```
{
  "apiVersion": "[variables('tmApiVersion')]",
  "type": "Microsoft.Network/trafficManagerProfiles",
  "name": "VMEndpointExample",
  "location": "global",
  "dependsOn": [
    "[concat('Microsoft.Network/publicIPAddresses/', variables('publicIPAddressName'), '0')]",
    "[concat('Microsoft.Network/publicIPAddresses/', variables('publicIPAddressName'), '1')]",
    "[concat('Microsoft.Network/publicIPAddresses/', variables('publicIPAddressName'), '2')]",
  ],
  "properties": {
    "profileStatus": "Enabled",
    "trafficRoutingMethod": "Weighted",
    "dnsConfig": {
      "relativeName": "[parameters('dnsname')]",
      "ttl": 30
    },
    "monitorConfig": {
      "protocol": "http",
      "port": 80,
      "path": "/"
    },
    "endpoints": [
      {
        "name": "endpoint0",
        "type": "Microsoft.Network/trafficManagerProfiles/azureEndpoints",
        "properties": {
          "targetResourceId": "[resourceId('Microsoft.Network/publicIPAddresses',concat(variables('publicIPAddressName'), 0))]",
          "endpointStatus": "Enabled",
          "weight": 1
        }
      },
      {
        "name": "endpoint1",
        "type": "Microsoft.Network/trafficManagerProfiles/azureEndpoints",
        "properties": {
          "targetResourceId": "[resourceId('Microsoft.Network/publicIPAddresses',concat(variables('publicIPAddressName'), 1))]",
          "endpointStatus": "Enabled",
          "weight": 1
        }
      },
      {
        "name": "endpoint2",
        "type": "Microsoft.Network/trafficManagerProfiles/azureEndpoints",
        "properties": {
          "targetResourceId": "[resourceId('Microsoft.Network/publicIPAddresses',concat(variables('publicIPAddressName'), 2))]",
          "endpointStatus": "Enabled",
          "weight": 1
        }
      }
    ]
  }
}
```

Additional resources

Read REST API documentation for Traffic Manager for more information.

Management interfaces

You can manage your Azure networking resources using different interfaces. In this document we will focus on tow of those interfaces: REST API, and templates.

REST API

As mentioned earlier, network resources can be managed via a variety of interfaces, including REST API,.NET SDK, Node.JS SDK, Java SDK, PowerShell, CLI, Azure Portal and templates.

The Rest API's conform to the HTTP 1.1 protocol specification. The general URI structure of the API is presented below:

```
https://management.azure.com/subscriptions/{subscription-id}/providers/{resource-provider-namespace}/locations/{region-location}/register?api-version={api-version}
```

And the parameters in braces represent the following elements:

- **subscription-id** - your Azure subscription id.
- **resource-provider-namespace** - namespace for the provider being used. THe value for the network resource provider is *Microsoft.Network*.
- **region-name** - the Azure region name

The following HTTP methods are supported when making calls to the REST API:

- **PUT** - used to create a resource of a given type, modify a resource property or change an association between resources.
- **GET** - used to retrieve information for a provisioned resource.
- **DELETE** - used to delete an existing resource.

Both the request and response conform to a JSON payload format. For more details, see Azure Resource Management APIs.

Resource Manager template language

In addition to managing resources imperatively (via APIs or SDK), you can also use a declarative programming style to build and manage network resources by using the Resource Manager Template Language.

A sample representation of a template is provided below –

```
{
 "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/deploymentTemplate.json",
 "contentVersion": "<version-number-of-template>",
 "parameters": { <parameter-definitions-of-template> },
 "variables": { <variable-definitions-of-template> },
 "resources": [ { <definition-of-resource-to-deploy> } ],
 "outputs": { <output-of-template> }
}
```

The template is primarily a JSON description of the resources and the instance values injected via parameters. The example below can be used to create a virtual network with 2 subnets.

```
{
  "$schema": "http://schema.management.azure.com/schemas/2014-04-01-preview/VNET.json",
  "contentVersion": "1.0.0.0",
  "parameters" : {
    "location": {
      "type": "String",
      "allowedValues": ["East US", "West US", "West Europe", "East Asia", "South East Asia"],
      "metadata" : {
        "Description" : "Deployment location"
      }
    },
    "virtualNetworkName":{
      "type" : "string",
      "defaultValue":"myVNET",
      "metadata" : {
        "Description" : "VNET name"
      }
    },
    "addressPrefix":{
      "type" : "string",
      "defaultValue" : "10.0.0.0/16",
      "metadata" : {
        "Description" : "Address prefix"
      }

    },
    "subnet1Name": {
      "type" : "string",
      "defaultValue" : "Subnet-1",
      "metadata" : {
        "Description" : "Subnet 1 Name"
      }
    },
    "subnet2Name": {
      "type" : "string",
      "defaultValue" : "Subnet-2",
      "metadata" : {
        "Description" : "Subnet 2 name"
      }
    },
    "subnet1Prefix" : {
      "type" : "string",
      "defaultValue" : "10.0.0.0/24",
      "metadata" : {
        "Description" : "Subnet 1 Prefix"
      }
    },
    "subnet2Prefix" : {
      "type" : "string",
      "defaultValue" : "10.0.1.0/24",
      "metadata" : {
        "Description" : "Subnet 2 Prefix"
      }
    }
  },
  "resources": [
  {
    "apiVersion": "2015-05-01-preview",
    "type": "Microsoft.Network/virtualNetworks",
    "name": "[parameters('virtualNetworkName')]",
    "location": "[parameters('location')]",
    "properties": {
      "addressSpace": {
        "addressPrefixes": [
          "[parameters('addressPrefix')]"
        ]
      },
      "subnets": [
        {
          "name": "[parameters('subnet1Name')]",
          "properties" : {
            "addressPrefix": "[parameters('subnet1Prefix')]"
          }
        },
        {
          "name": "[parameters('subnet2Name')]",
          "properties" : {
            "addressPrefix": "[parameters('subnet2Prefix')]"
          }
        }
      ]
    }
  }
  ]
}
```

You have the option of providing the parameter values manually when using a template, or you can use a parameter file. The example below shows a possible set of parameter values to be used with the template above:

```
{
 "location": {
    "value": "East US"
 },
 "virtualNetworkName": {
    "value": "VNET1"
 },
 "subnet1Name": {
    "value": "Subnet1"
 },
 "subnet2Name": {
    "value": "Subnet2"
 },
 "addressPrefix": {
    "value": "192.168.0.0/16"
 },
 "subnet1Prefix": {
    "value": "192.168.1.0/24"
 },
 "subnet2Prefix": {
    "value": "192.168.2.0/24"
 }
}
```

The main advantages of using templates are:

- You can build a complex infrastructure in a resource group in a declarative style. The orchestration of creating the resources, including dependency management, is handled by Resource Manager.
- The infrastructure can be created in a repeatable way across various regions and within a region by simply changing parameters.
- The declarative style leads to shorter lead time in building the templates and rolling out the infrastructure.

For sample templates, see Azure quickstart templates.

