Connect your on-premises network to Azure with VPN Gateway

A virtual private network (VPN) is a type of private interconnected network. VPNs use an encrypted tunnel within another network. They're typically deployed to connect two or more trusted private networks to one another over an untrusted network (typically the public Internet). Traffic is encrypted while traveling over the untrusted network to prevent eavesdropping or other attacks.

A VPN gateway is a type of Virtual Network Gateway. VPN gateways are deployed in Azure virtual networks and enable the following connectivity:

- Connect on-premises datacenters to Azure virtual networks through a site-to-site connection.
- Connect individual devices to Azure virtual networks through a point-to-site connection.
- Connect Azure virtual networks to other Azure virtual networks through a network-to-network connection.

Policy-based VPNs

Policy-based VPN gateways specify statically the IP address of packets that should be encrypted through each tunnel.

- Support for IKEv1 only.
- Use of *static routing*, where combinations of address prefixes from both networks control how traffic is encrypted and decrypted through the VPN tunnel. The source and destination of the tunneled networks are declared in the policy and don't need to be declared in routing tables.
- Policy-based VPNs must be used in specific scenarios that require them, such as for compatibility with legacy on-premises VPN devices.

Route-based VPNs

With route-based gateways, IPSec tunnels are modeled as a network interface or VTI (virtual tunnel interface). IP routing (static routes or dynamic routing protocols) decide across which one of these tunnel interfaces to send each packet. Route-based VPNs are the preferred connection method for on-premises devices, since they are more resilient to topology changes such as the creation of new subnets

Use a route-based VPN gateway for:

- Connections between virtual networks
- Point-to-site connections
- Multisite connections
- Coexistence with an Azure ExpressRoute gateway

Key features of route-based VPNs gateways in Azure include:

- Supports IKEv2.
- Uses any-to-any (wildcard) traffic selectors.
- Can use *dynamic routing protocols*, where routing/forwarding tables direct traffic to different IPSec tunnels. In this case, the source and destination networks are not statically defined as they are in policy-based VPNs or even in route-based VPNs with static routing. Instead, data packets are encrypted based on network routing tables that are created dynamically using routing protocols such as BGP (Border Gateway Protocol).

VPN gateway sizes

SKU	Site-to-site/VNet-to-VNet tunnels	Aggregate throughput benchmark	Border Gateway Protocol (BGP) support
Basic*	Maximum: 10	100 Mbps	Not supported
VpnGw1/Az	Maximum: 30	650 Mbps	Supported
VpnGw2/Az	Maximum: 30	1 Gbps	Supported
VpnGw3/Az	Maximum: 30	1.25 Gbps	Supported

High availability scenarios

There are several ways to ensure you have a fault-tolerant configuration.

Active/standby

By default, VPN gateways are deployed as two instances in an *active/standby* configuration, even if you only see one VPN gateway resource in Azure. Connections are interrupted during a failover, but they're typically restored within a few seconds for planned maintenance and within 90 seconds for unplanned disruptions.

Active/active

In this configuration, you assign a unique public IP address to each instance. You then create separate tunnels from the on-premises device to each IP address. You can extend the high availability by deploying an additional VPN device on-premises.

Enhance your service availability and data locality by using Azure Traffic Manager

A Traffic Manager doesn't see the traffic that passes between the clients and the service; it just gives clients the IP address of where they need to go.

Traffic Manager endpoints

- **Azure endpoints** are used for services hosted in Azure. These can be services like Azure App Service, as well as public IP resources that are associated with load balancers or virtual machines.
- **External endpoints** are used for IPv4/IPv6 addresses, FQDNs, or for services hosted outside Azure that can either be 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.

Weighted routing

Choose weighted when you want to distribute traffic across a set of endpoints, either evenly, or based on different weights. The weight is an integer from 1 to 1,000. 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.

Performance routing

If you have endpoints in different geographic locations, you can use performance routing to send users to the endpoint that has the best performance for the user. To choose the best endpoint to use, this routing method uses an internet latency table, which actively tracks network latencies to the endpoints from locations around the globe. When a user makes a request, Traffic Manager returns the best performing endpoint based on the location of the request.

Geographic routing

With the geographic routing method, users are directed to specific endpoints based on where their DNS query originates. Using this method enables you to geo-fence content to specific user regions. For example, European users can be directed to an endpoint in Europe that has specific terms and conditions for regional compliance.

Priority routing

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 isn't 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 endpoint, and so on

You don't have to do anything more than configure a Traffic Manager profile and select **performance** as the routing method. Endpoints don't need to be prioritized, Traffic Manager will route all the traffic automatically to the fastest responding server.

Improve application scalability and resiliency by using Azure Load Balancer

With Azure Load Balancer, you can spread user requests across multiple virtual machines or other services. That way, you can scale the app to larger sizes than a single virtual machine can support, and you ensure that users get service, even when a virtual machine fails.

Distribute traffic with Azure Load Balancer

Azure Load Balancer is a service you can use to distribute traffic across multiple virtual machines. Use Load Balancer to scale applications and create high availability for your virtual machines and services. Load balancers use a hash-based distribution algorithm. By default, a five-tuple hash is used to map traffic to available servers. The hash is made from the following elements:

- **Source IP**: The IP address of the requesting client.
- **Source port**: The port of the requesting client.
- **Destination IP**: The destination IP of the request.
- **Destination port**: The destination port of the request.
- **Protocol type**: The specified protocol type, TCP or UDP.

Load Balancer supports inbound and outbound scenarios, provides low latency and high throughput, and scales up to millions of flows for all TCP and UDP applications. Load balancers aren't physical instances. Load balancer objects are used to express how Azure configures its infrastructure to meet your requirements. To achieve high availability with Load Balancer, you can choose to use availability sets and availability zones to ensure that virtual machines are always available:

Configuration	SLA	Information
Availability set	99.95%	Protection from hardware failures within datacenters
Availability zone	99.99%	Protection from entire datacenter failure

Availability sets

An availability set is a logical grouping that you use to isolate virtual machine resources from each other when they're deployed. Azure ensures that the virtual machines you put in an availability set run across multiple physical servers, compute racks, storage units, and network switches. If there's a hardware or software failure, only a subset of your virtual machines is affected. Your overall solution stays operational. Availability sets are essential for building reliable cloud solutions.

Availability zones

An availability zone offers groups of one or more datacenters that have independent power, cooling, and networking. The virtual machines in an availability zone are placed in different physical locations within the same region. Use this architecture when you want to ensure that, when an entire datacenter fails, you can continue to serve users.

Select the right Load Balancer product

Two products are available when you create a load balancer in Azure: basic load balancers and standard load balancers.

Basic load balancers allow:

- Port forwarding
- Automatic reconfiguration
- Health probes
- Outbound connections through source network address translation (SNAT)
- Diagnostics through Azure Log Analytics for public-facing load balancers

Basic load balancers can be used only with availability sets.

Standard load balancers support all of the basic features. They also allow:

- HTTPS health probes
- Availability zones
- Diagnostics through Azure Monitor, for multidimensional metrics
- High availability (HA) ports
- Outbound rules
- A guaranteed SLA (99.99% for two or more virtual machines)

Internal and external load balancers

An external load balancer operates by distributing client traffic across multiple virtual machines. An external load balancer permits traffic from the internet.

An internal load balancer distributes a load from internal Azure resources to other Azure resources. No traffic is allowed from internet sources.

Load Balancer and Remote Desktop Gateway

Remote Desktop Gateway is a Windows service that you can use to enable clients on the internet to make Remote Desktop Protocol (RDP) connections through firewalls to Remote Desktop servers on your private network. The default five-tuple hash in Load Balancer is incompatible with this service. If you want to use Load Balancer with your Remote Desktop servers, use source IP affinity.

Load Balancer and media upload

Another use case for source IP affinity is media upload. In many implementations, a client initiates a session through a TCP protocol and connects to a destination IP address. This connection remains open throughout the upload to monitor progress, but the file is uploaded through a separate UDP protocol.

With the five-tuple hash, the load balancer likely will send the TCP and UDP connections to different destination IP addresses and the upload won't finish successfully. Use source IP affinity to resolve this issue.

Configure an internal load balancer

You can configure an internal load balancer in almost the same way as an external load balancer, but with these differences:

- When you create the load balancer, for the **Type** value, select **Internal**. When you select this setting, the front-end IP address of the load balancer isn't exposed to the internet.
- Assign a private IP address instead of a public IP address for the front end of the load balancer.
- Place the load balancer in the protected virtual network that contains the virtual machines you want to handle the requests.

Route traffic with Application Gateway

Application Gateway manages the requests that client applications can send to a web app. Application Gateway routes traffic to a pool of web servers based on the URL of a request. This is known as *application layer routing*. The pool of web servers can be Azure virtual machines, Azure virtual machine scale sets, Azure App Service, and even on-premises servers.

How Application Gateway routes requests

There are two primary methods of routing traffic, path-based routing and multiple site hosting. Let's take a look at the capabilities for each.

Path-based routing

Path-based routing enables you to send requests with different paths in the URL to a different pool of back-end servers. For example, you could direct requests with the path `/video/*` to a back-end pool containing servers that are optimized to handle video streaming, and direct `/images/*` requests to a pool of servers that handle image retrieval.

Multiple site hosting

Multiple site hosting enables you to configure more than one web application on the same application gateway instance. In a multi-site configuration, you register multiple DNS names (CNAMEs) for the IP address of the Application Gateway, specifying the name of each site. Application Gateway uses separate listeners to wait for requests for each site. Each listener passes the request to a different rule, which can route the requests to servers in a different back-end pool. For example, you could configure Application Gateway to direct all requests for `http://contoso.com` to servers in one back-end pool, and requests for `http://fabrikam.com` to another back-end pool.

Multi-site configurations are useful for supporting multi-tenant applications, where each tenant has its own set of virtual machines or other resources hosting a web application.

Other routing capabilities

Along with path-based routing and multiple site hosting, there are a few additional capabilities when routing with Application Gateway.

- **Redirection** - Redirection can be used to another site, or from HTTP to HTTPS.

- **Rewrite HTTP headers** - HTTP headers allow the client and server to pass additional information with the request or the response.
- **Custom error pages** - Application Gateway allows you to create custom error pages instead of displaying default error pages. You can use your own branding and layout using a custom error page.

Load balancing in Application Gateway

Application Gateway will automatically load balance requests sent to the servers in each back-end pool using a round-robin mechanism. However, you can configure session stickiness.

Load-balancing works with the OSI Layer 7 routing implemented by Application Gateway routing, which means that it load balances requests based on the routing parameters (host names and paths) used by the Application Gateway rules. In comparison, other load balancers, such as Azure Load Balancer, function at the OSI Layer 4 level, and distribute traffic based on the IP address of the target of a request.

Operating at OSI Layer 7 enables load balancing to take advantage of the other features that Application Gateway provides. These features include:

- Support for the HTTP, HTTPS, HTTP/2 and WebSocket protocols.
- A web application firewall to protect against web application vulnerabilities.
- End-to-end request encryption.
- Autoscaling, to dynamically adjust capacity as your web traffic load changes.

Application Gateway creation and configuration

Application Gateway comprises a series of components that combine to route requests to a pool of web servers and to check the health of these web servers.

Front-end IP address

Client requests are received through a *front-end IP address*. You can configure Application Gateway to have a public IP address, a private IP address, or both. Application Gateway can't have more than one public and one private IP address.

Listeners

Application Gateway uses one or more *listeners* to receive incoming requests. A listener accepts traffic arriving on a specified combination of protocol, port, host, and IP address. Each listener routes requests to a back-end pool of servers following routing rules that you specify. A listener

can be Basic or Multi-site. A Basic listener only routes a request based on the path in the URL. A Multi-site listener can also route requests using the hostname element of the URL.

Listeners also handle SSL certificates for securing your application between the user and Application Gateway.

Routing rules

A routing rule binds a listener to the back-end pools. A rule specifies how to interpret the hostname and path elements in the URL of a request, and direct the request to the appropriate back-end pool. A routing rule also has an associated set of HTTP settings. These settings indicate whether (and how) traffic is encrypted between Application Gateway and the back-end servers, and other configuration information such as:

- Protocol (HTTP or HTTPS).
- Session stickiness, to pass all requests in a client session to the same web server rather than distributing them across servers with load balancing.
- Connection draining, to enable the graceful removal of servers from a back-end pool.
- Request timeout period, in seconds.
- Health probes, specifying a probe URL, time out periods, and other parameters used to determine whether a server in the back-end pool is available.

Back-end pools

A *back-end pool* references a collection of web servers. You provide the IP address of each web server and the port on which it listens for requests when configuring the pool. Each pool can specify a fixed set of virtual machines, a virtual machine scale-set, an app hosted by Azure App Services, or a collection of on-premises servers. Each back-end pool has an associated load balancer that distributes work across the pool.

Web application firewall

The web application firewall (WAF) is an optional component that handles incoming requests before they reach a listener. The web application firewall checks each request for many common threats, based on the Open Web Application Security Project (OWASP). These include:

- SQL-injection
- Cross-site scripting
- Command injection
- HTTP request smuggling
- HTTP response splitting
- Remote file inclusion

- Bots, crawlers, and scanners
- HTTP protocol violations and anomalies

Health probes

Health probes are an important part in assisting the load balancer to determine which servers are available for load balancing in a back-end pool. Application Gateway uses a health probe to send a request to a server. If the server returns an HTTP response with a status code between 200 and 399, the server is deemed healthy.

If you don't configure a health probe, Application Gateway creates a default probe that waits for 30 seconds before deciding that a server is unavailable.

Application Gateway network requirements

Application Gateway requires a virtual network in which to operate. You must create this virtual network and a dedicated subnet before setting up Application Gateway.

Identify routing capabilities of an Azure virtual network

To control traffic flow within your virtual network, you must learn the purpose and benefits of custom routes. You must also learn how to configure the routes to direct traffic flow through a network virtual appliance (NVA).

Azure routing

Network traffic in Azure is automatically routed across Azure subnets, virtual networks, and on-premises networks. This routing is controlled by system routes, which are assigned by default to each subnet in a virtual network.

You can't create or delete system routes. But you can override the system routes by adding custom routes to control traffic flow to the next hop.

Every subnet has the following default system routes:

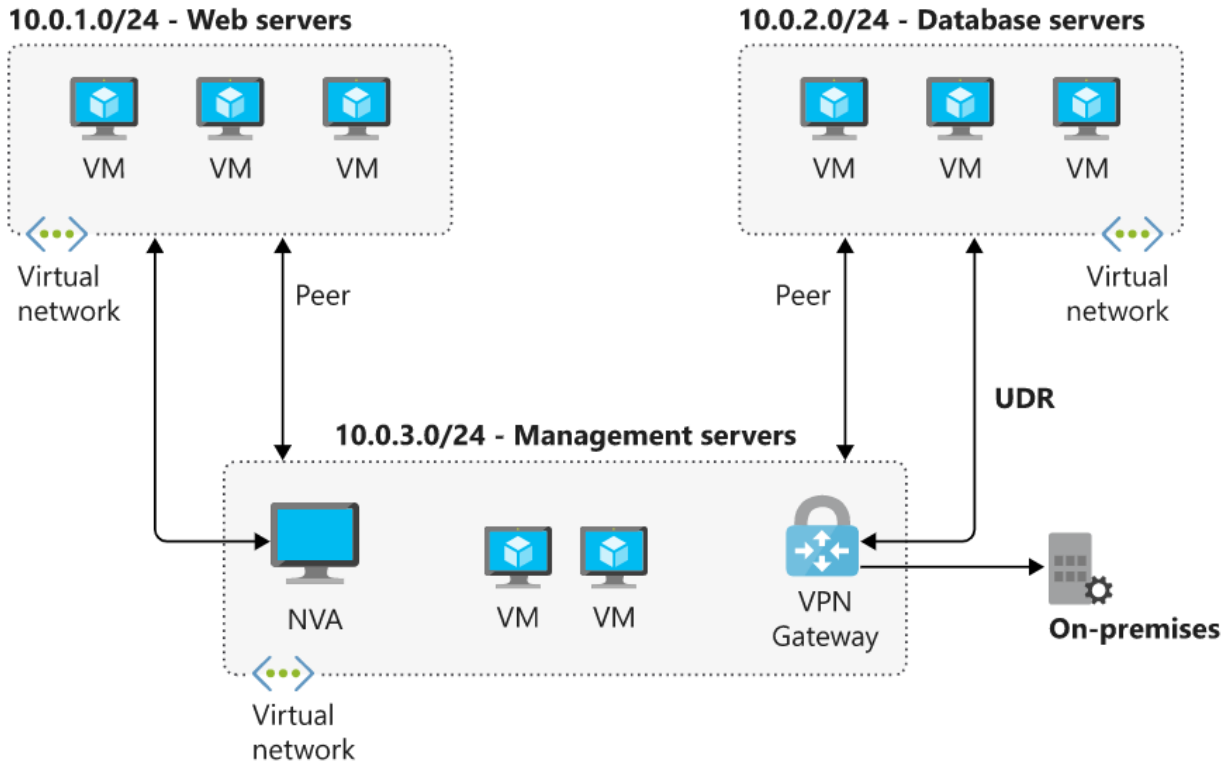
AZURE ROUTING	
Address prefix	**Next hop type**
Unique to the virtual network	Virtual network
0.0.0.0/0	Internet
10.0.0.0/8	None
172.16.0.0/12	None
192.168.0.0/16	None
100.64.0.0/10	None

The **Next hop type** column shows the network path taken by traffic sent to each address prefix. The path can be one of the following hop types:

- **Virtual network**: A route is created in the address prefix. The prefix represents each address range created at the virtual-network level. If multiple address ranges are specified, multiple routes are created for each address range.
- **Internet**: The default system route 0.0.0.0/0 routes any address range to the internet, unless you override Azure's default route with a custom route.
- **None**: Any traffic routed to this hop type is dropped and doesn't get routed outside the subnet. By default, the following IPv4 private-address prefixes are created: 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16. The prefix 100.64.0.0/10 for a shared address space is also added. None of these address ranges are globally routable.

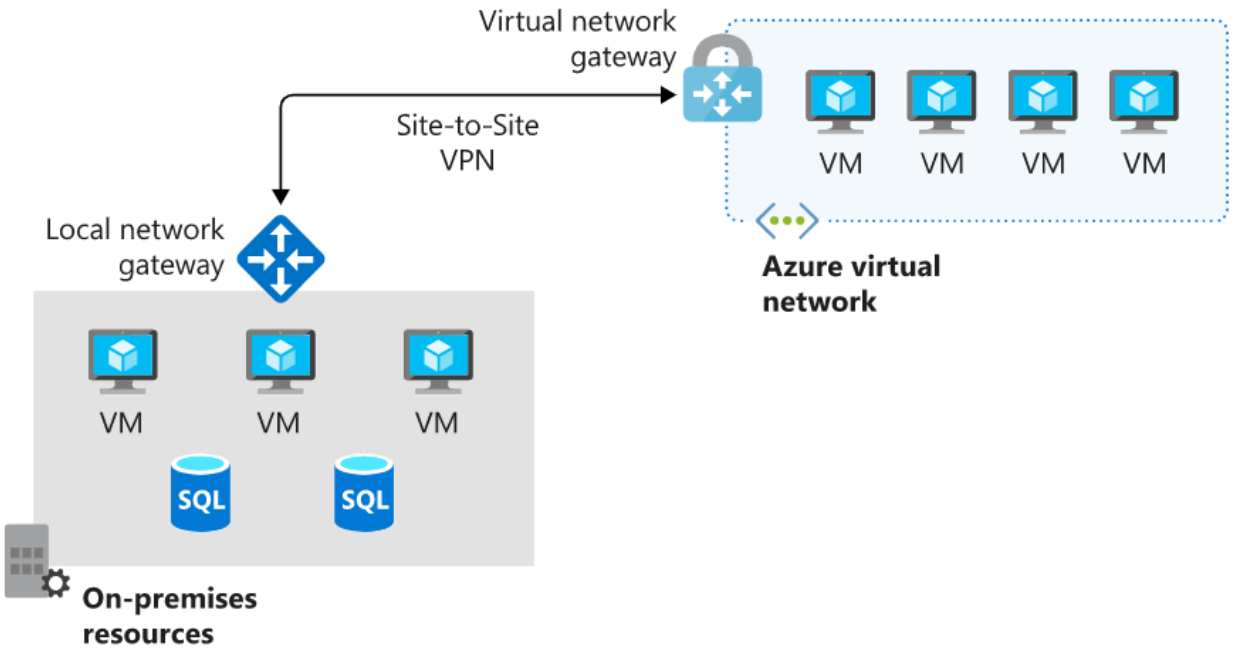
Virtual network peering and service chaining

Virtual network peering and service chaining let virtual networks within Azure be connected to one another. With this connection, virtual machines can communicate with each other within the same region or across regions. This communication in turn creates additional routes within the default route table. Service chaining lets you override these routes by creating user-defined routes between peered networks.



Virtual network gateway

Use a virtual network gateway to send encrypted traffic between Azure and on-premises over the internet and to send encrypted traffic between Azure networks. A virtual network gateway contains routing tables and gateway services.



Virtual network service endpoint

Virtual network endpoints extend your private address space in Azure by providing a direct connection to your Azure resources. This connection restricts the flow of traffic: your Azure virtual machines can access your storage account directly from the private address space and deny access from a public virtual machine. As you enable service endpoints, Azure creates routes in the route table to direct this traffic.

Custom routes

System routes might make it easy for you to quickly get your environment up and running. But there are many scenarios in which you'll want to more closely control the traffic flow within your network. For example, you might want to route traffic through an NVA or through a firewall from partners and others. This control is possible with custom routes.

You have two options for implementing custom routes: create a user-defined route or use Border Gateway Protocol (BGP) to exchange routes between Azure and on-premises networks.

User-defined routes

You use a user-defined route to override the default system routes so that traffic can be routed through firewalls or NVAs.

For example, you might have a network with two subnets and want to add a virtual machine in the perimeter network to be used as a firewall. You create a user-defined route so that traffic passes through the firewall and doesn't go directly between the subnets.

When creating user-defined routes, you can specify these next hop types:

- **Virtual appliance**: A virtual appliance is typically a firewall device used to analyze or filter traffic that is entering or leaving your network. You can specify the private IP address of a NIC attached to a virtual machine so that IP forwarding can be enabled. Or you can provide the private IP address of an internal load balancer.
- **Virtual network gateway**: Use to indicate when you want routes for a specific address to be routed to a virtual network gateway. The virtual network gateway is specified as a VPN for the next hop type.
- **Virtual network**: Use to override the default system route within a virtual network.
- **Internet**: Use to route traffic to a specified address prefix that is routed to the internet.
- **None**: Use to drop traffic sent to a specified address prefix.

With user-defined routes, you can't specify the next hop type **VirtualNetworkServiceEndpoint**, which indicates virtual network peering.

Border gateway protocol

A network gateway in your on-premises network can exchange routes with a virtual network gateway in Azure by using BGP. BGP is the standard routing protocol that is normally used to exchange routing and information among two or more networks. BGP is used to transfer data and information between different host gateways like on the internet or between autonomous systems.

You typically use BGP to advertise on-premises routes to Azure when you're connected to an Azure datacenter through Azure ExpressRoute. You can also configure BGP if you connect to an Azure virtual network by using a VPN site-to-site connection.

Route selection and priority

If multiple routes are available in a route table, Azure uses the route with the longest prefix match. For example, if a message is sent to the IP address 10.0.0.2, but two routes

are available with the 10.0.0.0/16 and 10.0.0.0/24 prefixes, Azure selects the route with the 10.0.0.0/24 prefix because it's more specific.

The longer the route prefix, the shorter the list of IP addresses available through that prefix. By using longer prefixes, the routing algorithm can select the intended address more quickly.

You can't configure multiple user-defined routes with the same address prefix.

If multiple routes share the same address prefix, Azure selects the route based on its type in the following order of priority:

1. User-defined routes
2. BGP routes
3. System routes

What is an NVA?

A network virtual appliance (NVA) is a virtual appliance that consists of various layers like:

- a firewall
- a WAN optimizer
- application-delivery controllers
- routers
- load balancers
- IDS/IPS
- proxies

You can deploy NVAs chosen from providers in Azure Marketplace. You can use an NVA to filter traffic inbound to a virtual network, to block malicious requests, and to block requests made from unexpected resources.

User-defined routes

For most environments, the default system routes already defined by Azure are enough to get the environments up and running. But in certain cases you should create a routing table and add custom routes. Examples include:

- Access to the internet via on-premises network using forced tunneling.
- Using virtual appliances to control traffic flow.

You can define multiple routing tables in Azure. Each routing table is associated with one or more subnets. But each subnet is associated with only one routing table.

Network virtual appliances in a highly available architecture

If traffic is routed through an NVA, the NVA becomes a critical piece of your infrastructure. Any NVA failures will directly affect the ability of your services to communicate. It's important to include a highly available architecture in your NVA deployment.

There are several methods of achieving high availability when using NVAs. At the end of this module, you can find more information about using NVAs in highly available scenarios.

Network IP addressing and integration

To integrate resources in an Azure virtual network with resources in your on-premises network, you must understand how you can connect those resources and how to configure IP addresses.

On-premises IP addressing

A typical on-premises network design includes these components:

- Routers
- Firewalls
- Switches
- Network segmentation

There are three ranges of non-routable IP addresses that are designed for internal networks that won't be sent over internet routers:

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.31.255.255
- 192.168.0.1 to 192.168.255.255

Azure IP addressing

Azure virtual networks use private IP addresses. The ranges of private IP addresses are the same as for on-premises IP addressing.

In a typical Azure network design, we usually have these components:

- Virtual networks
- Subnets
- Network security groups
- Firewalls
- Load balancers

Basic properties of Azure virtual networks

A virtual network is your network in the cloud. You can divide your virtual network into multiple subnets. Each subnet has a portion of the IP address space that is assigned to

your virtual network. You can add, remove, expand, or shrink a subnet if there are no VMs or services deployed in it.

By default, all subnets in an Azure virtual network can communicate with each other. However, you can use a network security group to deny communication between subnets

Integrate Azure with on-premises networks

Before you start integrating Azure with on-premises networks, it's important to identify the current private IP address scheme used in the on-premises network. There can be no IP address overlap for interconnected networks.

For example, you can't use 192.168.0.0/16 on your on-premises network and use 192.168.10.0/24 on your Azure virtual network. These ranges both contain the same IP addresses and won't be able to route traffic between each other.

You can, however, have the same class range for multiple networks. For example, you can use the 10.10.0.0/16 address space for your on-premises network and the 10.20.0.0/16 address space for your Azure network because they don't overlap.

It is vital to check for overlaps when you're planning an IP address scheme. If there's an overlap of IP addresses, you can't integrate your on-premises network with your Azure network.

Public and private IP addressing in Azure

IP address types

There are two types of IP addresses that you can use in Azure:

- **Public IP addresses**
- **Private IP addresses**

Both types of IP addresses can be allocated in one of two ways:

- **Dynamic**
- **Static**

Public IP addresses

Use a public IP address for public-facing services. A public address can be either static or dynamic. A public IP address can be assigned to a VM, an internet-facing load balancer, a VPN gateway, or an application gateway.

Dynamic public IP addresses are assigned addresses that can change over the lifespan of the Azure resource. The dynamic IP address is allocated when you create or start a VM. The IP address is released when you stop or delete the VM. In each Azure region, public IP addresses are assigned from a unique pool of addresses. The default allocation method is dynamic.

Static public IP addresses are assigned addresses that will not change over the lifespan of the Azure resource. To ensure that the IP address for the resource remains the same, you can set the allocation method explicitly to static. In this case, an IP address is assigned immediately. It is released only when you delete the resource or change the IP allocation method to dynamic.

Basic and Standard SKUs

For public IP addresses, there are two types of SKUs to choose from: **Basic** and **Standard**. All public IP addresses created before the introduction of SKUs are Basic SKU public IP addresses.

Basic

Basic public IPs can be assigned by using static or dynamic allocation methods. Basic IPs are open by default. We recommend that you use network security groups to restrict inbound or outbound traffic. Network security groups are recommended but optional for restricting inbound or outbound traffic.

Basic public IPs can be assigned to any Azure resource that can be assigned a public IP address.

Standard

Standard SKU public IP addresses always use the static allocation method. Standard IPs are secure by default and closed to inbound traffic. You must explicitly allow inbound traffic by using a network security group.

Standard IPs can be assigned to network interfaces, Standard public load balancers, application gateways, or VPN gateways.

Public IP address prefix

You can't bring your own public IP addresses from on-premises networks into Azure. Based on the location of the resource, an IP address is assigned from a pool of available addresses. Public IP addresses are allocated from a range that's unique to each region in each Azure cloud. Public IP addresses can't be moved between regions; all IP addresses are region-specific. If your business needs to have datacenters in different regions, you would have a different public IP address range for each region. You can use technology like Azure Traffic Manager to balance between region-specific instances.

To ensure a static range of public IP addresses, you can create a public IP address prefix. You can't specify the addresses when you create the prefix, but after the prefix is created, the addresses will be fixed. The IP addresses will be a contiguous range. The advantage of a public IP address prefix is that you can specify firewall rules for these IP addresses with the knowledge that they will not change. You can assign the addresses from a public IP address prefix to any resource in Azure that supports public IP addresses.

Private IP addresses

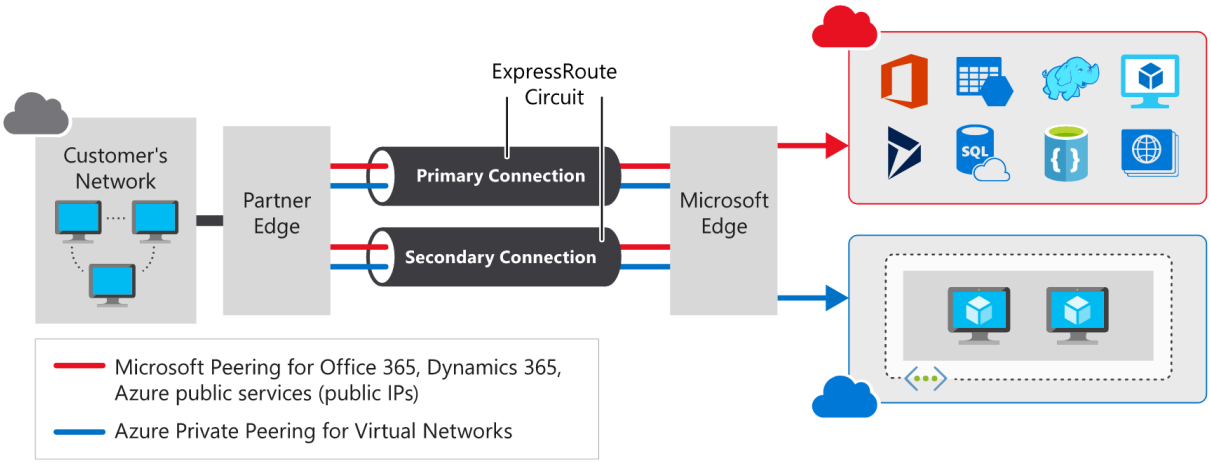
Private IP addresses are used for communication within a virtual network. Private IP addresses are used within Azure Virtual Network virtual networks and your on-premises networks. They can be set to dynamic (DHCP lease) or static (DHCP reservation).

Dynamic private IP addresses are assigned through a DHCP lease and can change over the lifespan of the Azure resource.

Static private IP addresses are assigned through a DHCP reservation and do not change throughout the lifespan of the Azure resource. Static private IP addresses persist if a resource is stopped or deallocated.

ExpressRoute overview

Azure ExpressRoute lets you seamlessly extend your on-premises networks into the Microsoft cloud. This connection between your organization and Azure is dedicated and private. Establishing an ExpressRoute connection enables you to connect to Microsoft cloud services like Azure, Office 365, and Dynamics 365. Security is enhanced, connections are more reliable, latency is minimal, and throughput is greatly increased.



Features and benefits of ExpressRoute

There are several benefits to using ExpressRoute as the connection service between Azure and on-premises networks.

Layer 3 connectivity

ExpressRoute provides Layer 3 (address-level) connectivity between your on-premises network and the Microsoft cloud through connectivity partners. These connections can be from a point-to-point, any-to-any network, or they can be virtual cross-connections through an exchange.

Built-in redundancy

Each connectivity provider uses redundant devices to ensure that connections established with Microsoft are highly available. You can configure multiple circuits to complement this feature. All redundant connections are configured with Layer 3 connectivity to meet SLAs.

Connectivity to Microsoft cloud services

ExpressRoute enables direct access to the following services in all regions:

- Microsoft Office 365
- Microsoft Dynamics 365
- Azure compute services, such as Azure Virtual Machines
- Azure cloud services, such as Azure Cosmos DB and Azure Storage

Office 365 was created to be accessed securely and reliably via the internet. Because of this, we recommend ExpressRoute for specific scenarios. The "Learn more" section at the end of this module includes a link about using ExpressRoute to access Office 365.

Across on-premises connectivity with ExpressRoute Global Reach

You can enable ExpressRoute Global Reach to exchange data across your on-premises sites by connecting your ExpressRoute circuits. For example, assume that you have a private datacenter in California connected to ExpressRoute in Silicon Valley. You have another private datacenter in Texas connected to ExpressRoute in Dallas. With ExpressRoute Global Reach, you can connect your private datacenters through two ExpressRoute circuits. Your cross-datacenter traffic will travel through the Microsoft network.

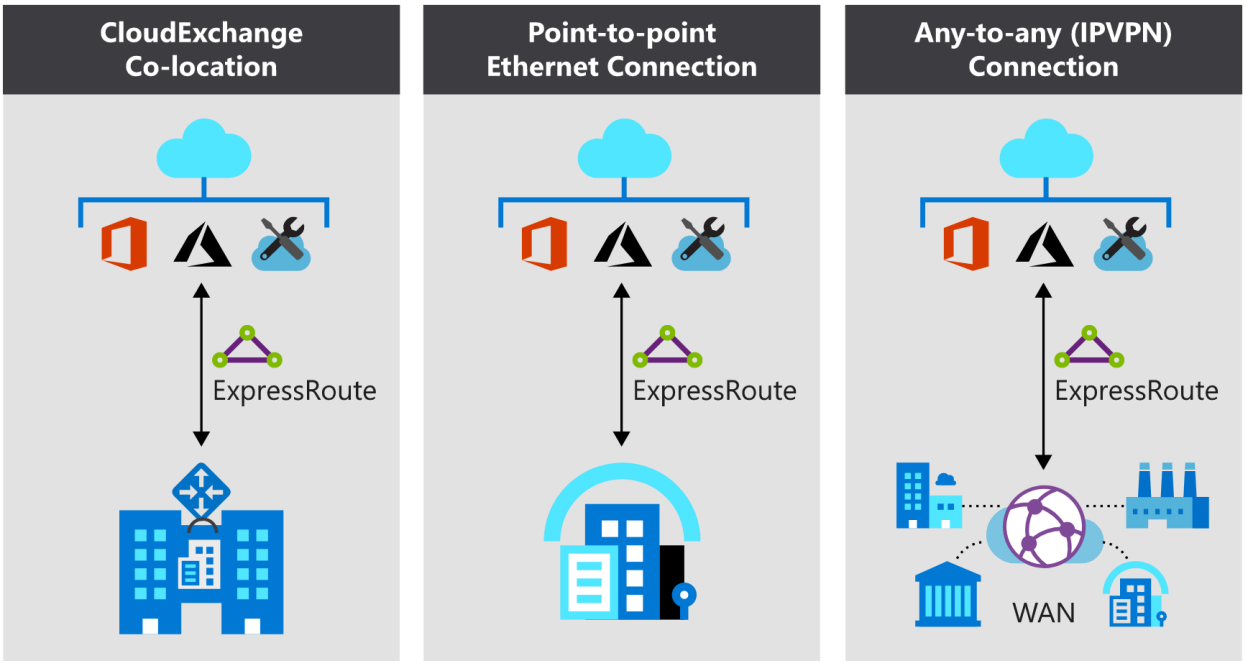
Dynamic routing

ExpressRoute uses the Border Gateway Protocol (BGP) routing protocol. BGP is used to exchange routes between on-premises networks and resources running in Azure. This protocol enables dynamic routing between your on-premises network and services running in the Microsoft cloud.

ExpressRoute connectivity models

ExpressRoute supports three models that you can use to connect your on-premises network to the Microsoft cloud:

- CloudExchange co-location
- Point-to-point Ethernet connection
- Any-to-any connection



Co-location at a cloud exchange

Co-located providers can normally offer both Layer 2 and Layer 3 connections between your infrastructure, which might be located in the co-location facility, and the Microsoft cloud. For example, if your datacenter is co-located at a cloud exchange such as an internet service provider (ISP), you can request a virtual cross-connection to the Microsoft cloud.

Point-to-point Ethernet connection

Point-to-point connections provide Layer 2 and Layer 3 connectivity between your on-premises site and Microsoft Azure. You can connect your offices or datacenters to Azure by using the point-to-point links. For example, if you have an on-premises datacenter, you can use a point-to-point Ethernet link to connect to Microsoft.

Any-to-any networks

With any-to-any connectivity, you can integrate your wide area network (WAN) with Microsoft Azure by providing connections to your offices and datacenters. Azure will integrate with your WAN connection to provide a seamless connection, just like you would have between your datacenter and any branch offices.

With any-to-any connections, all WAN providers offer Layer 3 connectivity. For example, if you already use Multiprotocol Label Switching (MPLS) to connect to your branch offices or other sites in your organization, an ExpressRoute connection to Microsoft will behave just like another location on your private WAN.

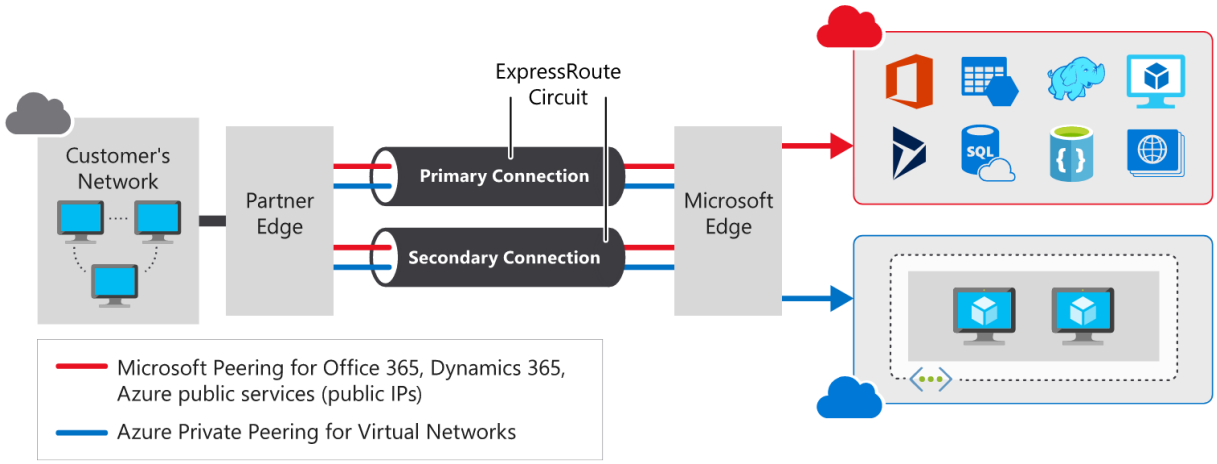
Security considerations

With ExpressRoute, your data doesn't travel over the public internet, so it's not exposed to the potential risks associated with internet communications. ExpressRoute is a private connection from your on-premises infrastructure to your Azure infrastructure. Even if you have an ExpressRoute connection, DNS queries, certificate revocation list checking, and Azure Content Delivery Network requests are still sent over the public internet.

Architecture of ExpressRoute

ExpressRoute is supported across all regions and locations. To implement ExpressRoute, you need to work with an ExpressRoute partner. The partner provides the *edge service*: an authorized and authenticated connection that operates through a partner-controlled router. The edge service is responsible for extending your network to the Microsoft cloud.

The partner sets up connections to an endpoint in an ExpressRoute location (implemented by a Microsoft edge router). These connections enable you to peer your on-premises networks with the virtual networks available through the endpoint. These connections are called *circuits*.



A circuit provides a physical connection for transmitting data through the ExpressRoute provider's edge routers to the Microsoft edge routers. A circuit is established across a private wire rather than the public internet. Your on-premises network is connected to the ExpressRoute provider's edge routers. The Microsoft edge routers provide the entry point to the Microsoft cloud.

Prerequisites for ExpressRoute

Before you can connect to Microsoft cloud services by using ExpressRoute, you need to have:

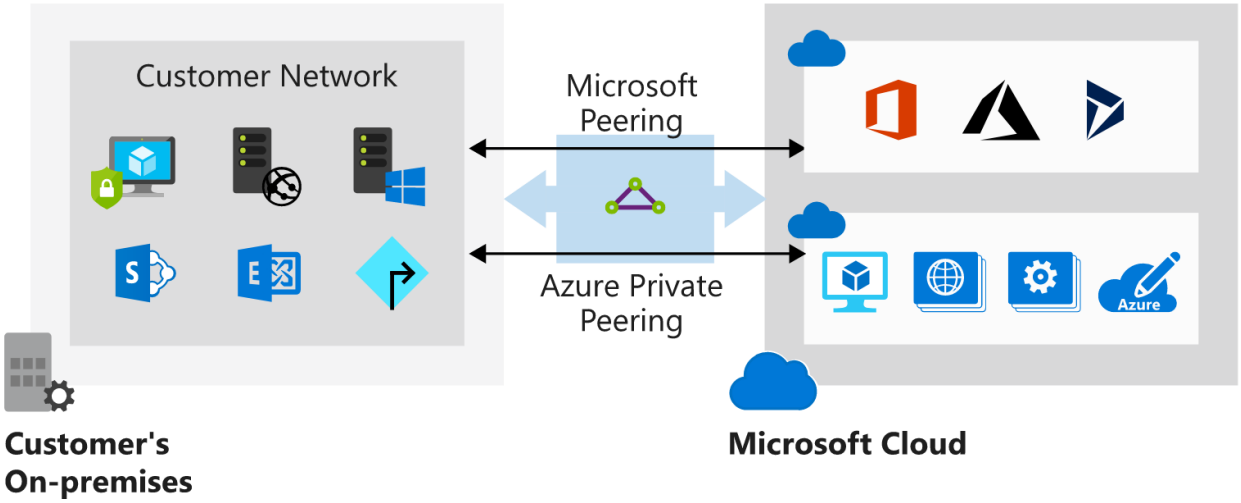
- An ExpressRoute connectivity partner or cloud exchange provider that can set up a connection from your on-premises networks to the Microsoft cloud.
- An Azure subscription that is registered with your chosen ExpressRoute connectivity partner.
- An active Microsoft Azure account that can be used to request an ExpressRoute circuit.
- An active Office 365 subscription, if you want to connect to the Microsoft cloud and access Office 365 services.

ExpressRoute works by peering your on-premises networks with networks running in the Microsoft cloud. Resources on your networks can communicate directly with resources hosted by Microsoft. To support these peerings, ExpressRoute has a number of network and routing requirements:

- Ensure that BGP sessions for routing domains have been configured. Depending on your partner, this might be their or your responsibility. Additionally, for each ExpressRoute circuit, Microsoft requires redundant BGP sessions between Microsoft's routers and your peering routers.
- You or your providers need to translate the private IP addresses used on-premises to public IP addresses by using a NAT service. Microsoft will reject anything except public IP addresses through Microsoft peering.
- Reserve several blocks of IP addresses in your network for routing traffic to the Microsoft cloud. You configure these blocks as either a /29 subnet or two /30 subnets in your IP address space. One of these subnets is used to configure the primary circuit to the Microsoft cloud, and the other implements a secondary circuit. You use the first address in these subnets to communicate with services in the Microsoft cloud. Microsoft uses the second address to establish a BGP session.

ExpressRoute supports two peering schemes:

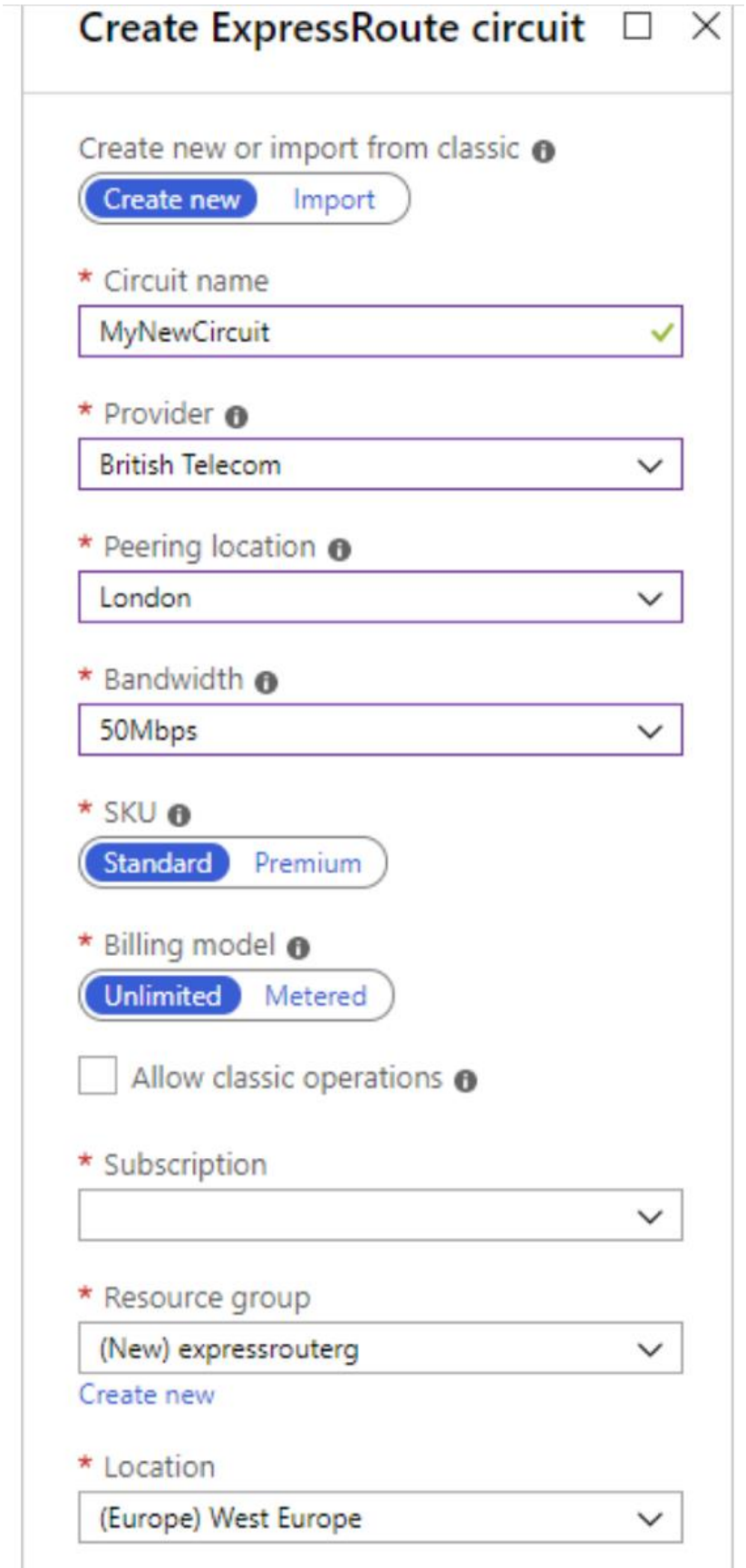
- Use private peering to connect to Azure IaaS and PaaS services deployed inside Azure virtual networks. The resources that you access must all be located in one or more Azure virtual networks with private IP addresses. You can't access resources through their public IP address over a private peering.
- Use Microsoft peering to connect to Azure PaaS services, Office 365 services, and Dynamics 365.



Create an ExpressRoute circuit and peering

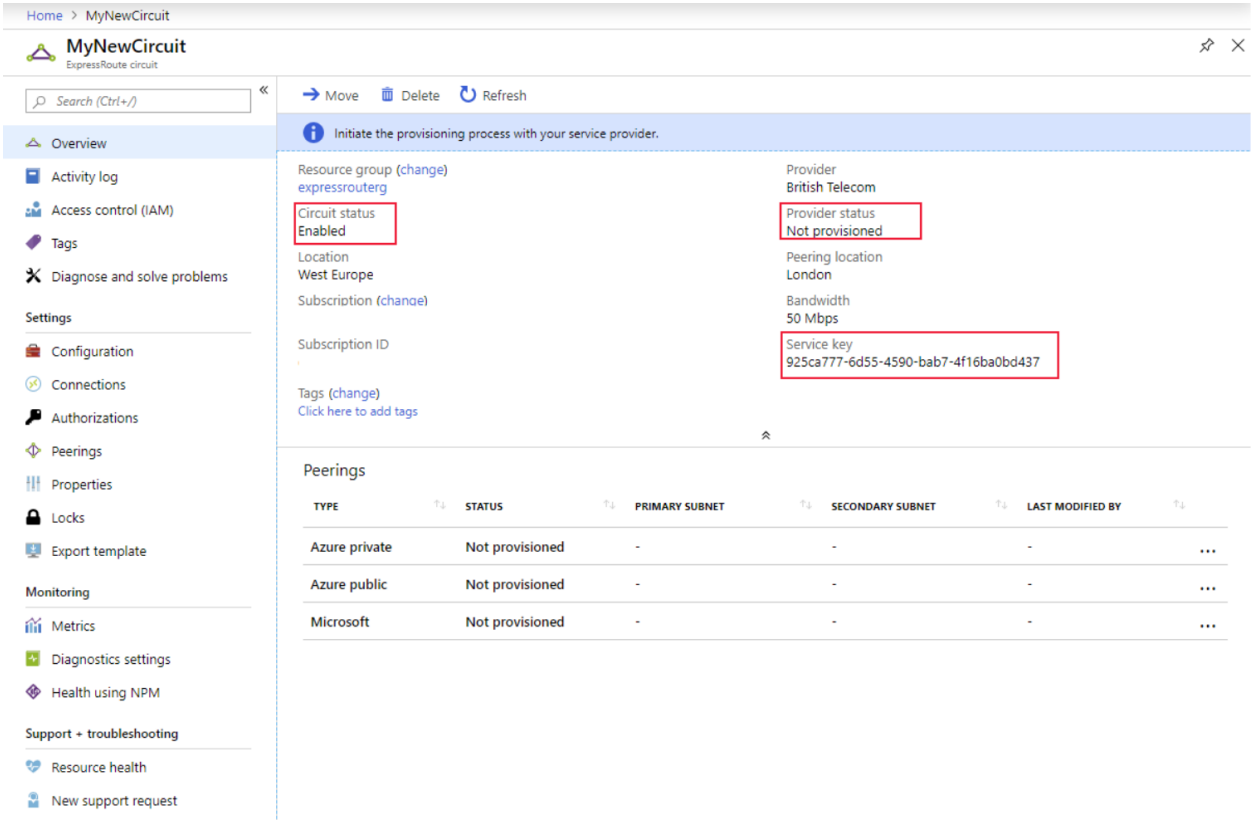
Create a circuit

When you're using the Azure portal, select **Create a resource** > **Networking** > **ExpressRoute**. The **Create ExpressRoute circuit** page.


Create ExpressRoute circuit
Create new or import from classic
Create new
Import
* Circuit name
MyNewCircuit
* Provider
British Telecom
* Peering location
London
* Bandwidth
50Mbps
* SKU
Standard
Premium
* Billing model
Unlimited
Metered
Allow classic operations
* Subscription
* Resource group
(New) expressrouterg
Create new
* Location
(Europe) West Europe

Circuit creation can take several minutes. After the circuit has been provisioned, you can use the Azure portal to view the properties. You'll see that **Circuit status** is enabled, meaning that the Microsoft side of the circuit is ready to accept connections. **Provider status** will be **Not provisioned** initially. This is because the provider hasn't configured their side of the circuit for connecting to your network.

You send the provider the value in the **Service key** field to enable them to configure the connection. This can take several days. You can revisit this page to check the provider status.



Create a peering configuration

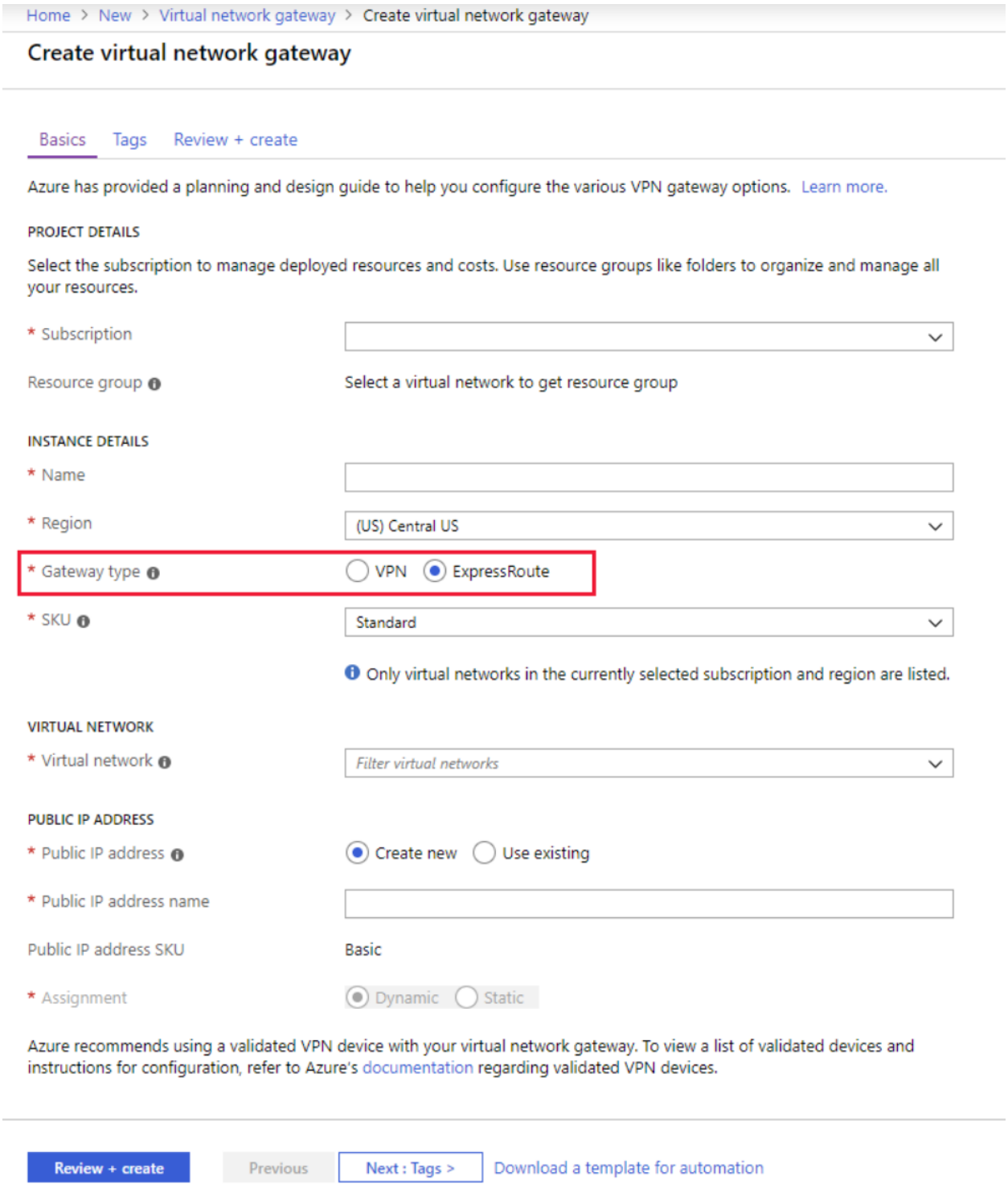
After the provider status is reported as **Provisioned**, you can configure the routing for the peerings. These steps apply only to circuits that are created with service providers who offer Layer 2 connectivity. For any circuits that operate at Layer 3, the provider might be able to configure the routing for you.

Connect a virtual network to an ExpressRoute circuit

After the ExpressRoute circuit has been established, Azure private peering is configured for your circuit, and the BGP session between your network and Microsoft is active, you can enable connectivity from your on-premises network to Azure.

Before you can connect to a private circuit, you must create an Azure virtual network gateway by using a subnet on one of your Azure virtual networks. The virtual network gateway provides the entry point to network traffic that enters from your on-premises network. It directs incoming traffic through the virtual network to your Azure resources.

You can configure network security groups and firewall rules to control the traffic that's routed from your on-premises network. You can also block requests from unauthorized addresses in your on-premises network.



Up to 10 virtual networks can be linked to an ExpressRoute circuit, but these virtual networks must be in the same geopolitical region as the ExpressRoute circuit. You can link a single virtual network to four ExpressRoute circuits if necessary. The ExpressRoute circuit can be in the same subscription to the virtual network, or in a different one.

If you're using the Azure portal, you connect a peering to a virtual network gateway as follows:

1. On the **ExpressRoute circuit** page for your circuit, select **Connections**.
2. On the **Connections** page, select **Add**.
3. On the **Add connection** page, give your connection a name, and then select your virtual network gateway. When the operation has finished, your on-premises network will be connected through the virtual network gateway to your virtual network in Azure. The connection will be made across the ExpressRoute connection.

High availability and failover with ExpressRoute

In each ExpressRoute circuit, there are two connections from the connectivity provider to two different Microsoft edge routers. This configuration occurs automatically. It provides a degree of availability within a single location.

Consider setting up ExpressRoute circuits in different peering locations to provide high availability and help protect against a regional outage.

You can also have multiple circuits across different providers to ensure that your network stays available even if an outage affects all circuits from a single approved provider. You can set the **Connection Weight** property to prefer one circuit to another.

ExpressRoute Direct and FastPath

Microsoft also provides an ultra-high-speed option called ExpressRoute Direct. This service enables dual 100-Gbps connectivity. It's suitable for scenarios that involve massive and frequent data ingestion. It's also suitable for solutions that require extreme scalability, such as banking, government, and retail.

ExpressRoute Direct supports FastPath. When FastPath is enabled, it sends network traffic directly to a virtual machine that's the intended destination. The traffic bypasses the virtual network gateway, improving the performance between Azure virtual networks and on-premises networks.

FastPath doesn't support virtual network peering (where you have virtual networks connected together). It also doesn't support user-defined routes on the gateway subnet.

When to use Azure ExpressRoute

Consider using the Azure ExpressRoute service in the following scenarios:

- Low-latency connectivity to services in the cloud. In these situations, eliminating or reducing the network overhead will have a significant impact on the performance of your applications.
- Accessing high-volume systems in the cloud that consume or produce massive volumes of data quickly. ExpressRoute can move data around rapidly, with high reliability.
- Consuming Microsoft Cloud Services, such as Office 365 and Dynamics 365. ExpressRoute is especially useful if your organization has a large number of users who need to access these services concurrently.
- Organizations that have migrated large-scale on-premises systems to Azure. Using ExpressRoute helps ensure that the results of the migrations are seamless for on-premises clients. They should notice no drop in performance. They might even experience some improvement if the previous on-premises systems were restricted by network bandwidth.
- Situations where data should not traverse the public internet for security reasons.
- Large datacenters, with a high number of users and systems accessing SaaS offerings.

Benefits of using ExpressRoute

ExpressRoute offers several advantages for building highly scalable, cloud-based solutions.

Predictable performance

Having a dedicated connection to the Microsoft cloud guarantees performance. There are no concerns over internet provider outages or spikes in internet traffic. With ExpressRoute, your providers are accountable to provide the necessary throughput and latency SLA.

Data privacy for your traffic

Traffic that's sent over ExpressRoute connection is as secure as using MPLS WAN links. There's no risk of internet monitoring or packet capture by malicious users.

High-throughput, low-latency connections

With ExpressRoute, you can obtain speeds of up to 10 Gbps when connecting to the Microsoft cloud. If you're using ExpressRoute Direct, you can achieve up to 100 Gbps. Latency is minimal, so your systems are highly responsive.

Availability and connectivity

Microsoft guarantees a minimum of 99.95 percent availability for an ExpressRoute dedicated circuit.

With ExpressRoute enabled, you can connect to Microsoft through one of several peering connections and have access to regions within the same geopolitical region. For example, if you connect to Microsoft through ExpressRoute in France, you'll have access to all Microsoft services hosted in Western Europe.

You can also enable ExpressRoute Premium, which provides cross-region accessibility. For example, if you access Microsoft through ExpressRoute in Germany, you'll have access to all Microsoft cloud services in all regions globally.

You can also take advantage of a feature called ExpressRoute Global Reach. It allows you to exchange data across all of your on-premises datacenters by connecting all of your ExpressRoute circuits.

Alternatives to ExpressRoute

ExpressRoute is one of three solutions that you can use to connect your on-premises network to Azure. The others are a virtual network site-to-site connection and a virtual network point-to-site connection.

Site-to-site VPN

An Azure site-to-site VPN connection enables you to connect your on-premises network to Azure over an IPsec tunnel to build a hybrid network solution. You configure an on-premises VPN device with a public IP address. You connect this device to an Azure virtual network through an Azure virtual network gateway.

Point-to-site VPN

With point-to-site VPN, you can establish a secure connection to a network from individual computers located on-premises. This solution is useful for someone who

wants to connect to Azure from remote locations such as a home or customer site. Point-to-site is useful if you have only a few clients that need to connect to a virtual network.

Azure ExpressRoute vs. site-to-site and point-to-site VPN connections

Connection	Azure services supported	Bandwidth	Protocols	Typical use case
Virtual network, point-to-site	Azure IaaS and PaaS services (through private endpoints)	Based on the gateway SKU	Active/passive	Dev, test, and lab environments for cloud services and virtual machines.
Virtual network, site-to-site	Azure IaaS and PaaS services (through private endpoints)	Typically < 1 Gbps aggregate	Active/passive	Dev, test, and lab environments. Small-scale production workloads and virtual machines.
ExpressRoute	Azure IaaS and PaaS services, Microsoft Office 365 services	50 Mbps up to 10 Gbps (100 Gbps for ExpressRoute Direct)	Active/active	Enterprise-class and mission-critical workloads. Big data solutions.

Use network security groups to control network access

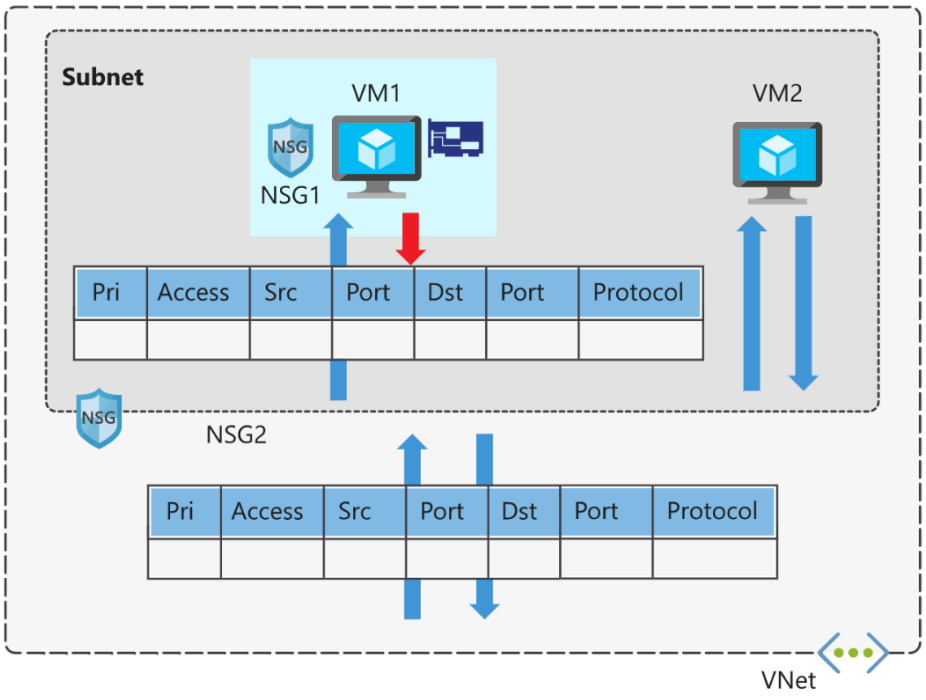
Network security groups

Network security groups filter network traffic to and from Azure resources. Network security groups contain security rules that you configure to allow or deny inbound and outbound traffic. You can use network security groups to filter traffic between VMs or subnets, both within a virtual network and from the internet.

Network security group assignment and evaluation

Network security groups are assigned to a network interface or a subnet. When you assign a network security group to a subnet, the rules apply to all network interfaces in that subnet. You can restrict traffic further by associating a network security group to the network interface of a VM.

When you apply network security groups to both a subnet and a network interface, each network security group is evaluated independently. Inbound traffic is first evaluated by the network security group applied to the subnet, and then by the network security group applied to the network interface. Conversely, outbound traffic from a VM is first evaluated by the network security group applied to the network interface, and then by the network security group applied to the subnet.



Applying a network security group to a subnet instead of individual network interfaces can reduce administration and management efforts. This approach also ensures that all VMs within the specified subnet are secured with the same set of rules.

Each subnet and network interface can have one network security group applied to it. Network security groups support TCP, UDP, and ICMP, and operate at Layer 4 of the OSI model.

In this manufacturing company scenario, network security groups can help you secure the network. You can control which computers can connect to your app servers. You configure the network security group so that only a specific range of IP addresses can connect to the servers. You can lock this down even more by only allowing access to or from specific ports, or from individual IP addresses. These rules can be applied to devices that are connecting remotely from on-premises networks, or between resources within Azure.

Property	**Explanation**
Name	A unique name within the network security group.
Priority	A number between 100 and 4096.
Source and destination	Any, or an individual IP address, classless inter-domain routing (CIDR) block (10.0.0.0/24, for example), service tag, or app security group.
Protocol	TCP, UDP, or Any.
Direction	Whether the rule applies to inbound, or outbound traffic.
Port range	An individual port or range of ports.
Action	Allow or deny the traffic.

etwork security group security rules are evaluated by priority, using the 5-tuple information (source, source port, destination, destination port, and protocol) to allow or deny the traffic. When the conditions for a rule match the device configuration, rule processing stops.

For example, suppose your company has created a security rule to allow inbound traffic on port 3389 (RDP) to your web servers, with a priority of 200. Next, suppose that another admin has created a rule to deny inbound traffic on port 3389, with a priority of 150. The deny rule takes precedence, because it's processed first. The rule with priority 150 is processed before the rule with priority 200.

With network security groups, the connections are stateful. Return traffic is automatically allowed for the same TCP/UDP session. For example, an inbound rule allowing traffic on port 80 also allows the VM to respond to the request (typically on an ephemeral port). You don't need a corresponding outbound rule.

Default security rules

When you create a network security group, Azure creates several default rules. These default rules can't be changed, but can be overridden with your own rules. These default rules allow connectivity within a virtual network and from Azure load balancers. They also allow outbound communication to the internet, and deny inbound traffic from the internet.

The default rules for inbound traffic are:

Priority	Rule name	Description
65000	AllowVnetInbound	Allow inbound coming from any VM to any VM within the subnet.
65001	AllowAzureLoadBalancerInbound	Allow traffic from the default load balancer to any VM within the subnet.
65500	DenyAllInBound	Deny traffic from any external source to any of the VMs.

The default rules for outbound traffic are:

Priority	Rule name	Description
65000	AllowVnetOutbound	Allow outbound going from any VM to any VM within the subnet.
65001	AllowInternetOutbound	Allow outbound traffic going to the internet from any VM.
65500	DenyAllOutBound	Deny traffic from any internal VM to a system outside the virtual network.

Augmented security rules

You use augmented security rules for network security groups to simplify the management of large numbers of rules. Augmented security rules also help when you need to implement more complex network sets of rules. Augmented rules let you add the following options into a single security rule:

- Multiple IP addresses
- Multiple ports
- Service tags

- App security groups

Suppose your company wants to restrict access to resources in your datacenter, spread across several network address ranges. With augmented rules, you can add all these ranges into a single rule, reducing the administrative overhead and complexity in your network security groups.

Service tags

You use service tags to simplify network security group security even further. You can allow or deny traffic to a specific Azure service, either globally or per region.

Service tags simplify security for VMs and Azure virtual networks, by allowing you to restrict access by resources or services. Service tags represent a group of IP addresses, and help simplify the configuration of your security rules. For resources that you can specify by using a tag, you don't need to know the IP address or port details.

You can restrict access to many services. Microsoft manages the service tags (you can't create your own). Some examples of the tags are:

- **VirtualNetwork** - Represents all virtual network addresses anywhere in Azure, and in your on-premises network if you're using hybrid connectivity.
- **AzureLoadBalancer** - Denotes Azure's infrastructure load balancer. The tag translates to the virtual IP address of the host (168.63.129.16) where Azure health probes originate.
- **Internet** - Represents anything outside the virtual network address that is publicly reachable, including resources that have public IP addresses. One such resource is the Web Apps feature of Azure App Service.
- **AzureTrafficManager** - Represents the IP address for Azure Traffic Manager.
- **Storage** - Represents the IP address space for Azure Storage. You can specify whether traffic is allowed or denied. You can also specify if access is allowed only to a specific region, but you can't select individual storage accounts.
- **SQL** - Represents the address for Azure SQL Database, Azure Database for MySQL, Azure Database for PostgreSQL, and Azure SQL Data Warehouse services. You can specify whether traffic is allowed or denied, and you can limit to a specific region.
- **AppService** - Represents address prefixes for Azure App Service.

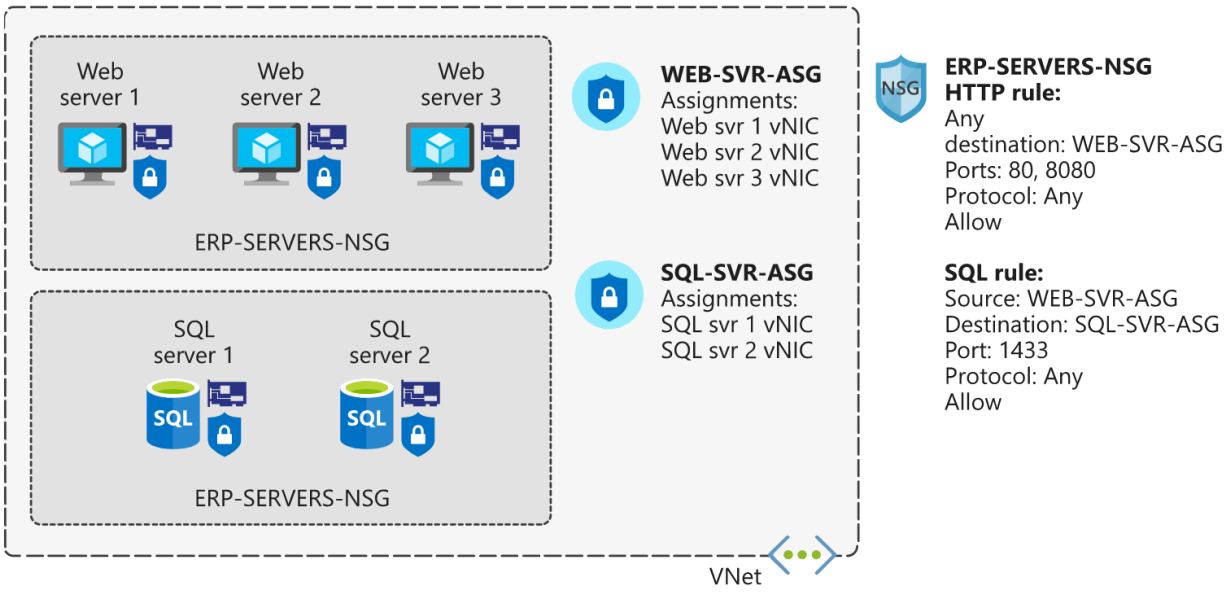
App security groups

App security groups let you configure network security for resources used by specific apps. You can group VMs logically, no matter what their IP address or subnet assignment.

Use app security groups within a network security group to apply a security rule to a group of resources. It's easier to deploy and scale up specific app workloads. You just add a new VM deployment to one or more app security groups, and that VM automatically picks up your security rules for that workload.

An app security group enables you to group network interfaces together. You can then use that app security group as a source or destination rule within a network security group.

For example, your company has a number of front-end servers in a virtual network. The web servers must be accessible over ports 80 and 8080. Database servers must be accessible over port 1433. You assign the network interfaces for the web servers to one app security group, and the network interfaces for the database servers to another app security group. You then create two inbound rules in your network security group. One rule allows HTTP traffic to all servers in the web server app security group. The other rule allows SQL traffic to all servers in the database server app security group.



Without app security groups, you'd need to create a separate rule for each VM.

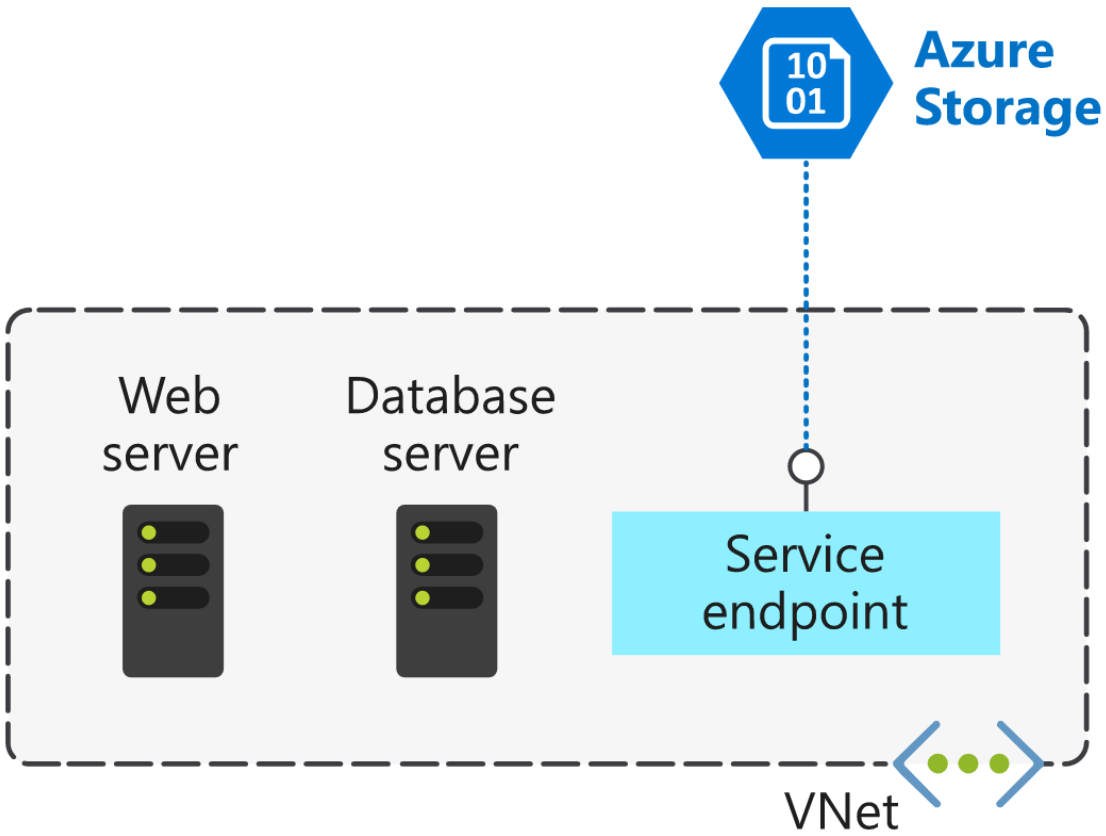
The key benefit of app security groups is that it makes administration easier. You can easily add and remove network interfaces to an app security group as you deploy or redeploy app servers. You can also dynamically apply new rules to an app security group, which are then automatically applied to all the VMs in that app security group.

When to use network security groups

As a best practice, you should always use network security groups to help protect your networked assets against unwanted traffic. Network security groups give you granular control access over the network layer, without the potential complexity of setting security rules for every VM or virtual network.

Virtual network service endpoints

Use virtual network service endpoints to extend your private address space in Azure by providing a direct connection to your Azure services. Service endpoints let you secure your Azure resources to only your virtual network. Service traffic will remain on the Azure backbone, and doesn't go out to the internet.



By default, Azure services are all designed for direct internet access. All Azure resources have public IP addresses, including PaaS services, such as Azure SQL Database and Azure Storage. Because these services are exposed to the internet, anyone can potentially access your Azure services.

Service endpoints can connect certain PaaS services directly to your private address space in Azure, so they act like they’re on the same virtual network. Use your private address space to access the PaaS services directly. Adding service endpoints doesn't remove the public endpoint. It simply provides a redirection of traffic.

Azure service endpoints are available for many services, such as:

- Azure Storage
- Azure SQL Database
- Azure Cosmos DB
- Azure Key Vault
- Azure Service Bus
- Azure Data Lake

For a service like SQL Database, which can't be accessed until you add IP addresses to its firewall, service endpoints should still be considered. Using a service endpoint for SQL Database restricts access to specific virtual networks, providing greater isolation and reducing the attack surface.

How service endpoints work

To enable a service endpoint, you must do the following two things:

1. Turn off public access to the service.
2. Add the service endpoint to a virtual network.

When you enable a service endpoint, you restrict the flow of traffic, and enable your Azure VMs to access the service directly from your private address space. Devices cannot access the service from a public network. On a deployed VM vNIC, if you look at **Effective routes**, you'll notice the service endpoint as the **Next Hop Type**.

This is an example route table, before enabling a service endpoint:

SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE
Default	Active	10.1.1.0/24	VNet
Default	Active	0.0.0.0./0	Internet
Default	Active	10.0.0.0/8	None
Default	Active	100.64.0.0./10	None
Default	Active	192.168.0.0/16	None

And here's an example route table after you've added two service endpoints to the virtual network:

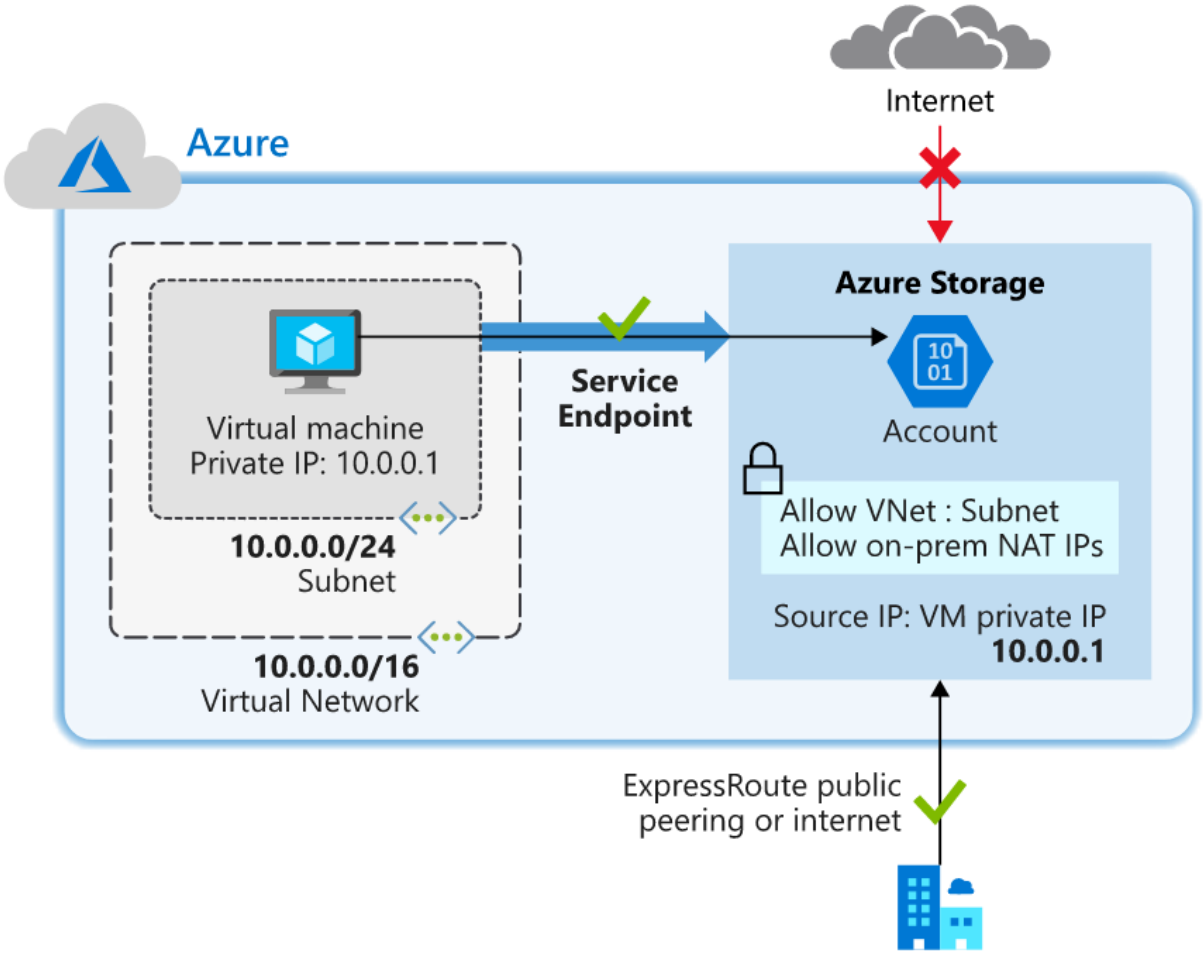
SOURCE	STATE	ADDRESS PREFIXES	NEXT HOP TYPE
Default	Active	10.1.1.0/24	VNet
Default	Active	0.0.0.0./0	Internet
Default	Active	10.0.0.0/8	None
Default	Active	100.64.0.0./10	None
Default	Active	192.168.0.0/16	None
Default	Active	20.38.106.0/23, 10 more	VirtualNetworkServiceEndpoint
Default	Active	20.150.2.0/23, 9 more	VirtualNetworkServiceEndpoint

All traffic for the service now is routed to the **VirtualNetworkServiceEndpoint**, and remains internal to Azure.

Service endpoints and hybrid networks

Service resources that you've secured by using virtual network service endpoints are not, by default, accessible from on-premises networks. To access resources from an on-premises network, use NAT IPs. If you use ExpressRoute for connectivity from on-premises to Azure, you have to identify the NAT IP addresses that are used by ExpressRoute. By default, each circuit uses two NAT IP addresses to connect to the Azure backbone network. You then need to add these IP addresses into the IP firewall configuration of the Azure service resource (for example, Azure Storage).

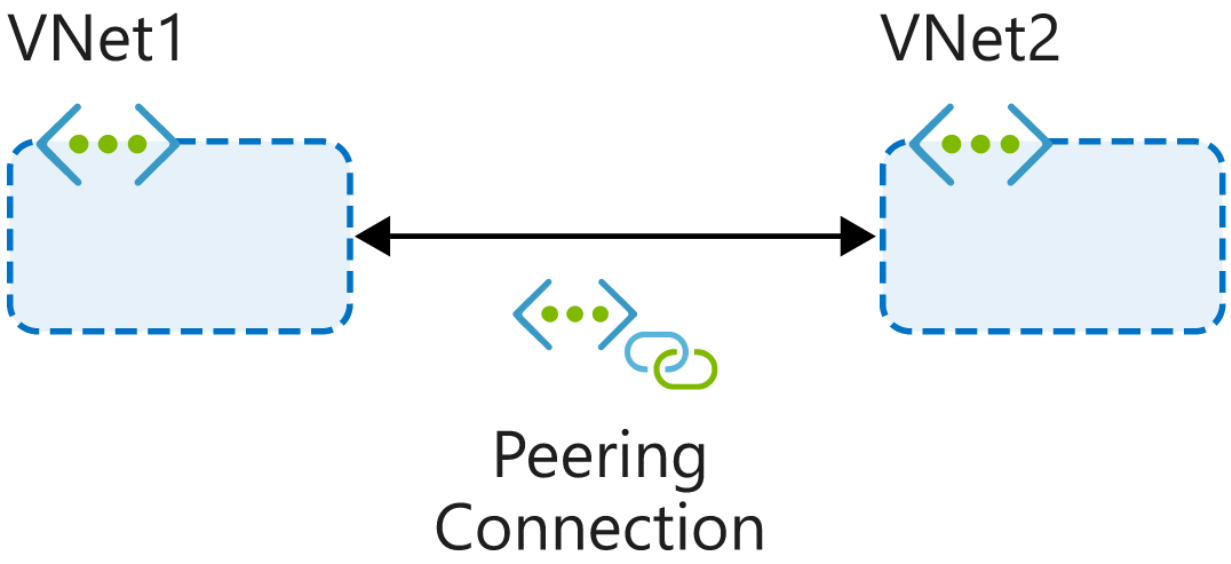
The following diagram shows how you can use a service endpoint and firewall configuration to enable on-premises devices to access Azure Storage resources.


Internet
Azure
Azure Storage
Virtual machine
Private IP: 10.0.0.1
Service
Endpoint
Account
Allow VNet : Subnet
Allow on-prem NAT IPs
10.0.0.0/24
Subnet
Source IP: VM private IP
10.0.0.1
10.0.0.0/16
Virtual Network
ExpressRoute public
peering or internet

Connect services by using virtual network peering

You can use virtual network peering to directly connect Azure virtual networks. When you use peering to connect virtual networks, virtual machines (VMs) in these networks can communicate with each other as if they were in the same network.

In peered virtual networks, traffic between virtual machines is routed through the Azure network. The traffic uses only private IP addresses. It doesn't rely on internet connectivity, gateways, or encrypted connections. The traffic is always private, and it takes advantage of the high bandwidth and low latency of the Azure backbone network.



The two types of peering connections are created in the same way:

- **Virtual network peering** connects virtual networks in the same Azure region, such as two virtual networks in North Europe.
- **Global virtual network peering** connects virtual networks that are in different Azure regions, such as a virtual network in North Europe and a virtual network in West Europe.

Virtual network peering doesn't affect or disrupt any resources that you've already deployed to the virtual networks.

Reciprocal connections

When you create a virtual network peering connection in only one virtual network to connect to a peer in another network, you're not connecting the networks together. To connect the networks by using virtual network peering, you have to create connections in each virtual network.

Cross-subscription virtual network peering

You can use virtual network peering even when both virtual networks are in different subscriptions. This might be necessary for mergers and acquisitions or to connect virtual networks in subscriptions that different departments manage. Virtual networks can be in different subscriptions, and the subscriptions can use the same or different Azure Active Directory tenants.

When you use virtual network peering across subscriptions, you might find that an administrator of one subscription doesn't administer the peer network's subscription. The administrator might not be able to configure both ends of the connection. To peer the virtual networks when both subscriptions are in different Azure Active Directory tenants, the administrators of each subscription must grant the peer subscription's administrator the `Network Contributor` role on their virtual network.

Transitivity

Virtual network peering is nontransitive. Only virtual networks that are directly peered can communicate with each other. The virtual networks can't communicate with the peers of their peers.

Suppose, for example, that your three virtual networks (A, B, C) are peered like this: A <-> B <-> C. Resources in A can't communicate with resources in C because that traffic can't transit through virtual network B. If you need communication between virtual network A and virtual network C, you must explicitly peer these two virtual networks.

Gateway transit

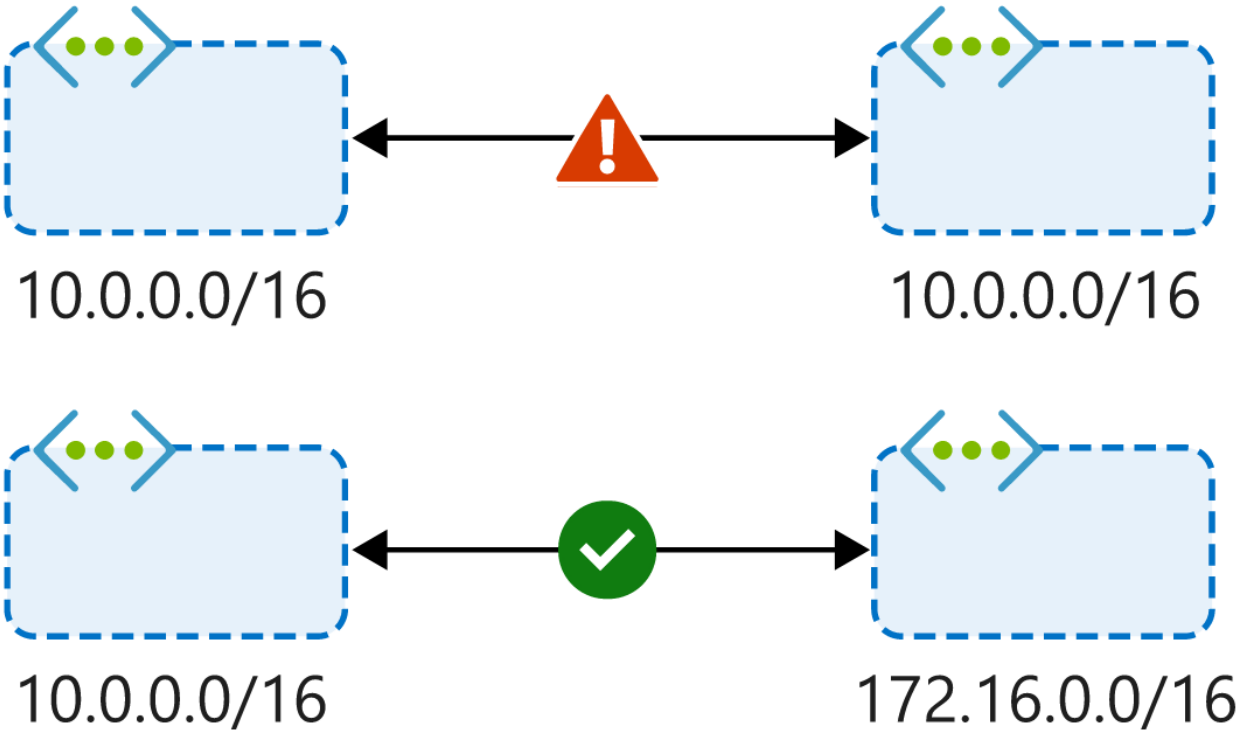
You can configure transitive connections on-premises if you use virtual network gateways as transit points. Using gateway transit, you can enable on-premises connectivity without deploying virtual network gateways to all your virtual networks. This method might reduce cost and complexity. By using gateway peering, you can

configure a single virtual network as a hub network. Connect this hub network to your on-premises datacenter and share its virtual network gateway with peers.

To enable gateway transit, configure the **Allow gateway transit** option in the hub virtual network where you deployed the gateway connection to your on-premises network. Also configure the **Use remote gateways** option in any spoke virtual networks.

Overlapping address spaces

IP address spaces of connected networks within Azure and between Azure and your on-premises system can't overlap. This is also true for peered virtual networks. Keep this rule in mind when you're planning your network design. In any networks you connect through virtual network peering, VPN, or ExpressRoute, assign different address spaces that don't overlap.



Alternative connectivity methods

Virtual network peering is the least complex way to connect virtual networks. Other methods focus primarily on connectivity between on-premises and Azure networks rather than connections between virtual networks.

You can also connect virtual networks together through the ExpressRoute circuit.

VPNs use the internet to connect your on-premises datacenter to the Azure backbone through an encrypted tunnel. You can use a site-to-site configuration to connect virtual networks together through VPN gateways. VPN gateways have higher latency than virtual network peering setups. They're more complex to manage, and they can cost more.

When virtual networks are connected through both a gateway and virtual network peering, traffic flows through the peering configuration.

When to choose virtual network peering

Virtual network peering can be a great way to enable network connectivity between services that are in different virtual networks. Because it's easy to implement and deploy, and it works well across regions and subscriptions, virtual network peering should be your first choice when you need to integrate Azure virtual networks.