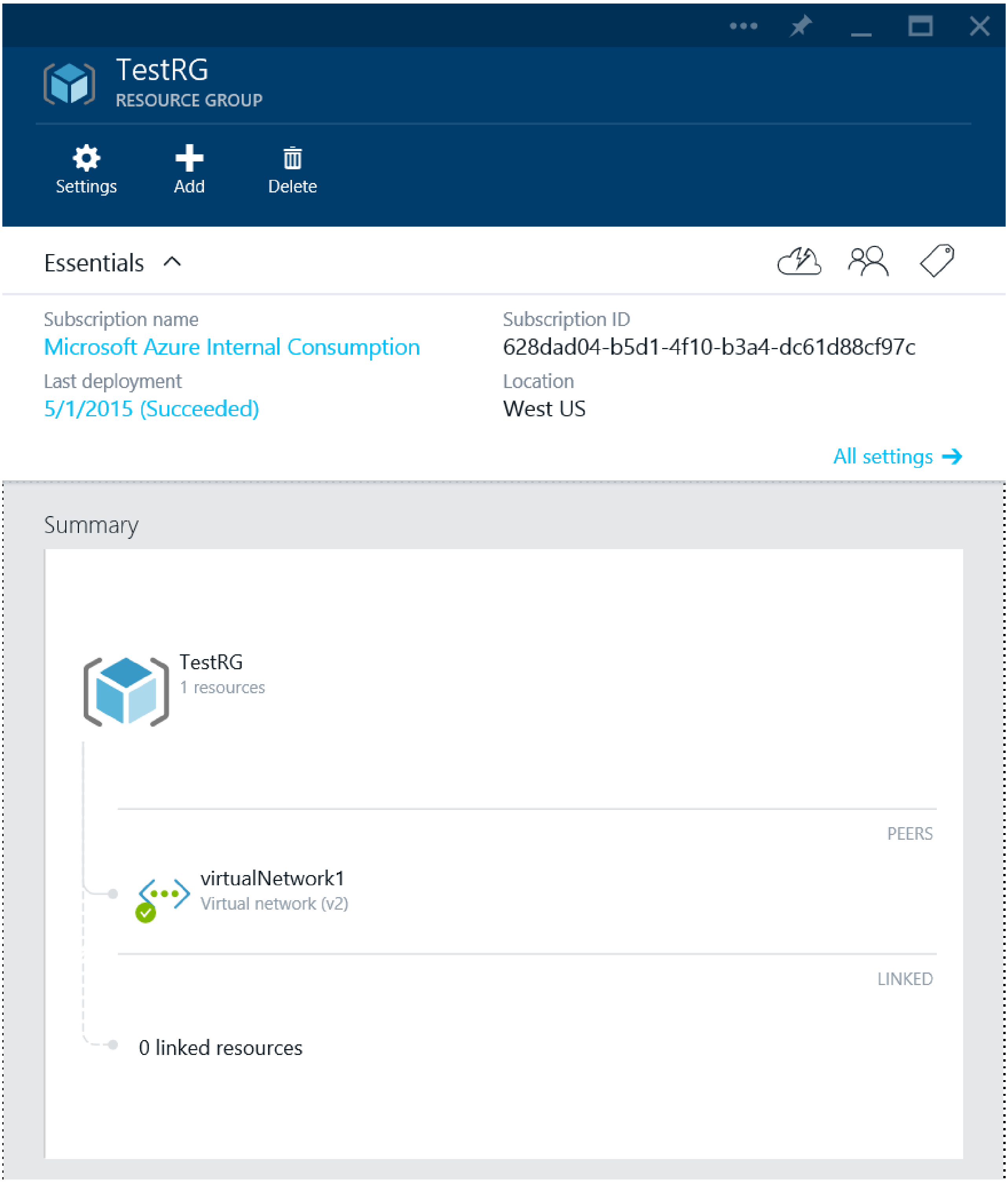
For more information on the Resource Manager Template Language, see Azure Resource Manager Template Language.

The sample template above uses the virtual network and subnet resources. There are other network resources you can use as listed below:

Using a template

You can deploy services to Azure from a template by using PowerShell, AzureCLI, or by performing a click to deploy from GitHub. To deploy services from a template in GitHub, execute the following steps:

1. Open the template3 file from GitHub. As an example, open Virtual network with two subnets.
2. Click on **Deploy to Azure**, and then sign in on to the Azure portal with your credentials.
3. Verify the template, and then click **Save**.
4. Click **Edit parameters** and select a location, such as *West US*, for the vnet and subnets.
5. If necessary, change the **ADDRESSPREFIX** and **SUBNETPREFIX** parameters, and then click **OK**.
6. Click **Select a resource group** and then click on the resource group you want to add the vnet and subnets to. Alternatively, you can create a new resource group by clicking **Or create new**.
7. Click **Create**. Notice the tile displaying **Provisioning Template deployment**. Once the deployment is done, you will see a screen similar to one below.



Subscription name
Microsoft Azure Internal Consumption

Subscription ID
628dad04-b5d1-4f10-b3a4-dc61d88cf97c

Last deployment
5/1/2015 (Succeeded)

Location
West US

Next steps

Azure Resource Manager Template Language

Azure Networking – commonly used templates

Azure Resource Manager vs. classic deployment

Azure Resource Manager Overview

Azure subscription and service limits, quotas, and constraints

1/17/2017 • 49 min to read • Edit on GitHub

This document lists some of the most common Microsoft Azure limits, which are also sometimes called quotas. This document doesn't currently cover all Azure services. Over time, the list will be expanded and updated to cover more of the platform.

Please visit Azure Pricing Overview to learn more about Azure pricing. There, you can estimate your costs using the Pricing Calculator or by visiting the pricing details page for a service (for example, Windows VMs).

> **NOTE**
>
> If you want to raise the limit or quota above the **Default Limit**, open an online customer support request at no charge. The limits can't be raised above the **Maximum Limit** value shown in the following tables. If there is no **Maximum Limit** column, then the resource doesn't have adjustable limits.
>
> Free Trial subscriptions are not eligible for limit or quota increases. If you have a Free Trial, you can upgrade to a Pay-As-You-Go subscription. For more information, see Upgrade Azure Free Trial to Pay-As-You-Go.

Limits and the Azure Resource Manager

It is now possible to combine multiple Azure resources in to a single Azure Resource Group. When using Resource Groups, limits that once were global become managed at a regional level with the Azure Resource Manager. For more information about Azure Resource Groups, see Azure Resource Manager overview.

In the limits below, a new table has been added to reflect any differences in limits when using the Azure Resource Manager. For example, there is a **Subscription Limits** table and a **Subscription Limits - Azure Resource Manager** table. When a limit applies to both scenarios, it is only shown in the first table. Unless otherwise indicated, limits are global across all regions.

> **NOTE**
>
> It is important to emphasize that quotas for resources in Azure Resource Groups are per-region accessible by your subscription, and are not per-subscription, as the service management quotas are. Let's use core quotas as an example. If you need to request a quota increase with support for cores, you need to decide how many cores you want to use in which regions, and then make a specific request for Azure Resource Group core quotas for the amounts and regions that you want. Therefore, if you need to use 30 cores in West Europe to run your application there; you should specifically request 30 cores in West Europe. But you will not have a core quota increase in any other region -- only West Europe will have the 30-core quota.
>
> As a result, you may find it useful to consider deciding what your Azure Resource Group quotas need to be for your workload in any one region, and request that amount in each region into which you are considering deployment. See troubleshooting deployment issues for more help discovering your current quotas for specific regions.

Service-specific limits

- Active Directory
- API Management
- App Service

- Application Gateway
- Application Insights
- Automation
- Azure Redis Cache
- Azure RemoteApp
- Backup
- Batch
- BizTalk Services
- CDN
- Cloud Services
- Data Factory
- Data Lake Analytics
- DNS
- DocumentDB
- Event Hubs
- IoT Hub
- Key Vault
- Media Services
- Mobile Engagement
- Mobile Services
- Monitoring
- Multi-Factor Authentication
- Networking
- Notification Hub Service
- Operational Insights
- Resource Group
- Scheduler
- Search
- Service Bus
- Site Recovery
- SQL Database
- Storage
- StorSimple System
- Stream Analytics
- Subscription
- Traffic Manager
- Virtual Machines
- Virtual Machine Scale Sets

Subscription limits

Subscription limits

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Cores per subscription [1]	20	10,000
Co-administrators per subscription	200	200

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Storage accounts per subscription[2]	200	250
Cloud services per subscription	20	200
Local networks per subscription	10	500
SQL Database servers per subscription	6	150
DNS servers per subscription	9	100
Reserved IPs per subscription	20	100
Hosted service certificates per subscription	400	400
Affinity groups per subscription	256	256
Batch accounts per region per subscription	1	50
Alert rules per subscription	250	250

[1] Extra Small instances count as one core towards the core limit despite using a partial core.

[2] This includes both Standard and Premium storage accounts. If you require more than 200 storage accounts, make a request through Azure Support. The Azure Storage team will review your business case and may approve up to 250 storage accounts.

Subscription limits - Azure Resource Manager

The following limits apply when using the Azure Resource Manager and Azure Resource Groups. Limits that have not changed with the Azure Resource Manager are not listed below. Please refer to the previous table for those limits.

For information about handling limits on Resource Manager requests, see Throttling Resource Manager requests.

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
VMs per subscription	20[1] per Region	10,000 per Region
VM total cores per subscription	20[1] per Region	10,000 per Region
VM per series (Dv2, F, etc.) cores per subscription	20[1] per Region	10,000 per Region
Co-administrators per subscription	Unlimited	Unlimited
Storage accounts per subscription	200	200[2]
Resource Groups per subscription	800	800
Availability Sets per subscription	2000 per Region	2000 per Region

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Resource Manager API Reads	15000 per hour	15000 per hour
Resource Manager API Writes	1200 per hour	1200 per hour
Resource Manager API request size	4194304 bytes	4194304 bytes
Cloud services per subscription	Not Applicable[3]	Not Applicable[3]
Affinity groups per subscription	Not Applicable[3]	Not Applicable[3]

[1] Default limits vary by offer Category Type, such as Free Trial, Pay-As-You-Go, and series, such as Dv2, F, G, etc.