Peering might not be your best option if you have existing VPN or ExpressRoute connections or services behind Azure Basic Load Balancers that would be accessed from a peered virtual network. In these cases, you should research alternatives.

On-premises network integration on Azure

About Azure Virtual Network

The Azure Virtual Network service has a specific set of tools and resources for building a cloud-based network architecture for your organization. Azure virtual networks provide a secure virtual communication channel for all permitted Azure resources within your subscription.

With an Azure virtual network, you can:

- Connect virtual machines to the internet.
- Provide secure communications between Azure resources that are hosted in various datacenters and regions.
- Isolate and manage Azure resources.
- Connect to on-premises computers.
- Manage network traffic.

By default, all Azure resources within a virtual network have outbound connectivity to the internet. External inbound communication must come through a public-facing endpoint. All internal resources use a private endpoint to access the virtual network.

A virtual network is composed of many elements including, but not limited to, network interfaces, load balancers, subnets, network security groups, and public IP addresses. These elements work together and enable secure, reliable network communication between your Azure resources, the internet, and on-premises networks.

Traffic routing on an Azure virtual network

Outbound traffic from a subnet is routed based on the destination IP address. A routing table defines how the traffic routes and what happens next. A destination IP address can exist across multiple routing table prefix definitions (for example, 10.0.0.0/16 and 10.0.0.0/24). The router uses a sophisticated algorithm to find the longest prefix match. Traffic that's heading for a 10.0.0.6 address would resolve to the 10.0.0.0/24 prefix and be routed accordingly.

There are two principal routing tables: system and custom.

System routing tables

Azure automatically creates a set of default routing tables for the virtual network and each subnet mask within the virtual network. These system routes are fixed and can't be edited or deleted. However, you can override the default settings by using a custom routing table.

A typical default routing table might look like this:

Source	Address prefixes	Next hop type
Default	Unique to the virtual network	Virtual network
Default	0.0.0.0/0	Internet
Default	10.0.0.0/8	None
Default	172.16.0.0/12	None
Default	192.168.0.0/16	None
Default	100.64.0.0/10	None

A routing table is made up of a source, an address prefix, and a *next hop* type. All traffic that leaves the subnet uses the routing table to find out where it should go next. In effect, the traffic looks for the next hop in its journey.

A next hop defines what happens to the traffic flow next, based on the prefix. There are three types of next hop:

- **Virtual network**: The traffic is routed according to the IP address within the virtual network.
- **Internet**: The traffic is routed to the internet.
- **None**: The traffic is dropped.

Custom routing tables

Apart from system-defined routing tables, you can also create custom routing tables. These user-defined routing tables override the default system table. There are limitations on the number of routing items you can have in a custom table.

A few of the many limitations that apply to virtual networks are listed in the following table:

Resource	Default or maximum number
Virtual networks	1,000
Subnets per virtual network	3,000
Virtual network peerings per virtual network	500
Private IP addresses per virtual network	65,536

Much like the system routing table, custom routing tables also have a next hop type. But the custom routing tables offer a few more options:

- **Virtual appliance**: This option is usually a virtual machine that runs a specific network application, such as a firewall.
- **Virtual network gateway**: Use this option when you want to send traffic to a virtual network gateway. A virtual network gateway type must be VPN. The type can't be Azure ExpressRoute, which requires setting a Border Gateway Protocol (BGP) routing process.
- **None**: This option drops the traffic rather than forwarding it.
- **Virtual network**: This option lets you override a default system routing.
- **Internet**: This option lets you specify that any prefix forwards traffic to the internet.

Connect Azure virtual networks

You can connect your virtual networks in any of several ways. You can use Azure VPN Gateway or ExpressRoute, or you can use the peering method directly.

Azure VPN Gateway

When you're working toward integrating your on-premises network with Azure, there needs to be a bridge between them. VPN Gateway is an Azure service that provides this functionality. A VPN gateway can send encrypted traffic between the two networks. VPN gateways support multiple connections, which enable them to route VPN tunnels that use any available bandwidth. A virtual network can have only one gateway assigned. VPN gateways can also be used for connections between virtual networks in Azure.

Implementing a VPN gateway requires two or more virtual machines to be deployed to the subnet that you create when you set up the virtual network. In this instance, the subnet is also known as the gateway subnet. Each virtual machine is assigned a default configuration for routing and gateway services, explicit to the provisioned gateway. You can't configure these virtual machines directly.

When you create a gateway, several topologies are available. These topologies, also known as gateway types, determine what's configured and the expected connection type.

Site-to-site

You use a site-to-site connection for cross-premises and hybrid-network configurations. This connection topology requires an on-premises VPN device to have a publicly accessible IP address, and must not be behind a NAT. The connection uses a secret ASCII string of up to 128 characters, to authenticate between the gateway and the VPN device.

Multisite

A multisite connection is similar to a site-to-site connection, but with a slight variation. Multisite supports multiple VPN connections to your on-premises VPN devices. This connection topology requires a RouteBased VPN known as a dynamic gateway. It's important to note that, with a multisite configuration, all connections route through and share all available bandwidth.

Point-to-site

A point-to-site connection is suited to a remote individual client device that connects to your network. You must authenticate the client device either through Azure Active Directory or by using Azure certificate authentication. This model suits home working scenarios.

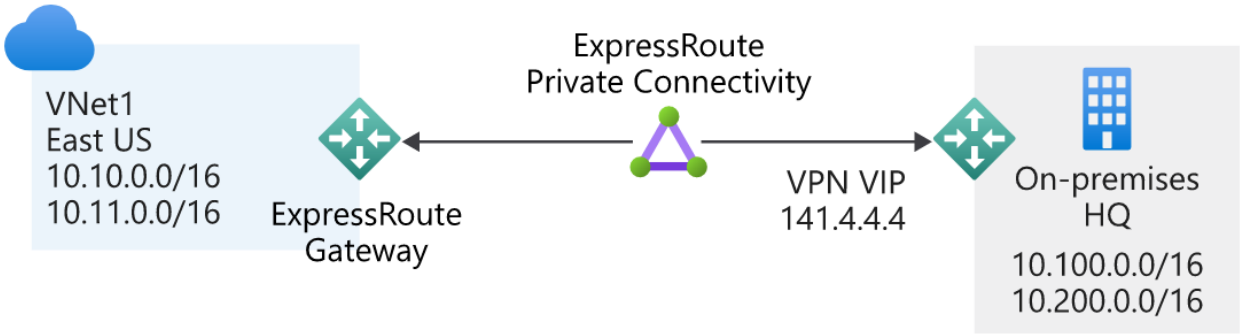
Network-to-network

You use a network-to-network connection to create connections between multiple Azure virtual networks. This connection topology, unlike the others, doesn't require a public IP or VPN device. It can also be used in a multisite configuration to establish combined cross-premises connections with inter-virtual network connectivity.

ExpressRoute

ExpressRoute creates a direct connection between your on-premises network and the Azure virtual network that doesn't use the internet. You use ExpressRoute to seamlessly extend your local network across to the Azure virtual network space. The ExpressRoute service is offered by many third-party connectivity providers. There are three different ExpressRoute connection types:

- CloudExchange colocation
- Point-to-point Ethernet connection
- Any-to-any (IPVPN) connection

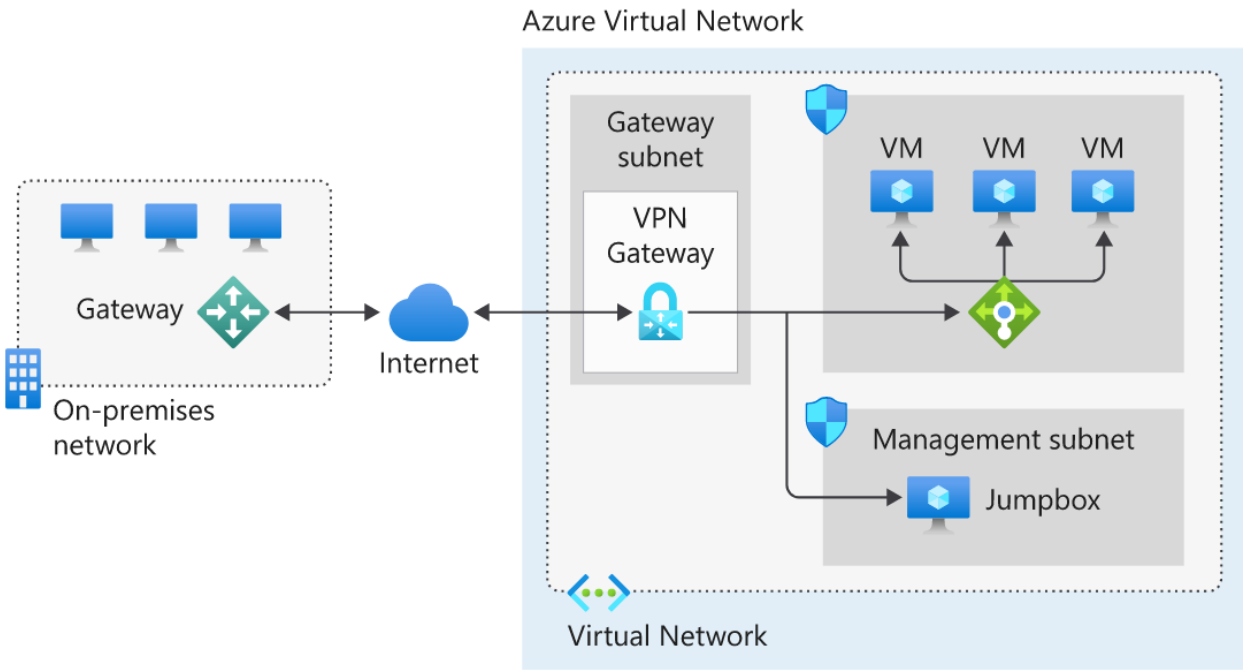


Peering

Virtual networks can peer across subscriptions and Azure regions. After the virtual networks are peered, resources in these networks communicate with each other as if they're in the same network. The traffic is routed between resources in a peered virtual network by using only private IP addresses. Routing is achieved by routing traffic through the Azure network and keeping the connection private as part of the Azure backbone network. The backbone network provides low latency and high-bandwidth network connections.

Site-to-site VPN gateway reference architecture

Although many reference architectures are available when you design a hybrid network, one popular architecture is the site-to-site configuration. The simplified reference architecture shown in the following diagram illustrates how you would connect an on-premises network to the Azure platform. The internet connection uses an IPsec VPN tunnel.



The architecture features several components:

- The **on-premises network** represents your on-premises Active Directory and any data or resources.
- The **gateway** is responsible for sending encrypted traffic to a virtual IP address when it uses a public connection.
- The **Azure virtual network** holds all your cloud applications and any Azure VPN gateway components.
- An **Azure VPN gateway** provides the encrypted link between the Azure virtual network and your on-premises network. An Azure VPN gateway is made up of these elements:
    - Virtual network gateway
    - Local network gateway
    - Connection
    - Gateway subnet
- **Cloud applications** are the ones you've made available through Azure.
- An **internal load balancer**, located in the front end, routes cloud traffic to the correct cloud-based application or resource.

Using this architecture offers several benefits, including:

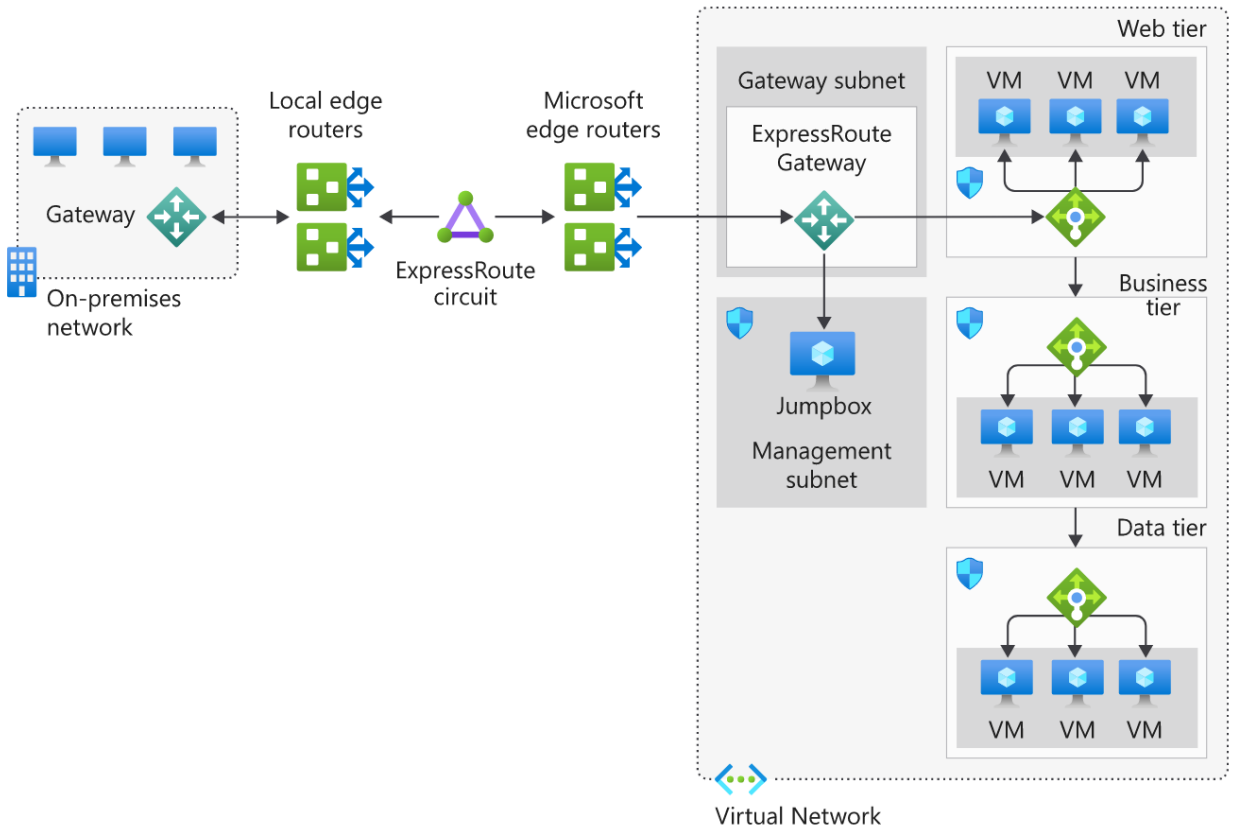
- Configuration and maintenance are simplified.
- Having a VPN gateway helps ensure that all data and traffic are encrypted between the on-premises gateway and the Azure gateway.

- The architecture can be scaled and extended to meet your organization's networking needs.

This architecture isn't applicable in all situations, because it uses an existing internet connection as the link between the two gateway points. Bandwidth constraints can cause latency issues that result from reuse of the existing infrastructure.

ExpressRoute reference architecture

The reference architecture that's illustrated in the following diagram shows how to connect your on-premises network to your Azure virtual networks.



The architecture model includes several components:

- The **on-premises network** is your local Active Directory-managed network.
- **Local edge routers** connect your on-premises network to the connectivity provider's circuit.

- An **ExpressRoute circuit**, supplied by your connectivity provider, operates as a layer 3 circuit. It provides the link between the Azure edge routers and your on-premises edge router.
- The **Microsoft edge routers** are the cloud-side connection between your on-premises network and the cloud. There are always two edge routers, which provides a highly available active-active connection.
- The **Azure virtual network** is where you'll segment your network and assets into tiers. Each application tier, or subnet, can manage specific business operations (for example, web, business, and data).

Is ExpressRoute right for you?

When you're evaluating whether to switch to ExpressRoute, consider the points in the following sections.

Benefits

Implementing ExpressRoute in your organization helps produce the following benefits:

- ExpressRoute is better suited to high-speed and critical business operations.
- ExpressRoute circuits support a maximum bandwidth of 100 Gbps.
- ExpressRoute provides dynamic scalability to help meet your organizational needs.
- ExpressRoute uses layer 3 connectivity and security standards.

Considerations

The following list identifies a few key considerations:

- The setup and configuration for ExpressRoute is more complex, and will require collaboration with the connectivity provider.
- ExpressRoute requires the on-premises installation of high-bandwidth routers.
- The ExpressRoute circuit is handled and managed by the connectivity provider.
- ExpressRoute doesn't support the Hot Standby Router Protocol (HSRP). You'll need to enable a Border Gateway Protocol (BGP) configuration.
- ExpressRoute operates on a layer 3 circuit and requires a network security appliance to manage threats.
- Monitoring the connectivity between your on-premises network and Azure must use the Azure Connectivity Toolkit.
- To improve network security, ExpressRoute requires network security appliances between the provider's edge routers and your on-premises network.

Work with hybrid networks on Azure

What is a hybrid network architecture?

Hybrid network is a term that's used when two different network topologies combine to form a single cohesive network. With Azure, a hybrid network represents the merging or combining of an on-premises network with an Azure virtual network. It allows the continued use of your existing infrastructure while giving you all the benefits of cloud-based computing and access.

There are several reasons why you might want to adopt a hybrid network solution. The two most common are:

- To migrate from a pure on-premises network to a pure cloud-based network.
- To extend your on-premises network and resources to support the cloud services.

Whatever your motivations for adding cloud services to your infrastructure, there are several architectures to consider. We covered ExpressRoute in the preceding unit. The other architectures are:

- Azure VPN Gateway
- ExpressRoute with VPN failover
- Hub-spoke network topology

Azure VPN Gateway

Azure VPN Gateway, a virtual network gateway service, allows site-to-site and point-to-site VPN connectivity between your on-premises network and Azure.

A VPN or virtual private network is a well-established, well-understood network architecture.

VPN Gateway uses your existing connection to the internet. However, all communication is encrypted using the Internet Key Exchange (IKE) and Internet Protocol Security (IPsec) protocols. You can have only one virtual network gateway per virtual network.

When you set up a virtual network gateway, you must specify whether it's a VPN gateway or an ExpressRoute gateway.

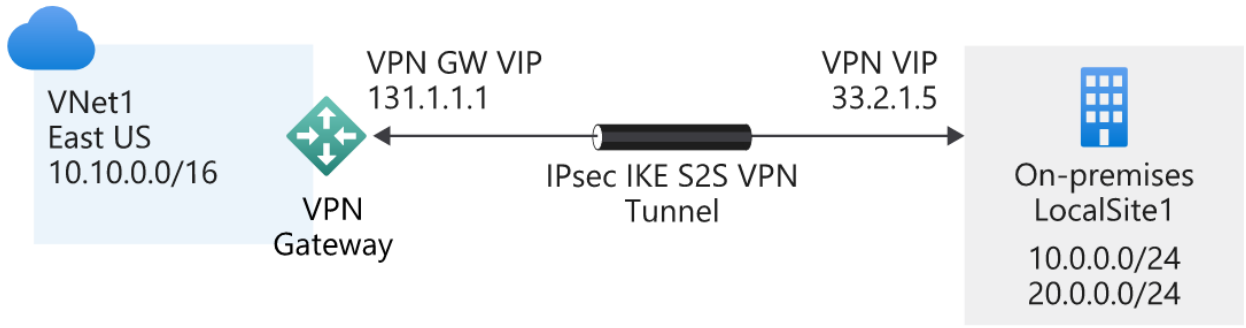
The VPN type depends on the type of connection topology you need. For example, if you want to create a point-to-site (P2S) or a point-to-point (P2P) gateway, you use a *RouteBased* type. There are two VPN types:

- **PolicyBased**: Uses an IPsec tunnel to encrypt data packets. Configuration of the policy uses address prefixes that are drawn from your Azure virtual network and your on-premises network.
- **RouteBased**: Uses the routing or IP forwarding tables to route data packets to the correct tunnel. Each tunnel encrypts and decrypts all packets.

After you've specified the *VPN type* for the virtual network gateway, it can't be altered. If you have to make a change, you need to delete the virtual network gateway and create it again.

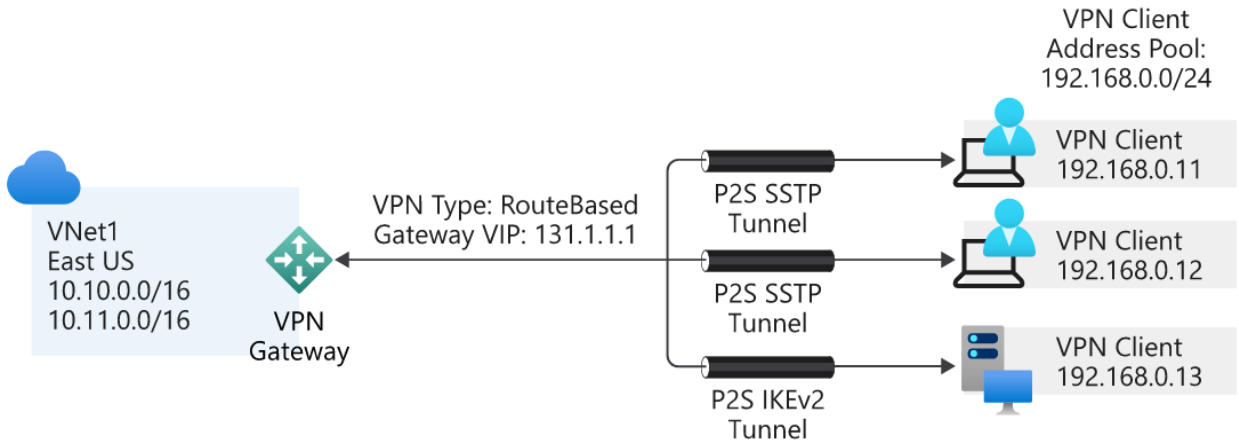
Site-to-site

All site-to-site gateway connections use an IPsec/IKE VPN tunnel to create a connection between Azure and your on-premises network. For a site-to-site connection to work, you'll need an on-premises VPN device with a publicly accessible IP address.



Point-to-site

A point-to-site gateway connection creates a secured connection between an individual device and your Azure virtual network. This gateway type is suited to remote workers; for example, users attending a conference or working from home. A point-to-site connection doesn't require a dedicated on-premises VPN device.



Benefits

Here are some of the benefits of using a VPN connection:

- It's a well-known technology, easy to configure and maintain.
- All data traffic is encrypted.
- It's better suited to lighter data-traffic loads.

Considerations

When you're evaluating the use of this hybrid architecture, consider the following points:

- A VPN connection uses the internet.
- Potential latency issues might exist, depending on bandwidth size and usage.
- Azure supports a maximum bandwidth of 1.25 Gbps.
- For site-to-site connections, you need a local VPN device.

ExpressRoute with VPN failover

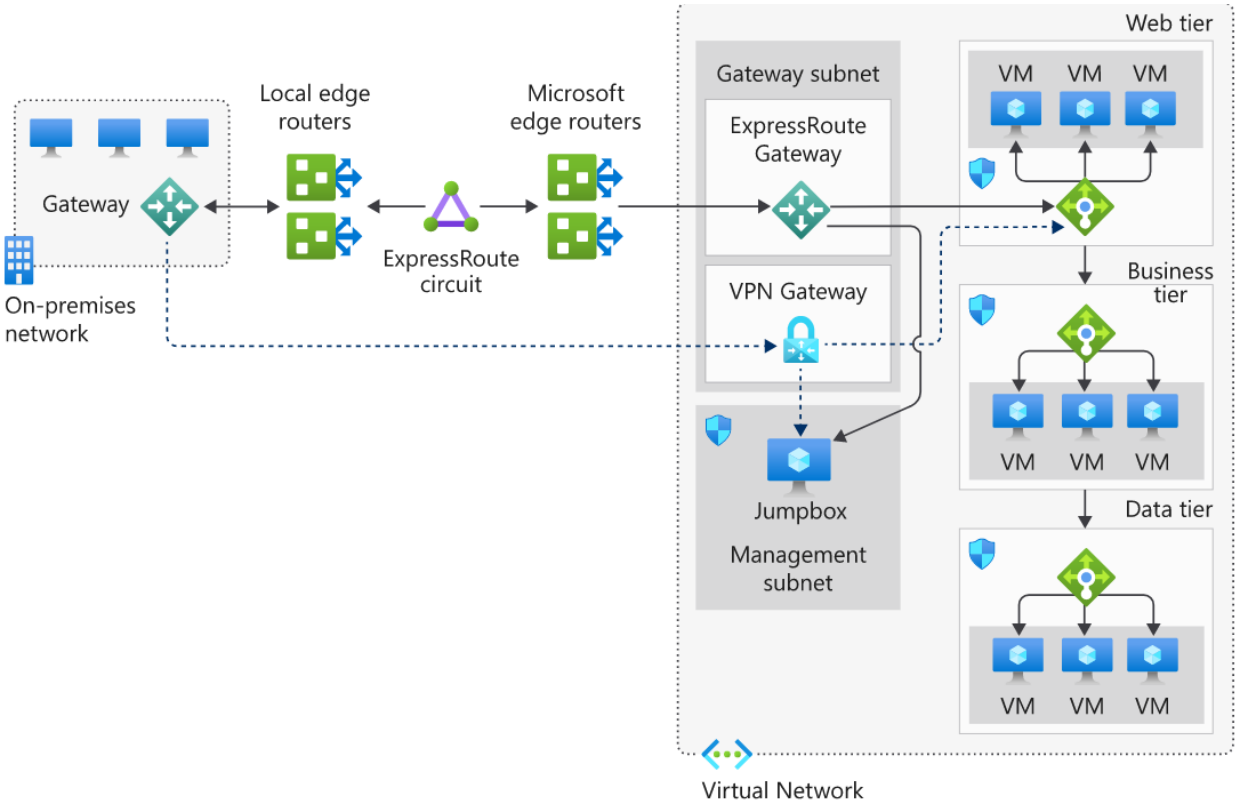
One of the guarantees of using ExpressRoute is that it provides a high level of availability. Each ExpressRoute circuit comes with dual ExpressRoute gateways. However, even with this level of resiliency built into the Azure side of the network, connectivity might be interrupted. One way to remedy this situation, and maintain connectivity, is to provide a VPN failover service.

The merging of the VPN connection and ExpressRoute improves the resiliency of your network connection. When it operates under normal conditions, ExpressRoute behaves

precisely like a regular ExpressRoute architecture, with the VPN connection remaining dormant. If the ExpressRoute circuit fails or goes offline, the VPN connection takes over. This action ensures network availability under all circumstances. When the ExpressRoute circuit is restored, all traffic reverts to using the ExpressRoute connection.

Reference architecture for ExpressRoute with VPN failover

The following diagram illustrates how to connect your on-premises network to Azure by using ExpressRoute with a VPN failover. The chosen topology in this solution is a VPN-based, site-to-site connection with high traffic flow.



n this model, all network traffic routes through the ExpressRoute private connection. When connectivity is lost on the ExpressRoute circuit, the gateway subnet automatically fails over to the site-to-site VPN gateway circuit. This scenario is indicated by the dotted line from the gateway to the VPN gateway in the Azure virtual network.

When the ExpressRoute circuit is restored, traffic automatically switches back from the VPN gateway.

Benefits

The following benefit is available when you implement ExpressRoute with a VPN failover:

- It creates a resilient, high availability network.

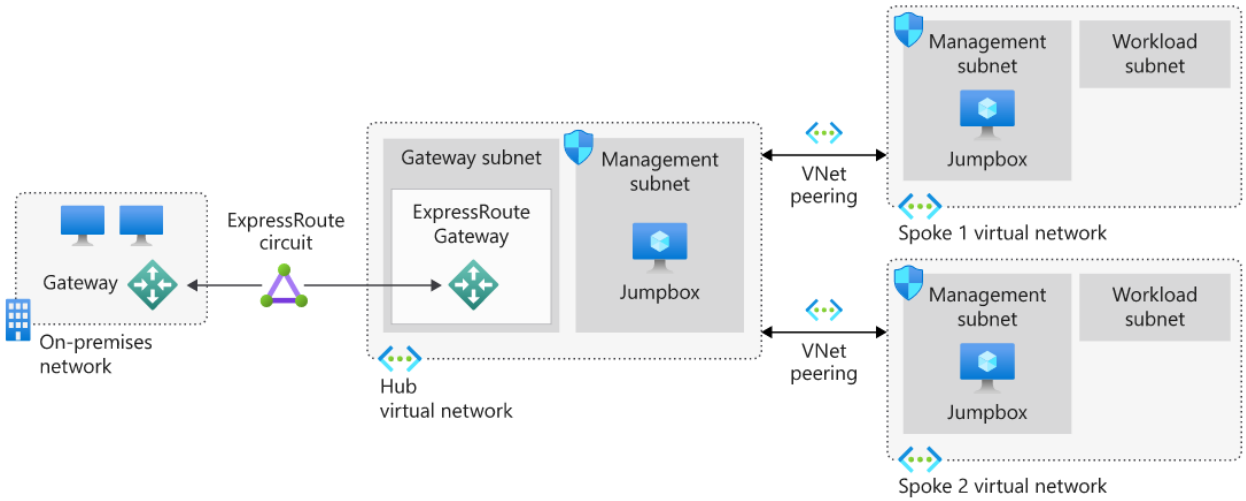
Considerations

When you implement an ExpressRoute with VPN failover architecture, consider the following points:

- When a failover occurs, bandwidth is reduced to VPN connection speeds.
- The ExpressRoute and VPN gateway resources must be in the same virtual network.
- There's a highly complex configuration.
- Implementation requires both an ExpressRoute connection and a VPN connection.
- Implementation requires a redundant VPN gateway and local VPN hardware.

Hub-spoke network topology

The hub-spoke network topology allows you to structure the workloads that are carried out by your servers. It uses a single virtual network as the hub, which is also connected to your on-premises network through either VPN or ExpressRoute. The spokes are other virtual networks that are peered with the hub. Each spoke can be assigned specific workloads, and the hub is used for shared services.



You can implement the hub and each spoke in separate subscriptions or resource groups and then peer them together.

This model uses one of the three previously discussed approaches: VPN, ExpressRoute, and ExpressRoute with VPN failover. The associated benefits and challenges are discussed in the following sections.

Benefits

Implementing a hub-spoke architecture has the following benefits:

- The use of sharing and centralized services on the hub might reduce the need for duplication on the spokes, which can reduce costs.
- Subscription limits are overcome by peering virtual networks.
- The hub-spoke model allows for the separation of organizational work areas into dedicated spokes, such as SecOps, InfraOps, and DevOps.

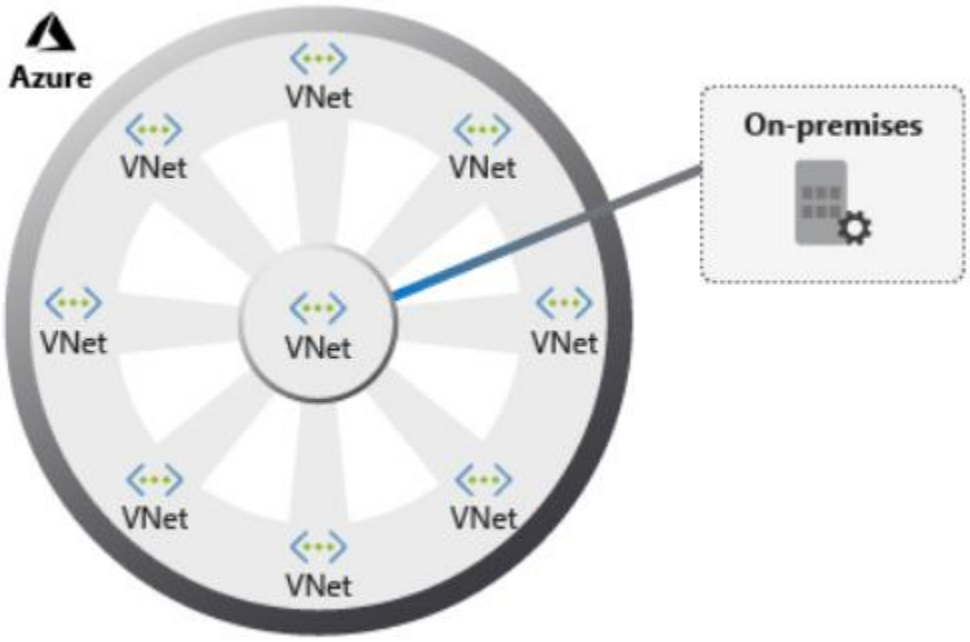
Considerations

When you're evaluating the use of this hybrid architecture, consider the following point:

- Look at the services that are shared on the hub and what remains on the spokes.

Implement a hub-spoke network topology on Azure

Hub and spoke architecture foundations



A hub and spoke consists of a centralized architecture (a hub) connecting to multiple points (spokes). When drawn, it looks similar to a wheel, with a hub at the center and spokes connected to it. This model in Azure organizes your network infrastructure into multiple connected virtual networks. This architecture provides an efficient way to manage common communication, security requirements, and potential subscription limitations.

Implementing a hub and spoke architecture can have the following benefits:

- A centrally managed connection to your on-premises environment.
- Integration of separate working environments into a central location for shared services.
- Traffic routing through the central hub, so workloads can be managed centrally.

Introduction to the hub-spoke topology

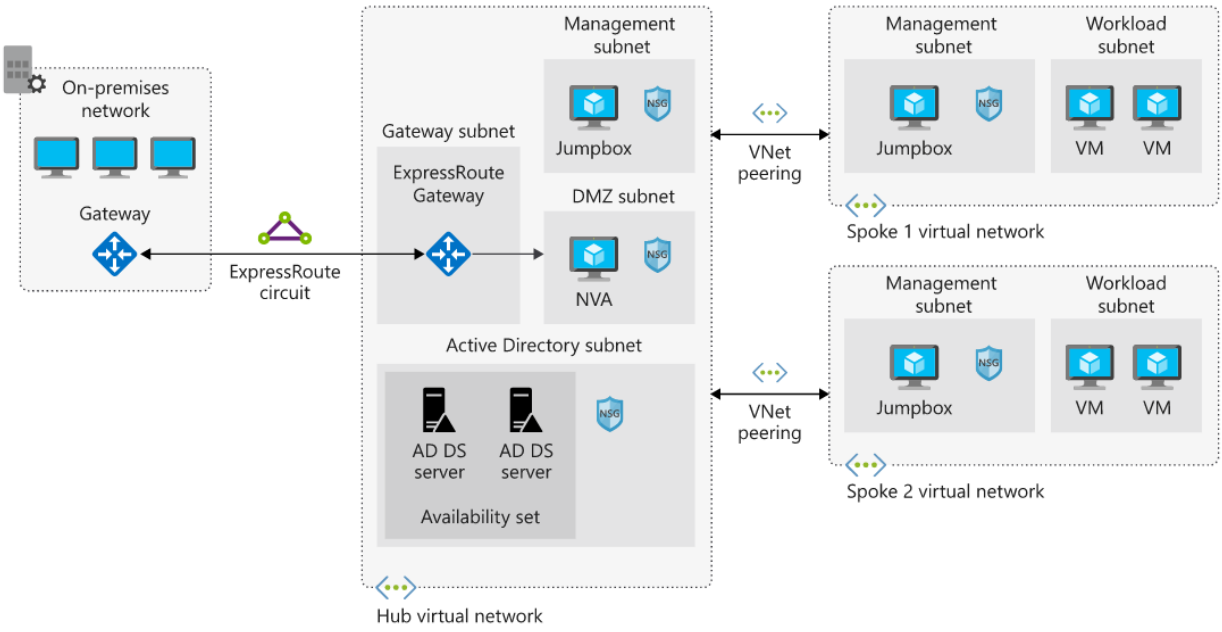
Hub-spoke networks are commonly used for hybrid cloud architectures, and can be simpler to implement and maintain in the long term. The hub is a virtual network that acts as a central location for managing external connectivity, and hosting services used by multiple workloads. The hub coordinates all communications to and from the spokes. IT rules or processes like security can inspect, route, and centrally manage traffic. The spokes are virtual networks that host workloads, and connect to the central hub through virtual network peering.

Hub and spoke topologies offer several business benefits:

- Increased business agility by standardizing on network connections. Organizations can adapt to changing markets, adding a new branch in a different geopolitical region, or a new business channel, as spokes.
- Liability reduction by maintaining a consistent architecture. As the business grows, or traffic volumes increase, it's simple to add more systems.
- Greater visibility into the business, with data flowing through the same place. The hub is the core of the business and provides the foundations for deeper business insights, as it processes every piece of information belonging to the organization.
- A single location in which to share centralized services by multiple workloads. This enables you to minimize redundant resources and the effort required to manage them.

Architectural components

Let's take a look at a reference architecture for a hub-spoke topology. The following image shows the proposed architecture of a pattern to extend your on-premises environment to Azure.



The hub is a virtual network in Azure that's the center point for your business connectivity. Shared services are hosted in their own subnets for sharing with the spokes, and a perimeter subnet acts as a security appliance.

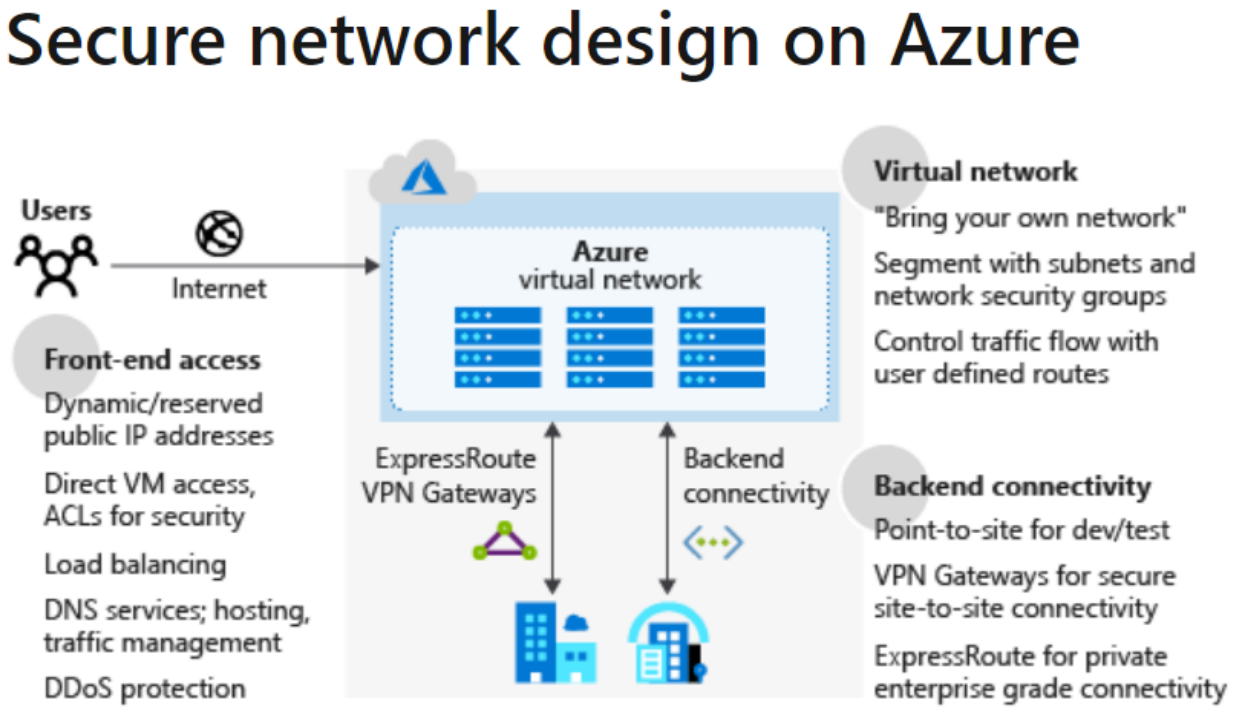
The spokes are also virtual networks in Azure, used to isolate individual workloads. The traffic flow between the on-premises headquarters and Azure is connected through ExpressRoute, connected to the hub virtual network. The virtual networks from the spokes to the hub are peered, and enable communication to on-premises resources. You can implement the hub and each spoke in separate subscriptions or resource groups.

The components included in this architecture are:

- **Azure Virtual Network**: Azure virtual networks are a representation of your own IT network. They're contained within the cloud, logically isolating dedicated organizational resources in your subscriptions.
- **Azure VPN Gateway**: VPN Gateway is the bridge between your on-premises network and Azure. VPN gateways are a special type of Virtual Network gateway that sends encrypted traffic between the two networks over the internet.
- **Azure ExpressRoute**: ExpressRoute is a service in Azure that allows you to extend on-premises networks over a private connection.

Secure your hub and spoke network

Secure network design on Azure



The preceding diagram shows the Azure network infrastructure, and the methods that enable you to connect more securely your on-premises environment, Azure hosted resources, and the public internet.

There are several features to consider as part of securing a network design:

- **Azure Virtual Network**: Provides a base layer of security by logically isolating your environments in Azure, to prevent unauthorized or unwanted access.
- **Azure DNS**: A hosting service for your domain names. Azure DNS is a secure service that manages and resolves domain names in your virtual network.
- **Azure Application Gateway**: A dedicated virtual appliance that provides an application delivery controller as a service, including a web application firewall (WAF).
- **Azure Traffic Manager**: A service to control the distribution of user traffic in Azure.
- **Azure Load Balancer**: Provides high availability and network performance to your Azure applications.
- **Perimeter network**: Segments assets between your Azure virtual network and the internet.

Additionally, consider incorporating some of the following into your network architecture to improve network security:

- Network access controls, to make sure that your Azure services are accessible to only the users and devices you want.
- Network security groups as a packet filtering firewall, to control virtual network traffic.
- Route control, and forced tunneling, to define custom routes through your infrastructure, and ensure services can't connect to an internet device.
- Enabling a virtual network security appliance through Azure Marketplace.
- Using Azure ExpressRoute for a dedicated WAN link, to securely extend your on-premises networks to Azure.
- Azure Security Center to prevent, detect, and respond to threats against your Azure services.
- Azure Firewall as a network security service.

There's a wide variety of security solutions for your organization, many of which complement each other to provide additional layers of security. Microsoft has recommended best practices that you should align with overall. You then implement any features needed to meet your organization's internal security requirements.

Base components of Azure security for hub-spoke topologies

You want to ensure that your resources are protected from unauthorized access, or attack, by controlling your network traffic. In the hub and spoke model, there are several components you need to implement:

Network security group

Each subnet within the topology has a network security group configured. The network security groups implement security rules to allow or deny network traffic to and from each resource in the topology.

Perimeter network

Configure a perimeter network in its own subnet in the hub virtual network for routing external traffic. The perimeter network is designed to host network virtual appliances to provide security functionality, such as firewalls and packet inspection. You can route the outbound traffic from the perimeter network through virtual appliances, so the traffic is monitored, secured, and audited.

Network virtual appliance

Network virtual appliances (NVAs) provide a secure network boundary by checking all inbound and outbound network traffic. Then the NVA passes only the traffic that meets network security rules, essentially acting as a firewall.

Azure Firewall can replace some components discussed in this article, to control access to Azure network resources. For more details, see the following section, "Azure Firewall."

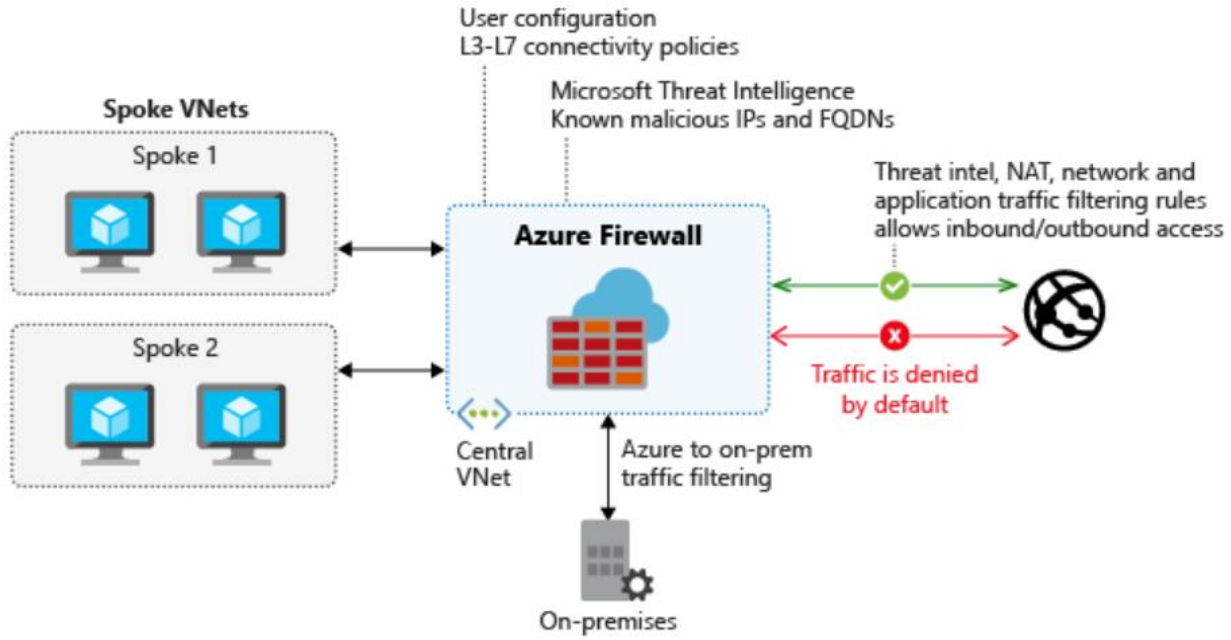
Azure ExpressRoute

ExpressRoute creates a dedicated private WAN link between on-premises resources and an Azure gateway subnet in the hub virtual network. You add a network security appliance between the on-premises network and the ExpressRoute provider edge routers. This restricts the flow of unauthorized traffic from the virtual network.

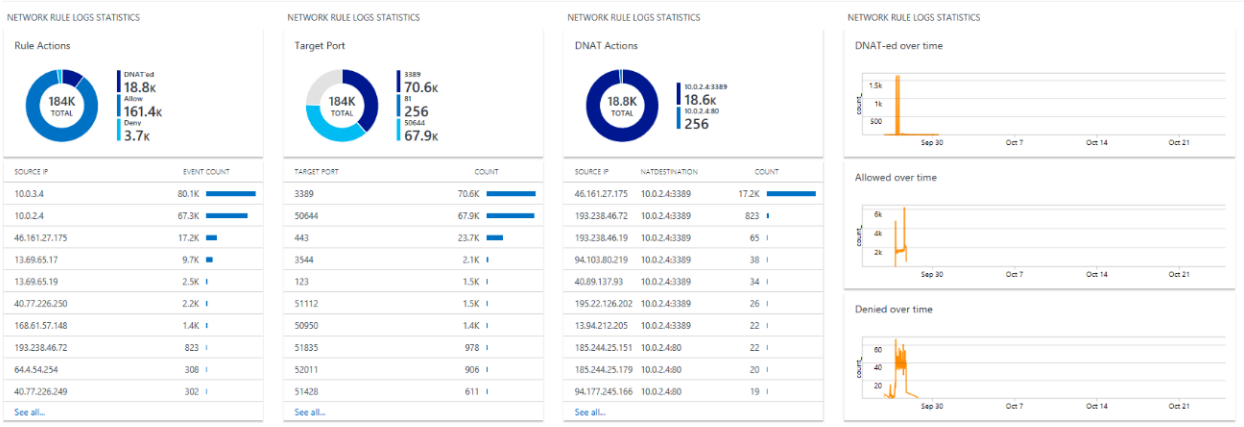
Azure Firewall

Microsoft manages this network security service. It protects Azure virtual networks and their resources by letting you manage and enforce connectivity policies centrally. Azure Firewall uses a static, public IP address for virtual network resources, allowing outside firewalls to identify your virtual network traffic.

Azure Firewall is a fully stateful network firewall that tracks the operating state, and the characteristics of network connections traversing it. Azure Firewall enables central control of all network communications through policy enforcement. These policies can be enforced across virtual networks, regions, and Azure subscriptions. In a hub and spoke topology, Azure Firewall is typically provisioned in the hub for complete control of traffic through the network.



The monitoring of Azure Firewall consists of reviewing the firewall and activity logs. Because Azure Firewall is integrated with Azure Monitor Logs, you can view the full logs there. Some logs are also available to view in the Azure portal.



The logs can be stored in an Azure Storage Account, streamed to an Azure Event Hub, or sent to Azure Monitor Logs.

Network security with network security groups

Network security groups (NSGs) enforce and control network traffic rules. Access is controlled by permitting or denying communication between workloads in a virtual network. NSGs are rules-based, and evaluate traffic using a 5-tuple method. NSGs

evaluate traffic using source IP, source port, destination IP, destination port, and protocol, to determine if traffic is allowed or denied.

Defining security rules

Security rules in an NSG provide the mechanism that defines the control of traffic flow. An NSG has a set of rules by default. These rules can't be deleted, but you can override them with your own custom rules. The default rules are:

- Traffic originating from, and ending in, a virtual network is allowed.
- Outbound traffic to the internet is allowed, but inbound traffic is blocked.
- Azure Load Balancer is allowed to probe the health of virtual machines, or role instances.

Additional security considerations

The ability to control how traffic is routed through your resources is an important security measure to take. Azure helps you improve the security of your overall infrastructure by offering other services:

- **Application security groups**: Provides central policy and security management for your applications. Use application security groups to define detailed network security policies by using a moniker. You can then use a zero-trust approach, where only specified flows are permitted.
- **Azure Network Watcher**: Enables insights into your network logging and diagnostics. Network Watcher allows you to understand the health and performance of your Azure networks.
- **Virtual network service endpoints**: Extends your virtual network private address space to make it available to Azure services. The endpoints allow you to restrict access to Azure resources.
- **Azure DDoS Protection**: Allows you to mitigate volumetric, protocol, and resource layer attacks.

Troubleshoot a network by using Network Watcher monitoring and diagnostic tools

What is Network Watcher?

Network Watcher is an Azure service that combines tools in a central place to diagnose the health of Azure networks. The Network Watcher tools are divided into two categories:

- Monitoring tools
- Diagnostic tools

With tools to monitor for and diagnose problems, Network Watcher gives you a centralized hub for identifying network glitches, CPU spikes, connectivity problems, memory leaks, and other issues before they affect your business.

Network Watcher monitoring tools

Network Watchers provides three monitoring tools:

- Topology
- Connection Monitor
- Network Performance Monitor

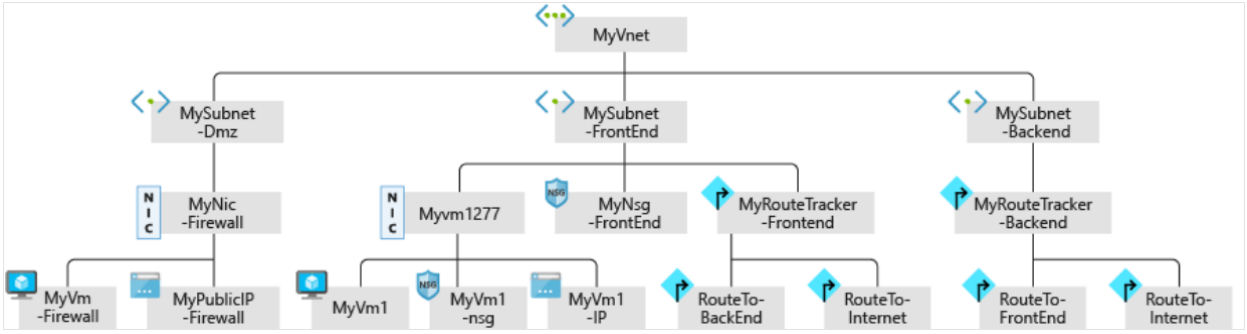
Let's look at each of these tools.

What is the topology tool?

The topology tool generates a graphical display of your Azure virtual network, its resources, its interconnections, and their relationships with each other.

1. Sign in to the Azure portal, and select **All services**. Then, search for **Network Watcher**.
2. Select **Topology**.
3. Select a subscription, the resource group of a virtual network, and then the virtual network itself.

Here's an example of a topology generated for a virtual network named MyVNet.



What is the Connection Monitor tool?

The Connection Monitor tool provides a way to check that connections work between Azure resources. To check that two VMs can communicate if you want them to, use this tool.

This tool also measures the latency between resources. It can catch changes that will affect connectivity, such as changes to the network configuration or changes to network security group (NSG) rules. It can probe VMs at regular intervals to look for failures or changes.

If there's an issue, Connection Monitor tells you why it occurred and how to fix it. Along with monitoring VMs, Connection Monitor can examine an IP address or fully qualified domain name (FQDN).

What is the Network Performance Monitor tool?

The Network Performance Monitor tool enables you to track and alert on latency and packet drops over time. It gives you a centralized view of your network.

When you decide to monitor your hybrid connections by using Network Performance Monitor, check that the associated workspace is in a supported region.

You can use Network Performance Monitor to monitor endpoint-to-endpoint connectivity:

- Between branches and datacenters.
- Between virtual networks.
- For your connections between on-premises and the cloud.
- For Azure ExpressRoute circuits.

Network Watcher diagnostic tools

Network Watcher includes six diagnostic tools:

- IP flow verify
- Next hop
- Effective security rules
- Packet capture
- Connection troubleshoot
- VPN troubleshoot

Let's examine each tool and find out how they can help you solve problems.

What is the IP flow verify tool?

The IP flow verify tool tells you if packets are allowed or denied for a specific virtual machine. If a network security group denies a packet, the tool tells you the name of that group so that you can fix the problem.

This tool uses a 5-tuple packet parameter-based verification mechanism to detect whether packets inbound or outbound are allowed or denied from a VM. Within the tool, you specify a local and remote port, the protocol (TCP or UDP), the local IP, the remote IP, the VM, and the VM's network adapter.

What is the next hop tool?

When a VM sends a packet to a destination, it might take multiple hops in its journey. For example, if the destination is a VM in a different virtual network, the next hop might be the virtual network gateway that routes the packet to the destination VM.

With the next hop tool, you can determine how a packet gets from a VM to any destination. You specify the source VM, source network adapter, source IP address, and destination IP address. The tool then determines the packet's destination. You can use this tool to diagnose problems caused by incorrect routing tables.

What is the effective security rules tool?

The effective security rules tool in Network Watcher displays all the effective NSG rules applied to a network interface.

Network security groups (NSGs) are used in Azure networks to filter packets based on their source and destination IP address and port numbers. NSGs are vital to security because they help you carefully control the surface area of the VMs that users can access. Keep in mind, though, that a mistakenly configured NSG rule might prevent legitimate communication. As a result, NSGs are a frequent source of network problems.

For example, if two VMs can't communicate because an NSG rule blocks them, it can be difficult to diagnose which rule is causing the problem. You'll use the effective security rules tool in Network Watcher to display all the effective NSG rules and help you diagnose which rule is causing the specific problem.

To use the tool, you choose a VM and its network adapter. The tool displays all the NSG rules that apply to that adapter. It's easy to determine a blocking rule by viewing this list.

You can also use the tool to spot vulnerabilities for your VM caused by unnecessary open ports.

What is the packet capture tool?

You use the packet capture tool to record all of the packets sent to and from a VM. You'll then review the capture to gather statistics about network traffic or diagnose anomalies, such as unexpected network traffic on a private virtual network.

The packet capture tool is a virtual machine extension that is remotely started through Network Watcher and happens automatically when you start a packet capture session.

Keep in mind that there is a limit to the amount of packet capture sessions allowed per region. The default usage limit is 100 packet capture sessions per region, and the overall limit is 10,000. These limits are for the number of sessions only, not saved captures. You can save packets captured in Azure Storage or locally on your computer.

Packet capture has a dependency on the *Network Watcher Agent VM Extension* installed on the VM. For links to instructions that detail the installation of the extension on both Windows and Linux VMs, see the "Learn more" section at the end of this module.

What is the connection troubleshoot tool?

You use the connection troubleshoot tool to check TCP connectivity between a source and destination VM. You can specify the destination VM by using an FQDN, a URI, or an IP address.

If the connection is successful, information about the communication appears, including:

- The latency in milliseconds.
- The number of probe packets sent.
- The number of hops in the complete route to the destination.

If the connection is unsuccessful, you'll see details of the fault. Fault types include:

- **CPU**. The connection failed because of high CPU utilization.
- **Memory**. The connection failed because of high memory utilization.
- **GuestFirewall**. The connection was blocked by a firewall outside Azure.
- **DNSResolution**. The destination IP address couldn't be resolved.
- **NetworkSecurityRule**. The connection was blocked by an NSG.
- **UserDefinedRoute**. There's an incorrect user route in a routing table.

What is the VPN troubleshoot tool?

You can use the VPN troubleshoot tool to diagnose problems with virtual network gateway connections. This tool runs diagnostics on a virtual network gateway connection and returns a health diagnosis.

When you start the VPN troubleshoot tool, Network Watcher diagnoses the health of the gateway or connection, and returns the appropriate results. The request is a long-running transaction.

Fault Type	Reason	Log
NoFault	No error is detected.	Yes
GatewayNotFound	A gateway can't be found or isn't provisioned.	No
PlannedMaintenance	A gateway instance is under maintenance.	No
UserDrivenUpdate	A user update is in progress. The update might be a resize operation.	No
VipUnResponsive	The primary instance of the gateway can't be reached because of a health probe failure.	No
PlatformInActive	There's an issue with the platform.	No

There are connectivity issues in a single-VM network

To troubleshoot this issue, use the IP flow verify tool. This tool lets you specify a local and remote port, the protocol (TCP/UDP), the local IP, and the remote IP to check the connection status. It also lets you specify the direction of the connection (inbound or outbound). IP flow verify runs a logical test on the rules in place on your network.

In this case, use IP flow verify to specify the VM's IP address and the RDP port 3389. Then, specify the remote VM's IP address and port. Choose the TCP protocol, and then select **Check**.

A VPN connection isn't working

To troubleshoot a VPN connection, use Azure VPN troubleshoot.

No servers are listening on designated destination ports

Use the connection troubleshoot tool to troubleshoot this issue. In this tool, you specify the local and remote VMs. In the probe setting, you can choose a specific port.

Suppose the results show the remote server is **Unreachable**, along with the message "Traffic blocked due to virtual machine firewall configuration." On the remote server, disable the firewall, and then test the connection again.

Suppose the server is now reachable. This result indicates that firewall rules on the remote server are the issue, and must be corrected to permit the connection.

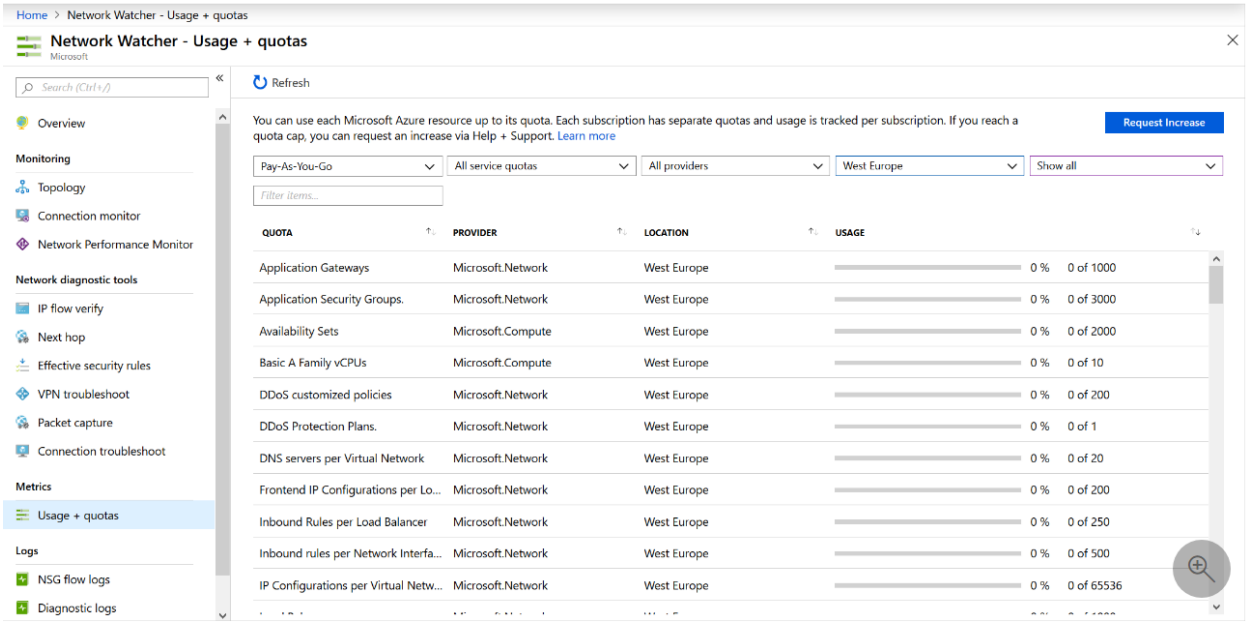
Troubleshoot a network by using Network Watcher metrics and logs

Usage and quotas

Each Microsoft Azure resource can be used up to its quota. Each subscription has separate quotas, and usage is tracked per subscription. Only one instance of Network Watcher is required per subscription per region. This instance gives you a view of usage and quotas so that you can see if you're at risk of hitting a quota.

To view the usage and quota information, go to **All Services** > **Networking** > **Network Watcher**, and then select **Usage and quotas**. You'll see granular data based on usage and resource location. Data for the following metrics is captured:

- Network interfaces
- Network security groups (NSGs)
- Virtual networks
- Public IP addresses



Logs

Network diagnostic logs provide granular data. You'll use this data to understand connectivity and performance issues better. There are three log display tools in Network Watcher:

- Flow logs
- Diagnostic logs
- Traffic analytics

Let's look at each of these tools.

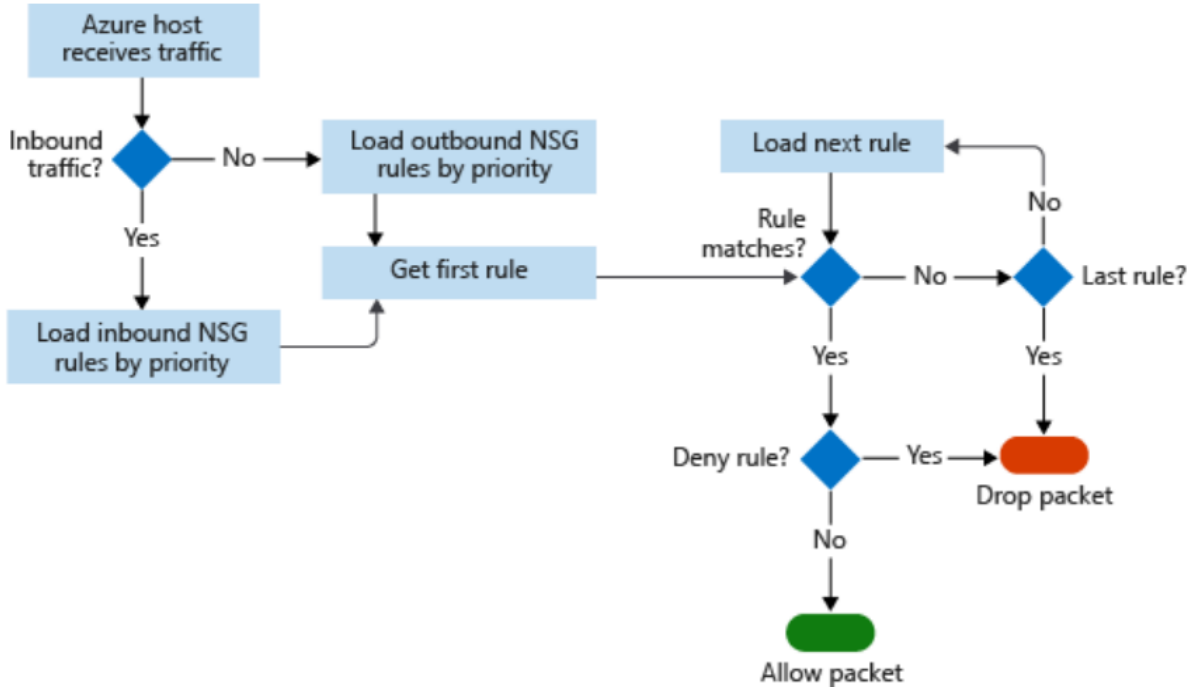
Flow logs

In flow logs, you can view information about ingress and egress IP traffic on network security groups. Flow logs show outbound and inbound flows on a per-rule basis, based

on the network adapter that the flow applies. NSG flow logs show whether traffic was allowed or denied based on the 5-tuple information captured. This information includes:

- Source IP
- Source port
- Destination IP
- Destination port
- Protocol

This diagram shows the workflow that the NSG follows.



Flow logs store data in a JSON file. It can be difficult to gain insights into this data by manually searching the log files, especially if you have a large infrastructure deployment in Azure. You can solve this problem by using Power BI.

In Power BI, you can visualize NSG flow logs by, for example:

- Top talkers (IP address)
- Flows by direction (inbound and outbound)
- Flows by decision (allowed and denied)
- Flows by destination port

You can also use open source tools to analyze your logs, such as Elastic Stack, Grafana, and Graylog.

Diagnostic logs

In Network Watcher, diagnostic logs are a central place to enable and disable logs for Azure network resources. These resources might include NSGs, public IPs, load balancers, and app gateways. After you've enabled the logs that interest you, you can use the tools to query and view log entries.

You can import diagnostic logs into Power BI and other tools to analyze them.

Traffic analytics

To investigate user and app activity across your cloud networks, use traffic analytics.

The tool gives insights into network activity across subscriptions. You can diagnose security threats such as open ports, VMs communicating with known bad networks, and traffic flow patterns. Traffic analytics analyzes NSG flow logs across Azure regions and subscriptions. You can use the data to optimize network performance.

This tool requires Log Analytics. The Log Analytics workspace must exist in a supported region.

Choose a data storage approach in Azure

What is a transaction?

A transaction is a logical group of database operations that execute together.

Here's the question to ask yourself regarding whether you need to use transactions in your application: Will a change to one piece of data in your dataset impact another? If the answer is yes, then you'll need support for transactions in your database service.

Transactions are often defined by a set of four requirements, referred to as ACID guarantees. ACID stands for **A**tomicity, **C**onsistency, **I**solation, and **D**urability:

- **Atomicity** means a transaction must execute exactly once and must be atomic; either all of the work is done, or none of it is. Operations within a transaction usually share a common intent and are interdependent.
- **Consistency** ensures that the data is consistent both before and after the transaction.
- **Isolation** ensures that one transaction is not impacted by another transaction.
- **Durability** means that the changes made due to the transaction are permanently saved in the system. Committed data is saved by the system so that even in the event of a failure and system restart, the data is available in its correct state.

When a database offers ACID guarantees, these principles are applied to any transactions in a consistent manner.

OLTP vs OLAP

Transactional databases are often called OLTP (Online Transaction Processing) systems. OLTP systems commonly support lots of users, have quick response times, and handle large volumes of data. They are also highly available (meaning they have very minimal downtime), and typically handle small or relatively simple transactions.

On the contrary, OLAP (Online Analytical Processing) systems commonly support fewer users, have longer response times, can be less available, and typically handle large and complex transactions.

The terms OLTP and OLAP aren't used as frequently as they used to be, but understanding them makes it easier to categorize the needs of your application.

Now that you're familiar with transactions, OLTP, and OLAP, let's walk through each of the data sets in the online retail scenario, and determine the need for transactions.

Create an Azure Storage account

Account kind

Storage account *kind* is a set of policies that determine which data services you can include in the account and the pricing of those services. There are three kinds of storage accounts:

- **StorageV2 (general purpose v2)**: the current offering that supports all storage types and all of the latest features
- **Storage (general purpose v1)**: a legacy kind that supports all storage types but may not support all features
- **Blob storage**: a legacy kind that allows only block blobs and append blobs

Microsoft recommends that you use the **General-purpose v2** option for new storage accounts.

There are a few special cases that can be exceptions to this rule. For example, pricing for transactions is lower in general purpose v1, which would allow you to slightly reduce costs if that matches your typical workload.

The core advice here is to choose the **Resource Manager** deployment model and the **StorageV2 (general purpose v2)** account kind for all your storage accounts. The other options still exist primarily to allow existing resources to continue operation. For new resources, there are few reasons to consider the other choices.

Connect an app to Azure Storage

Blob storage

Azure Storage supports three kinds of blobs:

Blob type	Description
Block blobs	Block blobs are used to hold text or binary files up to ~5 TB (50,000 blocks of 100 MB) in size. The primary use case for block blobs is the storage of files that are read from beginning to end, such as media files or image files for websites. They are named block blobs because files larger than 100 MB must be uploaded as small blocks. These blocks are then consolidated (or committed) into the final blob.
Page blobs	Page blobs are used to hold random-access files up to 8 TB in size. Page blobs are used primarily as the backing storage for the VHDs used to provide durable disks for Azure Virtual Machines (Azure VMs). They are named page blobs because they provide random read/write access to 512-byte pages.
Append blobs	Append blobs are made up of blocks like block blobs, but they are optimized for append operations. These blobs are frequently used for logging information from one or more sources into the same blob. For example, you might write all of your trace logging to the same append blob for an application running on multiple VMs. A single append blob can be up to 195 GB.

REST API endpoint

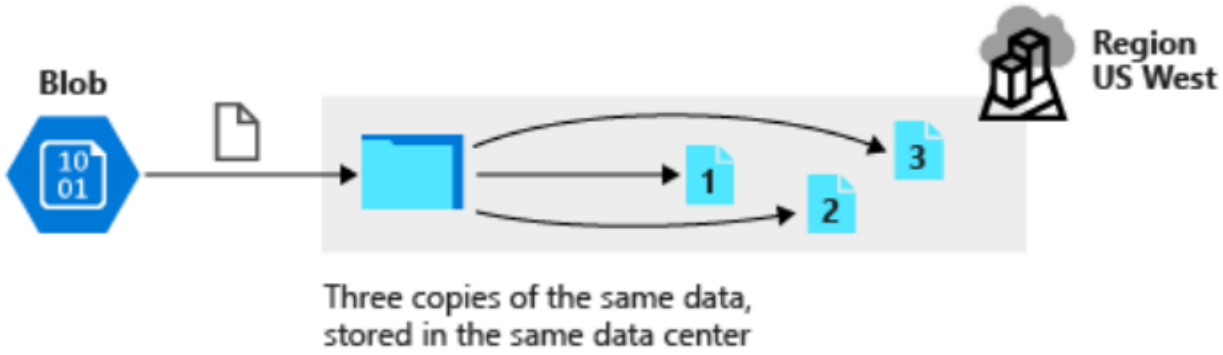
In addition to access keys for authentication to storage accounts, your app will need to know the storage service endpoints to issue the REST requests.

The REST endpoint is a combination of your storage account *name*, the data type, and a known domain. For example:

Data type	Example endpoint
Blobs	https://[name].blob.core.windows.net/
Queues	https://[name].queue.core.windows.net/
Table	https://[name].table.core.windows.net/
Files	https://[name].file.core.windows.net/

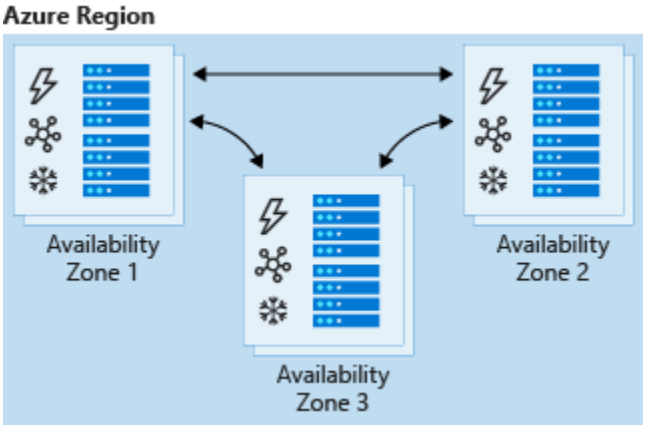
Make your application storage highly available with read-access geo-redundant storage

What is locally redundant storage (LRS)?



Locally redundant storage replicates data and stores three copies across fault domains, or racks of hardware, within a single datacenter facility in one region. Data is replicated so that if there's a hardware fault or maintenance work, your data is still available and accessible.

What is zone-redundant storage (ZRS)?

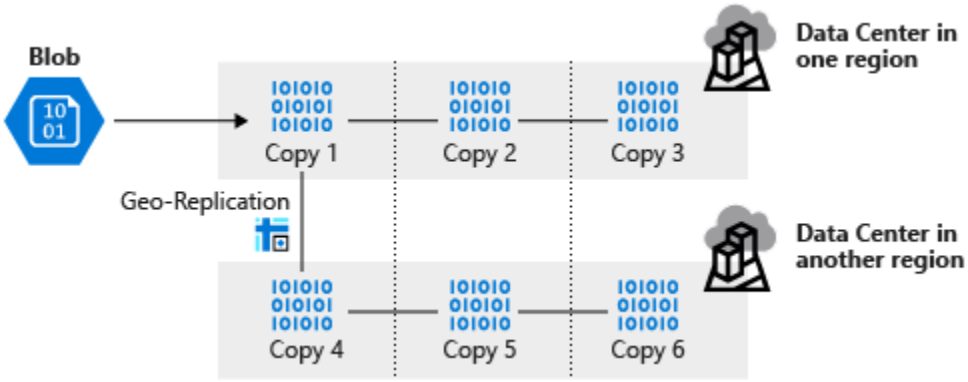


Zone-redundant storage replicates your data across three storage clusters in a region. Each cluster is physically separated from the other two, which means that each cluster is supplied by separate utilities, such as power or networking.

If there's an outage in a datacenter, you can still access your data from another availability zone in that region. Data is normally replicated to two or three availability zones, depending on the region.

ZRS offers 99.999999999999 percent durability of data. However, ZRS might not protect you from a regional outage, because all AZs reside in the same region.

What is geographically redundant storage (GRS)?

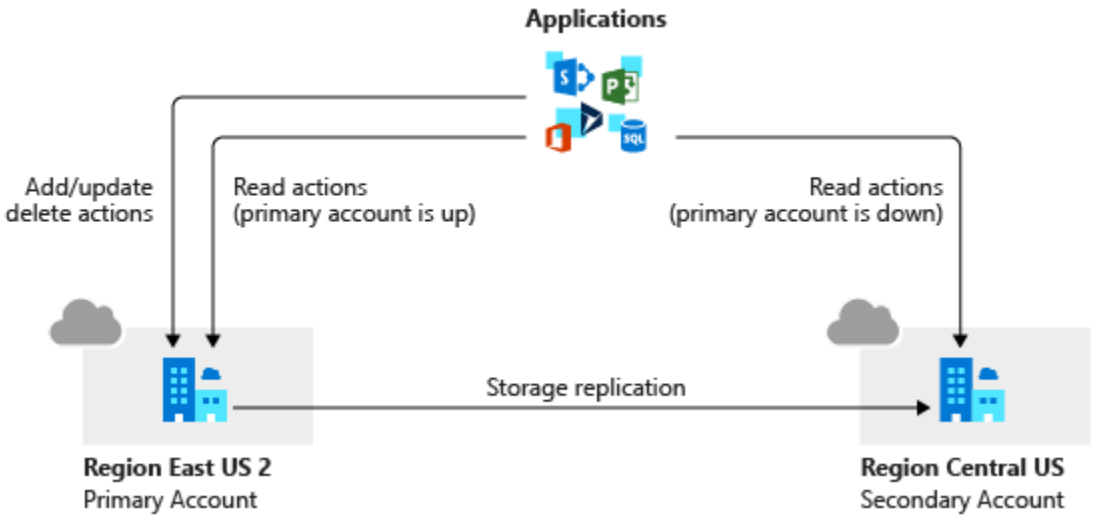


Geographically redundant, or geo-redundant, storage provides multiple levels of replication. Your data is replicated three times within the primary region, and then this set is replicated to a secondary region.

GRS provides the highest level of durability, because you finish with six copies of your data.

Keep in mind that your data in the secondary region is inaccessible until the primary region has failed across to the secondary region. At this point, the secondary region becomes the active region (primary), and your data becomes accessible.

What is read-access geo-redundant storage (RA-GRS)?



Geo-redundant storage provides 99.9999999999999999 percent durability, because it replicates data and objects to a secondary region. When failover starts, DNS entries that point to the primary region are updated to point to the secondary region. Microsoft currently controls the DNS failover process.

What is geo-zone-redundant storage (GZRS)?

Geo-zone-redundant storage (GZRS) combines the high availability benefits of ZRS with GRS. With this replication type, your data is copied across three availability zones in one region. Data is also replicated three times to another secondary region that's paired with it. This way, your zone-redundant data is also secure from regional level outage.

What is read-access geo-zone-redundant storage (RA-GZRS)?

Read-access geo-zone-redundant storage (RA-GZRS) uses the same replication method as GZRS but lets you read from the secondary region. If you want to read the data that's replicated to the secondary region, even if your primary isn't experiencing downtime, use RA-GZRS for your replication type.

When to use each type of redundant storage

The most appropriate use of each type of redundant storage is summarized in the following table:

Replication type	Copies	Use case
LRS	3	Data remains highly available, but for compliance reasons, isn't allowed to leave the local datacenter.
ZRS	3	Need redundancy in multiple physical locations, but because of compliance, data isn't allowed to leave a region.
GRS	6	App has access to the data, even if an entire region has an outage.
RA-GRS	6	App reads from multiple geographical locations, so you can serve users from a location that's closer to them.
GZRS	6	App can access data, even if the primary region has failed, and your secondary region has a datacenter that's experiencing an outage. But you don't want to read from the secondary region unless the primary region is down.
RA-GZRS	6	Regularly read data from your secondary region, perhaps to serve users from a location closer to them, even if a datacenter is up in your primary region.

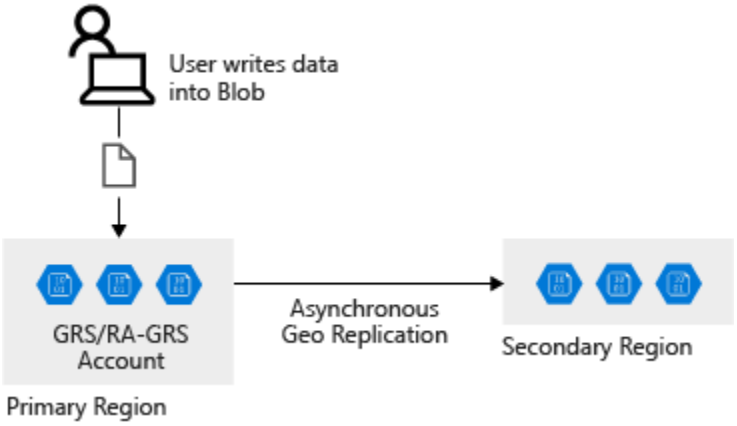
Change replication strategy

After you've created an Azure storage account, you can change the replication strategy. You can switch the replication status of a storage account from LRS to GRS, or LRS to RA-GRS, and back again. To change the replication strategy to GZRS, the process you use depends on the current replication strategy for your account.

Design HA applications to handle disaster recovery

How an account failover works

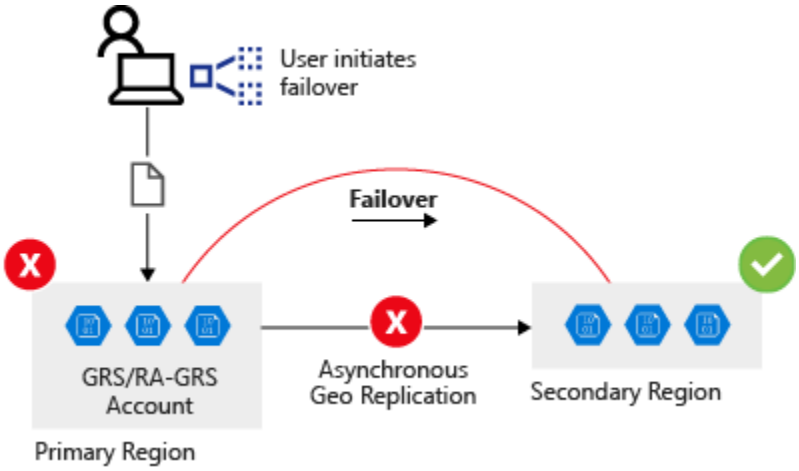
When you configure a storage account GRS or RA-GRS, the client writes data to the primary endpoint or region. The data is then automatically replicated across to the secondary region, as shown in the following image:



If the primary region that hosts your geo-redundant storage becomes unavailable, you can fail over to the secondary region.

When failover occurs, the secondary region becomes your new primary region, and all data is then accessible from this new primary region. All DNS records, which relate to your storage account, have their DNS endpoints updated to point to the new primary region. This redirection requires no changes to your application code.

A failure in the primary region is shown in the following image:



Implications of a storage account failover

When a storage account failover occurs, you could lose data. Data from the current primary region might not replicate across to the secondary region at the time of the failover. To determine whether there's likely to be data loss, you can check the **Last Sync Time** property. You used the command for finding this value in the previous exercise to review the storage account replication status.

Design a resilient application

When you design your application, consider the following factors:

- **Resiliency**: The ability to recover from a failure and continue to function, to avoid downtime and data loss.
- **High availability**: The ability to continue to function in a healthy state in the event of a hardware fault, a server fault, or network issues that affect one or more components of the application.
- **Disaster recovery**: The ability to recover if a major incident affects the services that host the application, such as a datacenter outage or complete regional outage. Disaster recovery includes manually failing over an application by using Azure Site Recovery. With Azure Site Recovery, you can fail over servers between Azure regions or Azure backups. You can then restore a database or application from a backup.
- **Eventual consistency**: RA-GRS works by replicating data from the primary endpoint to the secondary endpoint. The data, which is replicated between the regions, isn't immediately available at the secondary location. Eventual consistency means that all the transactions on the primary region will eventually appear in the secondary region. The data isn't lost, but there might be some lag.

  The following table shows the effects of eventual consistency on the healthcare system. When new or updated records are written to the primary region, the latest records are immediately available in the primary storage location. These updates are eventually propagated to the secondary regions, but there might be a delay before propagation occurs. An application that reads data from a secondary location might see out-of-date data for a short while.

Time	Transaction	Replication	Last sync time	Result
T0	Doctor adds patient record			Transaction is added but not replicated
T1		Record is replicated	T1	Last Sync Time field is updated
T2	Consultant updates patient record		T1	Record is updated in primary region but not replicated

T3	Read record from secondary region			Data from the secondary region is stale, because updates haven't yet been replicated from the primary region
T4		Record replicated	T1	Data at secondary region is now updated; Last Sync Time field is updated

Best practices for cloud-based applications with RA-GRS

When you develop applications for the cloud, consider the guidelines in the next sections.

Retry transient failures

Transient failures can be caused by a number of conditions from a disconnected database, temporary loss of network, or latency issues that cause slow response times from services. Applications must detect the faults and determine whether it's merely a blip in the service or a more severe outage. The application must have the capability to retry a service if it believes the fault is likely to be transient, before listing it as failed.

Handle failed writes

RA-GRS replicates writes across locations. If the primary location fails, you can direct read operations toward the secondary location. However, this secondary location is read-only. If a long-lasting outage (more than a few seconds) occurs at the primary location, your application must run in read-only mode. You can achieve read-only mode in several ways:

- Temporarily return an error from all write operations until write capability is restored.
- Buffer write operations, perhaps by using a queue, and enact them later when the write location becomes available.
- Write updates to a different storage account in another location. Merge these changes into the storage account at the primary location when it becomes available.
- Trigger the application to disable all write operations, and inform the user that the application is running in read-only mode. You can also use this mechanism if you need to upgrade the application and ensure that no-one is using the primary location while the upgrade is taking place.

An application that uses the Azure Storage client library can set the *LocationMode* of a read request to one of the following values:

- **PrimaryOnly**: The read request fails if the primary location is unavailable. This failure is the default behavior.
- **PrimaryThenSecondary**: Try the primary location first, and then try the secondary location if the primary location is unavailable. Fail if the secondary location is also unavailable.
- **SecondaryOnly**: Try only the secondary location, and fail if it's not available.
- **SecondaryThenPrimary**: Try the secondary location first, and then try the primary location.

Handle eventual consistency

Be prepared to handle stale data if it's read from a secondary region. As previously described, it takes time to replicate data between regions, and an outage can occur between the time when data is written to the primary location and it's replicated to each secondary location.

Use the Circuit Breaker pattern

In distributed environments, communication between remote resources can fail because of slow network connections, resources timeouts, resources being offline, or a transmission problem corrupting data in transit. Most of the time, these issues are transient and resolve themselves. If the application retries the same operation, it often succeeds.

In some situations, when an outage is severe, it makes sense for the application to stop retrying the operation and instead start failover to a secondary site.

To prevent an application from retrying operations that have failed, you can implement the Circuit Breaker pattern.

The Circuit Breaker pattern forces the application to fail over to the secondary site, which allows the application to resume its normal service. At the same time, the circuit breaker continues to check on whether the resources on the primary site are back online. And when they do come online, it allows the application to reconnect to the primary site. The circuit breaker acts as a proxy. It monitors the service, and if there's a failure in the service, it prevents the application from retrying that endpoint and forces it to go to an alternative endpoint.

The difference between the Circuit Breaker pattern and the Retry pattern is that the Retry pattern allows an application to keep retrying a connection to a resource, which might be offline. The Circuit Breaker pattern prevents this behavior and fails over the application to the secondary connection.

The purpose of implementing a Circuit Breaker pattern is to provide stability to your application while the system recovers from a failure.

Use the Circuit Breaker pattern to prevent an application from trying connections to resources that have failed and, instead, to minimize disruption by redirecting the connection to working resources. Don't use the Circuit Breaker pattern for accessing local or in-memory data structures, because circuit breakers would add overhead to the system.

When you implement the Circuit Breaker pattern, set the *LocationMode* of read requests appropriately. Most of the time, you should set this mode to *PrimaryThenSecondary*. If the read from the primary location times out, the secondary location is used. However, this process can slow down an application if it's done repeatedly. After the circuit breaker has detected that the primary location is unavailable, it should switch the mode to *SecondaryOnly*. This action ensures that read operations don't wait for a timeout from the primary location before trying the secondary location. When the circuit breaker estimates that the primary location has been repaired, it can revert back to the *PrimaryThenSecondary* mode.

Secure your Azure Storage account

Azure Storage accounts provide several high-level security benefits for the data in the cloud:

- Protect the data at rest
- Protect the data in transit
- Support browser cross-domain access
- Control who can access data
- Audit storage access

Encryption at rest

All data written to Azure Storage is automatically encrypted by Storage Service Encryption (SSE) with a 256-bit Advanced Encryption Standard (AES) cipher, and is FIPS 140-2 compliant. SSE automatically encrypts data when writing it to Azure Storage. When you read data from Azure Storage, Azure Storage decrypts the data before returning it. This process incurs no additional charges and doesn't degrade performance. It can't be disabled.

For virtual machines (VMs), Azure lets you encrypt virtual hard disks (VHDs) by using Azure Disk Encryption. This encryption uses BitLocker for Windows images, and it uses dm-crypt for Linux.

Azure Key Vault stores the keys automatically to help you control and manage the disk-encryption keys and secrets. So even if someone gets access to the VHD image and downloads it, they can't access the data on the VHD.

Encryption in transit

Keep your data secure by enabling *transport-level security* between Azure and the client. Always use *HTTPS* to secure communication over the public internet. When you call the REST APIs to access objects in storage accounts, you can enforce the use of HTTPS by requiring secure transfer for the storage account. After you enable secure transfer, connections that use HTTP will be refused. This flag will also enforce secure transfer over SMB by requiring SMB 3.0 for all file share mounts.

CORS support

Contoso stores several website asset types in Azure Storage. These types include images and videos. To secure browser apps, Contoso locks GET requests down to specific domains.

Azure Storage supports cross-domain access through cross-origin resource sharing (CORS). CORS uses HTTP headers so that a web application at one domain can access resources from a server at a different domain. By using CORS, web apps ensure that they load only authorized content from authorized sources.

CORS support is an optional flag you can enable on Storage accounts. The flag adds the appropriate headers when you use HTTP GET requests to retrieve resources from the Storage account.

Role-based access control

To access data in a storage account, the client makes a request over HTTP or HTTPS. Every request to a secure resource must be authorized. The service ensures that the client has the permissions required to access the data. You can choose from several access options. Arguably, the most flexible option is role-based access.

Azure Storage supports Azure Active Directory and role-based access control (RBAC) for both resource management and data operations. To security principals, you can assign RBAC roles that are scoped to the storage account. Use Active Directory to authorize resource management operations, such as configuration. Active Directory is supported for data operations on Blob and Queue storage.

To a security principal or a managed identity for Azure resources, you can assign RBAC roles that are scoped to a subscription, a resource group, a storage account, or an individual container or queue.

Auditing access

Auditing is another part of controlling access. You can audit Azure Storage access by using the built-in Storage Analytics service.

Storage Analytics logs every operation in real time, and you can search the Storage Analytics logs for specific requests. Filter based on the authentication mechanism, the success of the operation, or the resource that was accessed.

Types of shared access signatures

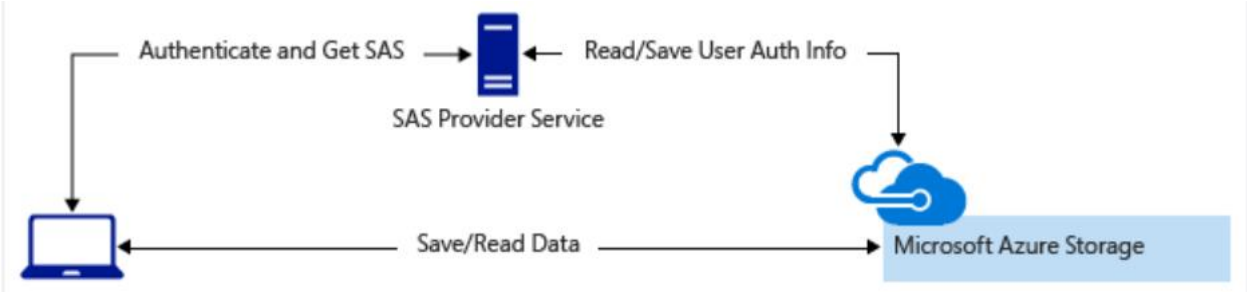
You can use a *service-level* SAS to allow access to specific resources in a storage account. You'd use this type of SAS, for example, to allow an app to retrieve a list of files in a file system, or to download a file.

Use an *account-level* SAS to allow access to anything that a service-level SAS can allow, plus additional resources and abilities. For example, you can use an account-level SAS to allow the ability to create file systems.

You'd typically use a SAS for a service where users read and write their data to your storage account. Accounts that store user data have two typical designs:

- Clients upload and download data through a front-end proxy service, which performs authentication. This front-end proxy service has the advantage of allowing validation of business rules. But, if the service must handle large amounts of data or high-volume transactions, you might find it complicated or expensive to scale this service to match demand.

- A lightweight service authenticates the client, as needed. Next, it generates a SAS. After receiving the SAS, the client can access storage account resources directly. The SAS defines the client's permissions and access interval. It reduces the need to route all data through the front-end proxy service.



Understand advanced threat protection for Azure Storage

Azure Defender for Storage provides an extra layer of security intelligence that detects unusual and potentially harmful attempts to access or exploit storage accounts. This layer of protection allows you to address threats without being a security expert or managing security monitoring systems.

Security alerts are triggered when anomalies in activity occur. These security alerts are integrated with Azure Security Center, and are also sent via email to subscription administrators, with details of suspicious activity and recommendations on how to investigate and remediate threats.

Azure Defender for Storage is currently available for Blob storage, Azure Files, and Azure Data Lake Storage Gen2. Account types that support Azure Defender include general-purpose v2, block blob, and Blob storage accounts. Azure Defender for Storage is available in all public clouds and US government clouds, but not in other sovereign or Azure Government cloud regions.

You can turn on Azure Defender for Storage in the Azure portal through the configuration page of the Azure Storage account, or in the advanced security section of the Azure portal. Follow these steps.

1. Launch the Azure portal.
2. Navigate to your storage account. Under **Settings**, select **Advanced security**.
3. Select **Enable Azure Defender for Storage**.