[2] This includes both Standard and Premium storage accounts. If you require more than 200 storage accounts, make a request through Azure Support. The Azure Storage team will review your business case and may approve up to 250 storage accounts.

[3] These features are no longer required with Azure Resource Groups and the Azure Resource Manager.

> **NOTE**
>
> It is important to emphasize that virtual machine cores have a regional total limit as well as a regional per size series (Dv2, F, etc.) limit that are separately enforced. For example, consider a subscription with a US East total VM core limit of 30, an A series core limit of 30, and a D series core limit of 30. This subscription would be allowed to deploy 30 A1 VMs, or 30 D1 VMs, or a combnation of the two not to exceed a total of 30 cores (e.g. 10 A1 VMs and 20 D1 VMs).

Resource Group limits

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Resources per resource group (per resource type)	800	Varies per resource type
Deployments per resource group	800	800
Resources per deployment	800	800
Management Locks (per unique scope)	20	20
Number of Tags (per resource or resource group)	15	15
Tag key length	512	512
Tag value length	256	256

Virtual Machines limits

Virtual Machine limits

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Virtual machines per cloud service[1]	50	50

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Input endpoints per cloud service[2]	150	150

[1] Virtual machines created in Service Management (instead of Resource Manager) are automatically stored in a cloud service. You can add more virtual machines to that cloud service for load balancing and availability. See How to Connect Virtual Machines with a Virtual Network or Cloud Service.

[2] Input endpoints allow communications to a virtual machine from outside the virtual machine's cloud service. Virtual machines in the same cloud service or virtual network can automatically communicate with each other. See How to Set Up Endpoints to a Virtual Machine.

Virtual Machines limits - Azure Resource Manager

The following limits apply when using the Azure Resource Manager and Azure Resource Groups. Limits that have not changed with the Azure Resource Manager are not listed below. Please refer to the previous table for those limits.

RESOURCE	DEFAULT LIMIT
Virtual machines per availability set	100
Certificates per subscription	Unlimited[1]

[1] With Azure Resource Manager, certificates are stored in the Azure Key Vault. Although the number of certificates is unlimited for a subscription, there is still a 1 MB limit of certificates per deployment (which consists of either a single VM or an availability set).

Virtual Machine Scale Sets limits

RESOURCE	MAXIMUM LIMIT
Maximum number of VMs in a scale set	100
Maximum number of scale sets in a region	200

Networking limits

ExpressRoute Limits

The following limits apply to ExpressRoute resources per subscription.

RESOURCE	DEFAULT LIMIT
ExpressRoute circuits per subscription	10
ExpressRoute circuits per region per subscription for ARM	10
Maximum number of routes for Azure private peering with ExpressRoute standard	4,000
Maximum number of routes for Azure private peering with ExpressRoute premium add-on	10,000
Maximum number of routes for Azure public peering with ExpressRoute standard	200

RESOURCE	DEFAULT LIMIT
Maximum number of routes for Azure public peering with ExpressRoute premium add-on	200
Maximum number of routes for Azure Microsoft peering with ExpressRoute standard	200
Maximum number of routes for Azure Microsoft peering with ExpressRoute premium add-on	200
Number of virtual network links allowed per ExpressRoute circuit	see table below

Number of Virtual Networks per ExpressRoute circuit

CIRCUIT SIZE	NUMBER OF VNET LINKS FOR STANDARD	NUMBER OF VNET LINKS WITH PREMIUM ADD-ON
50 Mbps	10	20
100 Mbps	10	25
200 Mbps	10	25
500 Mbps	10	40
1 Gbps	10	50
2 Gbps	10	60
5 Gbps	10	75
10 Gbps	10	100

Networking limits

The following limits apply only for networking resources managed through the classic deployment model per subscription.

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Virtual networks per subscription	50	100
Local network sites per subscription	20	contact support
DNS Servers per virtual network	20	100
Private IP Addresses per virtual network	4096	4096
Concurrent TCP connections for a virtual machine or role instance	500K	500K
Network Security Groups (NSG)	100	200

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
NSG rules per NSG	200	400
User defined route tables	100	200
User defined routes per route table	100	400
Public IP addresses (dynamic)	5	contact support
Reserved public IP addresses	20	contact support
Public VIP per deployment	5	contact support
Private VIP (ILB) per deployment	1	1
Endpoint Access Control Lists (ACLs)	50	50

Networking Limits - Azure Resource Manager

The following limits apply only for networking resources managed through Azure Resource Manager per region per subscription.

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Virtual networks per subscription	50	500
Subnets per virtual network	1,000	contact support
DNS Servers per virtual network	9	25
Private IP Addresses per virtual network	4096	4096
Concurrent TCP connections for a virtual machine or role instance	500K	500K
Network Interfaces (NIC)	300	10000
Network Security Groups (NSG)	100	400
NSG rules per NSG	200	500
User defined route tables	100	200
User defined routes per route table	100	400
Public IP addresses (dynamic)	60	contact support
Public IP addresses (Static)	20	contact support
Load balancers (internal and internet facing)	100	contact support
Load balancer rules per load balancer	150	150

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Public front end IP per load balancer	5	contact support
Private front end IP per load balancer	30	contact support
VNets peerings per Virtual Network	10	50
Point-to-Site Root Certificates per VPN Gateway	20	20

Contact support in case you need to increase limits from default.

Application Gateway limits

RESOURCE	DEFAULT LIMIT	NOTE
Application Gateway	50 per subscription	
Frontend IP Configurations	2	1 public and 1 private
Frontend Ports	20	
Backend Address Pools	20	
Backend Servers per pool	100	
HTTP Listeners	20	
HTTP load balancing rules	200	# of HTTP Listeners * n, n=10 Default
Backend HTTP settings	20	1 per Backend Address Pool
Instances per gateway	10	
SSL certificates	20	1 per HTTP Listeners
Request timeout min	1 second	
Request timeout max	24hrs	
Number of sites	20	1 per HTTP Listeners
URL Maps per listener	1	

Traffic Manager limits

RESOURCE	DEFAULT LIMIT
Profiles per subscription	100 [1]
Endpoints per profile	200

[1] Contact support in case you need to increase these limits.

DNS limits

RESOURCE	DEFAULT LIMIT
Zones per subscription	100 [1]
Record sets per zone	5000 [1]
Records per record set	20

[1] Contact Azure Support in case you need to increase these limits.

Storage limits

For additional details on storage account limits, see Azure Storage Scalability and Performance Targets.

Storage Service limits

RESOURCE	DEFAULT LIMIT
Number of storage accounts per subscription	200[1]
TB per storage account	500 TB
Max number of blob containers, blobs, file shares, tables, queues, entities, or messages per storage account	Only limit is the 500 TB storage account capacity
Max size of a single blob container, table, or queue	500 TB
Max number of blocks in a block blob or append blob	50,000
Max size of a block in a block blob	100 MB
Max size of a block blob	50,000 X 100 MB (approx. 4.75 TB)
Max size of a block in an append blob	4 MB
Max size of an append blob	50,000 X 4 MB (approx. 195 GB)
Max size of a page blob	1 TB
Max size of a table entity	1 MB
Max number of properties in a table entity	252
Max size of a message in a queue	64 KB
Max size of a file share	5 TB
Max size of a file in a file share	1 TB
Max number of files in a file share	Only limit is the 5 TB total capacity of the file share
Max 8 KB IOPS per share	1000

RESOURCE	DEFAULT LIMIT
Max number of files in a file share	Only limit is the 5 TB total capacity of the file share
Max number of blob containers, blobs, file shares, tables, queues, entities, or messages per storage account	Only limit is the 500 TB storage account capacity
Max number of stored access policies per container, file share, table, or queue	5
Total Request Rate (assuming 1 KB object size) per storage account	Up to 20,000 IOPS, entities per second, or messages per second
Target throughput for single blob	Up to 60 MB per second, or up to 500 requests per second
Target throughput for single queue (1 KB messages)	Up to 2000 messages per second
Target throughput for single table partition (1 KB entities)	Up to 2000 entities per second
Target throughput for single file share	Up to 60 MB per second
Max ingress[2] per storage account (US Regions)	10 Gbps if GRS/ZRS[3] enabled, 20 Gbps for LRS
Max egress[2] per storage account (US Regions)	20 Gbps if RA-GRS/GRS/ZRS[3] enabled, 30 Gbps for LRS
Max ingress[2] per storage account (European and Asian Regions)	5 Gbps if GRS/ZRS[3] enabled, 10 Gbps for LRS
Max egress[2] per storage account (European and Asian Regions)	10 Gbps if RA-GRS/GRS/ZRS[3] enabled, 15 Gbps for LRS

[1]This includes both Standard and Premium storage accounts. If you require more than 200 storage accounts, make a request through Azure Support. The Azure Storage team will review your business case and may approve up to 250 storage accounts.