Explore security anomalies

When storage activity anomalies occur, you receive an email notification with information about the suspicious security event. Details of the event include:

- Nature of the anomaly
- Storage account name
- Event time
- Storage type
- Potential causes
- Investigation steps
- Remediation steps
- Email also includes details about possible causes and recommended actions to investigate and mitigate the potential threat

Store and share files in your app with Azure Files

Choose your data access method

There are two built-in methods of data access supported by Azure Files. One method is direct access via a mounted drive in your operating system. The other method is to use a Windows server (either on-premises or in Azure) and install Azure File Sync to synchronize the files between local shares and Azure Files.

The most common scenario that might lead you to consider using Azure File Sync is to run your applications or compute resources locally on on-premises Windows machines. If the office also has a slow internet connection, it increases the need to run Azure File Sync for performance reasons.

The primary use of direct cloud access is where the apps are running on Azure and all locations that need access to the data have a fast internet connection.

Choose your file redundancy option

Because Azure Files stores files in a storage account, you can choose between standard or premium performance storage accounts:

- **Standard performance**: Double-digit ms latency, 10,000 IOPS, 300-MBps bandwidth
- **Premium performance**: Single-digit ms latency, 100,000 IOPS, 5-GBps bandwidth

Standard performance accounts use HDD to store data. With HDD, the costs are lower but so is the performance. SSD arrays back the premium storage account's performance, which comes with higher costs. Currently, premium accounts can only use file storage accounts with ZRS storage in a limited number of regions.

You need to balance the availability and performance requirements to decide on the account and redundancy options. The finance company is more concerned with the security of their data than performance, and they want the most resilience possible. As a result, the best choice is a standard GRS storage account. When GZRS leaves preview and becomes generally available, it will be the best option for them. This table compares the different characteristics of each storage option.

LRS	ZRS	GRS/GZRS		Result
You can easily re-create data, and cost is a priority.	✔			Transaction is added but not replicated
Data must be stored in a single known location.	✔			Last Sync Time field is updated
Premium performance is required.	✔	✔*		Record is updated in primary region but not replicated
Data needs to be highly available, and redundancy is a priority.		✔	✔	Data from the secondary region is stale, because updates haven't yet been replicated from the primary region
99.999999999% (11 nines) durability.	✔			Data at secondary region is now updated; Last Sync Time field is updated
99.9999999999% (12 nines) durability.		✔		
99.99999999999999% (16 nines) durability.			✔	

Choose your data migration solution

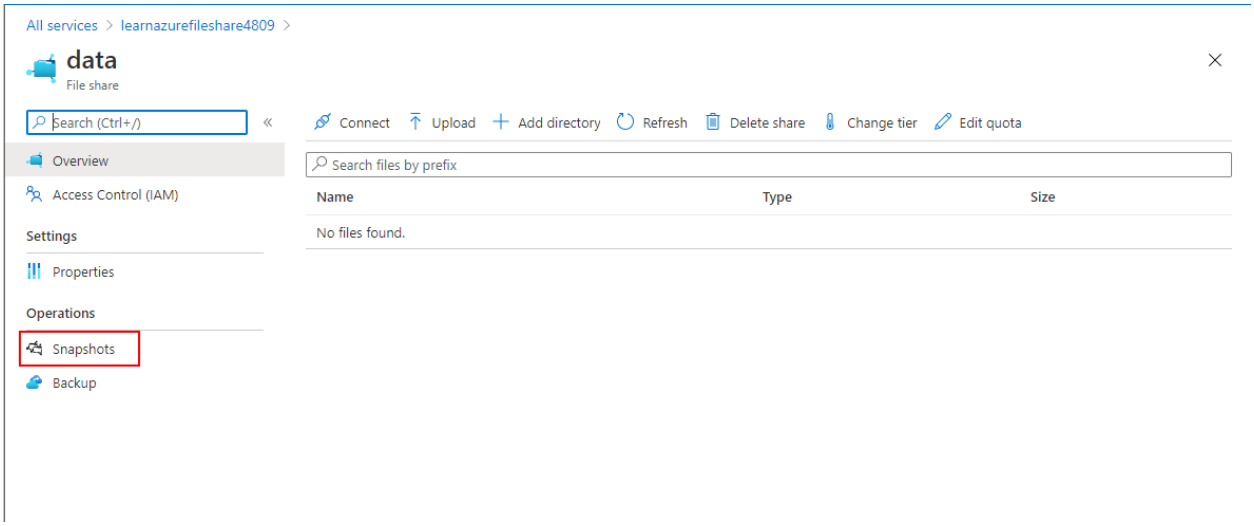
Utility	Description
AzCopy	Command-line tool that offers the best performance, especially for a low volume of small files.
Robocopy	Command-line tool shipped with Windows and Windows Server. AzCopy is written to be Azure aware and performs better.
Azure Storage Explorer	Graphical file management utility that runs on Windows, Linux, and macOS.
Azure portal	Use the portal to import files and folders.
Azure File Sync	Can be used to do the initial data transfer, and then uninstalled after the data is transferred.
Azure Data Box	If you have up to 35 TB of data and you need it imported in less than a week.

Use share snapshots to protect against accidental deletion

Share snapshots give you an extra level of security and help reduce the risk of data corruption or accidental deletion. You can also use them as a general backup for disaster recovery.

The snapshots are easy to create in the Azure portal or with the REST API, client libraries, the Azure CLI, and PowerShell.

Snapshots are at the root level of a file share and apply to all the folders and files contained in it.



Choose the right disk storage for your virtual machine workload

Disk roles

Each disk can take one of three roles in a virtual machine:

- **OS disk**. One disk in each virtual machine contains the operating system files. When you create a virtual machine, you select a virtual machine image and that fixes the operating system and the OS disk that's attached to the new machine. The OS disk has a maximum capacity of 2,048 GB.
- **Data disk**. You can add one or more data virtual disks to each virtual machine to store data. For example, database files, website static content, or custom application code should be stored on data disks. The number of data disks you can add depends on the virtual machine size. Each data disk has a maximum capacity of 32,767 GB.
- **Temporary disk**. Each virtual machine contains a single temporary disk, which is used for short-term storage applications such as page files and swap files. The contents of temporary disks are lost during maintenance events, so don't use these disks for critical data. These disks are local to the server and aren't stored in a storage account.

Ephemeral OS disks

An ephemeral OS disk is a virtual disk that saves data on the local virtual machine storage. An ephemeral disk has faster read-and-write latency than a managed disk. It's also faster to reset the image to the original boot state if you're using an ephemeral disk. However, an individual virtual machine failure might destroy all the data on an ephemeral disk and leave the virtual machine unable to boot. Because ephemeral disks reside locally to the host, they incur no storage costs and are free.

Ephemeral disks work well when you want to host a stateless workload, such as the business logic for a multitier website or a microservice. Such applications are tolerant of individual virtual machine failures, because requests can be rerouted to other virtual machines in the system. You can reset the failed virtual machine to its original boot state rapidly and get it back up and running faster than if it used managed disks.

Managed disks

Most disks that you use with virtual machines in Azure are managed disks. A managed disk is a virtual hard disk for which Azure manages all the required physical

infrastructure. Because Azure takes care of the underlying complexity, managed disks are easy to use. You can just provision them and attach them to virtual machines.

Virtual hard disks in Azure are stored as page blobs in an Azure Storage account, but you don't have to create storage accounts, blob containers, and page blobs yourself or maintain this infrastructure later. The benefits of managed disks include:

- **Simple scalability.** You can create up to 50,000 managed disks of each type in each region in your subscription.
- **High availability.** Managed disks support 99.999% availability by storing data three times. If there's a failure in one replica, the other two can maintain full read-write functionality.
- **Integration with availability sets and zones.** If you place your virtual machines into an availability set, Azure automatically distributes the managed disks for those machines into different fault domains so that your machines are resilient to localized failures. You can also use availability zones, which distribute data across multiple datacenters, for even greater availability.
- **Support for Azure Backup.** Azure Backup natively supports managed disks, which includes encrypted disks.
- **Granular access control.** You can use Azure role-based access control (RBAC) to grant access to specific user accounts for specific operations on a managed disk. For example, you could ensure that only an administrator can delete a disk.
- **Support for encryption.** To protect sensitive data on a managed disk from unauthorized access, you can encrypt it by using Azure Storage Service Encryption (SSE), which is provided with Azure Storage accounts. Alternatively, you can use Azure Disk Encryption (ADE), which uses BitLocker for Windows virtual machines, and DM-Crypt for Linux virtual machines.

Because of all these features, use managed disks by default.

Unmanaged disks

An unmanaged disk, like a managed disk, is stored as a page blob in an Azure Storage account. The difference is that with unmanaged disks, you create and maintain this storage account manually. This requirement means that you have to keep track of IOPS limits within a storage account and ensures that you don't overprovision throughput of your storage account. You must also manage the security and RBAC access at the storage account level, instead of at each individual disk with managed disks.

Because unmanaged disks don't support all of the scalability and management features that you've seen for managed disks, they're no longer widely used. Consider using them

only if you want to manually set up and manage the data for your virtual machine in the storage account.

In the portal, to use unmanaged disks, expand the **Advanced** section on the **Disks** page of the **Create a virtual machine** wizard.

Disk performance measures

To choose the right disk type, it's critical to understand its performance. Performance is expressed in two key measures:

- **Input/output operations per second (IOPS)**. IOPS measure the rate at which the disk can complete a mix of read and write operations. Higher performance disks have higher IOPS values.
- **Throughput**. Throughput measures the rate at which data can be moved onto the disk from the host computer and off the disk to the host computer. Throughput is also called *data transfer rate* and is measured in megabytes per second (MBps). Higher performance disks have higher throughput.

For physical disks, solid-state disks (SSDs) usually realize higher IOPS and throughput than hard disk drives (HDDs). Virtual disks that you can choose for an Azure virtual machine are based on SSDs of several types or HDDs. Their performance varies widely based on the disk type that you choose.

Ultra SSD

Ultra SSDs provide the highest disk performance available in Azure. Choose them when you need the fastest storage performance, which includes high throughput, high IOPS, and low latency.

Ultra disks can have capacities from 4 GB up to 64 TB. A unique feature of ultra disks is that you can adjust the IOPS and throughput values while they're running and without detaching them from the host virtual machine. Performance adjustments can take up to an hour to take effect.

Ultra disks are a new disk type and currently have some limitations:

- They're only available in a subset of Azure regions.
- They can only be attached to virtual machines that are in availability zones.
- They can only be attached to ES/DS v3 virtual machines.
- They can only be used as data disks and can only be created as empty disks.

- They don't support disk snapshots, virtual machine images, scale sets, Azure Disk Encryption, Azure Backup, or Azure Site Recovery.

Some workloads place intensive loads on disk storage. For example, top-tier databases and SAP HANA need fast performance and are transaction heavy. If you have such a workload, and if premium SSDs have caused performance bottlenecks, consider using ultra SSDs.

Premium SSD

Premium SSDs are the next tier down from ultra disks in terms of performance, but they still provide high throughput and IOPS with low latency. Premium disks don't have the current limitations of ultra disks. For example, they're available in all regions and can be used with virtual machines that are outside of availability zones.

You can't adjust performance without detaching these disks from their virtual machine. Also, you can only use premium SSDs with larger virtual machine sizes, which are compatible with premium storage.

With premium SSDs, these performance figures are guaranteed. There's no such guarantee for standard tier disks, which can be impacted occasionally by high demand.

If you need higher performance than standard disks provide, or if you can't sustain occasional drops in performance, use premium SSDs. Also use premium SSDs when you want the highest performance but can't use ultra disks because of their current limitations. Premium SSDs are a good fit for mission-critical workloads in medium and large organizations.

You can migrate a disk to a premium SSD at any time, if you've found that its performance isn't good enough.

Standard SSD

Standard SSDs in Azure are a cost-effective storage option for virtual machines that need consistent performance at lower speeds. Standard SSDs aren't as fast as premium or ultra SSDs, but they still have latencies in the range of 1 millisecond to 10 milliseconds and up to 6,000 IOPS. They're available to attach to any virtual machine, no matter what size it is.

Performance figures aren't guaranteed but are achieved 99% of the time.

Use standard SSDs when you have budgetary constraints and a workload that isn't disk intensive. For example, web servers, lightly used enterprise applications, and test servers can all run on standard SSDs.

Standard HDD

If you choose to use standard HDDs, data is stored on conventional magnetic disk drives with moving spindles. Disks are slower and speeds are more variable than for SSDs, but latencies are under 10 ms for write operations and 20 ms for reads. As for standard SSDs, you can use standard HDDs for any virtual machine.

Use standard HDDs when you want to minimize costs for less critical workloads and development or test environments.

Monitor, diagnose, and troubleshoot your Azure storage

Understand Storage Analytics metrics

Storage Analytics captures transactional metrics for all types of storage and capacity metrics for blob storage. The metrics provide an aggregated view of the work requested by applications in the storage account. If you need detail on individual requests, enable storage logging.

Transaction metrics

Storage operations are transactional. If a write operation fails, its effects are undone. If a write succeeds, the changes are committed to storage. The metrics summarize information about each type of request, whether it's a read, a write, or a delete operation, and whether the operation succeeded or failed, and if it failed, what error occurred. Storage analytics will also record information about rates of ingress and egress to and from storage, and storage availability. This information is useful if errors occur because storage is temporarily unavailable. Transactions are summarized for each type of storage service (blob, file, queue, table) by default, but you can also choose to summarize metrics for each storage API call. For example, if you enable API metrics, you'll see metrics for API calls such as `GetBlob`, `PutBlob`, `ListBlobs`, and so on, as well as the overall metrics for the service.

At the service level, operations are aggregated hourly, even with zero requests made to the service. At the API operation level, statistics are only recorded when invoking an operation. For example, if you do a `GetBlob` operation in your blob storage, Storage Analytics will log a request and include it in the aggregated data for the Blob service as well as `GetBlob` operation. However, if no `GetBlob` operation is requested within the hour, then no summary will be generated for that operation.

Capacity metrics

Capacity metrics are only available for blob storage. Capacity data is sent to the analytics service hourly, but the aggregated statistics are only updated daily.

The statistics reported are:

- **Capacity**, which records the sum total of the storage occupied by the blobs in your storage account.

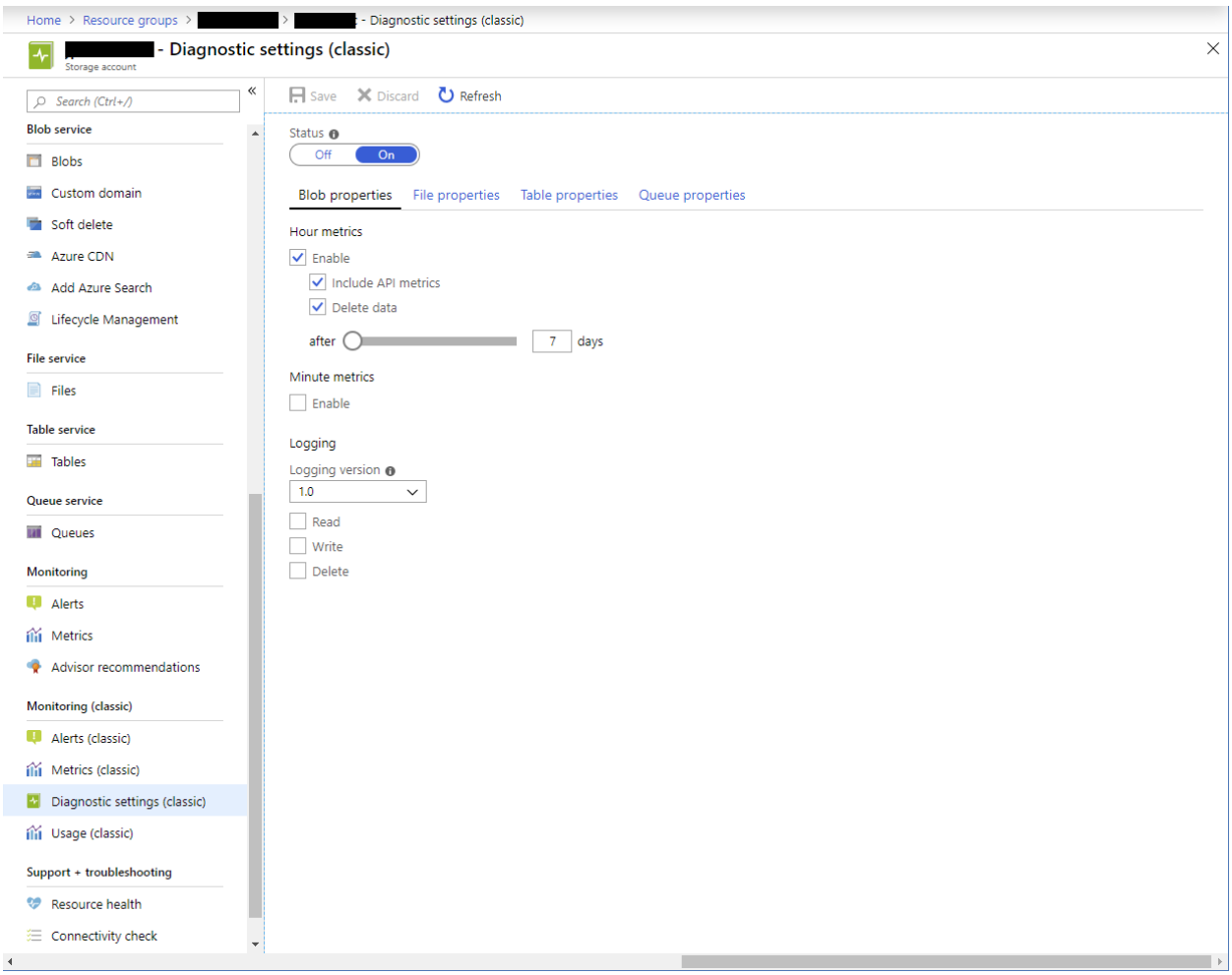
- **Container count**, which shows how many containers have been created across blob storage in your storage account.
- **Object count**, which displays the total number of blobs created across all of your containers.

Capture Storage Analytics metrics

Storage Analytics records aggregated transaction statistics and capacity data, about requests made to the Azure Storage service. The metrics data is stored in a set of tables in your storage account. Each table is named after the type of storage monitored and whether the transaction occurred in primary or secondary storage. For example, the metrics for blob storage are stored in tables named **$MetricsHourPrimaryTransactionsBlob** and **$MetricsHourSecondaryTransactionsBlob**. Similarly, the metrics for table storage are recorded in tables named **$MetricsHourPrimaryTransactionsTable** and **$MetricsHourSecondaryTransactionsTable**. Capacity metrics, which are only available for blob storage, are held in the **$MetricsCapacityBlob** table.

Metrics capture for all storage types (blob, file, table, queue) is automatically enabled when you create a new storage account. You can selectively disable (and re-enable) metrics captured in several ways:

- Using the Azure portal. Select **Diagnostic settings (classic)** on the page for your storage account, select the storage type and change the **Status** to **On** or **Off** as required. By default, the data captured includes the metrics for each type of storage API call. You can also set the retention period for the metrics data. When this period expires, the metrics are deleted from table storage.



- Using PowerShell. The `Set-AzureStorageServiceMetricsProperty` cmdlet enables you to enable and disable metrics capture for each type of storage in your account.

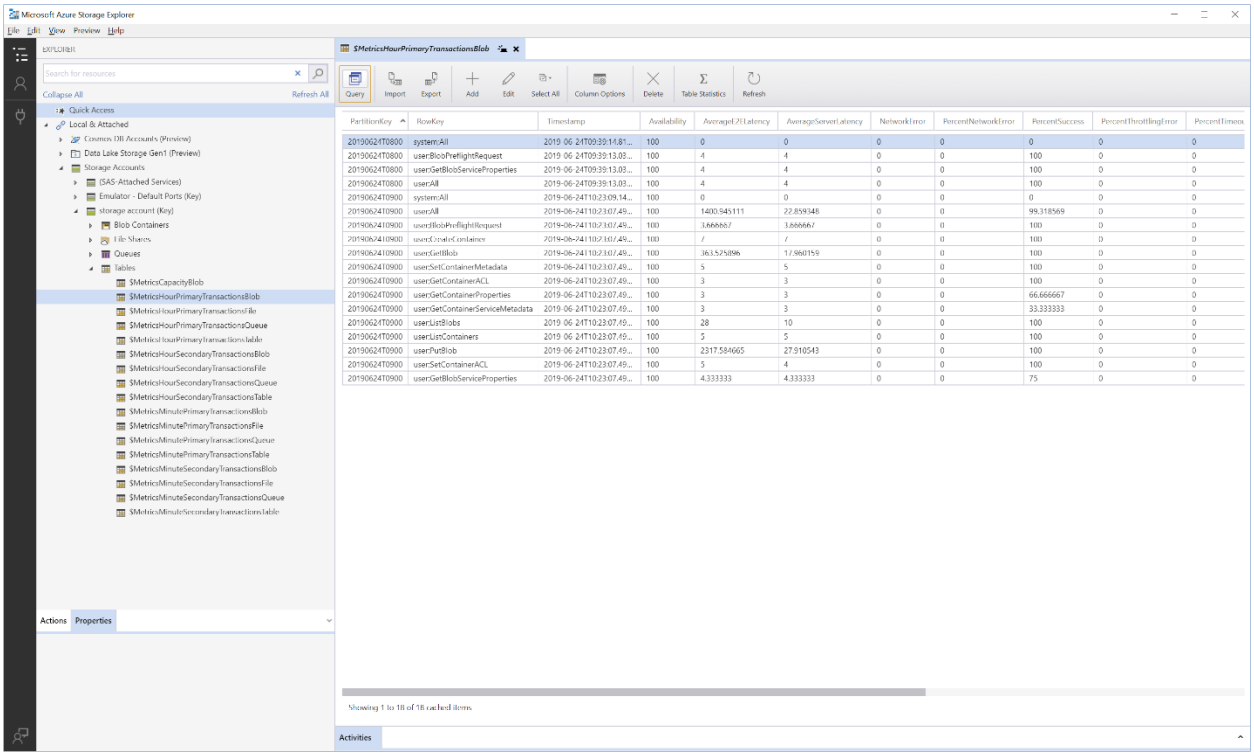
View Storage Analytics metrics

As described earlier, you can view the metrics gathered for a storage account using the **Metrics** page for the account in the Azure portal. This pane displays the information captured by Azure Monitor. This data is managed separately and isn't derived from the statics held in the **$Metrics** tables in your storage account.

The **Overview** page provides a useful overview of the performance of your storage account, but you may need more detail, especially if there are errors and failures in your applications. You can see a more detailed breakdown of the metrics by examining the data in the **$Metrics** tables directly. A useful tool for gaining a quick view of this data is the desktop version of Azure Storage Explorer. The statistics gathered are broken down

by requester (**user** for requests made by applications, and **system** for requests made by the Storage Analytics service). If you haven't captured metrics by API, you will see **user: All** and **system: All** for each hour's worth of metrics.

If you have chosen to capture metrics by API, you'll see aggregated data for each API for each hour (or minute, if you're collecting minute-level metrics). There are a large number of values gathered for each API, including the success and failure rate, and the reasons for failure, such as timeouts, throttling, network errors, authorization failure, and so on. This information can give you a good insight as to why the performance of your applications may be suffering. For example, frequent throttling and timeout errors can indicate a high level of contention occurring for limited resources, and you might need to rearchitect your system to the use **Premium** rather than the **Standard** tier for your storage accounts. You might also need to spread the load across multiple storage accounts or select a different organization for any blob containers and tables that your application is using.



Azure Storage Analytics logging

Storage Analytics logs detailed information about requests to your storage service. This information is used to monitor individual requests and to diagnose issues with a storage service. Log entries are created when requests are made against the service endpoint.

However, requests are recorded on a *best effort* basis. Occasionally, a small number of requests might not be logged, and others might be duplicated. You shouldn't treat the log as a definitive list of operations, rather a means of examining the workload for the storage account.

Storage Analytics logs record authenticated and anonymous requests made to a storage account. The following types of authenticated requests are logged:

- Successful requests
- Failed requests, including timeout, throttling, network, authorization, and other errors
- Requests using a Shared Access Signature (SAS) or OAuth, including failed and successful requests

Details for the following types of anonymous requests are also recorded:

- Successful requests
- Server errors
- Time out errors for both client and server
- Failed GET requests with error code 304 (*Not Modified*)

Requests made by the Storage Analytics service itself aren't recorded.

Diagnostic data collected by the Storage Analytics service is written to blobs in the **$logs** blob container in your storage account. The data in the container is structured as a hierarchical collection of blobs, organized by storage service (blob, file, queue, table), and the date, in the format **<storage service>/YYYY/MM/DD/hhmm/<counter>.log**.

Enable Storage Analytics logging

Unlike Storage Analytics metrics, Storage Analytics logging isn't enabled by default for your storage account. You can enable it in the Azure portal, from the **Diagnostic settings (classic)** page. You can specify whether you want to record read, write, and delete operations performed against your storage account. You enable logging individually for each service (blob, file, table, queue).

Explore Azure compute services

What is Azure Batch?

Azure Batch enables large-scale parallel and high-performance computing (HPC) batch jobs with the ability to scale to tens, hundreds, or thousands of VMs.

When you're ready to run a job, Batch does the following:

- Starts a pool of compute VMs for you.
- Installs applications and staging data.
- Runs jobs with as many tasks as you have.
- Identifies failures.
- Requeues work.
- Scales down the pool as work completes.

There might be situations in which you need raw computing power or supercomputer-level compute power. Azure provides these capabilities.

Azure Container Instances

Azure Container Instances offers the fastest and simplest way to run a container in Azure without having to manage any virtual machines or adopt any additional services. It's a platform as a service (PaaS) offering that allows you to upload your containers, which it runs for you.

When to use Azure Functions

Serverless computing includes the abstraction of servers, an event-driven scale, and micro-billing:

- **Abstraction of servers**: Serverless computing abstracts the servers you run on. You never explicitly reserve server instances. The platform manages that for you. Each function execution can run on a different compute instance. This execution context is transparent to the code. With serverless architecture, you deploy your code, which then runs with high availability.
- **Event-driven scale**: Serverless computing is an excellent fit for workloads that respond to incoming events. Events include triggers by:
    - Timers, for example, if a function needs to run every day at 10:00 AM UTC.

    - HTTP, for example, API and webhook scenarios.
    - Queues, for example, with order processing.
    - And much more.

  Instead of writing an entire application, the developer authors a function, which contains both code and metadata about its triggers and bindings. The platform automatically schedules the function to run and scales the number of compute instances based on the rate of incoming events. Triggers define how a function is invoked. Bindings provide a declarative way to connect to services from within the code.

- **Micro-billing**: Traditional computing bills for a block of time like paying a monthly or annual rate for website hosting. This method of billing is convenient but isn't always cost effective. Even if a customer's website gets only one hit a day, they still pay for a full day's worth of availability. With serverless computing, they pay only for the time their code runs. If no active function executions occur, they're not charged. For example, if the code runs once a day for two minutes, they're charged for one execution and two minutes of computing time.

Azure has two implementations of serverless compute:

- **Azure Functions**: Functions can execute code in almost any modern language.
- **Azure Logic Apps**: Logic apps are designed in a web-based designer and can execute logic triggered by Azure services without writing any code.

Functions vs. Logic Apps

Functions and Logic Apps can both create complex orchestrations. An orchestration is a collection of functions or steps that are executed to accomplish a complex task.

- With Functions, you write code to complete each step.
- With Logic Apps, you use a GUI to define the actions and how they relate to one another.

You can mix and match services when you build an orchestration, calling functions from logic apps and calling logic apps from functions. Here are some common differences between the two.

	Functions	Logic Apps
State	Normally stateless, but Durable Functions provide state.	Stateful.
Development	Code-first (imperative).	Designer-first (declarative).
Connectivity	About a dozen built-in binding types. Write code for custom bindings.	Large collection of connectors. Enterprise Integration Pack for B2B scenarios. Build custom connectors.
Actions	Each activity is an Azure function. Write code for activity functions.	Large collection of ready-made actions.
Monitoring	Azure Application Insights.	Azure portal, Log Analytics.
Management	REST API, Visual Studio.	Azure portal, REST API, PowerShell, Visual Studio.
Execution context	Can run locally or in the cloud.	Runs only in the cloud.

What is Windows Virtual Desktop?

Windows Virtual Desktop on Azure is a desktop and application virtualization service that runs on the cloud. It enables your users to use a cloud-hosted version of Windows from any location. Windows Virtual Desktop works across devices like Windows, Mac, iOS, Android, and Linux. It works with apps that you can use to access remote desktops and apps. You can also use most modern browsers to access Windows Virtual Desktop-hosted experiences.

Provide the best user experience

Users have the freedom to connect to Windows Virtual Desktop with any device over the internet. They use a Windows Virtual Desktop client to connect to their published Windows desktop and applications. This client could either be a native application on the device or the Windows Virtual Desktop HTML5 web client.

You can make sure your session host virtual machines (VMs) run near apps and services that connect to your datacenter or the cloud. This way your users stay productive and don't encounter long load times.

User sign-in to Windows Virtual Desktop is fast because user profiles are containerized by using FSLogix. At sign-in, the user profile container is dynamically attached to the

computing environment. The user profile is immediately available and appears in the system exactly like a native user profile.

You can provide individual ownership through personal (persistent) desktops. For example, you might want to provide personal remote desktops for members of an engineering team. Then they can add or remove programs without impacting other users on that remote desktop.

Enhance security

Windows Virtual Desktop provides centralized security management for users' desktops with Azure Active Directory (Azure AD). You can enable multifactor authentication to secure user sign-ins. You can also secure access to data by assigning granular role-based access controls (RBACs) to users.

With Windows Virtual Desktop, the data and apps are separated from the local hardware. Windows Virtual Desktop runs them instead on a remote server. The risk of confidential data being left on a personal device is reduced.

User sessions are isolated in both single and multi-session environments.

Windows Virtual Desktop also improves security by using reverse connect technology. This connection type is more secure than the Remote Desktop Protocol. We don't open inbound ports to the session host VMs.

Simplified management

Windows Virtual Desktop is an Azure service, so it will be familiar to Azure administrators. You use Azure AD and RBACs to manage access to resources. With Azure, you also get tools to automate VM deployments, manage VM updates, and provide disaster recovery. As with other Azure services, Windows Virtual Desktop uses Azure Monitor for monitoring and alerts. This standardization lets admins identify issues through a single interface.

Performance management

Windows Virtual Desktop gives you options to load balance users on your VM host pools. *Host pools* are collections of VMs with the same configuration assigned to multiple users. For the best performance, you can configure load balancing to occur as

users sign in (breadth mode). With breadth mode, users are sequentially allocated across the host pool for your workload. To save costs, you can configure your VMs for depth mode load balancing where users are fully allocated on one VM before moving to the next. Windows Virtual Desktop provides tools to automatically provision additional VMs when incoming demand exceeds a specified threshold.

Multi-session Windows 10 deployment

Windows Virtual Desktop lets you use Windows 10 Enterprise multi-session, the only Windows client-based operating system that enables multiple concurrent users on a single VM. Windows Virtual Desktop also provides a more consistent experience with broader application support compared to Windows Server-based operating systems.

Bring your own licenses

Windows Virtual Desktop is available to you at no additional cost if you have an eligible Microsoft 365 license. Just pay for the Azure resources used by Windows Virtual Desktop.

- Bring your eligible Windows or Microsoft 365 license to get Windows 10 Enterprise and Windows 7 Enterprise desktops and apps at no additional cost.
- If you're an eligible Microsoft Remote Desktop Services Client Access License customer, Windows Server Remote Desktop Services desktops and apps are available at no additional cost.

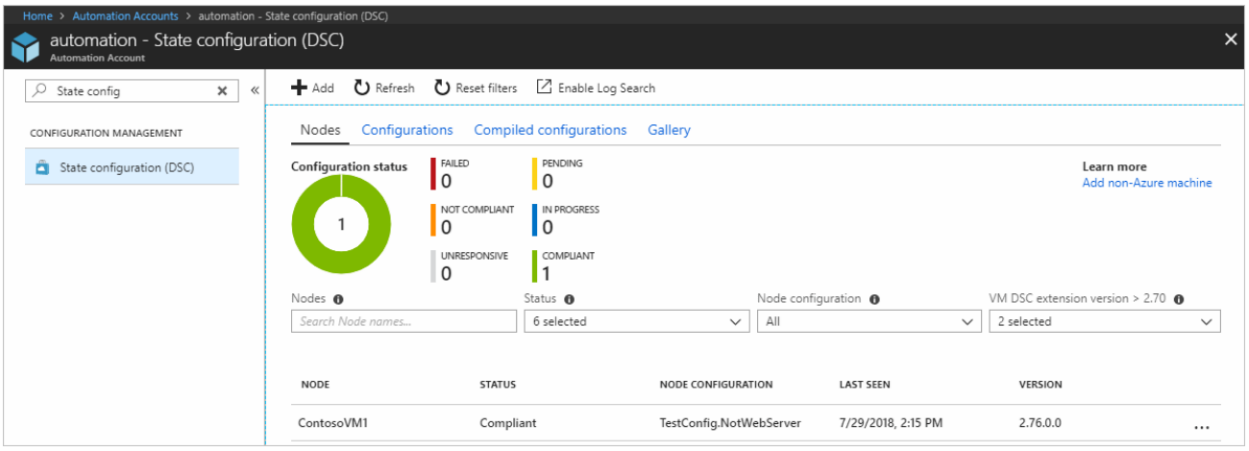
Save on compute costs

Buy one-year or three-year Azure Reserved Virtual Machine Instances to save you up to 72 percent versus pay-as-you-go pricing. You can pay for a reservation up front or monthly. Reservations provide a billing discount and don't affect the runtime state of your resources.

Choose a compute provisioning solution for your application

Azure automation state configuration

Azure automation state configuration is the service you use to make sure that your DSC configurations are managed properly and deployed across your nodes (virtual machines). Azure automation state configuration works with both Azure virtual machines and machines on-premises. It also works with machines on other cloud providers. Through an intuitive Azure portal process, you can apply configurations to all of your nodes.



Azure automation state configuration makes it possible for you to ensure that all target machines are assigned the correct configurations automatically. It also ensures that each machine reports back on what its current state is and shows whether it has achieved the desired state. You can send this information for reporting and for further decision making. You can interact with Azure automation state configuration through the Azure portal or through Azure PowerShell.

Custom script

- **Ease of setup.** The custom script extension is built into the Azure portal, so setup is easy.
- **Management.** The management of custom scripts can get tricky as your infrastructure grows and you accumulate different custom scripts for different resources.
- **Interoperability.** The custom script extension can be added into an Azure Resource Manager template. You can also deploy it through Azure PowerShell or the Azure CLI.

- **Configuration language.** You can write scripts by using many types of commands. You can use PowerShell and Bash.
- **Limitations and drawbacks.** Custom scripts aren't suitable if your script needs more than one and a half hours to apply your configuration. Avoid using custom scripts for any configuration that needs reboots.

Solution summary

The custom script extension is good for small configurations after provisioning. It's also good if you need to add or update some applications on a target machine quickly. It's imperative for ad-hoc cross-platform scripting.

Azure Desired State Configuration extensions

- **Ease of setup.** Desired State Configurations (DSCs) are easy to read, update, and store. Configurations define what state you want to achieve. The author doesn't need to know how that state is reached.
- **Management.** DSC democratizes configuration management across servers.
- **Interoperability.** DSCs are used with Azure Automation State Configuration. They can be configured through the Azure portal, Azure PowerShell, or Azure Resource Manager templates.
- **Configuration language.** Use PowerShell to configure DSC.
- **Limitations and drawbacks.** You can only use PowerShell to define configurations. If you use DSC without Azure Automation State Configuration, you have to take care of your own orchestration and management.

Solution summary

DSC is easy to read, update, and store. DSC configurations help you declare the state your machines should be in at the point they are provisioned, rather than having instructions that detail how to put the machines in a certain state. Without Azure Automation State Configuration, you have to manage your own DSC configurations and orchestration. DSC can achieve more when it's coupled with Azure Automation State Configuration.

Azure Automation State Configuration

- **Ease of setup.** Automation State Configuration isn't difficult to set up, but it requires the user to be familiar with the Azure portal.

- **Management.** The service manages all of the virtual machines for you automatically. Each virtual machine can send you detailed reports about its state, which you can use to draw insights from this data. Automation State Configuration also helps you to manage your DSC configurations more easily.
- **Interoperability.** Automation State Configuration requires DSC configurations. It works with your Azure virtual machines automatically, and any virtual machines that you have on-premises or on another cloud provider.
- **Configuration language.** Use PowerShell.
- **Limitations and drawbacks.** You can only use PowerShell to define configurations.

Solution summary

The Azure Automation State Configuration service is good for automating your DSC configurations, along with the management of machines that need those configurations, and getting centralized reporting back from each machine. You can use DSC without Azure Automation State Configuration, particularly if you want to administer a smaller number of machines. For larger and more complicated scenarios that need orchestration, Azure Automation State Configuration is the solution you need. All of the configurations and features that you need can be pushed to all of the machines, and applied equally, with minimal effort.

Azure Resource Manager templates

- **Ease of setup.** You can create Resource Manager templates easily. You have many templates available from the GitHub community, which you can use or build upon. Alternatively, you can create your own templates from the Azure portal.
- **Management.** Managing Resource Manager templates is straightforward because you manage JavaScript Object Notation (JSON) files.
- **Interoperability.** You can use other tools to provision Resource Manager templates, such as the Azure CLI, the Azure portal, PowerShell, and Terraform.
- **Configuration language.** Use JSON.
- **Limitations and drawbacks.** JSON has a strict syntax and grammar, and mistakes can easily render a template invalid. The requirement to know all of the resource providers in Azure and their options can be onerous.

Solution summary

Resource Manager templates can be used for small ad-hoc infrastructures. They're also ideal for deploying larger infrastructures with multiple services along with their dependencies. Resource templates can fit well into developers' workflows. You use the

same template to deploy your application repeatedly during every stage of the application lifecycle.

Chef

- **Ease of setup.** The Chef server runs on the master machine, and Chef clients run as agents on each of your client machines. You can also use hosted Chef and get started much faster, instead of running your own server.
- **Management.** The management of Chef can be difficult because it uses a Ruby-based domain-specific language. You might need a Ruby developer to manage the configuration.
- **Interoperability.** Chef server only works under Linux and Unix, but the Chef client can run on Windows.
- **Configuration language.** Chef uses a Ruby-based domain-specific language.
- **Limitations and drawbacks.** The language can take time to learn, especially for developers who aren't familiar with Ruby.

Solution summary

Chef is suitable for large-scale infrastructure deployment and configuration. Chef makes it easy for you to automate the deployment of an entire infrastructure, such as in the workflow of a development team.

Terraform

- **Ease of setup.** To get started with Terraform, download the version that corresponds with your operating system and install it.
- **Management.** Terraform's configuration files are designed to be easy to manage.
- **Interoperability.** Terraform supports Azure, Amazon Web Services, and Google Cloud Platform.
- **Configuration language.** Terraform uses Hashicorp Configuration Language (HCL). You can also use JSON.
- **Limitations and drawbacks.** Because Terraform is managed separately from Azure, you might find that you can't provision some types of services or resources.

Solution summary

With Terraform, you can plan the infrastructure as code and see a preview of what the code will create. You can have that code peer reviewed to minimize errors in

configuration. Terraform supports infrastructure configurations across different cloud service providers.

Build Azure Resource Manager templates

Why use Resource Manager templates?

Using Resource Manager templates will make your deployments faster and more repeatable. For example, you no longer have to create a VM in the portal, wait for it to finish, then create the next VM, and so on. Resource Manager takes care of the entire deployment for you.

Here are some other benefits to consider.

- **Templates improve consistency**

  Resource Manager templates provide a common language for you and others to describe your deployments. Regardless of the tool or SDK used to deploy the template, the structure, format, and expressions inside the template remain the same.

- **Templates help express complex deployments**

  Templates enable you to deploy multiple resources in the correct order. For example, you wouldn't want to deploy a virtual machine before creating OS disk or network interface. Resource Manager maps out each resource and its dependent resources and creates dependent resources first. Dependency mapping helps ensure that the deployment is carried out in the correct order.

- **Templates reduce manual, error-prone tasks**

  Manually creating and connecting resources can be time consuming, and it's easy to make mistakes along the way. Resource Manager ensures that the deployment happens the same way every time.

- **Templates are code**

  Templates express your requirements through code. Think of a template as a type of *infrastructure as code* that can be shared, tested, and versioned like any other piece of software. Also, because templates are code, you can create a "paper trail" that you can follow. The template code documents the deployment. Most users maintain their templates under some kind of

revision control, such as Git. When you change the template, its revision history also documents how the template (and your deployment) has evolved over time.

- **Templates promote reuse**

  Your template can contain parameters that are filled in when the template runs. A parameter can define a username or password, a domain name, and so on. Template parameters enable you to create multiple versions of your infrastructure, such as staging and production, but still utilize the exact same template.

- **Templates are linkable**

  Resource Manager templates can be linked together to make the templates themselves modular. You can write small templates that each define a piece of a solution and combine them to create a complete system.

What's in a Resource Manager template?

- Parameters
- Variables
- Functions
- Resources
- Outputs

Deploy Azure virtual machines from VHD templates

What is a virtual machine image?

If you consider a VHD to be similar to a physical disk, a virtual machine image is a template from which you can create the VHDs to run a virtual machine. The VHDs for a typical virtual machine image contain a preconfigured version of an operating system.

Azure Marketplace supplies many virtual machine images that you can use as a starting point for your own systems. Examples include:

- Various versions of Windows Server, optionally with SQL Server installed.
- Linux variants with software such as MySQL, MongoDB, Cassandra, or other databases already configured.

For a complete list, visit the Azure Marketplace virtual machine images page.

You can also build your own virtual machine images and VHDs from scratch by using Microsoft Hyper-V. You can then upload these images to Azure so that your virtual machines can use them.

You can extend an existing virtual machine image with your own software. You can then use the image as a basis for deploying additional virtual machines in your organization. For example, if you have an in-house system that you need to roll out across all of your virtual machines, you can create an image that includes this system, and then build your virtual machines from this image.

What is a generalized image?

You can create your own custom virtual machine image in one of two ways:

- If you're building an image from scratch by using Hyper-V, you first create a blank virtual disk, and then create a virtual machine with this disk. When you start the virtual machine, you install the operating system and any other additional software from source disks (typically DVDs) and other packages.
- If you're customizing an image from Azure Marketplace, you build a virtual machine by using an existing image. The image provides the operating system and base functionality. You add your own software, operating system updates, and other packages as required. Unit 3 describes this process in more detail.

After you build and customize a virtual machine, you can save the new image as a set of VHDs. However, you must do some cleaning up first. This is because as you create a virtual machine, the operating system data is updated with several items, including:

- The host name of your virtual machine.
- The username and credentials that you provided when you installed the operating system on the virtual machine.
- Log files.
- Security identifiers for various operating system services.

You must reset these items back to a default state before you use the image to create more virtual machines. Otherwise, you might end up with multiple virtual machines that have the same identities. The process of resetting this data is called *generalization*, and the result is a *generalized image*.

The tools for preparing a virtual machine for generalization vary according to the operating system that's being installed and configured. For Windows, use the Microsoft System Preparation (Sysprep) tool. For Linux, use the Windows Azure Linux Agent (waagent) tool.

When you create a new virtual machine by using a generalized image, you have to supply items such as the host name, user account details, and other information that the generalization process removed.

What is a specialized virtual image?

A specialized virtual image is a copy of a live virtual machine after it has reached a specific state. For example, a specialized image might contain a copy of the configured operating system, software, user accounts, databases, connection information, and other data for your system.

You can use a specialized virtual image as a backup of your system at a particular point in time. If you need to recover after a catastrophic failure, or you need to roll back the virtual machine, you can restore your virtual machine from this image.

If you use a specialized image to create a new virtual machine, the new virtual machine will retain all of the data from the image. That data includes the host name, user accounts, and other settings.

Build a scalable application with virtual machine scale sets

Reducing costs by using low-priority scale sets

A low-priority virtual machine scale set allows you to use Azure compute resources at cost savings of up to 80 percent. The global Azure infrastructure frequently has underused compute resources available. A low-priority scale set provisions VMs through this underused compute capability.

When you use these VMs, keep in mind that they're temporary. Availability depends on size, region, time of day, and so on. These VMs have no SLA.

When Azure needs the computing power again, you'll receive a notification about the VM that will be removed from your scale set. If you need to clean up or gracefully exit code on your VM, you can use Azure Scheduled Events to react to the notification within the VM. You can also make the scale set try to create another VM to replace the one that's being removed. The creation of the new VM is, however, not guaranteed.

In a low-priority scale set, you specify two kinds of removal:

- **Delete**: The entire VM is removed, including all of the underlying disks.
- **Deallocate**: The VM is stopped. The processing and memory resources are deallocated. Disks are left intact and data is kept. You're charged for the disk space while the VM isn't running.

Low-priority scale sets are useful for workloads that run with interruptions or when you need larger VMs at a much-reduced cost. Just keep in mind that you can't control when a VM might be removed.

Choose the best Azure service to automate your business processes

Logic Apps

Logic Apps is a service within Azure that you can use to automate, orchestrate, and integrate disparate components of a distributed application. By using the design-first approach in Logic Apps, you can draw out complex workflows that model complex business processes. The following screenshot shows the Logic Apps Designer and design canvas that you use to define your workflow.