[2]*Ingress* refers to all data (requests) being sent to a storage account. *Egress* refers to all data (responses) being received from a storage account.

[3]Azure Storage replication options include:

- **RA-GRS**: Read-access geo-redundant storage. If RA-GRS is enabled, egress targets for the secondary location are identical to those for the primary location.
- **GRS**: Geo-redundant storage.
- **ZRS**: Zone-redundant storage. Available only for block blobs.
- **LRS**: Locally redundant storage.

Virtual Machine disk limits

An Azure virtual machine supports attaching a number of data disks. For optimal performance, you will want to limit the number of highly utilized disks attached to the virtual machine to avoid possible throttling. If all disks are not being highly utilized at the same time, the storage account can support a larger number disks.

- **For standard storage accounts:** A standard storage account has a maximum total request rate of 20,000 IOPS. The total IOPS across all of your virtual machine disks in a standard storage account should not exceed this limit.

You can roughly calculate the number of highly utilized disks supported by a single standard storage account based on the request rate limit. For example, for a Basic Tier VM, the maximum number of highly utilized disks is about 66 (20,000/300 IOPS per disk), and for a Standard Tier VM, it is about 40 (20,000/500 IOPS per disk), as shown in the table below.

- **For premium storage accounts:** A premium storage account has a maximum total throughput rate of 50 Gbps. The total throughput across all of your VM disks should not exceed this limit.

See Virtual machine sizes for additional details.

Standard storage accounts

Virtual machine disks: per disk limits

VM TIER	BASIC TIER VM	STANDARD TIER VM
Disk size	1023 GB	1023 GB
Max 8 KB IOPS per persistent disk	300	500
Max number of disks performing max IOPS	66	40

Premium storage accounts

Virtual machine disks: per account limits

RESOURCE	DEFAULT LIMIT
Total disk capacity per account	35 TB
Total snapshot capacity per account	10 TB
Max bandwidth per account (ingress + egress[1])	<=50 Gbps

[1] *Ingress* refers to all data (requests) being sent to a storage account. *Egress* refers to all data (responses) being received from a storage account.

Virtual machine disks: per disk limits

PREMIUM STORAGE DISK TYPE	P10	P20	P30
Disk size	128 GiB	512 GiB	1024 GiB (1 TB)
Max IOPS per disk	500	2300	5000
Max throughput per disk	100 MB per second	150 MB per second	200 MB per second
Max number of disks per storage account	280	70	35

Virtual machine disks: per VM limits

RESOURCE	DEFAULT LIMIT
Max IOPS Per VM	80,000 IOPS with GS5 VM[1]

RESOURCE	DEFAULT LIMIT
Max throughput per VM	2,000 MB/s with GS5 VM[1]

[1]Refer to VM Size for limits on other VM sizes.

Storage Resource Provider limits

The following limits apply when using the Azure Resource Manager and Azure Resource Groups only.

RESOURCE	DEFAULT LIMIT
Storage account management operations (read)	800 per 5 minutes
Storage account management operations (write)	200 per hour
Storage account management operations (list)	100 per 5 minutes

Cloud Services limits

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Web/worker roles per deployment[1]	25	25
Instance Input Endpoints per deployment	25	25
Input Endpoints per deployment	25	25
Internal Endpoints per deployment	25	25

[1]Each Cloud Service with Web/Worker roles can have two deployments, one for production and one for staging. Also note that this limit refers to the number of distinct roles (configuration) and not the number of instances per role (scaling).

App Service limits

The following App Service limits include limits for Web Apps, Mobile Apps, API Apps, and Logic Apps.

RESOURCE	FREE	SHARED (PREVIEW)	BASIC	STANDARD	PREMIUM (PREVIEW)
Web, mobile, or API apps per App Service plan[1]	10	100	Unlimited[2]	Unlimited[2]	Unlimited[2]
Logic apps per App Service plan[1]	10	10	10	20 per core	20 per core
App Service plan	1 per region	10 per resource group	100 per resource group	100 per resource group	100 per resource group
Compute instance type	Shared	Shared	Dedicated[3]	Dedicated[3]	Dedicated[3]

RESOURCE	FREE	SHARED (PREVIEW)	BASIC	STANDARD	PREMIUM (PREVIEW)
Scale-Out (max instances)	1 shared	1 shared	3 dedicated[3]	10 dedicated[3]	20 dedicated (50 in ASE)[3,4]
Storage[5]	1 GB[5]	1 GB[5]	10 GB[5]	50 GB[5]	500 GB[4,5]
CPU time (5 min)[6]	3 minutes	3 minutes	Unlimited, pay at standard rates	Unlimited, pay at standard rates	Unlimited, pay at standard rates
CPU time (day)[6]	60 minutes	240 minutes	Unlimited, pay at standard rates	Unlimited, pay at standard rates	Unlimited, pay at standard rates
Memory (1 hour)	1024 MB per App Service plan	1024 MB per app	N/A	N/A	N/A
Bandwidth	165 MB	Unlimited, data transfer rates apply	Unlimited, data transfer rates apply	Unlimited, data transfer rates apply	Unlimited, data transfer rates apply
Application architecture	32-bit	32-bit	32-bit/64-bit	32-bit/64-bit	32-bit/64-bit
Web Sockets per instance[7]	5	35	350	Unlimited	Unlimited
Concurrent debugger connections per application	1	1	1	5	5
azurewebsites.net subdomain with FTP/S and SSL	X	X	X	X	X
Custom domain support		X	X	X	X
Custom domain SSL support			Unlimited	Unlimited, 5 SNI SSL and 1 IP SSL connections included	Unlimited, 5 SNI SSL and 1 IP SSL connections included
Integrated Load Balancer		X	X	X	X
Always On			X	X	X
Scheduled Backups				Once per day	Once every 5 minutes[8]
Auto Scale			X	X	X
WebJobs[9]	X	X	X	X	X

RESOURCE	FREE	SHARED (PREVIEW)	BASIC	STANDARD	PREMIUM (PREVIEW)
Azure Scheduler support		X	X	X	X
Endpoint monitoring			X	X	X
Staging Slots				5	20
Custom domains per app		500	500	500	500
SLA			99.9%	99.95%[10]	99.95%[10]

[1] Apps and storage quotas are per App Service plan unless noted otherwise.
[2] The actual number of apps that you can host on these machines depends on the activity of the apps, the size of the machine instances, and the corresponding resource utilization.
[3] Dedicated instances can be of different sizes. See App Service Pricing for more details.
[4] Premium tier allows up to 50 computes instances (subject to availability) and 500 GB of disk space when using App Service Environments, and 20 compute instances and 250 GB storage otherwise.
[5] The storage limit is the total content size across all apps in the same App Service plan. More storage options are available in App Service Environment
[6] These resources are constrained by physical resources on the dedicated instances (the instance size and the number of instances).
[7] If you scale an app in the Basic tier to two instances, you have 350 concurrent connections for each of the two instances.
[8] Premium tier allows backup intervals down up to every 5 minutes when using App Service Environments, and 50 times per day otherwise.
[9] Run custom executables and/or scripts on demand, on a schedule, or continuously as a background task within your App Service instance. Always On is required for continuous WebJobs execution. Azure Scheduler Free or Standard is required for scheduled WebJobs. There is no predefined limit on the number of WebJobs that can run in an App Service instance, but there are practical limits that depend on what the application code is trying to do.
[10] SLA of 99.95% provided for deployments that use multiple instances with Azure Traffic Manager configured for failover.

Scheduler limits

The following table describes each of the major quotas, limits, defaults, and throttles in Azure Scheduler.

RESOURCE	LIMIT DESCRIPTION
Job size	Maximum job size is 16K. If a PUT or a PATCH results in a job larger than these limits, a 400 Bad Request status code is returned.
Request URL size	Maximum size of the request URL is 2048 chars.
Aggregate header size	Maximum aggregate header size is 4096 chars.
Header count	Maximum header count is 50 headers.
Body size	Maximum body size is 8192 chars.

RESOURCE	LIMIT DESCRIPTION
Recurrence span	Maximum recurrence span is 18 months.
Time to start time	Maximum “time to start time” is 18 months.
Job history	Maximum response body stored in job history is 2048 bytes.
Frequency	The default max frequency quota is 1 hour in a free job collection and 1 minute in a standard job collection. The max frequency is configurable on a job collection to be lower than the maximum. All jobs in the job collection are limited the value set on the job collection. If you attempt to create a job with a higher frequency than the maximum frequency on the job collection then request will fail with a 409 Conflict status code.
Jobs	The default max jobs quota is 5 jobs in a free job collection and 50 jobs in a standard job collection. The maximum number of jobs is configurable on a job collection. All jobs in the job collection are limited the value set on the job collection. If you attempt to create more jobs than the maximum jobs quota, then the request fails with a 409 Conflict status code.
Job collections	Maximum number of job collection per subscription is 200,000.
Job history retention	Job history is retained for up to 2 months or up to the last 1000 executions.
Completed and faulted job retention	Completed and faulted jobs are retained for 60 days.
Timeout	There’s a static (not configurable) request timeout of 60 seconds for HTTP actions. For longer running operations, follow HTTP asynchronous protocols; for example, return a 202 immediately but continue working in the background.

Batch limits

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Cores per Batch account	20	N/A[1]
Jobs and job schedules[2] per Batch account	20	10,000
Pools per Batch account	20	5000

[1] The number of cores per Batch account can be increased, but the maximum number is unspecified. Contact customer support to discuss increase options.

[2] Includes run-once active jobs and active job schedules. Completed jobs and job schedules are not limited.

BizTalk Services limits

The following table shows the limits for Azure Biztalk Services.

RESOURCE	FREE (PREVIEW)	DEVELOPER	BASIC	STANDARD	PREMIUM
Scale out	N/A	N/A	Yes, in increments of 1 Basic Unit	Yes, in increments of 1 Standard Unit	Yes, in increments of 1 Premium Unit
Scale Limit	N/A	N/A	Up to 8 units	Up to 8 units	Up to 8 units
EAI Bridges per Unit	N/A	25	25	125	500
EDI Agreements per Unit	N/A	10	50	250	1000
Hybrid Connections per Unit	5	5	10	50	100
Hybrid Connection Data Transfer (GBs) per Unit	5	5	50	250	500
Number of connections using BizTalk Adapter Service per Unit	N/A	1	2	5	25
Archiving	N/A	Available	N/A	N/A	Available
High Availability	N/A	N/A	Available	Available	Available

DocumentDB limits

DocumentDB is a global scale database in which throughput and storage can be scaled to handle whatever your application requires. If you have any questions about the scale DocumentDB provides, please send email to askdocdb@microsoft.com.

Mobile Engagement limits

RESOURCE	MAXIMUM LIMIT
App Collection Users	5 per App Collection
Average Data points	200 per Active User/Day
Average App-Info set	50 per Active User/Day
Average Messages pushed	20 per Active User/Day
Segments	100 per app
Criteria per segment	10
Active Push Campaigns	50 per app

RESOURCE	MAXIMUM LIMIT
Total Push Campaigns (includes Active & Completed)	1000 per app

Search limits

Pricing tiers determine the capacity and limits of your search service. Tiers include:

- *Free* multi-tenant service, shared with other Azure subscribers, intended for evaluation and small development projects.
- *Basic* provides dedicated computing resources for production workloads at a smaller scale, with up to three replicas for highly available query workloads.
- *Standard (S1, S2, S3, S3 High Density)* is for larger production workloads. Multiple levels exist within the standard tier so that you can choose a resource configuration that best matches your workload profile.

Limits per subscription

You can create multiple services within a subscription, each one provisioned at a specific tier, limited only by the number of services allowed at each tier. For example, you could create up to 12 services at the Basic tier and another 12 services at the S1 tier within the same subscription. For more information about tiers, see Choose a SKU or tier for Azure Search.

Maximum service limits can be raised upon request. Contact Azure Support if you need more services within the same subscription.

RESOURCE	FREE	BASIC	S1	S2	S3	S3 HD [1]
Maximum services	1	12	12	6	6	6
Maximum scale in SU [2]	N/A [3]	3 SU [4]	36 SU	36 SU	36 SU	36 SU

[1] S3 HD does not support indexers at this time.

[2] Search units (SU) are billing units, allocated as either a *replica* or a *partition*. You need both resources for storage, indexing, and query operations. To learn more about how search units are computed, plus a chart of valid combinations that stay under the maximum limits, see Scale resource levels for query and index workloads.

[3] Free is based on shared resources used by multiple subscribers. At this tier, there are no dedicated resources for an individual subscriber. For this reason, maximum scale is marked as not applicable.

[4] Basic has one fixed partition. At this tier, additional SUs are used for allocating more replicas for increased query workloads.

Limits per search service

Storage is constrained by disk space or by a hard limit on the *maximum number* of indexes or documents, whichever comes first.

RESOURCE	FREE	BASIC	S1	S2	S3	S3 HD
Service Level Agreement (SLA)	No [1]	Yes	Yes	Yes	Yes	Yes

RESOURCE	FREE	BASIC	S1	S2	S3	S3 HD
Storage per partition	50 MB	2 GB	25 GB	100 GB	200 GB	200 GB
Partitions per service	N/A	1	12	12	12	3 [2]
Partition size	N/A	2 GB	25 GB	100 GB	200 GB	200 GB
Replicas	N/A	3	12	12	12	12
Maximum indexes	3	5	50	200	200	1000 per partition or 3000 per service
Maximum documents	10,000	1 million	15 million per partition or 180 million per service	60 million per partition or 720 million per service	120 million per partition or 1.4 billion per service	1 million per index or 200 million per partition
Estimated queries per second (QPS)	N/A	~3 per replica	~15 per replica	~60 per replica	~60 per replica	>60 per replica

[1] Free and Preview SKUs do not come with service level agreements (SLAs). SLAs are enforced once a SKU becomes generally available.

[2] S3 HD has a hard limit of 3 partitions, which is lower than the partition limit for S3. The lower partition limit is imposed because the index count for S3 HD is substantially higher. Given that service limits exist for both computing resources (storage and processing) and content (indexes and documents), the content limit is reached first.

To learn more about limits on a more granular level, such as document size, queries per second, keys, requests, and responses, see Service limits in Azure Search.

Media Services limits

> **NOTE**
>
> For resources that are not fixed, you may ask for the quotas to be raised, by opening a support ticket. Do **not** create additional Azure Media Services accounts in an attempt to obtain higher limits.

RESOURCE	DEFAULT LIMIT
Azure Media Services (AMS) accounts in a single subscription	25 (fixed)
Assets per AMS account	1,000,000
Chained tasks per job	30 (fixed)
Assets per task	50
Assets per job	100

RESOURCE	DEFAULT LIMIT
Jobs per AMS account	50,000[2]
Unique locators associated with an asset at one time	5[4]
Live channels per AMS account	5
Programs in stopped state per channel	50
Programs in running state per channel	3
Streaming endpoints in running state per AMS account	2
Streaming units per streaming endpoint	10
Media Reserved Units (RUs) per AMS account	25 (S1, S2) 10 (S3) [1]
Storage accounts	1,000[5] (fixed)
Policies	

[1] S3 RUs are not available in India West.

[2] This number includes queued, finished, active, and canceled jobs. It does not include deleted jobs. You can delete the old jobs using **IJob.Delete** or the **DELETE** HTTP request.

[3] When making a request to list Job entities, a maximum of 1,000 will be returned per request. If you need to keep track of all submitted Jobs, you can use top/skip as described in OData system query options.

[4] Locators are not designed for managing per-user access control. To give different access rights to individual users, use Digital Rights Management (DRM) solutions. For more information, see this section.

[5] The storage accounts must be from the same Azure subscription.

[6] There is a limit of 1,000,000 policies for different AMS policies (for example, for Locator policy or ContentKeyAuthorizationPolicy).

NOTE

You should use the same policy ID if you are always using the same days / access permissions / etc.