Microsoft Power Automate

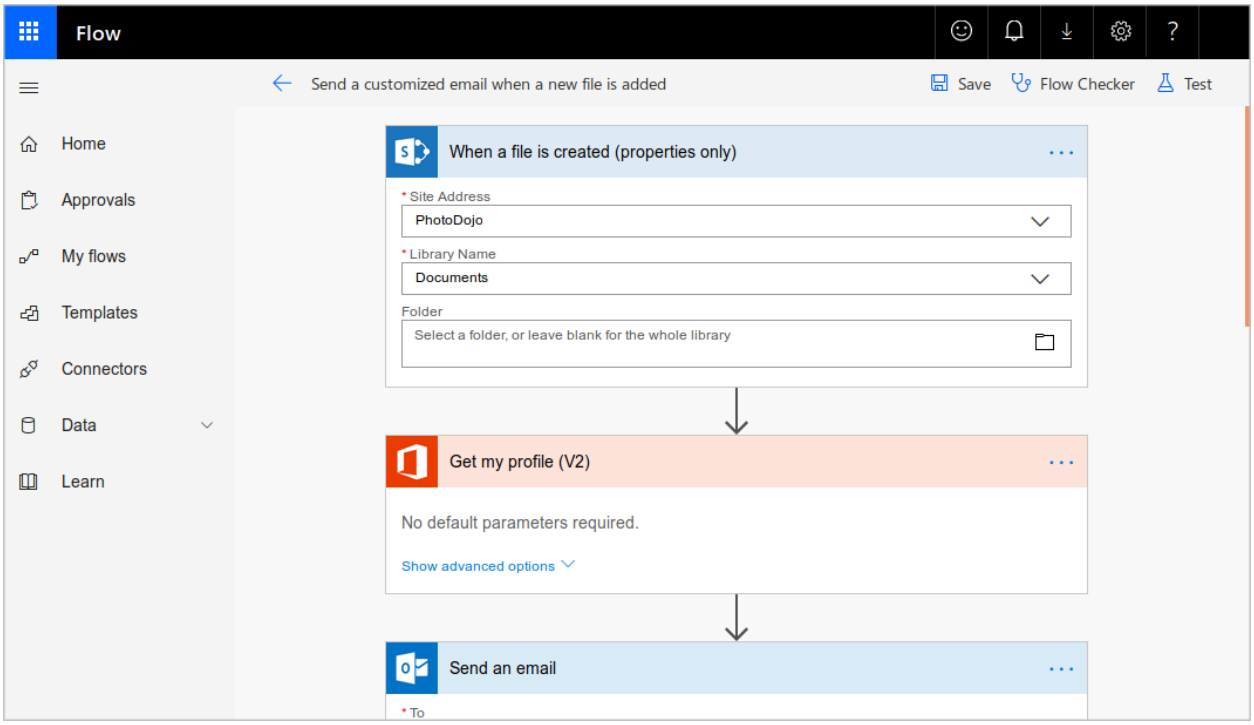
Microsoft Power Automate is a service that you can use to create workflows even when you have no development or IT Pro experience. You can create workflows that integrate and orchestrate many different components by using the website or the Microsoft Power Automate mobile app.

There are four different types of flow that you can create:

- **Automated**: A flow that is started by a trigger from some event. For example, the event could be the arrival of a new tweet or a new file being uploaded.
- **Button**: Use a button flow to run a repetitive task with a single click from your mobile device.
- **Scheduled**: A flow that executes on a regular basis such as once a week, on a specific date, or after 10 hours.
- **Business process**: A flow that models a business process such as the stock ordering process or the complaints procedure. The flow process can have:

notification to required people; with their approval recorded; calendar dates for steps; and recorded time of flow steps.

Microsoft Power Automate provides an easy-to-use design surface that anyone can use to create flows of the above types. As the following screenshot illustrates, the designer makes it easy to design and layout your process.



Design-first technologies compared

As you can see from the following table, Microsoft Power Automate is more appropriate for use by non-technical staff. If your workflow designers are IT professionals, developers, or DevOps practitioners, Logic Apps are usually a better fit:

	Microsoft Power Automate	**Logic Apps**
Intended users	Office workers and business analysts	Developers and IT pros
Intended scenarios	Self-service workflow creation	Advanced integration projects
Design tools	GUI only. Browser and mobile app	Browser and Visual Studio designer. Code editing is possible
Application Lifecycle Management	Power Automate includes testing and production environments	Logic Apps source code can be included in Azure DevOps and source code management systems

Code-first technologies

The developers on your team will likely prefer to write code when they want to orchestrate and integrate different business applications into a single workflow. This is the case when you need more control over the performance of your workflow or need to write custom code as part of the business process. For such people, Azure includes WebJobs and Functions.

WebJobs and the WebJobs SDK

The Azure App Service is a cloud-based hosting service for web applications, mobile back-ends, and RESTful APIs. These applications often need to perform some kind of background task. For example, in your bike rental system, when a user uploads a photo of a bike, you may need to generate a smaller thumbnail photograph.

WebJobs are a part of the Azure App Service that you can use to run a program or script automatically. There are two kinds of WebJob:

- **Continuous.** These WebJobs run in a continuous loop. For example, you could use a continuous WebJob to check a shared folder for a new photo.
- **Triggered.** These WebJobs run when you manually start them or on a schedule.

To determine what actions your WebJobs takes, you can write code in several different languages. For example, you can script the WebJob by writing code in a Shell Script (Windows, PowerShell, Bash). Alternatively, you can write a program in PHP, Python, Node.js, or JavaScript. These WebJOBS do have a few limitations, such as only supporting ASP.NET / SDK 2.x; however SDK 3.x supports .NET Core.

You can also program a WebJob by using the .NET Framework or the .NET Core Framework and a .NET language such as C# or VB.NET. In this case, you can also use the WebJobs SDK to make the task easier. The SDK includes a range of classes, such as `JobHostConfiguration` and `HostBuilder`, which reduce the amount of code required to interact with the Azure App Service.

The WebJobs SDK only supports C# and the NuGet package manager.

Azure Functions

An Azure Function is a simple way for you to run small pieces of code in the cloud, without having to worry about the infrastructure required to host that code. You can write the Function in C#, Java, JavaScript, PowerShell, Python, or any of the languages that are listed in the Supported languages in Azure Functions article. In addition, with the consumption plan option, you only pay for the time when the code runs. Azure automatically scales your function in response to the demand from users.

When you create an Azure Function, you can start by writing the code for it in the portal. Alternatively, if you need source code management, you can use GitHub or Azure DevOps Services.

To create an Azure Function, choose from the range of templates. The following list is an example of some of the templates available to you.

- **HTTPTrigger**. Use this template when you want the code to execute in response to a request sent through the HTTP protocol.
- **TimerTrigger**. Use this template when you want the code to execute according to a schedule.
- **BlobTrigger**. Use this template when you want the code to execute when a new blob is added to an Azure Storage account.
- **CosmosDBTrigger**. Use this template when you want the code to execute in response to new or updated documents in a NoSQL database.

Azure Functions can integrate with many different services both within Azure and from third parties. These services can trigger your function, or send data input to your function, or receive data output from your function.

Code-first technologies compared

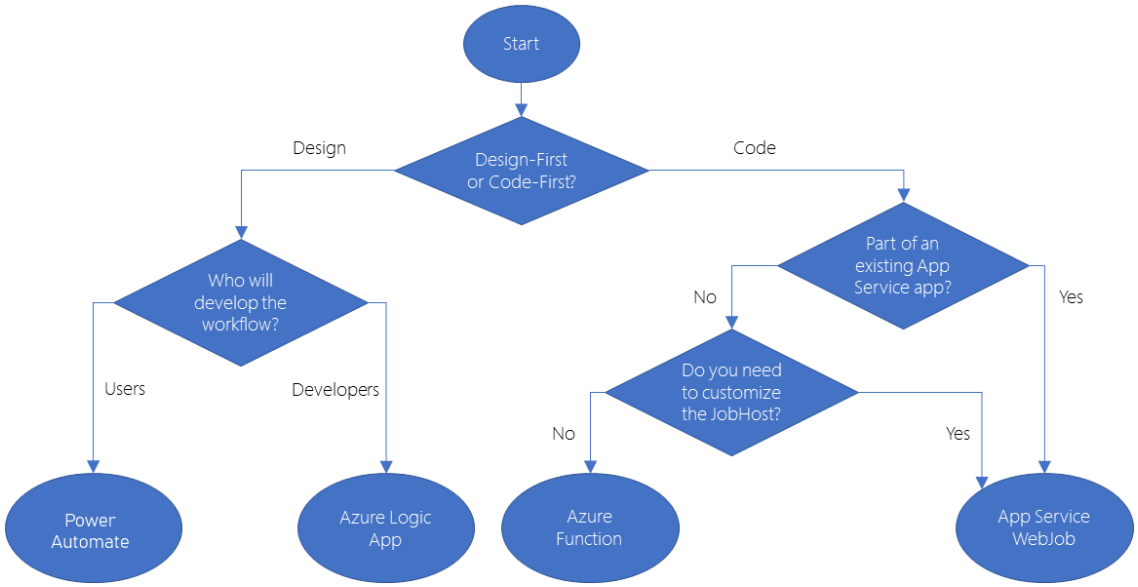
In most cases, the simple administration and more flexible coding model provided by Azure Functions may lead you to choose them in preference to WebJobs. However, you may choose WebJobs for the following reasons:

- You want the code to be a part of an existing App Service application and to be managed as part of that application, for example in the same Azure DevOps environment.
- You need close control over the object that listens for events that trigger the code. This object in question is the `JobHost` class, and you have more flexibility to modify its behavior in WebJobs.

	Azure WebJobs	**Azure Functions**
Supported languages	C# if you are using the WebJobs SDK	C#, Java, JavaScript, PowerShell, etc.
Automatic scaling	No	Yes
Development and testing in a browser	No	Yes
Pay-per-use pricing	No	Yes
Integration with Logic Apps	No	Yes
Package managers	NuGet if you are using the WebJobs SDK	Nuget and NPM
Can be part of an App Service application	Yes	No
Provides close control of `JobHost`	Yes	No

How to choose a service

The following diagram shows a simplified flow chart that you can use to choose the best technology to use for your business process:



Choosing a design-first technology

If you choose to use a design-first approach, you must also choose from Microsoft Power Automate and Azure Logic Apps.

The principal question here is who will design the workflow: will it be developers or users?

In Logic Apps, there is a GUI designer on which you draw out the workflow. It is intuitive and easy to use but you also have the opportunity to delve under the hood and edit the source code for a workflow. This tool is designed for people with development skills.

In Microsoft Power Automate, extra help and templates are provided for common types of workflow. There is no way to edit the source code that the tool creates. This tool is designed for users who have a good understanding of the business process but no coding skills.

Choosing a code-first technology

If you choose to use a code-first approach, your next choice is between WebJobs and Azure Functions.

Because of the extra features that are included with Azure Functions, including wider ranges of trigger events and supported languages, the ability to develop test code in the browser, and the pay-per-use price model, consider Azure Functions to be your default choice. There are two situations in which WebJobs might be a better choice:

- You have an existing Azure App Service application, and you want to model the workflow within the application. This requirement means that the workflow can also be managed as part of the application, for example in an Azure DevOps environment.
- You have specific customizations that you want to make to the `JobHost` that are not supported by Azure Functions. For example, in a WebJob, you can create a custom retry policy for calls to external systems. This kind of policy can't be configured in an Azure Function.
- Webjobs only supports C# on Microsoft Windows.

Mixing Technologies

Remember that there is no requirement for you to use the same technology for different workflows: if your requirements differ, you are likely to reach a different answer at the end of your decision-making process. Furthermore, you can also call one workflow from another. For example, a workflow implemented in Microsoft Power Automate can easily call another that is built as an Azure Function.

One reason to mix the technologies used in your business processes would be to give users control over a small section of a complete workflow. Do this by implementing that section in Microsoft Power Automate, then call that flow from a Logic App, Web Job, or Function.

Introduction to high-performance computing (HPC) on Azure

Azure Batch

What is HPC?

There are many different industries that require very powerful computing resources for specialized tasks. For example:

- In genetic sciences, gene sequencing.
- In oil and gas exploration, reservoir simulations.
- In finance, market modeling.
- In engineering, physical system modeling.
- In meteorology, weather modeling.

These tasks require processors that can carry out instructions extremely fast. It's also helpful to run many processors in parallel, to obtain answers within a practical time duration. On-premises HPC systems have many powerful CPUs and, for graphics-intensive tasks, GPUs. They also require fast disks and high-speed memory.

Azure enables you to perform HPC tasks in the cloud, without building your own expensive HPC hardware. An Azure HPC system also has the advantage that you can dynamically add resources as they are needed, and remove them when demand falls. Azure makes it easy to coordinate an HPC task across many virtual machines, and it supports powerful VM sizes.

Azure Batch

Azure Batch is a service for working with large-scale parallel and computationally intensive tasks on Azure. Unlike the other options you'll see in this module, Batch is a managed service. You provide data and applications, and you specify whether to run on Windows or Linux, how many machines to use, and what rules apply to autoscaling. Batch handles provisioning of the compute capacity and optimizes the way the work is done in parallel. You only pay for the underlying compute, networking, and storage you use. The Batch scheduling and management service is free.

Batch is ideally suited to heavy workloads, such as financial risk modeling, 3D rendering, media transcoding, and genetic sequence analysis. Think of Batch as a flexible management and scheduling service layer on top of the huge scale of Azure. For example, you might spin up 100 or 1000 virtual machines to support heavy workloads

without the aid of Batch. However, you'd then be responsible for all the scheduling of the VMs, and for distributing the work according to available capacity.

Components of Azure Batch

Batch has several components that act together. An *Azure Batch Account* forms a container for all of the main Batch elements. Within the Batch account, you typically create Batch *pools* of VMs, which are often called nodes, running either Windows or Linux. You set up Batch *jobs* that work like logical containers with configurable settings for the real unit of work in Batch, known as Batch *tasks*. This unit of work is highly flexible and can run either command-line instructions or entire applications. Optionally, you might associate an Azure Storage account with the Azure Batch account. You then upload and download data inputs and outputs. You also provide application installers for the Batch tasks that need them.

This diagram shows a client application or hosted service interacting with Batch to upload input, create jobs, monitor tasks, and download output.

Azure Batch in action

When you create Batch tasks, the scheduling and management engine determines the optimal plan for allocating and scheduling tasks across the specified compute capacity. This plan happens during the initial creation of a Batch pool.

In a series of 100 tasks and 10 nodes, for example, Batch schedules the first 10 tasks onto those 10 nodes. Batch immediately allocates later tasks when nodes finish processing. For spiky workloads, you can configure scaling rules, which Batch also handles automatically. If you provision 100 VMs with no Batch context, you must code these scheduling and work allocation mechanisms by hand.

Because of how this managed scheduling engine works, Batch is well-suited to highly parallel workloads, which are sometimes called *embarrassingly parallel* workloads. For example, the engineering company needs to generate highly detailed 3D models of the facilities they design. These models can be extraordinarily compute-intensive and take hours or days to complete. Using Batch, you allocate the rendering of different sections of the structures to separate compute nodes. For example, if 100 nodes are provisioned, it should be possible to complete the rendering in approximately 1/100th of the time.

What is HPC Pack?

In researching options for the engineering organization, you've looked at Azure Batch and Azure HPC Instances. But what if you want to have full control of the management and scheduling of your clusters of VMs? What if you have significant investment in on-premises infrastructure in your datacenter? HPC Pack offers a series of installers for Windows that allows you to configure your own control and management plane, and highly flexible deployments of on-premises and cloud nodes. By contrast with the exclusively cloud-based Batch, HPC Pack has the flexibility to deploy to on-premises and the cloud. It uses a hybrid of both to expand to the cloud when your on-premises reserves are insufficient.

Think of Microsoft HPC Pack as a version of the Batch management and scheduling control layer, over which you have full control, and for which you have responsibility. Deployment of HPC Pack requires Windows Server 2012 or later, and takes careful consideration to implement.

Plan for HPC Pack

Typically, you should prepare for the installation of HPC Pack with a full review of requirements. You need SQL Server and an Active Directory controller. You must also plan a topology. How many head or control nodes should there be, and how many worker nodes? Do you need to expand up to Azure? If so, you pre-provision Azure nodes as part of the cluster. The size of the main machines that make up the control plane (head and control nodes, SQL Server, and Active Directory domain controller) will depend on the projected cluster size.

When you install HPC Pack, it shows a job scheduler with support for both HPC and parallel jobs. The scheduler appears in the Microsoft Message Passing Interface. HPC Pack is highly integrated with Windows, so you can use Visual Studio for parallel debugging. You'll see all the application, networking, and operating system events from the compute nodes in the cluster in a single, debugger view.

HPC Pack also offers an advanced job scheduler. You can rapidly deploy, even to nodes not exclusively dedicated to HPC Pack, to Linux-based nodes, and Azure nodes. That means you can use spare capacity within your datacenter. HPC Pack provides an ideal

way to use existing infrastructure investments, and keep more discrete control over how work gets divided up than is possible with Batch.

Use a mix of technologies

The options you're considering in this module aren't mutually exclusive. You can use the H-Series VMs, which you looked at in the last unit, as possible Azure nodes in an HPC configuration. While you've concentrated on hybrid use cases to highlight the differences with Batch, HPC Pack is flexible. It allows for both exclusively on-premises deployments and exclusively cloud-based deployments. This flexibility is useful when you want more granular control than Batch offers.

Control and organize Azure resources with Azure Resource Manager

Use tags for organization

The above example is just one example of where you can use tags to organize your resources. With their flexibility, there are several ways you can use tags to your advantage.

You can use tags to group your billing data. For example, if you're running multiple VMs for different organizations, use the tags to group usage by cost center. You can also use tags to categorize costs by runtime environment, such as the billing usage for VMs running in the production environment. When exporting billing data or accessing it through billing APIs, tags are included in that data and can be used to further slice your data from a cost perspective.

You can retrieve all the resources in your subscription with a specific tag name or value. Tags enable you to retrieve related resources from different resource groups. This approach is helpful when you need to organize resources for billing or management.

Tagging resources can also help in monitoring to track down impacted resources. Monitoring systems could include tag data with alerts, giving you the ability to know exactly who is impacted. In our example above, you applied the **Department** tag with a value of **Finance** to the **msftlearn-vnet1** resource. If an alarm was thrown on **msftlearn-vnet1** and the alarm included the tag, you'd know that the finance department may be impacted by the condition that triggered the alarm. This contextual information can be valuable if an issue occurs.

It's also common for tags to be used in automation. If you want to automate the shutdown and startup of virtual machines in development environments during off-hours to save costs, you can use tags to assist in this automation. Add a **shutdown:6PM** and **startup:7AM** tag to the virtual machines, then create an automation job that looks for these tags, and shuts them down or starts them up based on the tag value. There are several solutions in the Azure Automation Runbooks Gallery that use tags in a similar manner to accomplish this result.

What are resource locks?

Resource locks are a setting that can be applied to any resource to block modification or deletion. Resource locks can set to either **Delete** or **Read-only**. **Delete** will allow all operations against the resource but block the ability to delete it. **Read-only** will only allow read activities to be performed against it, blocking any modification or deletion of the resource. Resource locks can be applied to subscriptions, resource groups, and to individual resources, and are inherited when applied at higher levels.

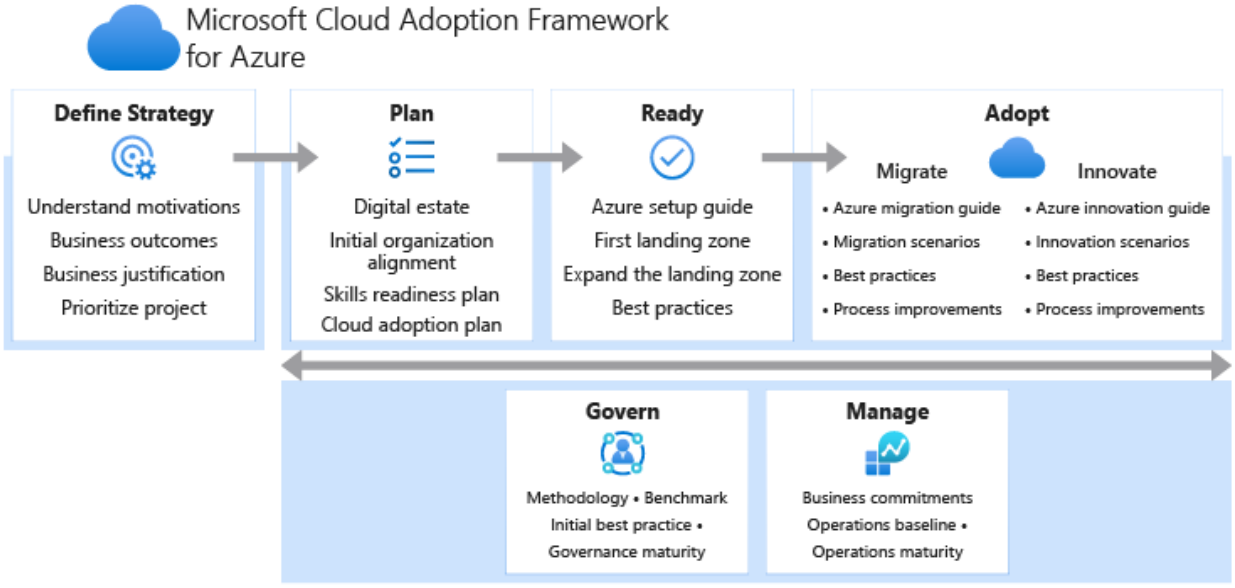
When a resource lock is applied, you must first remove the lock in order to perform that activity. By putting an additional step in place before allowing the action to be taken on the resource, it helps protect resources from inadvertent actions, and helps protect your admins from doing something they may not have intended to do. Resource locks apply regardless of RBAC permissions. Even if you are an owner of the resource, you must still remove the lock before you'll actually be able to perform the blocked activity.

Build a cloud governance strategy on Azure

What's in the Cloud Adoption Framework?

The Cloud Adoption Framework consists of tools, documentation, and proven practices. The Cloud Adoption Framework includes these stages:

1. Define your strategy.
2. Make a plan.
3. Ready your organization.
4. Adopt the cloud.
5. Govern and manage your cloud environments.



The govern stage focuses on cloud governance. You can refer back to the Cloud Adoption Framework for recommended guidance as you build your cloud governance strategy.

To help build your adoption strategy, the Cloud Adoption Framework breaks out each stage into further exercises and steps. Let's take a brief look at each stage.

Create a subscription governance strategy

Billing

You can create one billing report per subscription. If you have multiple departments and need to do a "chargeback" of cloud costs, one possible solution is to organize subscriptions by department or by project.

Resource tags can also help. You'll explore tags later in this module. When you define how many subscriptions you need and what to name them, take into account your internal billing requirements.

Access control

A subscription is a deployment boundary for Azure resources. Every subscription is associated with an Azure Active Directory tenant. Each tenant provides administrators the ability to set granular access through defined roles by using Azure role-based access control.

When you design your subscription architecture, consider the deployment boundary factor. For example, do you need separate subscriptions for development and for production environments? With separate subscriptions, you can control access to each one separately and isolate their resources from one another.

Subscription limits

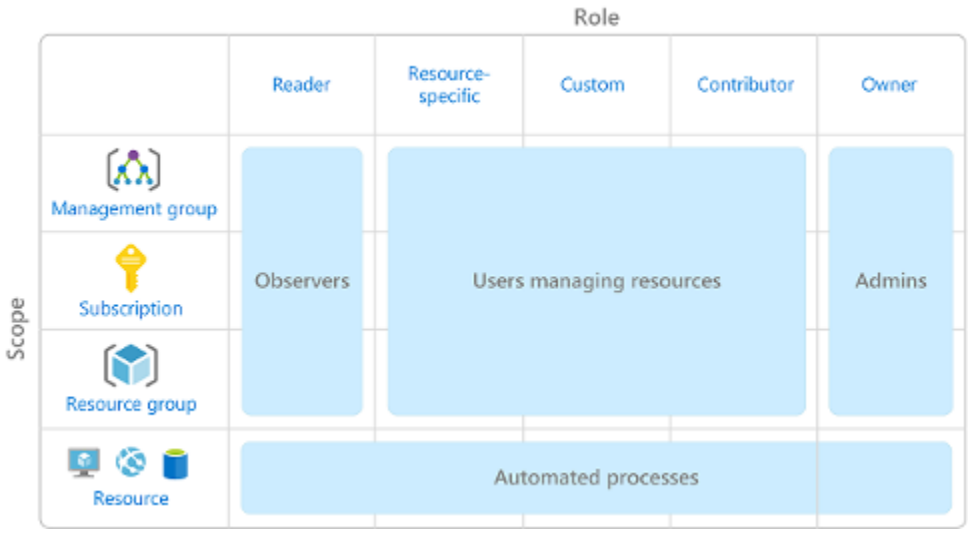
Subscriptions also have some resource limitations. For example, the maximum number of network Azure ExpressRoute circuits per subscription is 10. Those limits should be considered during your design phase. If you'll need to exceed those limits, you might need to add more subscriptions. If you hit a hard limit maximum, there's no flexibility to increase it.

Management groups are also available to assist with managing subscriptions. A management group manages access, policies, and compliance across multiple Azure subscriptions

How is role-based access control applied to resources?

Role-based access control is applied to a *scope*, which is a resource or set of resources that this access applies to.

Here's a diagram that shows the relationship between roles and scopes.



Scopes include:

- A management group (a collection of multiple subscriptions).
- A single subscription.
- A resource group.
- A single resource.

When you grant access at a parent scope, those permissions are inherited by all child scopes. For example:

- When you assign the Owner role to a user at the management group scope, that user can manage everything in all subscriptions within the management group.
- When you assign the Reader role to a group at the subscription scope, the members of that group can view every resource group and resource within the subscription.
- When you assign the Contributor role to an application at the resource group scope, the application can manage resources of all types within that resource group, but not other resource groups within the subscription.

When should I use Azure RBAC?

Use Azure RBAC when you need to:

- Allow one user to manage VMs in a subscription and another user to manage virtual networks.
- Allow a database administrator group to manage SQL databases in a subscription.
- Allow a user to manage all resources in a resource group, such as virtual machines, websites, and subnets.
- Allow an application to access all resources in a resource group.

These are just a few examples. You'll find the complete list of built-in roles at the end of this module.

How is Azure RBAC enforced?

Azure RBAC is enforced on any action that's initiated against an Azure resource that passes through Azure Resource Manager. Resource Manager is a management service that provides a way to organize and secure your cloud resources.

You typically access Resource Manager from the Azure portal, Azure Cloud Shell, Azure PowerShell, and the Azure CLI. Azure RBAC doesn't enforce access permissions at the application or data level. Application security must be handled by your application.

RBAC uses an *allow model*. When you're assigned a role, RBAC *allows* you to perform certain actions, such as read, write, or delete. If one role assignment grants you read permissions to a resource group and a different role assignment grants you write permissions to the same resource group, you have both read and write permissions on that resource group.

Combine resource locks with Azure Blueprints

What if a cloud administrator accidentally deletes a resource lock? If the resource lock is removed, its associated resources can be changed or deleted.

To make the protection process more robust, you can combine resource locks with Azure Blueprints. Azure Blueprints enables you to define the set of standard Azure resources that your organization requires. For example, you can define a blueprint that specifies that a certain resource lock must exist. Azure Blueprints can automatically replace the resource lock if that lock is removed.

An example tagging structure

A resource tag consists of a name and a value. You can assign one or more tags to each Azure resource.

After reviewing its business requirements, Tailwind Traders decides on the following tags.

Name	Value
AppName	The name of the application that the resource is part of.
CostCenter	The internal cost center code.
Owner	The name of the business owner who's responsible for the resource.
Environment	An environment name, such as "Prod," "Dev," or "Test."
Impact	How important the resource is to business operations, such as "Mission-critical," "High-impact," or "Low-impact."

Here's an example that shows these tags as they're applied to a virtual machine during provisioning.

Name ⓘ		Value ⓘ	Resource
AppName	:	SpecialOrders	Virtual machine
CostCenter	:	0224 - Infrastructure R&D	Virtual machine
Owner	:	tim@tailwindtraders.com	Virtual machine
Environment	:	Test	Virtual machine
Impact	:	High-impact	Virtual machine

How does Azure Policy define policies?

Azure Policy enables you to define both individual policies and groups of related policies, known as *initiatives*. Azure Policy evaluates your resources and highlights resources that aren't compliant with the policies you've created. Azure Policy can also prevent noncompliant resources from being created.

Azure Policy comes with a number of built-in policy and initiative definitions that you can use, under categories such as Storage, Networking, Compute, Security Center, and Monitoring.

For example, say you define a policy that allows only a certain stock-keeping unit (SKU) size of virtual machines (VMs) to be used in your environment. After you enable this policy, that policy is applied when you create new VMs or resize existing VMs. Azure Policy also evaluates any current VMs in your environment.

In some cases, Azure Policy can automatically remediate noncompliant resources and configurations to ensure the integrity of the state of the resources. For example, if all resources in a certain resource group should be tagged with the **AppName** tag and a value of "SpecialOrders," Azure Policy can automatically reapply that tag if it has been removed.

Azure Policy also integrates with Azure DevOps by applying any continuous integration and delivery pipeline policies that apply to the pre-deployment and post-deployment phases of your applications.

What are Azure Policy initiatives?

An Azure Policy initiative is a way of grouping related policies into one set. The initiative definition contains all of the policy definitions to help track your compliance state for a larger goal.

For example, Azure Policy includes an initiative named **Enable Monitoring in Azure Security Center**. Its goal is to monitor all of the available security recommendations for all Azure resource types in Azure Security Center.

Under this initiative, the following policy definitions are included:

- **Monitor unencrypted SQL Database in Security Center**

  This policy monitors for unencrypted SQL databases and servers.

- **Monitor OS vulnerabilities in Security Center**

  This policy monitors servers that don't satisfy the configured OS vulnerability baseline.

- **Monitor missing Endpoint Protection in Security Center**

  This policy monitors for servers that don't have an installed endpoint protection agent.

In fact, the **Enable Monitoring in Azure Security Center** initiative contains over 100 separate policy definitions.

Azure Policy also includes initiatives that support regulatory compliance standards such as HIPAA and ISO 27001.

Govern multiple subscriptions by using Azure Blueprints

Azure Blueprints orchestrates the deployment of various resource templates and other artifacts, such as:

- Role assignments
- Policy assignments
- Azure Resource Manager templates
- Resource groups

Azure Blueprints in action

When you form a cloud center of excellence team or a cloud custodian team, that team can use Azure Blueprints to scale their governance practices throughout the organization.

Implementing a blueprint in Azure Blueprints involves these three steps:

1. Create an Azure blueprint.
2. Assign the blueprint.
3. Track the blueprint assignments.

With Azure Blueprints, the relationship between the blueprint definition (what should be deployed) and the blueprint assignment (what was deployed) is preserved. In other words, Azure creates a record that associates a resource with the blueprint that defines it. This connection helps you track and audit your deployments.

Blueprints are also versioned. Versioning enables you to track and comment on changes to your blueprint.

What are blueprint artifacts?

Each component in the blueprint definition is known as an *artifact*.

It is possible for artifacts to have no additional parameters (configurations). An example is the **Deploy threat detection on SQL servers** policy, which requires no additional configuration.

Artifacts can also contain one or more parameters that you can configure. The following screenshot shows the **Allowed locations** policy.

You can specify a parameter's value when you create the blueprint definition or when you assign the blueprint definition to a scope. In this way, you can maintain one standard blueprint but have the flexibility to specify the relevant configuration parameters at each scope where the definition is assigned.

Design a holistic monitoring strategy on Azure

What is Azure Security Center?

Azure Security Center is a service that manages the security of your infrastructure from a centralized location. Use Security Center to monitor the security of your workloads, whether they're on-premises or in the cloud.

Attacks are becoming more intelligent, and the number of people with the right security skills is low. Security Center helps you deal with these challenges because it provides you with tools that improve your protection against security threats. Use Security Center to monitor the health of your resources and implement recommendations.

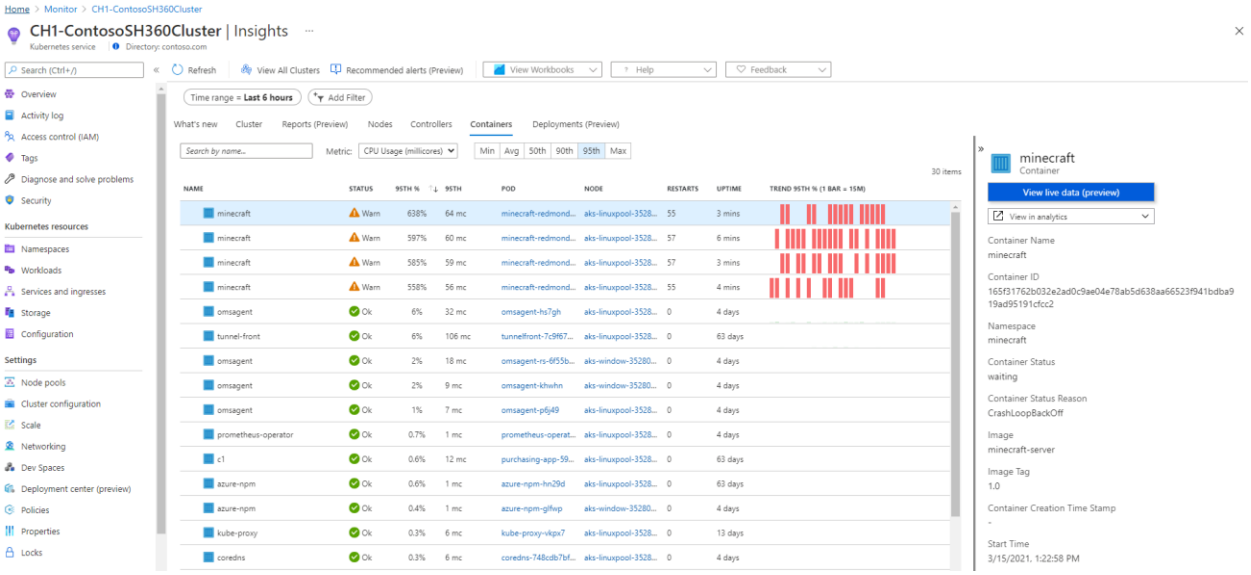
What is Azure Sentinel?

You use Azure Sentinel to collect data on the devices, users, infrastructure, and applications across your enterprise. Built-in threat intelligence for detection and investigation can help reduce false positives. Use Sentinel to proactively hunt for threats and anomalies, and respond by using orchestration and automation.

You connect your data sources to Sentinel. These sources include Microsoft services such as Office 365 and Azure Advanced Threat Protection. These sources can also include external solutions, such as AWS CloudTrail or on-premises sources. The dashboard shows detailed information collected from your sources.

Monitor Kubernetes clusters at-scale

You can view the performance and resource utilization of your Kubernetes workloads from different perspectives. You may want to investigate an over-utilized node, view the state of pods by controller, or look at the number of restarts and CPU or memory utilization of a specific container. Azure Monitor container insights also allow you to filter by namespace, access logs and enable commonly used alerts.



Criteria for assessing Azure Monitor virtual machine insights

You use Azure Monitor VM insights to:

- View the health and performance of your VMs
- Monitor your VMs at-scale across multiple subscriptions and resource groups.
- Want a topology view that shows the processes, and network connection details of your VMs and scale sets.

Improve incident response with alerting on Azure

When to use activity log alerts

So far, you've seen two different types of alerts supported in Azure Monitor. Metric alerts are ideally suited to monitoring for threshold breaches or spotting trends. Log alerts allow for greater analytical monitoring of historical data.

Activity log alerts are designed to work with Azure resources. Typically, you create this type of log to receive notifications when specific changes occur on a resource within your Azure subscription.

There are two types of activity log alerts:

- **Specific operations**: Apply to resources within your Azure subscription, and often have a scope with specific resources or a resource group. You use this type when you need to receive an alert that reports a change to an aspect of your subscription. For example, you can receive an alert if a VM is deleted or new roles are assigned to a user.
- **Service health events**: Include notice of incidents and maintenance of target resources.

What are smart groups?

Smart groups are an automatic feature of Azure Monitor. By using machine learning algorithms, Azure Monitor joins alerts based on repeat occurrence or similarity. Smart groups enable you to address a group of alerts instead of each alert individually.

The name of the smart group (its taxonomy), is assigned automatically, and is the name of the first alert in the group. It's important to assign meaningful names to each alert that you create, because the name of the smart group can't be changed or amended.

When to use smart groups

Think of smart groups as a dynamic filter applied to all the alerts in Azure Monitor. The machine learning algorithm in Azure Monitor joins alerts based on information, such as historical patterns, similar properties, or structure. Using smart groups can reduce alert noise by more than 90 percent.

The power of smart groups is that they show you all related alerts and give improved analytics. They can often identify a previously unseen root cause.

Smart group states

Smart groups, like regular alerts, have their own state. The state shows the progress of the smart group in the resolution process. Changing the state of a smart group doesn't alter the state of the individual alerts.

To change the state, select **Change smart group state**.

The states are:

- **New**: The smart group has been created with a collection of alerts, but it hasn't yet been addressed.
- **Acknowledged**: When an admin starts the resolution process, they change the state to this.
- **Closed**: When the source of the alert is fixed, the admin changes the state to this.

Changing the state of the smart group doesn't affect the underlying alert. Each alert member shown in the smart group can have a different state.

Analyze your Azure infrastructure by using Azure Monitor logs

Azure Monitor collects data automatically from a range of components. For example:

- **Application data**: Data that relates to your custom application code.
- **Operating system data**: Data from the Windows or Linux virtual machines that host your application.
- **Azure resource data**: Data that relates to the operations of an Azure resource, such as a web app or a load balancer.
- **Azure subscription data**: Data that relates to your subscription. It includes data about Azure health and availability.
- **Azure tenant data**: Data about your Azure organization-level services, such as Azure Active Directory.

Because Azure Monitor is an automatic system, it begins to collect data from these sources as soon as you create Azure resources such as virtual machines and web apps. You can extend the data that Azure Monitor collects by:

- **Enabling diagnostics**: For some resources, such as Azure SQL Database, you receive full information about a resource only after you have enabled diagnostic logging for it. You can use the Azure portal, the Azure CLI, or PowerShell to enable diagnostics.
- **Adding an agent**: For virtual machines, you can install the Log Analytics agent and configure it to send data to a Log Analytics workspace. This agent increases the amount of information that's sent to Azure Monitor.

Analyzing logs by using Kusto

To retrieve, consolidate, and analyze data, you specify a query to run in Azure Monitor logs. You write a log query with the Kusto query language, which is also used by Azure Data Explorer.

Log queries can be tested in the Azure portal so you can work with them interactively. You typically start with basic queries and then progress to more advanced functions as your requirements become more complex.

In the Azure portal, you can create custom dashboards, which are targeted displays of resources and data. Each dashboard is built from a set of tiles. Each tile might show a set of resources, a chart, a table of data, or some custom text. Azure Monitor provides tiles

that you can add to dashboards. For example, you might use a tile to display the results of a Kusto query in a dashboard.

In the example scenario, the operations team can consolidate its data by visualizing monitoring data such as charts and tables. These tools are effective for summarizing data and presenting it to different audiences.

By using Azure dashboards, you can combine various kinds of data, including both logs and metrics, into a single pane in the Azure portal. For example, you might want to create a dashboard that combines tiles that show a graph of metrics, a table of activity logs, charts from Azure Monitor, and the output of a log query.

Explore Azure database and analytics services

Explore Azure Database for PostgreSQL

Single Server

The Single Server deployment option delivers:

- Built-in high availability with no additional cost (99.99 percent SLA).
- Predictable performance and inclusive, pay-as-you-go pricing.
- Vertical scale as needed, within seconds.
- Monitoring and alerting to assess your server.
- Enterprise-grade security and compliance.
- Ability to protect sensitive data at-rest and in-motion.
- Automatic backups and point-in-time-restore for up to 35 days.

All those capabilities require almost no administration, and all are provided at no additional cost. You can focus on rapid application development and accelerating your time to market, rather than having to manage virtual machines and infrastructure. You can continue to develop your application with the open-source tools and platform of your choice, without having to learn new skills.

The Single Server deployment option offers three pricing tiers: Basic, General Purpose, and Memory Optimized. Each tier offers different resource capabilities to support your database workloads. You can build your first app on a small database for a few dollars a month, and then adjust the scale to meet the needs of your solution. Dynamic scalability enables your database to transparently respond to rapidly changing resource requirements. You only pay for the resources you need, and only when you need them.

Hyperscale (Citus)

The Hyperscale (Citus) option horizontally scales queries across multiple machines by using sharding. Its query engine parallelizes incoming SQL queries across these servers for faster responses on large datasets. It serves applications that require greater scale and performance, generally workloads that are approaching, or already exceed, 100 GB of data.

The Hyperscale (Citus) deployment option supports multi-tenant applications, real-time operational analytics, and high throughput transactional workloads. Applications built

for PostgreSQL can run distributed queries on Hyperscale (Citus) with standard connection libraries and minimal changes.

Explore big data and analytics

Azure Synapse Analytics

Azure Synapse Analytics (formerly Azure SQL Data Warehouse) is a limitless analytics service that brings together enterprise data warehousing and big data analytics. You can query data on your terms by using either serverless or provisioned resources at scale. You have a unified experience to ingest, prepare, manage, and serve data for immediate BI and machine learning needs.

Azure HDInsight

Azure HDInsight is a fully managed, open-source analytics service for enterprises. It's a cloud service that makes it easier, faster, and more cost-effective to process massive amounts of data. You can run popular open-source frameworks and create cluster types such as Apache Spark, Apache Hadoop, Apache Kafka, Apache HBase, Apache Storm, and Machine Learning Services. HDInsight also supports a broad range of scenarios such as extraction, transformation, and loading (ETL), data warehousing, machine learning, and IoT.

Azure Databricks

Azure Databricks helps you unlock insights from all your data and build artificial intelligence solutions. You can set up your Apache Spark environment in minutes, and then autoscale and collaborate on shared projects in an interactive workspace. Azure Databricks supports Python, Scala, R, Java, and SQL, as well as data science frameworks and libraries including TensorFlow, PyTorch, and scikit-learn.

Azure Data Lake Analytics

Azure Data Lake Analytics is an on-demand analytics job service that simplifies big data. Instead of deploying, configuring, and tuning hardware, you write queries to transform your data and extract valuable insights. The analytics service can handle jobs of any scale instantly by setting the dial for how much power you need. You only pay for your job when it's running, making it more cost-effective.

Scale multiple Azure SQL Databases with SQL elastic pools

What is a SQL elastic pool?

SQL elastic pools are a resource allocation service used to scale and manage the performance and cost of a group of Azure SQL databases. Elastic pools allow you to purchase resources for the group. You set the amount of resources available to the pool, add databases to the pool, and set minimum and maximum resource limits for the databases within the pool.

The pool resource requirements are set based on the overall needs of the group. The pool allows the databases within the pool to share the allocated resources. SQL elastic pools are used to manage the budget and performance of multiple SQL databases.

When to use an elastic pool?

SQL elastic pools are ideal when you have several SQL databases that have a low average utilization, but have infrequent, high utilization spikes. In this scenario, you can allocate enough capacity in the pool to manage the spikes for the group, but the total resources can be less than the sum of all of the peak demand of all of the databases. Since the spikes are infrequent, a spike from one database will be unlikely to impact the capacity of the other databases in the pool.

In our fitness company scenario, the individual locations may run promotions at different times of year or see spikes in demand during regional holidays.

How many databases to add to a pool?

The general guidance is, if the combined resources you would need for individual databases to meet capacity spikes is more than 1.5 times the capacity required for the elastic pool, then the pool will be cost effective.

At a minimum, it is recommended to add at least two S3 databases or fifteen S0 databases to a single pool for it to have potential cost savings.

Depending on the performance tier, you can add up to 100 or 500 databases to a single pool.

DTU-based pricing model

A database transaction unit (DTU) is a unit of measurement for the performance of a service tier in Azure and is based on a bundled measure of compute, storage, and IO resources. Compute sizes are expressed in terms of Database Transaction Units (DTUs) for single databases or elastic Database Transaction Units (eDTUs) for elastic pools.

If demand exceeds the available resources for your tier for any resource (CPU, storage or IO), the performance of your database is throttled. This model is best for customers who want simple, pre-configured resource options available in three tiers: **basic**, **standard**, and **premium**.

vCore-based pricing model

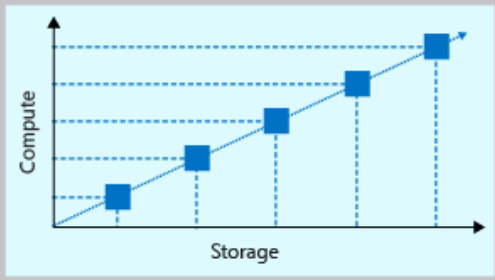
A virtual core (vCore) represents the logical CPU offered with an option to choose between generations of hardware and physical characteristics of hardware (for example, number of cores, memory, storage size).

The vCore-based purchasing model gives you flexibility and control over individual resource consumption and a straightforward way to translate on-premises workload requirements to the cloud. This model allows you to choose compute, memory, and storage based upon their workload needs within the **general purpose** or **business critical** service tier.

The chart below provides a comparison of the two pricing models:

DTU model

Simple, Preconfigured



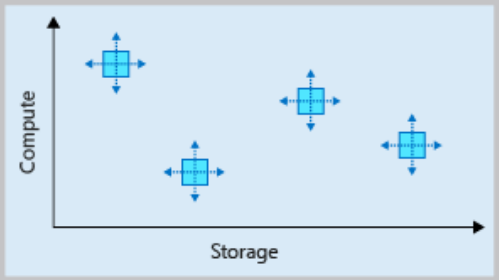
Database Transaction Unit (DTU)-based model

- Bundled measure of compute, storage and IO resources
- Best for customers who want simple, pre-configured resource options

OR

vCore model

Independent scalability



vCore-based model

- Independent scaling of compute, storage and IO resources
- Best for customers who value flexibility, control and transparency
- Use with Azure Hybrid Benefit for SQL Server to gain cost savings

Migrate your relational data stored in SQL Server to Azure SQL Database

Pre-migration

The pre-migration phase begins with *discovery*, or taking inventory of your existing databases and the tools and apps that rely on them. For this simple exercise, we're concerned with only a single social database. In practice, it can be a much more complex step.