CDN limits

RESOURCE	SOFT LIMIT
CDN profiles	8
CDN endpoints per profile	10
Custom domains per endpoint	10

Request an update to your subscription's soft limits by opening a support ticket.

Mobile Services limits

TIER:	FREE	BASIC	STANDARD
API Calls	500 K	1.5 M / unit	15 M / unit
Active Devices	500	Unlimited	Unlimited
Scale	N/A	Up to 6 units	Unlimited units
Push Notifications	Notification Hubs Free Tier included, up to 1 M pushes	Notification Hubs Basic Tier included, up to 10 M pushes	Notification Hubs Standard Tier included, up to 10 M pushes
Real time messaging/ Web Sockets	Limited	350 / mobile service	Unlimited
Offline synchronizations	Limited	Included	Included
Scheduled jobs	Limited	Included	Included
SQL Database (required) Standard rates apply for additional capacity	20 MB included	20 MB included	20 MB included
CPU capacity	60 minutes / day	Unlimited	Unlimited
Outbound data transfer	165 MB per day (daily Rollover)	Included	Included

For additional details on these limits and for information on pricing, see Mobile Services Pricing.

Monitoring limits

RESOURCE	LIMIT
Autoscale Settings	100 per region per subscription

Notification Hub Service limits

TIER:	FREE	BASIC	STANDARD
Included Pushes	1 Million	10 Million	10 Million
Active Devices	500	Unlimited	Unlimited
Tag quota per installation/registration	60	60	60

For additional details on these limits and for information on pricing, see Notification Hubs Pricing.

Event Hubs limits

The following table lists quotas and limits specific to Azure Event Hubs. For information about Event Hubs pricing, see Event Hubs Pricing.

LIMIT	SCOPE	TYPE	BEHAVIOR WHEN EXCEEDED	VALUE
Number of Event Hubs per namespace	Namespace	Static	Subsequent requests for creation of a new namespace will be rejected.	10
Number of partitions per Event Hub	Entity	Static	-	32
Number of consumer groups per Event Hub	Entity	Static	-	20
Number of AMQP connections per namespace	Namespace	Static	Subsequent requests for additional connections will be rejected and an exception will be received by the calling code.	5,000
Maximum size of Event Hubs event	System-wide	Static	-	256KB
Maximum size of an Event Hub name	Entity	Static	-	50 characters
Number of non-epoch receivers per consumer group	Entity	Static	-	5
Maximum retention period of event data	Entity	Static	-	1-7 days
Maximum throughput units	Namespace	Static	Exceeding the throughput unit limit will cause your data to be throttled and generate a **ServerBusyException**. You can request a larger number of throughput units for a Standard tier by filing a support ticket. Additional throughput units are available in blocks of twenty on a committed purchase basis.	20

Service Bus limits

The following table lists quota information specific to Service Bus messaging. For information about pricing and other quotas for Service Bus, see the Service Bus Pricing overview.

QUOTA NAME	SCOPE	TYPE	BEHAVIOR WHEN EXCEEDED	VALUE
Maximum number of basic / standard namespaces per Azure subscription	Namespace	Static	Subsequent requests for additional basic / standard namespaces will be rejected by the portal.	100
Maximum number of premium namespaces per Azure subscription	Namespace	Static	Subsequent requests for additional premium namespaces will be rejected by the portal.	10
Queue/topic size	Entity	Defined upon creation of the queue/topic.	Incoming messages will be rejected and an exception will be received by the calling code.	1, 2, 3, 4 or 5 GB. If partitioning is enabled, the maximum queue/topic size is 80 GB.
Number of concurrent connections on a namespace	Namespace	Static	Subsequent requests for additional connections will be rejected and an exception will be received by the calling code. REST operations do not count towards concurrent TCP connections.	NetMessaging: 1,000 AMQP: 5,000
Number of concurrent connections on a queue/topic/subscription entity	Entity	Static	Subsequent requests for additional connections will be rejected and an exception will be received by the calling code. REST operations do not count towards concurrent TCP connections.	Capped by the limit of concurrent connections per namespace.
Number of concurrent receive requests on a queue/topic/subscription entity	Entity	Static	Subsequent receive requests will be rejected and an exception will be received by the calling code. This quota applies to the combined number of concurrent receive operations across all subscriptions on a topic.	5,000

QUOTA NAME	SCOPE	TYPE	BEHAVIOR WHEN EXCEEDED	VALUE
Number of topics/queues per service namespace	System-wide	Static	Subsequent requests for creation of a new topic or queue on the service namespace will be rejected. As a result, if configured through the Azure portal, an error message will be generated. If called from the management API, an exception will be received by the calling code.	10,000 The total number of topics plus queues in a service namespace must be less than or equal to 10,000. This is not applicable to Premium as all entities are partitioned.
Number of partitioned topics/queues per service namespace	System-wide	Static	Subsequent requests for creation of a new partitioned topic or queue on the service namespace will be rejected. As a result, if configured through the Azure portal, an error message will be generated. If called from the management API, a **QuotaExceededException** exception will be received by the calling code.	Basic and Standard Tiers - 100 Premium - 1,000 Each partitioned queue or topic counts towards the quota of 10,000 entities per namespace.
Maximum size of any messaging entity path: queue or topic	Entity	Static	-	260 characters
Maximum size of any messaging entity name: namespace, subscription, or subscription rule	Entity	Static	-	50 characters

QUOTA NAME	SCOPE	TYPE	BEHAVIOR WHEN EXCEEDED	VALUE
Message size for a queue/topic/subscription entity	System-wide	Static	Incoming messages that exceed these quotas will be rejected and an exception will be received by the calling code.	Maximum message size: 256KB (Standard tier) / 1MB (Premium tier). **Note** Due to system overhead, this limit is usually slightly less. Maximum header size: 64KB Maximum number of header properties in property bag: **byte/int.MaxValue** Maximum size of property in property bag: No explicit limit. Limited by maximum header size.
Message property size for a queue/topic/subscription entity	System-wide	Static	A **SerializationException** exception is generated.	Maximum message property size for each property is 32K. Cumulative size of all properties cannot exceed 64K. This applies to the entire header of the BrokeredMessage, which has both user properties as well as system properties (such as SequenceNumber, Label, MessageId, and so on).
Number of subscriptions per topic	System-wide	Static	Subsequent requests for creating additional subscriptions for the topic will be rejected. As a result, if configured through the portal, an error message will be shown. If called from the management API an exception will be received by the calling code.	2,000

QUOTA NAME	SCOPE	TYPE	BEHAVIOR WHEN EXCEEDED	VALUE
Number of SQL filters per topic	System-wide	Static	Subsequent requests for creation of additional filters on the topic will be rejected and an exception will be received by the calling code.	2,000
Number of correlation filters per topic	System-wide	Static	Subsequent requests for creation of additional filters on the topic will be rejected and an exception will be received by the calling code.	100,000
Size of SQL filters/actions	System-wide	Static	Subsequent requests for creation of additional filters will be rejected and an exception will be received by the calling code.	Maximum length of filter condition string: 1024 (1K). Maximum length of rule action string: 1024 (1K). Maximum number of expressions per rule action: 32.
Number of SharedAccessAuthorizationRule rules per namespace, queue, or topic	Entity, namespace	Static	Subsequent requests for creation of additional rules will be rejected and an exception will be received by the calling code.	Maximum number of rules: 12. Rules that are configured on a Service Bus namespace apply to all queues and topics in that namespace.

IoT Hub limits

The following table lists the limits associated with the different service tiers (S1, S2, S3, F1). For information about the cost of each *unit* in each tier, see IoT Hub Pricing.

RESOURCE	S1 STANDARD	S2 STANDARD	S3 STANDARD	F1 FREE
Messages/day	400,000	6,000,000	300,000,000	8,000
Maximum units	200	200	200	1

> **NOTE**
>
> If you anticipate using more than 200 units with an S1 or S2 or S3 tier hub, please contact Microsoft support.

The following table lists the limits that apply to IoT Hub resources:

RESOURCE	LIMIT
Maximum paid IoT hubs per Azure subscription	10
Maximum free IoT hubs per Azure subscription	1
Maximum number of device identities returned in a single call	1000
IoT Hub message maximum retention for device-to-cloud messages	7 days
Maximum size of device-to-cloud message	256 KB
Maximum size of device-to-cloud batch	256 KB
Maximum messages in device-to-cloud batch	500
Maximum size of cloud-to-device message	64 KB
Maximum TTL for cloud-to-device messages	2 days
Maximum delivery count for cloud-to-device messages	100
Maximum delivery count for feedback messages in response to a cloud-to-device message	100
Maximum TTL for feedback messages in response to a cloud-to-device message	2 days

NOTE

If you need more than 10 paid IoT hubs in an Azure subscription, please contact Microsoft support.

The IoT Hub service throttles requests when the following quotas are exceeded:

THROTTLE	PER-HUB VALUE
Identity registry operations (create, retrieve, list, update, delete), individual or bulk import/export	5000/min/unit (for S3) 100/min/unit (for S1 and S2).
Device connections	6000/sec/unit (for S3), 120/sec/unit (for S2), 12/sec/unit (for S1). Minimum of 100/sec.
Device-to-cloud sends	6000/sec/unit (for S3), 120/sec/unit (for S2), 12/sec/unit (for S1). Minimum of 100/sec.
Cloud-to-device sends	5000/min/unit (for S3), 100/min/unit (for S1 and S2).
Cloud-to-device receives	50000/min/unit (for S3), 1000/min/unit (for S1 and S2).

THROTTLE	PER-HUB VALUE
File upload operations	5000 file upload notifications/min/unit (for S3), 100 file upload notifications/min/unit (for S1 and S2). 10000 SAS URIs can be out for an Azure Storage account at one time. 10 SAS URIs/device can be out at one time.

Data Factory limits

Data factory is a multi-tenant service that has the following default limits in place to make sure customer subscriptions are protected from each other's workloads. Many of the limits can be easily raised for your subscription up to the maximum limit by contacting support.

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
data factories in an Azure subscription	50	Contact support
pipelines within a data factory	2500	Contact support
datasets within a data factory	5000	Contact support
concurrent slices per dataset	10	10
bytes per object for pipeline objects [1]	200 KB	200 KB
bytes per object for dataset and linked service objects [1]	100 KB	2000 KB
HDInsight on-demand cluster cores within a subscription [2]	60	Contact support
Cloud data movement unit [3]	8	Contact support
Retry count for pipeline activity runs	1000	MaxInt (32 bit)

[1] Pipeline, dataset, and linked service objects represent a logical grouping of your workload. Limits for these objects do not relate to amount of data you can move and process with the Azure Data Factory service. Data factory is designed to scale to handle petabytes of data.

[2] On-demand HDInsight cores are allocated out of the subscription that contains the data factory. As a result, the above limit is the Data Factory enforced core limit for on-demand HDInsight cores and is different from the core limit associated with your Azure subscription.

[3] Cloud data movement unit (DMU) is being used in a cloud-to-cloud copy operation. It is a measure that represents the power (a combination of CPU, memory, and network resource allocation) of a single unit in Data Factory. You can achieve higher copy throughput by leveraging more DMUs for some scenarios. Refer to Cloud data movement units section on details.

RESOURCE	DEFAULT LOWER LIMIT	MINIMUM LIMIT
Scheduling interval	15 minutes	15 minutes
Interval between retry attempts	1 second	1 second

RESOURCE	DEFAULT LOWER LIMIT	MINIMUM LIMIT
Retry timeout value	1 second	1 second

Web service call limits

Azure Resource Manager has limits for API calls. You can make API calls at a rate within the Azure Resource Manager API limits.

Data Lake Analytics Limits

Data Lake Analytics makes the complex task of managing distributed infrastructure and complex code easy. It dynamically provisions resources and lets you do analytics on exabytes of data. When the job completes, it winds down resources automatically, and you pay only for the processing power used. As you increase or decrease the size of data stored or the amount of compute used, you don’t have to rewrite code. Many of the default limits can be easily raised for your subscription by contacting support.

RESOURCE	DEFAULT LIMIT	COMMENTS
max concurrent jobs	3	
Max parallelism per account	60	Use any combination of up to a maximum of 60 units of parallelism across three jobs.

Stream Analytics limits

LIMIT IDENTIFIER	LIMIT	COMMENTS
Maximum number of Streaming Units per subscription per region	50	A request to increase streaming units for your subscription beyond 50 can be made by contacting Microsoft Support.
Maximum throughput of a Streaming Unit	1MB/s*	Maximum throughput per SU depends on the scenario. Actual throughput may be lower and depends upon query complexity and partitioning. Further details can be found in the Scale Azure Stream Analytics jobs to increase throughput article.
Maximum number of inputs per job	60	There is a hard limit of 60 inputs per Stream Analytics job.
Maximum number of outputs per job	60	There is a hard limit of 60 outputs per Stream Analytics job.
Maximum number of functions per job	60	There is a hard limit of 60 functions per Stream Analytics job.
Maximum number of jobs per region	1500	Each subscription may have up to 1500 jobs per geographical region.

Active Directory limits

Here are the usage constraints and other service limits for the Azure Active Directory service.

CATEGORY	LIMITS
Directories	A single user can only be associated with a maximum of 20 Azure Active Directory directories. Examples of possible combinations: • A single user creates 20 directories. • A single user is added to 20 directories as a member. • A single user creates 10 directories and later is added by others to 10 different directories.
Objects	• A maximum of 500,000 objects can be used in a single directory by users of the Free edition of Azure Active Directory. • A non-admin user can create no more than 250 objects.
Schema extensions	• String type extensions can have maximum of 256 characters. • Binary type extensions are limited to 256 bytes. • 100 extension values (across ALL types and ALL applications) can be written to any single object. • Only “User”, “Group”, “TenantDetail”, “Device”, “Application” and “ServicePrincipal” entities can be extended with “String” type or “Binary” type single-valued attributes. • Schema extensions are available only in Graph API-version 1.21-preview. The application must be granted write access to register an extension.
Applications	A maximum of 10 users can be owners of a single application.
Groups	• A maximum of 10 users can be owners of a single group. • Any number of objects can be members of a single group in Azure Active Directory. • The number of members in a group you can synchronize from your on-premises Active Directory to Azure Active Directory is limited to 15K members, using Azure Active Directory Directory Synchronization (DirSync). • The number of members in a group you can synchronize from your on-premises Active Directory to Azure Active Directory using Azure AD Connect is limited to 50K members.
Access Panel	• There is no limit to the number of applications that can be seen in the Access Panel per end user, for users assigned licenses for Azure AD Premium or the Enterprise Mobility Suite. • A maximum of 10 app tiles (examples: Box, Salesforce, or Dropbox) can be seen in the Access Panel for each end user for users assigned licenses for Free or Azure AD Basic editions of Azure Active Directory. This limit does not apply to Administrator accounts.

CATEGORY	LIMITS
Reports	A maximum of 1,000 rows can be viewed or downloaded in any report. Any additional data is truncated.

Azure RemoteApp limits

RESOURCE	DEFAULT LIMIT
Collections per user	1
Published apps per collection	100
Trial collection duration	30 days
Trial collections	2 per subscription
Users per trial collection	10
Trial template images	25
Paid collections	3
Paid template images	25
Users - basic tier*	400 (default)/ 800 (maximum)
Users - standard tier*	250 (default)/ 500 (maximum)
Users- premium tier	100 default.
Users - premium plus tier	50 default.
Concurrent connections across all collections in a subscription	5000
User data storage (UPD) per user per collection	50 GB
Idle timeout	4 hours
Disconnected timeout	4 hours

*User limits in basic and standard tiers cannot be increased beyond the maximum limit listed above.

The number of users is determined by the number of VMs used for your collection:

- Basic = 16 users per VM
- Standard = 10 users per VM
- Premium = 4 users per VM
- Premium plus = 2 users per VM

StorSimple System limits

LIMIT IDENTIFIER	LIMIT	COMMENTS
Maximum number of storage account credentials	64	
Maximum number of volume containers	64	
Maximum number of volumes	255	
Maximum number of schedules per bandwidth template	168	A schedule for every hour, every day of the week (24*7).
Maximum size of a tiered volume on physical devices	64 TB for 8100 and 8600	8100 and 8600 are physical devices.
Maximum size of a tiered volume on virtual devices in Azure	30 TB for 8010 64 TB for 8020	8010 and 8020 are virtual devices in Azure that use Standard Storage and Premium Storage respectively.
Maximum size of a locally pinned volume on physical devices	9 TB for 8100 24 TB for 8600	8100 and 8600 are physical devices.
Maximum number of iSCSI connections	512	
Maximum number of iSCSI connections from initiators	512	
Maximum number of access control records per device	64	
Maximum number of volumes per backup policy	24	
Maximum number of backups retained per backup policy	64	
Maximum number of schedules per backup policy	10	
Maximum number of snapshots of any type that can be retained per volume	256	This includes local snapshots and cloud snapshots.
Maximum number of snapshots that can be present in any device	10,000	
Maximum number of volumes that can be processed in parallel for backup, restore, or clone	16	• If there are more than 16 volumes, they will be processed sequentially as processing slots become available. • New backups of a cloned or a restored tiered volume cannot occur until the operation is finished. However, for a local volume, backups are allowed after the volume is online.