You need to identify everything that uses your existing database. Apps, SQL Server Report Services reports, Power BI reports, and export jobs written in PowerShell are all examples of things to note so you can update them, after the migration, to point to the new Azure SQL Database.

The second step in the pre-migration phase is the *assessment*. During the assessment, you examine the database for any incompatibilities between it and the Azure SQL Database platform. Because this can be a difficult task to perform manually, Microsoft has provided Data Migration Assistant. You can use Data Migration Assistant to automatically examine your source database for any compatibility issues with Azure SQL Database.

Data Migration Assistant provides a report that you can use as a guide to update your database. As you make changes, you can rerun Data Migration Assistant to track your progress and to uncover any new issues that might arise as you make changes. The assessment phase is covered in steps 1 and 2 of the migration workflow previously illustrated.

The final stage in the pre-migration is *convert*. In the convert phase, you make any changes for compatibility that Data Migration Assistant has recommended. Then, you create the SQL scripts for deploying to the Azure SQL Database. Data Migration Assistant can be of help to you here as well. It generates all of the SQL scripts needed to deploy your schema to the target Azure SQL Database.

Migration

The migration phase involves migrating two elements: *schema* and *data*. In the convert phase of pre-migration, the Data Migration Assistant tool generated all of the code. Data Migration Assistant can run these scripts for you. Or, you can save these scripts,

and run them on your own by using a tool such as SQL Server Management Studio, Azure Data Studio, or the sqlcmd utility. The schema migration can be found in step 4 of the migration workflow.

After your database schema has been migrated, you're ready to migrate your data (steps 3 and 5 in the workflow). For this step, you'll use Azure Database Migration Service to move your data up to the Azure SQL Database Service.

Database Migration Service can be run in two modes, online and offline. When it's running in online mode, there are two additional steps. The first is *sync*, in which any changes made to the data in the source system after the migration are brought into the target database. The other is *cutover*, in which the source database is taken offline, and the new Azure SQL Database becomes available.

Post-migration

The post-migration phase is a process that consists of several steps. First, you need to remediate any applications, updating any affected by the database changes. For example, you might need to update the connection strings to point to the new Azure SQL Database.

In addition, make sure there's thorough and complete testing. Validation testing will ensure that your application did not break because of changes at the database level. Construct tests to return data from both the source and target. Compare the data to ensure that queries are returning from the Azure SQL Database as they would with the original source database. Next, create performance tests that will:

- Validate that your application returns data in the times required by your organization.
- Enable you to do further optimizations, if necessary.

The post-migration phase is critical because it ensures that your data is both accurate and complete. In addition, it alerts you to any performance issues that might arise with the workload in the new environment.

Data migration tools in Azure

The core of data migration in Azure is the Azure Database Migration Service. You can use this service to move bulk amounts of data in a timely way. As part of Database

Migration Service, Microsoft provides Data Migration Assistant. Just as its name implies, Data Migration Assistant *assists* the service by preparing the target database.

Data Migration Assistant

Data Migration Assistant is a client-side tool that you can install on a Windows-compatible workstation or server.

First, it assesses your existing database and identifies any incompatibilities between that database and Azure SQL Database. It then generates a report of the things you need to fix before you can migrate. As you make changes, you can rerun Data Migration Assistant to generate an updated report of changes that you need to make. This capability helps you to not only track your progress, but also catch any new issues that might have been introduced during your coding phase.

After Data Migration Assistant completes the assessment and you've made any changes, you need to migrate the database schema to Azure SQL Database. Data Migration Assistant can help with this as well. It generates the required SQL, and then gives you the option of running the code, or saving it so you can run it yourself later.

Using Data Migration Assistant is not a requirement to use Azure Database Migration Service. You have the option of coding your new database in the Azure SQL Database service manually without trying to convert an existing database.

As an example, let's say you're creating a staging database in Azure SQL Database that will later feed data into Azure Synapse Analytics. The staging database will be sourced from multiple systems, but it will migrate only small portions of the source data. In this situation, you might be better off manually crafting the new database directly on the Azure SQL Database service rather than trying to automate the job.

Azure Database Migration Service

After you've migrated your database schema by using Data Migration Assistant, or created a target database manually, you're ready to move your data. To do that, you'll use Azure Database Migration Service.

Azure Database Migration Service is a fully-managed Azure service that provides automated, seamless data migrations from multiple sources into the Azure data platforms.

Database Migration Service runs on the Azure platform, as opposed to being a client application like Data Migration Assistant. It's capable of moving large amounts of data quickly and is not dependent upon installation of a client application. Database Migration Service can operate in two modes, offline and online.

In offline mode, no more changes can be made to your source database. Data is migrated, and then your applications can begin using the new Azure SQL Database.

In online mode, your source database can remain in use while the bulk of the data is migrated. At the end of the migration, you'll take the source system offline momentarily while any final changes to the source are synced to the new Azure SQL Database. At this point, your applications can cut over to use the SQL database.

Migration with downtime

There are two options for doing a migration: *offline* and *online*.

Offline

With offline mode, you take the source database offline. You place it in a state where no more updates are being made. You can then use Azure Database Migration Service to do the migration.

Online

During an online migration, the service takes a backup of the source database and migrates the data to the target platform. This enables the source database to continue to receive incoming transactions and return data. When the main part of the migration is complete, the user starts the cutover process. The source database is then taken offline, rendering it unavailable for use.

Database Migration Service then reads the data from the transaction log to bring the target database into sync. When this process is complete, the new target database becomes available for use. This is why the online option is sometimes called *minimal downtime*. There is some downtime, but it's minimal compared to doing an offline migration.

Although the online option looks attractive, there's a major downside - cost. The online option requires creating a SQL Server instance that's based on the Premium price tier.

This can become cost prohibitive, especially when you don't need any of the features of the Premium tier except its support of online migrations.

Because of this downside, we recommend that you first test by using the offline option to see whether it can run in an acceptable time frame.

Create an Azure Cosmos DB database built to scale

Provisioning throughput for containers and databases

In Azure Cosmos DB, you provision throughput for your containers to run writes, reads, updates, and deletes. You can provision throughput for an entire database and have it shared among containers within the database. You can also provision throughput dedicated to specific containers.

Setting provisioned throughput at a container is the most frequently used option. Throughput is reserved only for that container and it's evenly distributed among its physical partitions.

Throughput on an Azure Cosmos database is shared across all the containers in the database. An exception is if you specified a provisioned throughput on specific containers in the database. No predictable throughput guarantees for a particular container in that database are provided.

To scale throughput strategically, you need to estimate your throughput needs by estimating the number of operations you'll have to support at different times. If your requests consume all of the provisioned throughput, Azure Cosmos DB will rate-limit your requests. Operations will have to wait and retry, likely causing higher latency.

What is a request unit?

Azure Cosmos DB measures throughput using something called a **request unit (RU)**. Request unit usage is measured per second, so the unit of measure is **request units per second (RU/s)**. You must reserve the number of RU/s you want Azure Cosmos DB to provision in advance, so it can handle the load you've estimated, and you can scale your RU/s up or down at any time to meet current demand.

Request unit basics

A single request unit, one RU, is equal to the approximate cost of performing a single GET request on a 1-KB document using a document's ID. Performing a GET by using a document's ID is an efficient means for retrieving a document, and thus the cost is small. Creating, replacing, or deleting the same item requires additional processing by the service, and therefore requires more request units.

The number of request units consumed for an operation changes depending on the document size, the number of properties in the document, the operation being performed, and some additional concepts such as consistency and indexing policy.

When provisioning throughput, you should understand how many RUs your most common operations consume. You can obtain the request unit charge for any operation on your Azure Cosmos DB containers. Multiply the number of consumed RUs of each operation by the estimated number of times each operation (write, read, update, and delete) will be executed per second. If you run several different queries on your data, you should understand how many RUs each query will consume. By summing the number of consumed RUs for each operation, you will be able to accurately estimate how many RUs to provision.

Azure Cosmos DB guarantees that the number of RUs for a given database operation for the same dataset is deterministic. Understanding your application's throughput requirements and the factors that affect RU charges will allow you run your application cost effectively.

You provision the number of RUs on a per-second basis and you can change the value at any time in increments or decrements of 100 RUs. You can make your changes either programmatically or by using the Azure portal. You're billed on an hourly basis.

Request Unit considerations

While you estimate the number of RUs per second to provision, consider the following factors:

- **Item size**: As the size of an item increases, the number of RUs consumed to read or write the item also increases.
- **Item indexing**: By default, each item is automatically indexed. Fewer RUs are consumed if you choose not to index some of your items in a container.
- **Item property count**: Assuming the default indexing is on all properties, the number of RUs consumed to write an item increases as the item property count increases.
- **Indexed properties**: An index policy on each container determines which properties are indexed by default. To reduce the RU consumption for write operations, limit the number of indexed properties.
- **Data consistency**: The strong and bounded staleness consistency levels consume approximately two times more RUs on read operations when compared to that of other relaxed consistency levels.

- **Query patterns**: The complexity of a query affects how many RUs are consumed for an operation. Factors that affect the cost of query operations include:
    - The number of query results
    - The number of predicates
    - The nature of the predicates
    - The number of user-defined functions
    - The size of the source data
    - The size of the result set
    - Projections

  Azure Cosmos DB guarantees that the same query on the same data always costs the same number of RUs on repeated executions.

- **Script usage**: As with queries, stored procedures and triggers consume RUs based on the complexity of their operations. As you develop your application, inspect the request charge header to better understand how much RU capacity each operation consumes.

Exceeding throughput limits

If you attempt to use throughput higher than the one provisioned, your request will be rate-limited. When a request is rate-limited, the request has to be retried again after a specified interval. The .NET SDK will automatically retry your request after waiting the amount of time specified in the *retry-after* header.

Creating an account built to scale

You can change the number of request units provisioned to a database at any time. So, during heavy volume periods, you can scale up to accommodate those high demands, and then reduce provisioned throughput during off peak times to reduce costs.

When you create an account, you can provision a minimum of 400 RU/s, or a maximum of 250,000 RU/s in the portal. If you need even more throughput, fill out a ticket in the Azure portal.

Choose a partition key in Azure Cosmos DB

What is a partition strategy?

If you continue to add new data to a single server or a single partition, it will eventually run out of space. A partitioning strategy enables you to add more partitions to your database when need them. This scaling strategy is called **scale out** or **horizontal scaling**.

A partition key defines the partition strategy, it's set when you create a container and can't be changed. Selecting the right partition key is an important decision to make early in your development process.

In this unit, you'll learn how to choose a partition key that's right for your scenario, which will enable you to take advantage of Azure Cosmos DB autoscaling.

What is a partition key?

A partition key is the value by which Azure organizes your data into logical divisions. It should aim to evenly distribute operations across the database to avoid hot partitions. A hot partition is a single partition that receives many more requests than the others, which can create a throughput bottleneck.

In our online retail scenario, using the `userID` or `productId` value as the partition key is a good choice because it will be unique and likely used to look up records. `userID` is a good choice, as your application frequently needs to retrieve the personalization settings, shopping cart, order history, and profile information for the user, just to name a few. `productId` is also a good choice, as your application needs to query inventory levels, shipping costs, color options, warehouse locations, and more.

Using the current time would be a poor choice of partition key, because all the incoming data would go to a single partition key. `userID` or `productId` would be better, as all the users on your site would likely be adding and updating their shopping cart or profile information at about the same frequency, which distributes the reads and writes across all the user and product partitions.

The amount of required RU's and storage determines the number of required physical partitions for the container, which are completely managed by Azure Cosmos DB. When

additional physical partitions are needed, Cosmos DB automatically creates them by splitting existing ones. There is no downtime or performance impact for the application.

The storage space for the data associated with each partition key can't exceed 20 GB, which is the size of one physical partition in Azure Cosmos DB. So, if your single `userID` or `productId` record is going to be larger than 20 GB, think about using a composite key instead so that each record is smaller. An example of a composite key would be `userID-date`, which would look like **CustomerName-08072018**. This composite key approach would enable you to create a new partition for each day a user visited the site.

Best practices

When you're trying to determine the right partition key and the solution isn't obvious, here are a few tips to keep in mind.

- Don't be afraid of choosing a partition key that has a large number of values. The more values your partition key has, the more scalability you have.
- To determine the best partition key for a read-heavy workload, review the top three to five queries you plan on using. The value most frequently included in the WHERE clause is a good candidate for the partition key.
- For write-heavy workloads, you'll need to understand the transactional needs of your workload, because the partition key is the scope of multi-document transactions.

Review: Choosing a Partition Key

For each Azure Cosmos DB container, you should specify a partition key that satisfies the following core properties:

- Have a high cardinality. This option allows data to distribute evenly across all physical partitions.
- Evenly distribute requests. Remember the total number of RU/s is evenly divided across all physical partitions.
- Evenly distribute storage. Each partition can grow up to 20 GB in size.

Distribute your data globally with Azure Cosmos DB

Write to multiple regions

What is multi-master support?

Multi-master support is an option that can be enabled on new Azure Cosmos DB accounts. Once the account is replicated in multiple regions, each region is a master region that equally participates in a write-anywhere model, also known as an active-active pattern.

Azure Cosmos DB regions operating as master regions in a multi-master configuration automatically work to converge data written to all replicas and ensure global consistency and data integrity.

With Azure Cosmos DB multi-master support, you can perform writes on any container in a write-enabled region world-wide. Written data is propagated to all other regions immediately.

What are the benefits of multi-master support?

The benefits of multi-master support are:

- Single-digit write latency – Multi-master accounts have an improved write latency of <10 ms for 99% of writes, up from <15 ms for non-multi-master accounts.
- 99.999% read-write availability - The write availability multi-master accounts increases to 99.999%, up from the 99.99% for non-multi-master accounts.
- Unlimited write scalability and throughput – With multi-master accounts, you can write to every region, providing unlimited write scalability and throughput to support billions of devices.
- Built-in conflict resolution – Multi-master accounts have three methods for resolving conflicts to ensure global data integrity and consistency.

Conflict resolution

With the addition of multi-master support comes the possibility of encountering conflicts for writes to different regions. Conflicts are rare in Azure Cosmos DB and can only occur when an item is simultaneously changed in multiple regions, before

propagation between the regions has happened. Given the speed with which replication happens globally, you should not experience conflicts often, if at all. However, Azure Cosmos DB does provide conflict resolution modes that allow users to decide how to handle scenarios where the same record is updated simultaneously by different writers in two or more regions.

There are three conflict resolution modes offered by Azure Cosmos DB.

- **Last-Writer-Wins (LWW)**, in which conflicts are resolved based on the value of a user-defined integer property in the document. By default _ts is used to determine the last written document. Last-Writer-Wins is the default conflict handling mechanism.
- **Custom - User-defined function**, in which you can fully control conflict resolution by registering a User-defined function to the collection. A User-defined function is a special type of stored procedure with a specific signature. If the User-defined function fails or does not exist, Azure Cosmos DB will add all conflicts into the read-only conflicts feed they can be processed asynchronously.
- **Custom - Async**, in which Azure Cosmos DB excludes all conflicts from being committed and registers them in the read-only conflicts feed for deferred resolution by the user's application. The application can perform conflict resolution asynchronously and use any logic or refer to any external source, application, or service to resolve the conflict.

Failover basics

In the rare event of an Azure regional outage or data center outage, Azure Cosmos DB automatically triggers failovers of all Azure Cosmos DB accounts with a presence in the affected region.

What happens if a read region has an outage?

Azure Cosmos DB accounts with a read region in one of the affected regions are automatically disconnected from their write region and marked offline. The Azure Cosmos DB SDKs implement a regional discovery protocol that allows them to automatically detect when a region is available and redirect read calls to the next available region in the preferred region list. If none of the regions in the preferred region list is available, calls automatically fall back to the current write region. No changes are required in your application code to handle regional failovers. During this entire process, consistency guarantees continue to be honored by Azure Cosmos DB.

Once the affected region recovers from the outage, all the affected Azure Cosmos DB accounts in the region are automatically recovered by the service. Azure Cosmos DB accounts that had a read region in the affected region will then automatically sync with current write region and turn online. The Azure Cosmos DB SDKs discover the availability of the new region and evaluate whether the region should be selected as the current read region based on the preferred region list configured by the application. Subsequent reads are redirected to the recovered region without requiring any changes to your application code.

What happens if a write region has an outage?

If the affected region is the current write region and automatic failover is enabled for the Azure Cosmos DB account, then the region is automatically marked as offline. Then, an alternative region is promoted as the write region for the affected Azure Cosmos DB account.

During automatic failovers, Azure Cosmos DB automatically chooses the next write region for a given Azure Cosmos DB account based on the specified priority order. Applications can use the WriteEndpoint property of DocumentClient class to detect the change in write region.

Once the affected region recovers from the outage, all the affected Azure Cosmos DB accounts in the region are automatically recovered by the service.

Now let's modify the read region for your database.

Use of consistency levels

About 73% of Azure Cosmos DB tenants use session consistency and 20% prefer bounded staleness. Approximately 3% of Azure Cosmos DB customers experiment with various consistency levels initially before settling on a specific consistency choice for their application. Only 2% of Azure Cosmos DB tenants override consistency levels on a per request basis.

Design a data warehouse with Azure Synapse Analytics

Analytics capabilities using Azure Synapse SQL through either dedicated SQL pools or SQL Serverless pools

Azure Synapse SQL is a distributed query system that enables you to implement data warehousing and data virtualization scenarios using standard T-SQL experiences familiar to data engineers.

Synapse SQL offers both serverless and dedicated resource models to work with both descriptive and diagnostic analytical scenarios. For predictable performance and cost, you can create dedicated SQL pools to reserve processing power for data stored in SQL tables. For unplanned or ad hoc workloads, you can use the always-available, serverless SQL endpoint.

Apache Spark pool with full support for Scala, Python, SparkSQL, and C#

You can develop big data engineering and machine learning solutions using Apache Spark for Azure Synapse. You can take advantage of the big data computation engine to deal with complex compute transformations that would take too long in a data warehouse.

For machine learning workloads, you can use SparkML algorithms and AzureML integration for Apache Spark 2.4 with built-in support for Linux Foundation Delta Lake.

There is a simple model for provisioning and scaling the Spark clusters to meet your compute needs, regardless of the operations that you are performing on the data.

Integrate your data with Azure Synapse pipelines

Azure Synapse pipelines leverages the capabilities of Azure Data Factory. It's the cloud-based ETL and data integration service that enables you to create data-driven workflows for orchestrating data movement, and transforming data at scale. Using Azure Synapse pipelines, you can create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores. You can build complex ETL processes that transform data visually with data flows, or by using compute services such as Azure HDInsight Hadoop, or Azure Databricks.

Use Azure Synapse Link to perform operational analytics with near real-time hybrid transactional and analytical processing

Azure Synapse Analytics enables you to reach out to operational data using Azure Synapse Link, and is achieved without impacting the performance of the transactional data store. For this to happen, you have to enable the feature within both Azure Synapse Analytics, and within the data store to which Azure Synapse Analytics will connect, such as Azure Cosmos DB.

In the case of Azure Cosmos DB, this will create an analytical data store. As data changes in the transactional system, the changed data is fed to the analytical store in a Column store format from which Azure Synapse Link can query with no disruption to the source system.

A single Web UI to access all Azure Synapse Analytics capabilities

While the Azure portal will enable you to manage some aspects of the product, Azure Synapse Studio is the best place to centrally work with all the capabilities.

Azure Synapse Studio is a single web UI that enables you to:

- Explore your data estate.
- Develop TSQL scripts and notebooks to interact with the analytical engines.
- Build data integration pipelines for managing data movement.
- Monitor the workloads within the service.
- Manage the components of the service.

Integrate with a variety of Azure Data Platform technologies

For organizations that have existing analytical solutions, Azure Synapse Analytics can integrate with a wide variety of technologies to complement them. For example, if you are already using Azure Data Factory to build data integration pipelines, these can be used to load data into Azure Synapse Analytics.

You can also integrate existing data preparation or data science projects that you may hold in Azure Databricks. Integration is also available with many of Azure security components to ensure that you meet security and compliance requirements within your organization.

Upon initially deploying Azure Synapse Analytics, a few resources deploy along with it, including the Azure Synapse Workspace and an Azure Data Lake Storage Gen2 (ADLS Gen2) account that acts as the primary storage for the workspace.

Within the Azure portal, there are links to configure your workspace, manage access through Access control (IAM), firewalls, managed identities, and private endpoint connections, as well as view metrics.

The portal also contains important information about your Synapse Analytics environment, such as:

- The **Primary ADLS Gen2 account URL (1)**, which identifies the primary data lake storage account.
- The **SQL endpoint** and **SQL on-demand endpoint (2)**, which are used to integrate with external tools, such as SQL Server Management Studio (SSMS), Azure Data Studio, and Power BI.
- The **Workspace web URL (3)**, a direct link to Synapse Studio for the workspace.
- Available resources, such as **SQL pools** and **Apache Spark pools (4)**.

Azure Synapse Analytics features

Workload management

Azure Synapse Analytics provides the capability to prioritize the query workloads that take place on the server using Workload Management. Workload Management is managed by three related areas:

- Workload groups
- Workload classification
- Workload importance

Workload groups

Workload groups enable you to define the resources to isolate and reserve resources for its use. It creates the following benefits:

- Reserves resources for a group of requests.
- Limits the amount of resources a group of requests can consume.
- Accesses shared resources based on importance level.

- Sets query timeout value. Gets DBAs out of the business of terminating runaway queries.

Workload classification

Using T-SQL, you can create a workload classifier to map queries to a specific classifier. A classifier can define the level of importance of the request, so that it can be mapped to a specific workload group that has an allocation of specific resources during the execution of the query.

Workload importance

Workload importance is defined in the CREATE WORKLOAD CLASSIFIER command, and enables higher priority queries to receive resources ahead of lower priority queries that are in the queue. By default, queries are released from the queue on a first-in, first-out basis as resources become available, but workload importance overrides this qualifier.

Result-set cache

In scenarios where the same results are requested on a regular basis, result-set caching can improve the performance of the queries that retrieve these results. When result-set caching is enabled, the results of the query are cached in the SQL pool storage.

Result-set cache enables interactive response times for repetitive queries against tables with infrequent data changes. The result-set cache persists even if SQL pool is paused and resumed later, although the query cache is invalidated and refreshed when the underlying table data or query code changes. To ensure that the cache is fresh, the result cache is evicted on a regular basis on a time-aware least recently used algorithm (TLRU).

Materialized views

A materialized view can pre-compute, store, and maintain data like a table. These views are automatically updated when data in underlying tables are changed. Updating materialized views is a synchronous operation that occurs as soon as the data is changed. This auto-caching functionality enables Azure Synapse Analytics Query Optimizer to consider using an indexed view, even if the view is not referenced in the

query. They also support the following aggregations: **MAX, MIN, AVG, COUNT, COUNT_BIG, SUM, VAR, STDEV**.

Continuous Integration/Continuous Delivery (CI/CD) support through SQL Server Data Tools (SSDT)

Database project support in SQL Server Data Tools (SSDT) enables teams of developers to collaborate over a version-controlled Azure Synapse Analytics, and track, deploy, and test schema changes.

Other supported features include:

- Ordered ColumnStore
- JSON support
- Dynamic data masking
- Row-level security

Types of solution scenarios

Descriptive analytics

Descriptive analytics answers the question "What is happening in my business?" The data to answer this question is typically found through the creation of a data warehouse. Azure Synapse Analytics uses the dedicated SQL Pool capability that enables you to create a persisted data warehouse to perform this type of analysis. You can also make use of SQL Serverless to prepare data from files to create a data warehouse interactively to answer the question too.

Diagnostic analytics

Diagnostic analytics deals with answering the question "Why is it happening?" This type of analytics may involve exploring information that already exists in a data warehouse, but typically involves a wider search of your data estate to find more data to support this type of analysis.

You can use the same SQL serverless capability within Azure Synapse Analytics that enables you to interactively explore data within a data lake. This can quickly enable a user to search for other data that may help them to understand "Why is it happening?"

Predictive analytics

Azure Synapse Analytics also enables you to answer the question “What is likely to happen in the future based on previous trends and patterns?” by using its integrated Apache Spark engine. This can also be used with other services, such as Azure Machine Learning Services, or Azure Databricks.

Prescriptive analytics

This type of analytics looks at executing actions based on real-time or near real-time analysis of data, using predictive analytics. Azure Synapse Analytics provides this capability through both Apache Spark, Azure Synapse Link, and by integrating streaming technologies such as Azure Stream Analytics.

Azure Synapse Analytics gives users of the service the freedom to query data on their own terms, using either serverless or dedicated resources at scale. Azure Synapse Analytics brings these two worlds together with a unified data integration experience to ingest, prepare, manage, and serve data using Azure Synapse Pipelines. In addition, you can visualize the data in the form of dashboards and reports for immediate analysis using Power BI, which is integrated into the service too.

These types of analytics are typically implemented in three common types of solution workloads as follows:

Modern Data Warehouse workloads

A Modern Data Warehouse is a centralized data store that provides analytics and decision support services across the whole enterprise using structured, unstructured, or streaming data sources. Data flows into the warehouse from multiple transactional systems, relational databases, and other data sources on a periodic basis. The stored data is used for historical and trend analysis reporting. The data warehouse acts as a central repository for many subject areas and contains the "single source of truth."

SQL Analytics stores data in relational tables with columnar storage. This format significantly reduces the data storage costs, and improves query performance. After data is stored, you can run analytics at massive scale. Compared to traditional database systems, analysis queries finish in seconds instead of minutes, or hours instead of days. A data warehouse is useful to reduce stress on production systems and to reorganize, index, or filter multiple data sources into a single storage area. To accomplish this goal,

deliberate thought and design needs to be put into how you organize and structure the data. It's not a dumping ground for tables from various sources without any design put into it.

Advanced Analytical workloads

You can perform advanced analytics in the form of predictive or preemptive analytics using Azure Synapse Analytics. Using the tight integration with Spark, and the hybrid data integration engine with Data Integration. In a cloud data solution, data is ingested into big data stores from various sources. Once in a big data store, Hadoop, Spark, and machine learning algorithms prepare and train the data. When the data is ready for complex analysis, SQL Analytics uses PolyBase to query the big data stores.

PolyBase uses standard T-SQL queries to bring the data into SQL Analytics tables. As a result, the worlds of the Modern Data Warehouse and the Advanced Analytical workloads are brought together.

Real-time analytics

Real-time analytics enables you to capture data continuously from devices or applications and to process it in near real time. This enables you to get live insights, and can be set up with ease from a range of devices.

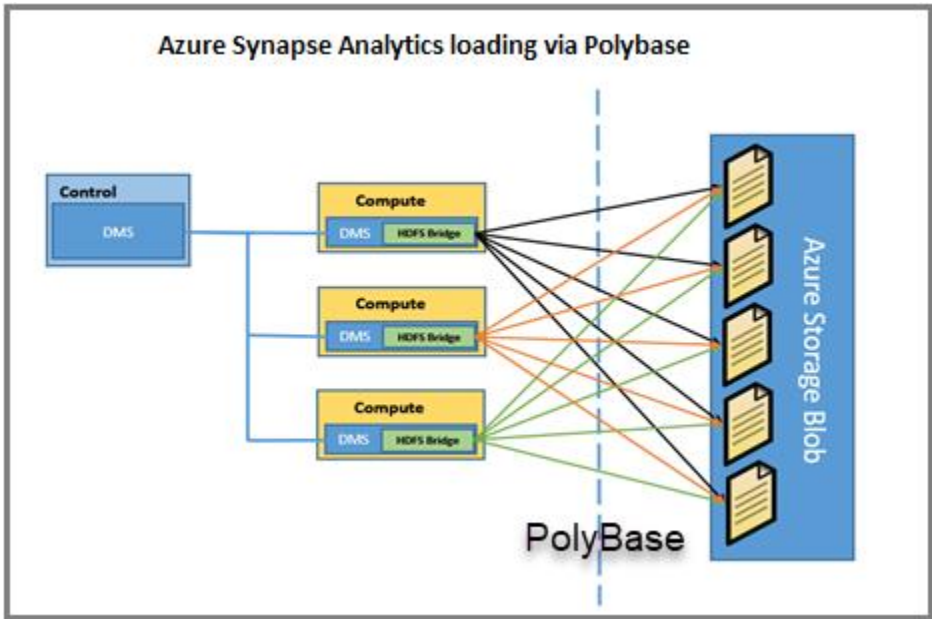
Import data into Azure Synapse Analytics by using PolyBase

The data warehouse component of Azure Synapse Analytics service is a relational big data store that uses a massively parallel processing (MPP) architecture. It takes advantage of the on-demand elastic scale of Azure compute and storage resources to load and process petabytes of data. With Azure Synapse Analytics, you get quicker access to the critical information you need to make good business decisions.

A key feature of Azure Synapse Analytics is that you pay for only the processing you need. You can decide how much parallelism is needed for your work. You also can pause the compute nodes when it's not in use. In this way, you pay for only the CPU time you use.

Azure Synapse Analytics supports many loading methods. These methods include non-PolyBase options such as Bulk Copy Utility (BCP) and the SQL Bulk Copy API. The fastest and most scalable way to load data is through PolyBase. PolyBase is a technology that accesses external data stored in Azure Blob storage, Hadoop, or Azure Data Lake Store via the Transact-SQL language.

The following architecture diagram shows how loading is achieved with each Hadoop Distributed File System (HDFS) bridge of the data movement service (DMS) on every compute node that connects to an external resource such as Azure Blob storage. PolyBase then bidirectionally transfers data between Azure Synapse Analytics and the external resource to provide the fast load performance.



Use PolyBase to extract, load, and transform data

Follow these steps to implement a PolyBase extract, load, and transform process for Azure Synapse Analytics:

1. Extract the source data into text files.
2. Load the data into Azure Blob storage, Hadoop, or Azure Data Lake Store.
3. Import the data into Azure Synapse Analytics staging tables by using PolyBase.
4. Transform the data (optional).
5. Insert the data into production tables.

Create serverless logic with Azure Functions

Benefits of a serverless compute solution

Serverless compute is a great option for hosting business logic code in the cloud. With serverless offerings such as Azure Functions, you can write your business logic in the language of your choice. You get automatic scaling, you have no servers to manage, and you are charged based on what is used — not on reserved time. Here are some additional characteristics of a serverless solution for you to consider.

Avoids over-allocation of infrastructure

Suppose you've provisioned VM servers and configured them with enough resources to handle your peak load times. When the load is light, you are potentially paying for infrastructure you're not using. Serverless computing helps solve the allocation problem by scaling up or down automatically, and you're only billed when your function is processing work.

Stateless logic

Stateless functions are great candidates for serverless compute; function instances are created and destroyed on demand. If state is required, it can be stored in an associated storage service.

Event driven

Functions are *event driven*. This means they run only in response to an event (called a "trigger"), such as receiving an HTTP request, or a message being added to a queue. You configure a trigger as part of the function definition. This approach simplifies your code by allowing you to declare where the data comes from (trigger/input binding) and where it goes (output binding). You don't need to write code to watch queues, blobs, hubs, etc. You can focus purely on the business logic.

Functions can be used in traditional compute environments

Functions are a key component of serverless computing, but they are also a general compute platform for executing any type of code. Should the needs of your app change,

you can take your project and deploy it in a non-serverless environment, which gives you the flexibility to manage scaling, run on virtual networks, and even completely isolate your functions.

Drawbacks of a serverless compute solution

Serverless compute will not always be the appropriate solution to hosting your business logic. Here are a few characteristics of functions that may affect your decision to host your services in serverless compute.

Execution time

By default, functions have a timeout of 5 minutes. This timeout is configurable to a maximum of 10 minutes. If your function requires more than 10 minutes to execute, you can host it on a VM. Additionally, if your service is initiated through an HTTP request and you expect that value as an HTTP response, the timeout is further restricted to 2.5 minutes. Finally, there's also an option called **Durable Functions** that allows you to orchestrate the executions of multiple functions without any timeout.

Execution frequency

The second characteristic is execution frequency. If you expect your function to be executed continuously by multiple clients, it would be prudent to estimate the usage and calculate the cost of using functions accordingly. It might be cheaper to host your service on a VM.

While scaling, only one function app instance can be created every 10 seconds, for up to 200 total instances. Keep in mind, each instance can service multiple concurrent executions, so there is no set limit on how much traffic a single instance can handle. Different types of triggers have different scaling requirements, so research your choice of trigger and investigate its limits.

Route and process data automatically using Logic Apps

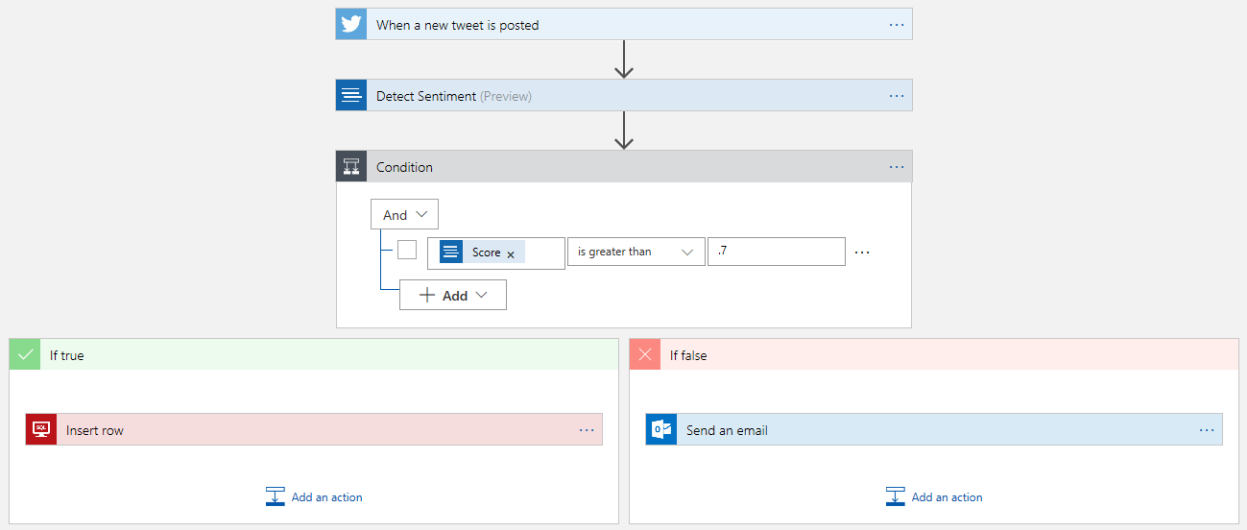
Map your steps to Logic Apps components

Let's be more formal about the definitions of the component types:

- A *trigger* is an event that occurs when a specific set of conditions is satisfied. Triggers activate automatically when the conditions are right. For example, when a timer expires or data becomes available. Every logic app must start with a trigger. In our example, we'll trigger the app when a new tweet mentions our product.
- An *action* is an operation that executes one of the tasks in your business process. Actions run when a trigger activates or another action completes. Our social-media monitor app has three actions: detect sentiment, insert database row, and send email.
- *Control actions* are special built-in actions that let you add decisions and loops to your app. Our example will use a control action to branch based on the sentiment score.

Define your app using the Logic Apps Designer

The Logic Apps Designer is a graphical tool for creating your workflows. It lets you pick from a gallery of connectors that contain the triggers and actions you can use in your app. You'll use the graphical Logic Apps Designer to arrange the trigger, actions, and control actions. The following screenshot shows the designer with the completed application.



When you save your app, it will be live and will run automatically whenever the trigger activates.

Trigger types

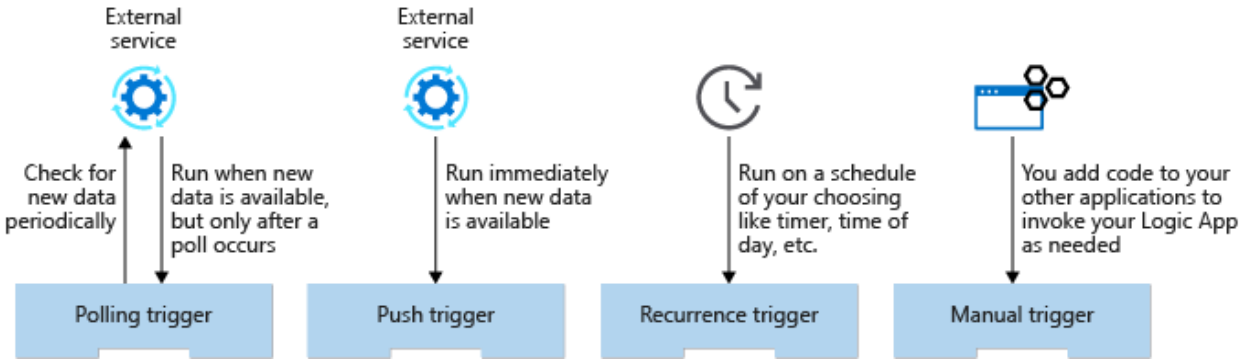
Think about the different conditions that businesses might use to launch their Logic Apps.

Most of the examples we've seen are in the *data becomes available* category. For example, a new tweet is posted, a new row is inserted into a database, a new email arrives, or a new file is uploaded to your cloud storage. This category doesn't cover all cases though.

Suppose you wanted to launch your logic app every Saturday at midnight? This trigger would be great for administrative tasks like running backups or archiving old data. Logic Apps provides a built-in *recurrence* trigger to help you do exactly this type of thing.

There's one more case to consider: suppose you wanted total control? Imagine you need to launch your logic app using code in your web or mobile applications? You can use the built-in *manual request* trigger to do this action.

This discussion shows that we have three broad categories of triggers: data, time, and manual. Data triggers use two different techniques to detect that new data is available: some use *polling* and some rely on the external service to *push* a notification. These two types of data triggers are so different, that we should think of them as separate categories. Altogether, we have four types of triggers, the following illustration shows a summary of the cases.

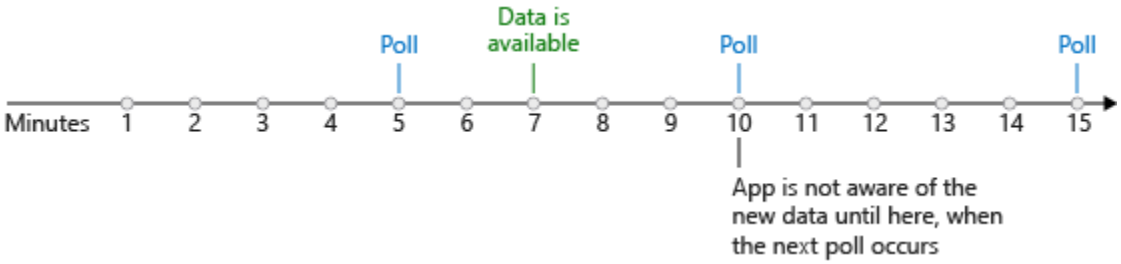


What is a polling trigger?

A *polling trigger* periodically checks an external service for new data. For example, the trigger that looks for new posts in an RSS feed is implemented using polling.

When you create a polling trigger, you set the **Frequency** and an **Interval** to control how often the trigger will run. The frequency is the unit of measurement and has values like **Second**, **Minute**, and **Hour**. Interval is a number that represents how often to execute. For example, a polling trigger with a frequency of **Minute** and an interval of **5** would run every five minutes.

Polling triggers force you to make a choice between how much they cost and how quickly they respond to new data. There is often a delay between when new data becomes available and when it is detected by the app. The following illustration shows the issue.

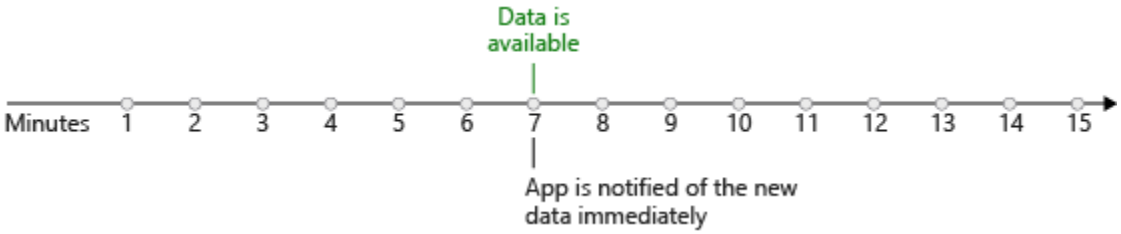


In the worst case, the potential delay for detecting new data is equal to the polling interval. So why not use a smaller interval? To check for new data, the Logic Apps execution engine needs to run your app, which means you incur a cost. In general, the shorter the interval, the higher the cost but the quicker you respond to new data. The best polling interval for your logic app depends on your business process and its tolerance for delay.

What is a push trigger?

A *push trigger* subscribes to an event offered by the external service to get notified immediately when data is available. For example, the trigger that detects when a message is added to an Azure Service Bus queue is a push trigger.

The nice thing about push triggers is that they don't incur any costs polling for data when none is available. They also respond immediately when new data is ready. The following illustration shows this immediate response.



If push triggers respond more quickly and cost less than polling triggers, then why not use them all the time? The reason is that not every connector offers a push trigger. Sometimes the trigger author chose not to implement push and sometimes the external service didn't support push. Generally, you'll find a connector offers either push or polling triggers but not both. In the rare cases where both options are available, consider using the push trigger because it should be more efficient.

In this module, we're going to focus on polling triggers. These triggers are the most common and are perfect for the "route and process data" scenarios that we've been discussing.

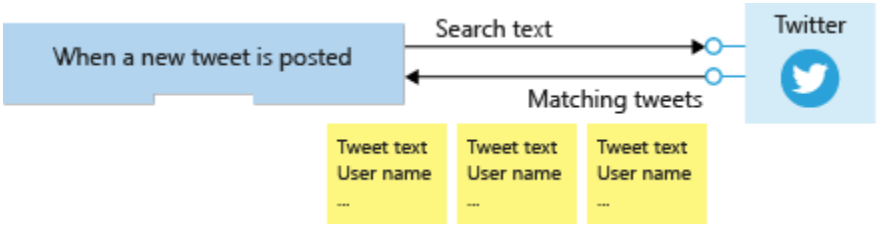
Trigger parameters and return values

You can think of trigger operations as function calls that have parameters and return values.

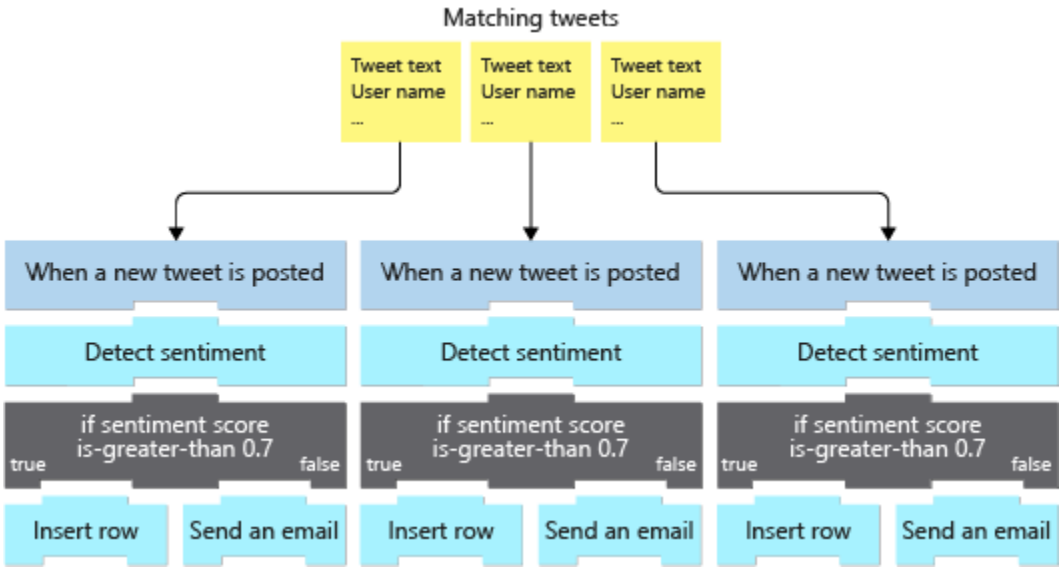
Trigger *parameters* let you configure the operation. The Twitter **When-a-new-tweet-is-posted** trigger has a parameter called **Search text** that it uses to select matching tweets for us. Some operations have a mix of required and optional parameters. The SQL Server **When an item is created** trigger has one required parameter named **Table name** and several optional parameters like **Order By** and **Select Query**.

Trigger *return values* are the results of the operation. The bitbucket connector has a **When a pull request is merged** trigger. The trigger returns an object containing

things like the identity of the **Repository** and the **Actor** who approved the merge. Most triggers actually return a collection instead of a single object. The Twitter **When a new tweet is posted** trigger returns an array of **TweetModel** objects. Each object contains values like the **Tweet text**, **User name**, and **Followers count**. The following illustration shows a collection being returned from a trigger.



You can use a loop to process each item or you can ask the trigger to split the array up for you. The default behavior for most triggers, including the Twitter trigger, is to automatically split the array. The Logic Apps execution engine will create one instance of your logic app for each data item and the instances will run in parallel. The following illustration shows how each item in the returned array is sent to a different instance of the logic app.



How to create a logic app in the Azure portal

You can use the Azure portal to create a logic app. You select the **Logic App** resource type and enter the standard resource properties **Name**, **Subscription**, **Resource group**, and **Location**. After deployment completes, you can navigate to the Logic Apps resource that you created.

The Logic Apps team has created several *templates* for common application types. For example, there are templates for apps like **Post to Slack if a new tweet matches with some hashtag** and **Get daily reminders emailed to you**.