LIMIT IDENTIFIER	LIMIT	COMMENTS
Restore and clone recover time for tiered volumes	< 2 minutes	• The volume is made available within 2 minutes of restore or clone operation, regardless of the volume size. • The volume performance may initially be slower than normal as most of the data and metadata still resides in the cloud. Performance may increase as data flows from the cloud to the StorSimple device. • The total time to download metadata depends on the allocated volume size. Metadata is automatically brought into the device in the background at the rate of 5 minutes per TB of allocated volume data. This rate may be affected by Internet bandwidth to the cloud. • The restore or clone operation is complete when all the metadata is on the device. • Backup operations cannot be performed until the restore or clone operation is fully complete.

LIMIT IDENTIFIER	LIMIT	COMMENTS
Restore recover time for locally pinned volumes	< 2 minutes	• The volume is made available within 2 minutes of the restore operation, regardless of the volume size. • The volume performance may initially be slower than normal as most of the data and metadata still resides in the cloud. Performance may increase as data flows from the cloud to the StorSimple device. • The total time to download metadata depends on the allocated volume size. Metadata is automatically brought into the device in the background at the rate of 5 minutes per TB of allocated volume data. This rate may be affected by Internet bandwidth to the cloud. • Unlike tiered volumes, in the case of locally pinned volumes, the volume data is also downloaded locally on the device. The restore operation is complete when all the volume data has been brought to the device. • The restore operations may be long and the total time to complete the restore will depend on the size of the provisioned local volume, your Internet bandwidth and the existing data on the device. Backup operations on the locally pinned volume are allowed while the restore operation is in progress.
Thin-restore availability	Last failover	
Maximum client read/write throughput (when served from the SSD tier)*	920/720 MB/s with a single 10GbE network interface	Up to 2x with MPIO and two network interfaces.
Maximum client read/write throughput (when served from the HDD tier)*	120/250 MB/s	
Maximum client read/write throughput (when served from the cloud tier)*	11/41 MB/s	Read throughput depends on clients generating and maintaining sufficient I/O queue depth.

* Maximum throughput per I/O type was measured with 100 percent read and 100 percent write scenarios. Actual throughput may be lower and depends on I/O mix and network conditions.

Operational Insights limits

The following limits apply to Operational Insights subscriptions.

	FREE	STANDARD	PREMIUM
Daily data transfer limit	500 MB[1]	None	None
Data retention period	7 days	1 month	12 months
Data storage limit	500 MB * 7 days = 3.5 GB	unlimited	unlimited

[1]When customers reach their 500MB daily data transfer limit, data analysis stops and resumes at the start of the next day. A day is based on UTC.

Backup limits

The following limits apply to Azure Backup.

LIMIT IDENTIFIER	DEFAULT LIMIT
Number of servers/machines that can be registered against each vault	50 for Windows Server/Client/SCDPM 200 for IaaS VMs
Size of a data source for data stored in Azure vault storage	54400 GB max[1]
Number of backup vaults that can be created in each Azure subscription	25(Backup vaults) 25 Recovery Services vault per region
Number of times backup can be scheduled per day	3 per day for Windows Server/Client 2 per day for SCDPM Once a day for IaaS VMs
Data disks attached to an Azure virtual machine for backup	16

- [1]The 54400 GB limit does not apply to IaaS VM backup.

Site Recovery limits

The following limits apply to Azure Site Recovery:

LIMIT IDENTIFIER	DEFAULT LIMIT
Number of vaults per subscription	25
Number of servers per Azure vault	250
Number of protection groups per Azure vault	No limit
Number of recovery plans per Azure vault	No limit
Number of servers per protection group	No limit
Number of servers per recovery plan	50

Application Insights limits

There are some limits on the number of metrics and events per application (that is, per instrumentation key).

Limits depend on the pricing plan that you choose.

RESOURCE	DEFAULT LIMIT	NOTE
Total data per day	500 GB	You can reduce by setting a cap. If you need more, mail AIDataCap@microsoft.com
Free data per month (Basic price plan)	1 GB	Additional data charged per GB
Throttling	16 k events/second	Measured over a minute.
Data retention	90 days	for Search, Analytics and Metrics explorer
Availability multi-step test detailed results retention	90 days	Detailed results of each step
Property and Metric[2] name count	200	
Property and metric name length	150	
Property value string length	8192	
Distinct values for properties[3,4]	100	>100 => can't use property as filter in Metrics Explorer
Trace and Exception message length	10000	
Availability tests count per app	10	

1. All these numbers are per instrumentation key.
2. Metric names are defined both in TrackMetric and in the measurement parameter of other Track*() calls. Metric names are global per instrumentation key.
3. Properties can be used for filtering and group-by only while they have less than 100 unique values for each property. After the number of unique values exceeds 100, you can still search the property, but no longer use it for filters or group-by.
4. Standard properties such as Request Name and Page URL are limited to 1000 unique values per week. After 1000 unique values, additional values are marked as "Other values." The original values can still be used for full text search and filtering.

About pricing and quotas in Application Insights

API Management limits

RESOURCE	LIMIT
API Calls (per unit of scale)	32 million per day[1]
Data transfer (per unit of scale)	161 GB per day[1]
Cache	5 GB[1]
Units of scale	Unlimited[1]

RESOURCE	LIMIT
Azure Active Directory Integration	Unlimited User Accounts[1]

[1]API Management limits are different for each pricing tier. To see the pricing tiers and their associated limits and scaling options, see API Management Pricing.

Azure Redis Cache limits

RESOURCE	LIMIT
Cache size	530 GB (contact us for more)
Databases	64
Max connected clients	40,000
Redis Cache replicas (for high availability)	1
Shards in a premium cache with clustering	10

Azure Redis Cache limits and sizes are different for each pricing tier. To see the pricing tiers and their associated sizes, see Azure Redis Cache Pricing.

For more information on Azure Redis Cache configuration limits, see Default Redis server configuration.

Because configuration and management of Azure Redis Cache instances is done by Microsoft, not all Redis commands are supported in Azure Redis Cache. For more information, see [Redis commands not supported in Azure Redis Cache]((redis-cache/cache-configure.md#redis-commands-not-supported-in-azure-redis-cache).

Key Vault limits

TRANSACTIONS TYPE	MAX TRANSACTIONS ALLOWED IN 10 SECONDS, PER VAULT PER REGION[1]
HSM- CREATE KEY	5
HSM- other transactions	1000
Soft-key CREATE KEY	10
Soft-key other transactions	1500
All secrets, vault related transactions	2000

[1] There is a subscription-wide limit for all transaction types, that is 5x per key vault limit. For example, HSM- other transactions per subscription are limited to 5000 transactions in 10 seconds per subscription.

Multi-Factor Authentication

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Max number of Trusted IP addresses/ranges per subscription[1]	0	12

RESOURCE	DEFAULT LIMIT	MAXIMUM LIMIT
Remember my devices - number of days	14	60
Max number of app passwords?	0	No Limit
Allow **X** attempts during MFA call	1	99
Two-way Text message Timeout Seconds	60	600
Default one-time bypass seconds	300	1800
Lock user account after **X** consecutive MFA denials	Not Set	99
Reset account lockout counter after **X** minutes	Not Set	9999
Unlock account after **X** minutes	Not Set	9999

[1] This is expected to increase in the future.

Automation limits

RESOURCE	MAXIMUM LIMIT
Max number of new jobs that can be submitted every 30 seconds per Automation Account (non Scheduled jobs)	100
Max number of concurrent running jobs at the same instance of time per Automation Account (non Scheduled jobs)	200
Max number of modules that can be imported every 30 seconds per Automation Account	5
Max size of a Module	100 MB
Job Run Time - Free tier	500 minutes per subscription per calendar month
Max amount of memory given to a job	400 MB
Max number of network sockets allowed per job	1000

SQL Database limits

For SQL Database limits, see SQL Database Resource Limits.

See also

Understanding Azure Limits and Increases

Virtual Machine and Cloud Service Sizes for Azure

Sizes for Cloud Services