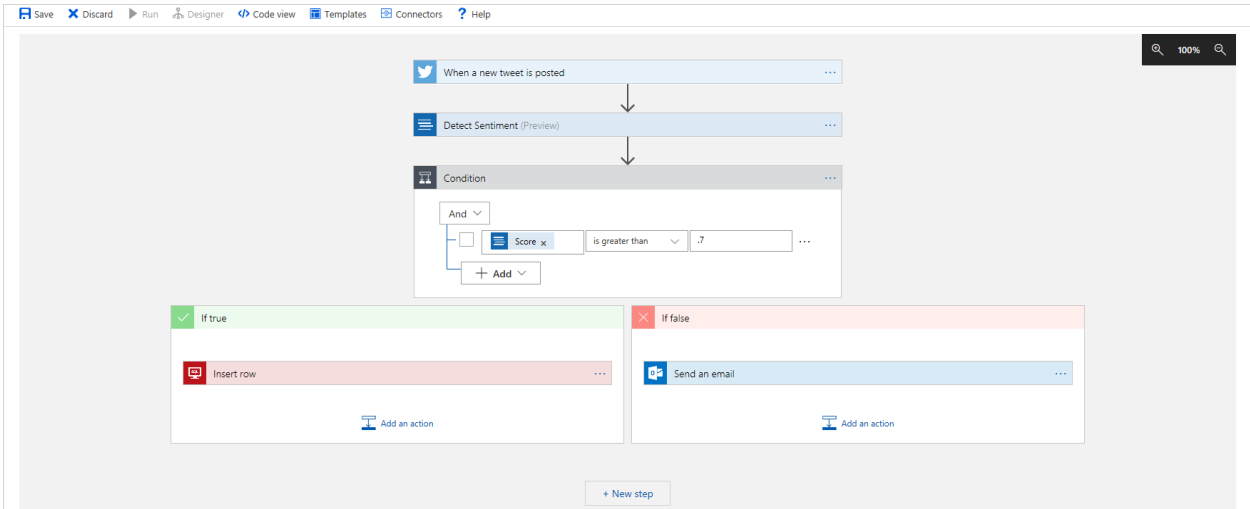
When you first navigate to your newly deployed logic app, you'll find a getting-started page. This page can add a common trigger to your app or generate an entire app for you from one of the templates. If any of these templates matches what you're working on, they can save you some time in getting your app set up. To do all the work yourself, there's also a **Blank Logic App** template.

After you select a starting template, you'll automatically navigate to the Logic Apps Designer.

How to add a trigger using the designer?

The Logic Apps Designer lets you pick from a gallery of connectors that contain the triggers and actions you can use in your app. The typical strategy is to use the search feature to locate the connector you are interested in. Then you look through the triggers supplied by the connector to find the one you want. In our case, we will use Twitter's **When-a-new-tweet-is-posted** trigger.

Once you've added the trigger, the designer gives you a GUI to set its properties. We'll set the **Search text**, **Frequency**, and **Interval** parameters. The following screenshot shows the social-media monitor logic app displayed in the designer; notice that it begins with the Twitter trigger.



Choose a messaging model in Azure to loosely connect your services

How to choose messages or events

A single application is likely to use events for some purposes and messages for others. Before you choose, you must analyze your application's architecture and all its use cases, to identify all the different purposes where its components have to communicate with each other.

Events are more likely to be used for broadcasts and are often ephemeral, meaning a communication might not be handled by any receiver if none is currently subscribing. Messages are more likely to be used where the distributed application requires a guarantee that the communication will be processed.

For each communication, consider the following question: **Does the sending component expect the communication to be processed in a particular way by the destination component?**

If the answer is *yes*, choose to use a message. If the answer is *no*, you may be able to use events.

Understanding how your components need to communicate will help you to choose how your components will communicate. Let's start with messages.

What are Azure Service Bus Topics?

Azure Service Bus topics are like queues, but can have multiple subscribers. When a message is sent to a topic instead of a queue, multiple components can be triggered to do their work. Imagine in a music-sharing application, a user is listening to a song. The mobile app might send a message to the "Listened" topic. That topic will have a subscription for "UpdateUserListenHistory", and a different subscription for "UpdateArtistsFanList". Each of those functions is handled by a different component that receives its own copy of the message.

Internally, topics use queues. When you post to a topic, the message is copied and dropped into the queue for each subscription. The queue means that the message copy will stay around to be processed *by each subscription branch* even if the component processing that subscription is too busy to keep up.

Benefits of queues

Queue infrastructures can support many advanced features that make them useful in the following ways.

Increased reliability

Queues are used by distributed applications as a temporary storage location for messages pending delivery to a destination component. The source component can add a message to the queue and destination components can retrieve the message at the front of the queue for processing. Queues increase the reliability of the message exchange because, at times of high demand, messages can wait until a destination component is ready to process them.

Message delivery guarantees

Queuing systems usually guarantee delivery of each message in the queue to a destination component. However, these guarantees can take different approaches:

- **At-Least-Once Delivery**: In this approach, each message is guaranteed delivery to at least one of the components that retrieve messages from the queue. Note, however, that in certain circumstances, it is possible that the same message may be delivered more than once. For example, if there are two instances of a web app retrieving messages from a queue, ordinarily each message goes to only one of those instances. However, if one instance takes a long time to process the message, and a time-out expires, the message may be sent to the other instance as well. Your web app code should be designed with this possibility in mind.
- **At-Most-Once Delivery**: In this approach, each message is not guaranteed for delivery, and there is a small chance that it may not arrive. However, unlike At-Least-Once delivery, there is no chance that the message will be delivered twice. This is sometimes referred to as *automatic duplicate detection*.
- **First-In-First-Out (FIFO)**: In most messaging systems, messages usually leave the queue in the same order in which they were added, but you should consider whether this delivery is guaranteed. If your distributed application requires that messages are processed in precisely the correct order, you must choose a queue system that includes a FIFO guarantee.

Transactional support

Some closely related groups of messages may cause problems when delivery fails for one message in the group.

For example, consider an e-commerce application. When the user clicks the **Buy** button, a series of messages might be generated and sent off to various processing destinations:

- A message with the order details is sent to a fulfillment center
- A message with the total and payment details is sent to a credit card processor.
- A message with the receipt information is sent to a database to generate an invoice for the customer

In this case, we want to make sure *all* messages get processed, or none of them are processed. We won't be in business long if the credit card message is not delivered, and all our orders are fulfilled without payment! You can avoid these kinds of problems by grouping the two messages into a transaction. Message transactions succeed or fail as a single unit - just like in the database world. If the credit card details message delivery fails, so will the order details message.

Which service should I choose?

Having understood that the communication strategy for this architecture should be a message, you must choose whether to use Azure Storage queues or Azure Service Bus, both of which can be used to store and deliver messages between your components. Each has a slightly different feature set, which means you can choose one or the other, or use both, depending on the problem you are solving.

Use Service Bus topics if you:

- Need multiple receivers to handle each message

Use Service Bus queues if you:

- Need an At-Most-Once delivery guarantee.
- Need a FIFO guarantee.
- Need to group messages into transactions.
- Want to receive messages without polling the queue.
- Need to provide a role-based access model to the queues.
- Need to handle messages larger than 64 KB but less than 256 KB.

- Queue size will not grow larger than 80 GB.
- Want to publish and consume batches of messages.

Queue storage isn't quite as feature rich, but if you don't need any of those features, it can be a simpler choice. In addition, it's the best solution if your app has any of the following requirements.

Use Queue storage if you:

- Need an audit trail of all messages that pass through the queue.
- Expect the queue to exceed 80 GB in size.
- Want to track progress for processing a message inside of the queue.

A queue is a simple, temporary storage location for messages sent between the components of a distributed application. Use a queue to organize messages and gracefully handle unpredictable surges in demand.

Use Storage queues when you want a simple and easy-to-code queue system. For more advanced needs, use Service Bus queues. If you have multiple destinations for a single message, but need queue-like behavior, use Service Bus topics.

What is Azure Event Grid?

Azure Event Grid is a fully-managed event routing service running on top of Azure Service Fabric. Event Grid distributes *events* from different sources, such as Azure Blob storage accounts or Azure Media Services, to different handlers, such as Azure Functions or Webhooks. Event Grid was created to make it easier to build event-based and serverless applications on Azure.

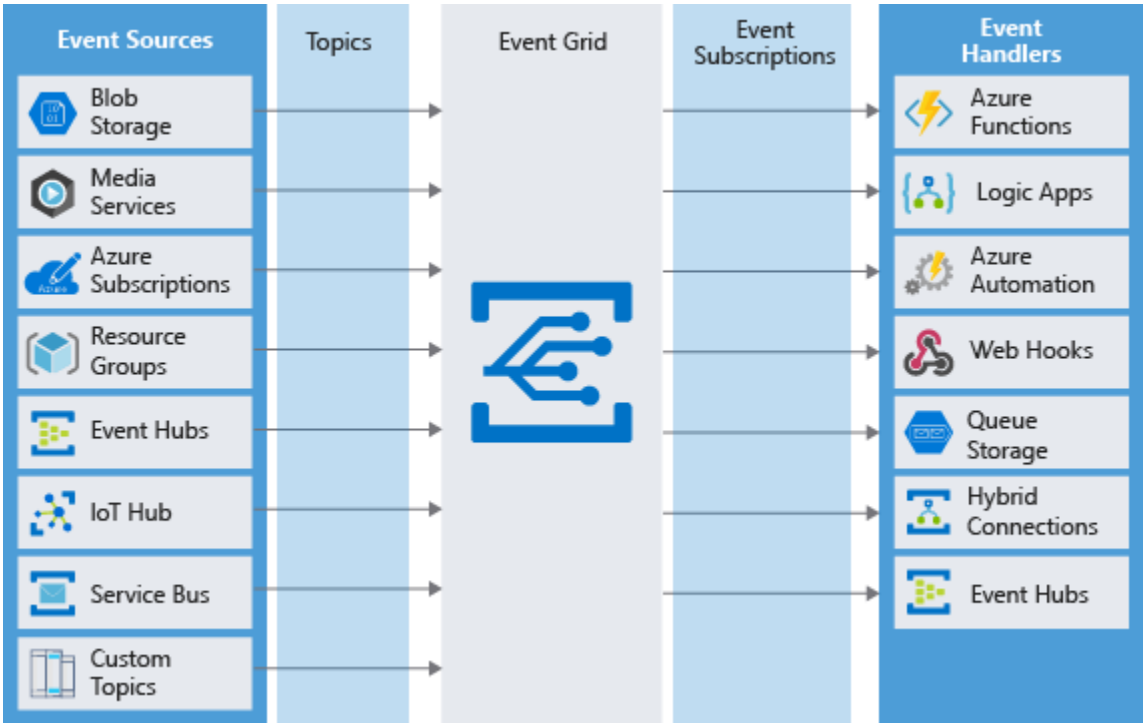
Event Grid supports most Azure services as a publisher or subscriber and can be used with third-party services. It provides a dynamically scalable, low-cost, messaging system that allows publishers to notify subscribers about a status change. The following illustration shows Azure Event Grid receiving messages from multiple sources and distributing them to event handlers based on subscription.

There are several concepts in Azure Event Grid that connect a source to a subscriber:

- **Events:** What happened.
- **Event sources:** Where the event took place.
- **Topics:** The endpoint where publishers send events.

- **Event subscriptions:** The endpoint or built-in mechanism to route events, sometimes to multiple handlers. Subscriptions are also used by handlers to filter incoming events intelligently.
- **Event handlers:** The app or service reacting to the event.

The following illustration shows an Azure Event Grid positioned between multiple event sources and multiple event handlers. The event sources send events to the Event Grid and the Event Grid forwards relevant events to the subscribers. Event Grid use topics to decide which events to send to which handlers. Events sources tag each event with one or more topics, and event handlers subscribe to the topics they are interested in.



What is an event topic?

Event topics categorize events into groups. Topics are represented by a public endpoint and are where the event source sends events *to*. When designing your application, you can decide how many topics to create. Larger solutions will create a custom topic for each category of related events, while smaller solutions might send all events to a single topic. For example, consider an application that sends events related to modifying user accounts and processing orders. It's unlikely any event handler wants both categories of events. Create two custom topics and let event handlers subscribe to the one that interests them. Event subscribers can filter for the event types they want from a specific topic.

Topics are divided into **system** topics, and **custom** topics.

System topics

System topics are built-in topics provided by Azure services. You don't see system topics in your Azure subscription because the publisher owns the topics, but you can subscribe to them. To subscribe, you provide information about the resource you want to receive events from. As long as you have access to the resource, you can subscribe to its events.

Custom topics

Custom topics are application and third-party topics. When you create or are assigned access to a custom topic, you see that custom topic in your subscription.

What is an event subscription?

Event Subscriptions define which events on a topic an event handler wants to receive. A subscription can also filter events by their type or subject, so you can ensure an event handler only receives relevant events.

What is an event handler?

An event handler (sometimes referred to as an event "subscriber") is any component (application or resource) that can receive events from Event Grid. For example, Azure Functions can execute code in response to the new song being added to the Blob storage account. Subscribers can decide which events they want to handle and Event Grid will efficiently notify each interested subscriber when a new event is available - no polling required.

Should you use Event Grid?

Use Event Grid when you need these features:

- **Simplicity:** It is straightforward to connect sources to subscribers in Event Grid.
- **Advanced filtering:** Subscriptions have close control over the events they receive from a topic.
- **Fan-out:** You can subscribe to an unlimited number of endpoints to the same events and topics.

- **Reliability:** Event Grid retries event delivery for up to 24 hours for each subscription.
- **Pay-per-event:** Pay only for the number of events that you transmit.

Event Grid is a simple but versatile event distribution system. Use it to deliver discrete events to subscribers, which will receive those events reliably and quickly. We have one more messaging model to examine - what if we want to deliver a large *stream* of events? In this scenario, Event Grid isn't a great solution because it's designed for one-event-at-a-time delivery. Instead, we need to turn to another Azure service: Event Hubs.

What is Azure Event Hubs?

Event Hubs is an intermediary for the publish-subscribe communication pattern. Unlike Event Grid, however, it is optimized for extremely high throughput, a large number of publishers, security, and resiliency.

Whereas Event Grid fits perfectly into the publish-subscribe pattern in that it simply manages subscriptions and routes communications to those subscribers, Event Hubs performs quite a few additional services. These additional services make it look more like a service bus or message queue, than a simple event broadcaster.

Partitions

As Event Hubs receives communications, it divides them into partitions. Partitions are buffers into which the communications are saved. Because of the event buffers, events are not completely ephemeral, and an event isn't missed just because a subscriber is busy or even offline. The subscriber can always use the buffer to "catch up." By default, events stay in the buffer for 24 hours before they automatically expire.

The buffers are called partitions because the data is divided amongst them. Every event hub has at least two partitions, and each partition has a separate set of subscribers.

Capture

Event Hubs can send all your events immediately to Azure Data Lake or Azure Blob storage for inexpensive, permanent persistence.

Authentication

All publishers are authenticated and issued a token. This means Event Hubs can accept events from external devices and mobile apps, without worrying that fraudulent data from pranksters could ruin our analysis.

Using Event Hubs

Event Hubs has support for pipelining event streams to other Azure services. Using it with Azure Stream Analytics, for instance, allows complex analysis of data in near real time, with the ability to correlate multiple events and look for patterns. In this case, Stream Analytics would be considered a subscriber.

Which service should I choose?

Just like our queue choice, selecting between these two event delivery services can seem tricky at first. Both support *At Least Once* semantics.

Choose Event Hubs if:

- You need to support authenticating a large number of publishers.
- You need to save a stream of events to Data Lake or Blob storage.
- You need aggregation or analytics on your event stream.
- You need reliable messaging or resiliency.

Otherwise, if you need a simple event publish-subscribe infrastructure, with trusted publishers (for instance, your own web server), you should choose Event Grid.

Event Hubs lets you build a big data pipeline capable of processing millions of events per second with low latency. It can handle data from concurrent sources and route it to a variety of stream-processing infrastructures and analytics services. It enables real-time processing and supports repeated replay of stored raw data.

Communicate between applications with Azure Queue storage

Settings for queues

When you create a storage account that will contain queues, you should consider the following settings:

- Queues are only available as part of Azure general-purpose storage accounts (v1 or v2). You cannot add them to Blob storage accounts.
- The **Access tier** setting which is shown for StorageV2 accounts applies only to Blob storage and does not affect queues.
- You should choose a location that is close to either the source components or destination components or (preferably) both.
- Data is always replicated to multiple servers to guard against disk failures and other hardware problems. You have a choice of replication strategies: **Locally Redundant Storage (LRS)** is low-cost but vulnerable to disasters that affect an entire data center while **Geo-Redundant Storage (GRS)** replicates data to other Azure data centers. Choose the replication strategy that meets your redundancy needs.
- The performance tier determines how your messages are stored: **Standard** uses magnetic drives while **Premium** uses solid-state drives. Choose Standard if you expect peaks in demand to be short. Consider Premium if queue length sometimes becomes long and you need to minimize the time to access messages.
- Require secure transfer if sensitive information may pass through the queue. This setting ensures that all connections to the queue are encrypted using Secure Sockets Layer (SSL).

Enable reliable messaging for Big Data applications using Azure Event Hubs

What is an Azure Event Hub?

Azure Event Hubs is a cloud-based, event-processing service that can receive and process millions of events per second. Event Hubs acts as a front door for an event pipeline, to receive incoming data and stores this data until processing resources are available.

An entity that sends data to the Event Hubs is called a *publisher*, and an entity that reads data from the Event Hubs is called a *consumer* or a *subscriber*. Azure Event Hubs sits between these two entities to divide the production (from the publisher) and consumption (to a subscriber) of an event stream. This decoupling helps to manage scenarios where the rate of event production is much higher than the consumption. The following illustration shows the role of an Event Hub.

Publishers and subscribers

Event publishers are any app or device that can send out events using either HTTPS or Advanced Message Queuing Protocol (AMQP) 1.0.

For publishers that send data frequently, AMQP has better performance. However, it has a higher initial session overhead, because a persistent bidirectional socket and transport-level security (TLS) or SSL/TLS has to be set up first.

For more intermittent publishing, HTTPS is the better option. Though HTTPS requires additional overhead for each request, there isn't the session initialization overhead.

Pricing

There are three pricing tiers for Azure Event Hubs: Basic, Standard, and Dedicated. The tiers differ in terms of supported connections, the number of available Consumer groups, and throughput. When using Azure CLI to create an Event Hubs namespace, if you don't specify a pricing tier, the default of **Standard** (20 Consumer groups, 1000 Brokered connections) is assigned.

Configure a new Event Hub

After you create the Event Hubs namespace, you can create an Event Hub. When creating a new Event Hub, there are several mandatory parameters.

The following parameters are required to create an Event Hub:

- **Event Hub name** - Event Hub name that is unique within your subscription and:
    - Is between 1 and 50 characters long.
    - Contains only letters, numbers, periods, hyphens, and underscores.
    - Starts and ends with a letter or number.
- **Partition Count** - The number of partitions required in an Event Hub (between 2 and 32). The partition count should be directly related to the expected number of concurrent consumers and can't be changed after the hub has been created. The partition separates the message stream so that consumer or receiver apps only need to read a specific subset of the data stream. If not defined, this value defaults to *4*.
- **Message Retention** - The number of days (between 1 and 7) that messages will remain available if the data stream needs to be replayed for any reason. If not defined, this value defaults to *7*.

You can also optionally configure an Event Hub to stream data to an Azure Blob storage or Azure Data Lake Store account.

What are the minimum Event Hub application requirements?

To configure an application to send messages to an Event Hub, you must provide the following information, so that the application can create connection credentials:

- Event Hub namespace name
- Event Hub name
- Shared access policy name

- Primary shared access key

To configure an application to receive messages from an Event Hub, provide the following information, so that the application can create connection credentials:

- Event Hub namespace name
- Event Hub name
- Shared access policy name
- Primary shared access key
- Storage account name
- Storage account connection string
- Storage account container name

If you have a receiver application that stores messages in Azure Blob Storage, you'll also need to configure a storage account.

How can you test Event Hub resilience?

Azure Event Hubs keeps received messages from your sender application, even when the hub is unavailable. Messages received after the hub becomes unavailable are successfully transmitted to our application as soon as the hub becomes available.

To test this functionality, you can use the Azure portal to disable your Event Hub.

When you re-enable your Event Hub, you can rerun your receiver application, and use Event Hubs metrics for your namespace to check whether all sender messages are successfully transmitted and received.

Other useful metrics available in the Event Hubs include:

- Throttled Requests: The number of throttled requests because the throughput exceeded unit usage.
- ActiveConnections: The number of active connections on a namespace or Event Hub.
- Incoming/Outgoing Bytes: The number of bytes sent to/received from the Event Hubs service over a specified period.

Implement message-based communication workflows with Azure Service Bus

Service Bus topics, queues, and relays

Azure Service Bus can exchange messages in three different ways: queues, topics, and relays.

What is a queue?

A **queue** is a simple temporary storage location for messages. A sending component adds a message to the queue. A destination component picks up the message at the front of the queue. Under ordinary circumstances, each message is received by only one receiver.

Queues decouple the source and destination components to insulate destination components from high demand.

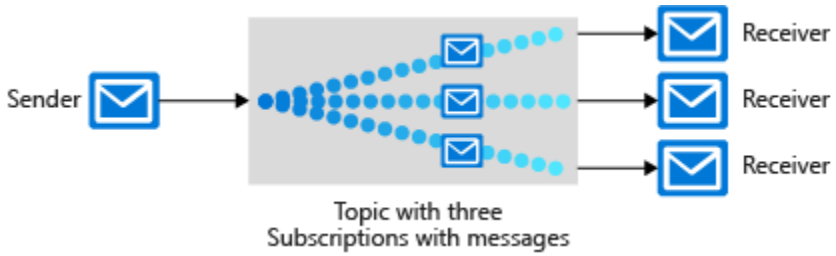
During peak times, messages may come in faster than destination components can handle them. Because source components have no direct connection to the destination, the source is unaffected and the queue will grow. Destination components will remove messages from the queue as they are able to handle them. When demand drops, destination components can catch up and the queue shortens.

A queue responds to high demand like this without needing to add resources to the system. However, for messages that need to be handled relatively quickly, adding additional instances of your destination component can allow them to share the load. Each message would be handled by only one instance. This is an effective way to scale your entire application while only adding resources to the components that actually need it.

What is a topic?

A **topic** is similar to a queue but can have multiple subscriptions. This means that multiple destination components can subscribe to a single topic, so each message is delivered to multiple receivers. Subscriptions can also filter the messages in the topic to receive only messages that are relevant. Subscriptions provide the same decoupled communications as queues and respond to high demand in the same way. Use a topic if you want each message to be delivered to more than one destination component.

Topics are not supported in the Basic pricing tier.



What is a relay?

A **relay** is an object that performs synchronous, two-way communication between applications. Unlike queues and topics, it is not a temporary storage location for messages. Instead, it provides bidirectional, unbuffered connections across network boundaries such as firewalls. Use a relay when you want direct communications between components as if they were located on the same network segment but separated by network security devices.

How to choose a communications technology

We've seen the different concepts and the implementations Azure provides. Let's discuss what our decision process should look like for each of our communications.

Consider the following questions:

1. Is the communication an event? If so, consider using Event Grid or Event Hubs.
2. Should a single message be delivered to more than one destination? If so, use a Service Bus topic. Otherwise, use a queue.

If you decide that you need a queue:

Choose Service Bus queues if:

- You need an at-most-once delivery guarantee
- You need a FIFO guarantee
- You need to group messages into transactions
- You want to receive messages without polling the queue
- You need to provide role-based access to the queues
- You need to handle messages larger than 64 KB but smaller than 256 KB
- Your queue size will not grow larger than 80 GB
- You would like to be able to publish and consume batches of messages

Choose queue storage if:

- You need a simple queue with no particular additional requirements
- You need an audit trail of all messages that pass through the queue
- You expect the queue to exceed 80 GB in size
- You want to track progress for processing a message inside of the queue

Although the components of a distributed application can communicate directly, you can often increase the reliability of that communication by using an intermediate communication platform such as Azure Service Bus or Azure Event Grid.

Event Grid is designed for events, which notify recipients only of an event and do not contain the raw data associated with that event. Azure Event Hubs is designed for high-flow analytics types of events. Azure Service Bus and storage queues are for messages, which can be used for binding the core pieces of any application workflow.

If your requirements are simple, if you want to send each message to only one destination, or if you want to write code as quickly as possible, a storage queue may be the best option. Otherwise, Service Bus queues provide many more options and flexibility.

If you want to send messages to multiple subscribers, use a Service Bus topic.

Introduction to Azure Kubernetes Service

Creating an AKS cluster

At its core, an AKS cluster is a cloud hosted Kubernetes cluster. Unlike a custom Kubernetes installation, AKS streamlines the installation process and takes care of most of the underlying cluster management tasks.

You have two options when you create an AKS cluster. You either use the Azure portal or Azure CLI. Both options require you to configure basic information about the cluster. For example:

- The Kubernetes cluster name
- The version of Kubernetes to install
- A DNS prefix to make the master node publicly accessible
- The initial node pool size

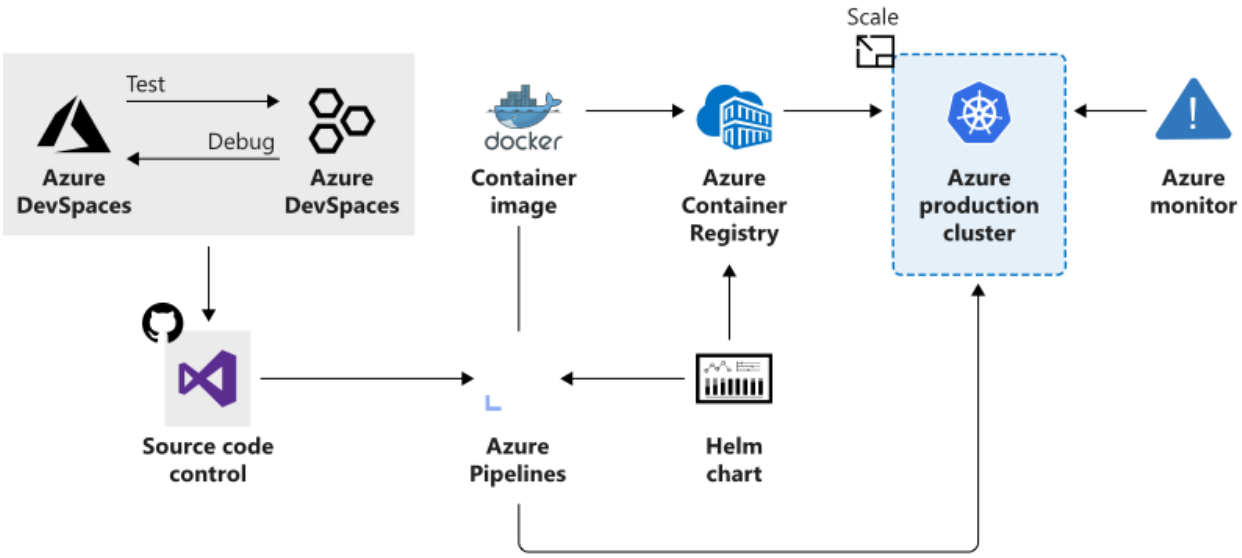
The initial node pool size defaults to two nodes, however it's recommended that at least three nodes are used for a production environment.

Note

The master node in your cluster is free. You only pay for node VMs, storage and networking resources consumed in your cluster.

Unless specified, the Azure service creation workflow creates a Kubernetes cluster using default configuration for scaling, authentication, networking and monitoring. Creating an AKS cluster typically takes a few minutes. Once complete, you can change any of the default AKS cluster properties. Access and management of your cluster is done through the Azure portal or from the command line.

How workloads are developed and deployed to AKS



AKS supports the Docker image format that means that you can use any development environment to create a workload, package the workload as a container and deploy the container as a Kubernetes pod.

Here you use the standard Kubernetes command-line tools or the Azure CLI to manage your deployments. The support for the standard Kubernetes tools ensures that you don't need to change your current workflow to support an existing Kubernetes migration to AKS.

AKS also supports all the popular development and management tools such as Helm, Draft, Kubernetes extension for Visual Studio Code and Visual Studio Kubernetes Tools.

Azure Dev Spaces

Setting up a local Kubernetes cluster on a developer machine can be complex and most solutions offers a single node configuration. It's also common to mock or replicate dependencies between developer teams when working on microservices projects.

Azure Dev Spaces helps your development teams be more productive on Kubernetes and allows you to:

- Minimize the local dev machine setup for each team member as developers can work directly in AKS
- Rapidly iterate and debug code directly in Kubernetes using Visual Studio or Visual Studio Code

- Generate Docker and Kubernetes configuration-as-code assets to use from development through to production
- Develop your code in isolation, and do integrated testing with other components without replicating or mocking up dependencies

Important

Azure Dev Spaces is supported only by AKS clusters in specific regions.

Deployment Center

Deployment center simplifies setting up a DevOps pipeline for your application. You can use this configured DevOps pipeline to set up a continuous integration (CI) and continuous delivery (CD) pipeline to your AKS Kubernetes cluster.

With Azure DevOps Projects you can:

- Automatically create Azure resources, such as an AKS cluster
- Create an Azure Application Insights resource for monitoring an AKS cluster
- Enable Azure Monitor for containers to monitor performance for the container workloads on an AKS cluster

You can add richer DevOps capabilities by extending the default configured DevOps pipeline. For example, you can add approvals before deploying, provision additional Azure resources, run scripts or upgrade workloads.

Azure Service Integration

AKS allows us to integrate any Azure service offering and use it as part of an AKS cluster solution.

For example, remember that Kubernetes doesn't provide middleware and storage systems. Suppose you need to add a processing queue to the fleet management data processing service. You can easily integrate Storage queues using Azure Storage to extend the capacity of the data processing service.

Publish and manage your APIs with Azure API Management

Why use API Management?

For developers, API Management provides a range of benefits.

- **API documentation**. Documentation of APIs enables calling clients to quickly integrate their solutions. API Management allows you to quickly expose the structure of your API to calling clients through modern standards like Open API. You can have more than one version of an API. With multiple versions, you can stage app updates as your consuming apps don't have to use the new version straight away.
- **Rate limiting access**. If your API could potentially access a large amount of data, it's a good idea to limit the rate at which clients can request data. Rate limiting helps maintain optimal response times for every client. API Management let you set rate limits as a whole or for specific individual clients.
- **Health monitoring**. APIs are consumed by remote clients. So it can be difficult to identify potential problems or errors. API Management lets you view error responses and log files, and filter by types of responses.
- **Modern formats like JSON**. APIs have used many different data exchange formats over the years from XML to CSV and many more. API Management enables you to expose these formats using modern data models like JSON.
- **Connections to any API**. In many businesses, APIs are located across different countries and use different formats. API Management lets you add all of these disparate APIs into single modern interface.
- **Analytics**. As you develop your APIs, it's useful to see how often your APIs are being called and by which types of systems. API Management allows you to visualize this data within the Azure portal.
- **Security**. Security is paramount when dealing with system data. Unauthorized breaches can cost companies money, time lost in reworking code, and reputational loss. Security tools that you can use with Azure API management include OAuth 2.0 user authorization, and integration with Azure Active Directory.

Pricing tiers

When you create an Azure API management gateway, you must select from one of several pricing tiers:

- **Developer**. You use the Developer tier for evaluating the API management service. You shouldn't use this tier for production deployments.

- **Basic**. Entry level production use. 99.9% SLA. 1000 requests/sec. Two scale units.

Note

A scale unit enables you to scale up a service. The more scale units you have, the more you can scale up the service.

- **Standard**. Medium level production use. 99.9% SLA. 2500 requests/sec. Four scale units.
- **Premium**. Multi region deployment. High volume use. 99.95% SLA. 4000 requests/sec. 10 scale units per region.
- **Consumption**. The serverless consumption tier plan lets you pay for what you use, rather than having dedicated resources. You can quickly set up ad-hoc testing, and you can scale up API access when demand increases. The consumption tier has built-in high availability and autoscaling. Because it is serverless, you can provision a consumption tier gateway in a lot less time than the other server-based tiers.

Protect your Azure infrastructure with Azure Site Recovery

Test failover of individual machines

A test failover enables you to simulate a disaster and see its effects. Failover tests can be started from the Site Recovery dashboard, or directly from the disaster recovery menu on a specific VM. You'll start by choosing a recovery point. You can choose from either the last processed, the latest app-consistent point, or a custom recovery point.

What is failback?

Failback is the reverse of a failover. It's where a completed failover to a secondary region has been committed, and is now the production environment. Reprotection has completed for the failed-over environment, and the source environment is now its replica. In a failback scenario, Site Recovery will fail over back to the source VMs.

The process to complete a failback is the same as a failover, even down to reusing the recovery plan. Selecting failover on your recovery plan has **from** set to the target region, and the **to** set to the source region.

Manage failovers

Site Recovery can run failovers on demand. Test failovers are isolated, meaning they don't impact production services. This flexibility enables you to run a failover without interrupting the users of that system. The flexibility works the other way too, allowing failback on-demand either as part of a planned test or as part of a fully-invoked DR process.

The recovery plans in Site Recovery also allow for customizing and sequencing failover and failback. The plans enable you to group machines and workloads.

Flexibility can also apply to how you trigger the failover process. Manual failovers are easy to do via the Azure portal. PowerShell scripting or using runbooks in Azure Automation also provide automation options.

Fix issues with a failover

Even though Site Recovery is automated, errors can still happen. The following list shows the three most common issues observed. For a full list of issues and how to troubleshoot them, see the link in the **Summary** unit.

Azure resource quota issues

Site Recovery must create resources in different regions. If our subscription isn't able to do this, the replication fails. This error also occurs if our subscription doesn't have the right quota limits to create VMs that match the size of the source VMs.

You can correct this by contacting Azure billing support, and requesting that they create the correct size VMs in the needed target region.

One or more disk(s) are available for protection

This error occurs if you've finished setting up Site Recovery for your VMs. Subsequently, you've added or initialized, additional disks.

To fix this error, you can add replication for the newly-added disks, or you can choose to ignore the disk warning.

Trusted root certificates

Check that the latest root certificates are installed to allow Site Recovery to communicate and authenticate VMs for replication securely. You can see this error if your VMs don't have the latest updates applied. Before Site Recovery can enable replication, you must update both Windows and Linux VMs.

The correction is different for each operating system. Windows is as simple as ensuring automatic Windows update is switched on, and updates are applied. For each Linux distribution, you'll need to follow the guidance provided by the distributor.

Protect your virtual machines by using Azure Backup

Azure Backup versus Azure Site Recovery

Both Backup and Site Recovery aim to make the system more resilient to faults and failures. However, while the primary goal of backup is to maintain copies of stateful data that allow you to go back in time, site-recovery replicates the data in almost real time and allows for a failover.

In that sense, if there are issues like network or power outages, you can use availability zones. For a region-wide disaster (such as natural disasters), Site Recovery is used. Backups are used in cases of accidental data loss, data corruption, or ransomware attacks.

Additionally, the choice of a recovery approach depends on the criticality of the application, RPO and RTO requirements, and the cost implications.

Why use Azure Backup?

Traditional backup solutions, such as disk and tape, don't offer the highest level of integration with cloud-based solutions. Azure Backup has several benefits over more traditional backup solutions:

Zero-infrastructure backup: Azure Backup eliminates the need to deploy and manage any backup infrastructure or storage. This means there's no overhead in maintaining backup servers or scaling the storage up or down as the needs vary.

Long-term retention: Meet rigorous compliance and audit needs by retaining backups for many years, beyond which the recovery points will be pruned automatically by the built-in lifecycle management capability.

Security: Azure Backup provides security to your backup environment - both when your data is in transit and at rest.

- **Azure role-based access control**: RBAC allows you to segregate duties within your team and grant only the amount of access to users necessary to do their jobs.
- **Encryption of backups**: Backup data is automatically encrypted using Microsoft-managed keys. Alternatively, you can also encrypt your backed-up data using customer-managed keys stored in the Azure Key Vault.

- **No internet connectivity required**: When using Azure VMs, all the data transfer happens only on the Azure backbone network without needing to access your virtual network. So no access to any IPs or FQDNs is required.
- **Soft delete**: With soft delete, the backup data is retained for 14 additional days even after the deletion of the backup item. This protects against accidental deletion or malicious deletion scenarios, allowing the recovery of those backups with no data loss.

Azure Backup also offers the ability to back up virtual machines encrypted with Azure Disk Encryption.

High availability: Azure Backup offers three types of replication - LRS, GRS, and RA-GRS (to enable customer-controlled cross region restore) to keep your backup data highly available.

Centralized monitoring and management: Azure Backup provides built-in monitoring and alerting capabilities in a Recovery Services vault. These capabilities are available without any additional management infrastructure.

Azure Backup supported scenarios

- **Azure VMs** - Back up Windows or Linux Azure virtual machines
  Azure Backup provides independent and isolated backups to guard against unintended destruction of the data on your VMs. Backups are stored in a Recovery Services vault with built-in management of recovery points. Configuration and scaling are simple, backups are optimized, and you can easily restore as needed.
- **On-premises** - Back up files, folders, and system state using the Microsoft Azure Recovery Services (MARS) agent. Or use Microsoft Azure Backup Server (MABS) or Data Protection Manager (DPM) server to protect on-premises VMs (Hyper-V and VMWare) and other on-premises workloads.
- **Azure Files shares** - Azure Files - Snapshot management by Azure Backup
- **SQL Server in Azure VMs** and **SAP HANA databases in Azure VMs** - Azure Backup offers stream-based, specialized solutions to back up SQL Server or SAP HANA running in Azure VMs. These solutions take workload-aware backups that support different backup types such as full, differential and log, 15-minute RPO, and point-in-time recovery.

General Notes

- ExpressRoute Premium is required if the on-premises data centers were in different geopolitical regions.
- You can not use Event Hub for streaming data from IoT devices to Azure Stream Analytics.
- Recovery Service Vaults should be in the same region as the data source
- SSO with PTA requires you to install a pass-through authentication agent on an on-premises server.
- An Azure Log Analytics workspace is required to enable sending Azure AD activity logs to Azure Monitor
- AAD Connect needs a full GUI to be installed and at least Windows Server 2012.
- When unassigning an existing blueprint, all blueprint resource locking is removed
- Azure Data Factory to transfer data between Data Lake Storage Gen1 and Gen2. AzCopy is used to copy blobs and files from or to Azure storage accounts.
- Use B-series VMs for low-cost and burstable workloads.
- Basically, all data solutions have a default backup retention of seven days. Standard and Premium DBs can be configured for a retention period of up to 35 days without having to configure long-term-retention (LTR). LTR of up to 10 years can be configured for all Azure DBs.
- You must install Azure Migrate appliance only in one on-premises location, even if you have several.
- Each Key Vault and the VMs it supports for Disk Encryption must be co-located in the same region.
- Business Critical DBs support up to 4 TB but do not provide a read-write replica of the DB, it supports read-only replicates that are made read-write after a failure of the primary database.
- Hyperscale DBs support up to 100 TB and support one read-write replica and up to four read-only replicas.
- Automatic data replication via geo-redundant storage (GRS) guarantees that the data remains within the same geo-political boundaries.
- Azure Event Hubs are designed to support telemetry and distributed data streaming and are the best fit to a real-time streaming solution. Event Grid is designed to react to status changes.
- An Azure Log Analytics workspace is required to enable sending AAD activity logs to Azure Monitor.
- General-purpose V2 and Blob Storage are the only storage accounts that support hot, cool, and archive storage tiers.
- VM scale sets support tup to 600 images for a custom VM image and up to 1000 VM images when based on a platform instance.

- You can not back up secrets from a Key Vault in one geography and restore them to a Key Vault in a different geography.
- Basic elastic pools have a limit of 1600 DTUs, and Standard elastic pools have a limit of 3000 DTUs.
- Archive storage tier does not support RA-GRS reads.
- Linux-based Azure Fabric clusters do not support Application Insights.
- NVMe storage is only available in the LV2-series
- Azure SQL managed instance only supports vCore pricing, not DTU-based pricing.
- Web apps can only scale to 50 instances of a web server.
- Azure migrate can migrate servers on premises Hyper-V VMs, physical servers, and public cloud VMs to Azure VMs, web apps to Azure and SQL Server databases
- Azure Storage Explorer is a free management tool that lets you manage your Azure Cloud data resources. Azure Data Explorer is a managed data analytics service that lets you perform real-time analysis on large data stream.
- F-series is optimized for batch processing, analytics and gaming. It can optimally run a very compute intensive workload.
- The E-series is intended for heavy in-memory operations.
- Azure AD Connect Health is a premium feature and requires AAD P1 or P2.
- An availability set can have a maximum of three fault domains (default is two).
- Azure Managed instance in Azure SQL Database allows you to host multiple databases as separate instances or on a single instance. It is useful when you want near 100% compatibility with your on-premises servers.
- Azure Data Lake can load data directly into the Azure SQL Data Warehouse.
- Azure Analysis Services is a solution that provides enterprise-grade data models in the cloud.
- Managed Identities can be assigned directly to applications.
- Key vault contents are replicated within the region and to a second region in the same geography.
- Managed identities can be used to control an application's access Keys Vault secrets.
- The backup of a key from a Key Vault can be restored to a Key Vault in any region in the same geography, as long as both regions are also in the same subscription.
- Use advanced access policy for Key Vault to enable the Key Vault to support template deployments.
- Premium DTU-based tier supports in-memory OLTP and columnstore indexes.
- Azure Data Flow can help to develop graphical data transformation logic without writing code.
- You can bring your own key to encrypt Azure Blob storage and Azure Files.

- Only Azure Key Vault Premium Supports HSM-protected keys.
- Azure SQL Database and Azure SQL Managed Instance automatic tuning provides peak performance and stable workloads through continuous performance tuning based on AI and machine learning.
  Automatic tuning is a fully managed intelligent performance service that uses built-in intelligence to continuously monitor queries executed on a database, and it automatically improves their performance.
- The contents of your key vault are replicated within the region and to a secondary region at least 150 miles away but within the same geography. This maintains high durability of your keys and secrets.
- Premium storage is currently not supported for audit logs.
- Deploying Azure API Management in a virtual network is only available in the Premium and Developer tiers of API Management.
- Named network locations for conditional access may include locations like an organization's headquarters network ranges, VPN network ranges, or ranges that you wish to block. Named locations can be defined by IPv4/IPv6 address ranges or by countries.
- When a blueprint is first created, it's considered to be in Draft mode. When it's ready to be assigned, it needs to be Published.
- When Azure Policy runs the template in the deployIfNotExists policy definition, it does so using a managed identity.
- Azure Active Directory Seamless Single Sign-On (Azure AD Seamless SSO) automatically signs users in when they are on their corporate devices connected to your corporate network (with password hash or PTA).
- Log Analytics or Application Insights send signal type as Log.
- DB Server and elastic pool must be in the same region
- SQL Managed Instance allows existing SQL Server customers to lift and shift their on-premises applications to the cloud with minimal application and database changes.
- VMSS only support 1 region.
- Azure Front Door supports rate-limiting.
- Traffic Manager also provides your public endpoints with high availability and quick responsiveness. It does not provide rate limiting.
- ACI has networking controls (public, private) and supports Restart policies set to "OnFailure" by default.
- Deterministic encryption always generates the same encrypted value for any given plain text value.
- Always Encrypted can be used to prevent cloud admins and db_owners from viewing protected data.
- PTA does not support smart card authentication.

- Federation with ADFS does support smart card authentication.
- Azure Front Door can use latency-based traffic routing.
- In Cosmos DB SQL API, you can query data using Select but you can not use Create to create a container.
- Cassandra uses a SQL-like query language named Cassandra Query Language (CQL).
- Premium storage supports only LRS as storage redundancy.
- PTA uses the same security and password policies as your on-prem AD.
- AAD roles and Azure resource roles can be reviewed using AAD PIM.
- DSC needs an Automation Account configured to run the script.
- Use az container create to deploy an image to ACI.
- Conditional Access needs P1.
- You can use per-user MFA for all AADs.
- Azure Disk Encryption is not supported for the Basic tier.
- You must encrypt the boot volume before you can encrypt any data volumes on a Windows VM. Azure Disk Encryption does not let you encrypt a data volume unless you first encrypt the OS volume. This is different for Linux VMs, which let you encrypt data without first encrypting the OS volume.
- You can not use an on-premises key management service to safeguard encryption keys. You are required to use Azure Key Vault.
- Azure Bastion is a service that you can provision in your VNet to provide RDP and SSH connectivity to your Azure VMs without needing a public IP.
- The user risk policy will analyse user sign-in and learn typical user behavior.
- You can use RBAC to access Azure Blobs and Queues.
- You can not use AAD MIs to access Azure Tables.
- Azure Service Health alerts you about service issues in an Azure region.
- You need two Cosmos DB accounts if one DB needs multi-region writes and the other one only single-region writes.
- Use a proximity placement group to provision resources like Azure VMs or VMSS that are physically located close to each other.
- Deploy one Azure Migrate appliance for each cluster in your environment.
- Provision one Microsoft Azure Recovery Service (MARS) agent per node.
- To have Azure verify your custom domain, Add a TXT record to your company's DNS servers.
- App Service Environment (ASE) is an isolated environment for App Services.
- You can use AzCopy to move data from Table storage to Cosmos DB.

- Use the Telegraf agent to collect performance counter data from Linux VMs and send it to Azure Monitor Metrics.
- Use the Premium plan to run your function on warm instances and avoid cold starts.
- You can assign user assigned managed identities with Azure VMs form different regions, resource groups, and subscriptions, as long these subscriptions are in the same AAD tenant.
- You can not migrate VMs that have BitLocker enabled.
- The OS disk size is limited to 2 TB and the data disk size is limited to 4 TB when you want to migrate from a Hyper-V cluster.
- When migrating from VMware the OS disk size is limited to 2 TB and data disk size is limited to 32 TB.
- You need to install SQL Server 2019 for the AAD Connect database when you have more than 100K objects on-premises. For a smaller number of objects, you can use the default database installation, which is LocalDb.
- It is not possible to update the partition key in an existing container in Azure Cosmos DB.
- Azure Load Balancer needs two rules if you have HTTP and HTTPS traffic.
- The Backup and Restore feature requires the App Service plan to be in the Standard, Premium or Isolated tier.
- The secondary SQL Database cannot be in the same region as the primary.
- The SMB protocol requires TCP port 445 to be open.
- On scale out, autoscale runs if any rule is met. On scale-in, autoscale requires all rules to be met.