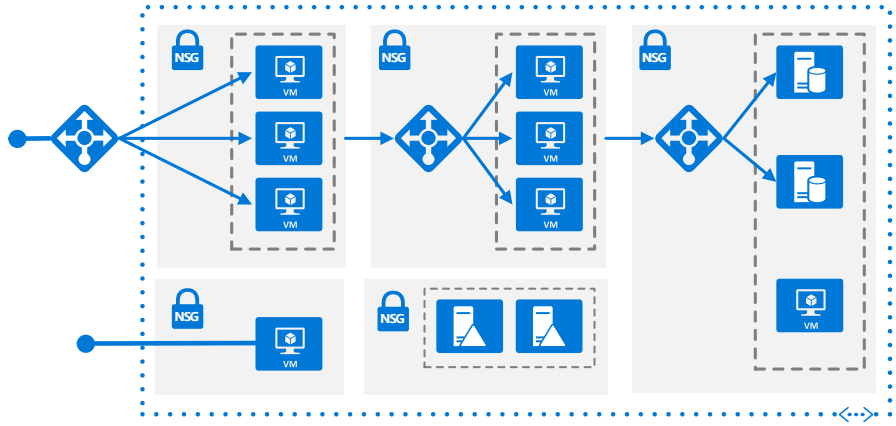
Table of Contents

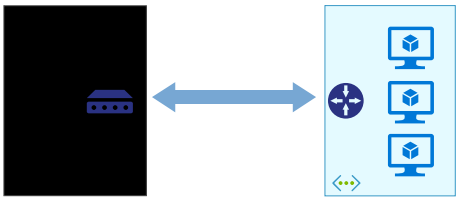
Our reference architectures are arranged by scenario, with related architectures grouped together. Each architecture includes recommended practices, along with considerations for scalability, availability, manageability, and security. Most also include a deployable solution.



N-tier application

Deploy an N-tier application to Azure, for Windows or Linux.

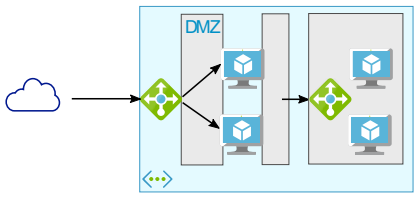
Configurations are shown for SQL Server and Apache Cassandra. For high availability, deploy an active-passive configuration in two regions.



Hybrid network

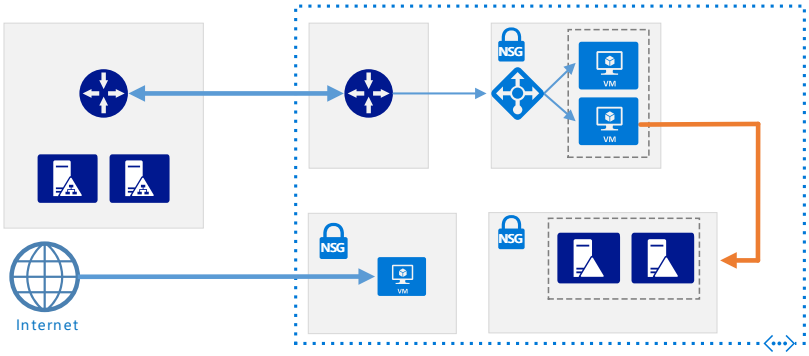
This series shows options for creating a network connection between an on-premises network and Azure.

Configurations include site-to-site VPN or Azure ExpressRoute for a private, dedicated connection.



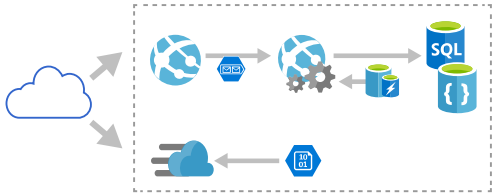
Network DMZ

This series shows how to create a network DMZ to protect the boundary between an Azure virtual network and an on-premises network or the Internet.



Identity management

This series show options for integrating your on-premises Active Directory (AD) environment with an Azure network.



App Service web application

This series shows best practices for web applications that use Azure App Service.

Jenkins build server

Deploy and operate a scalable, enterprise-grade Jenkins server on Azure.

SharePoint Server 2016 farm

Deploy and run a high availability SharePoint Server 2016 farm on Azure with SQL Server Always On Availability Groups.

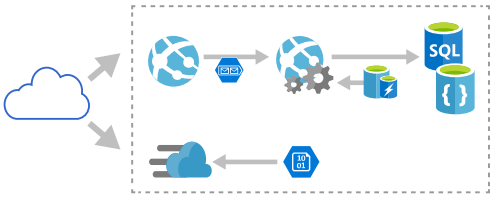
SAP NetWeaver and SAP HANA

Deploy and run SAP NetWeaver and SAP HANA in a high availability environment on Azure.

These reference architectures show proven practices for web applications that use Azure App Service and other managed service in Azure.

Basic web application

A basic web application that uses Azure App Service and Azure SQL Database.



Improved scalability

Improve scalability and performance by adding cache, CDN, and WebJobs for background tasks.

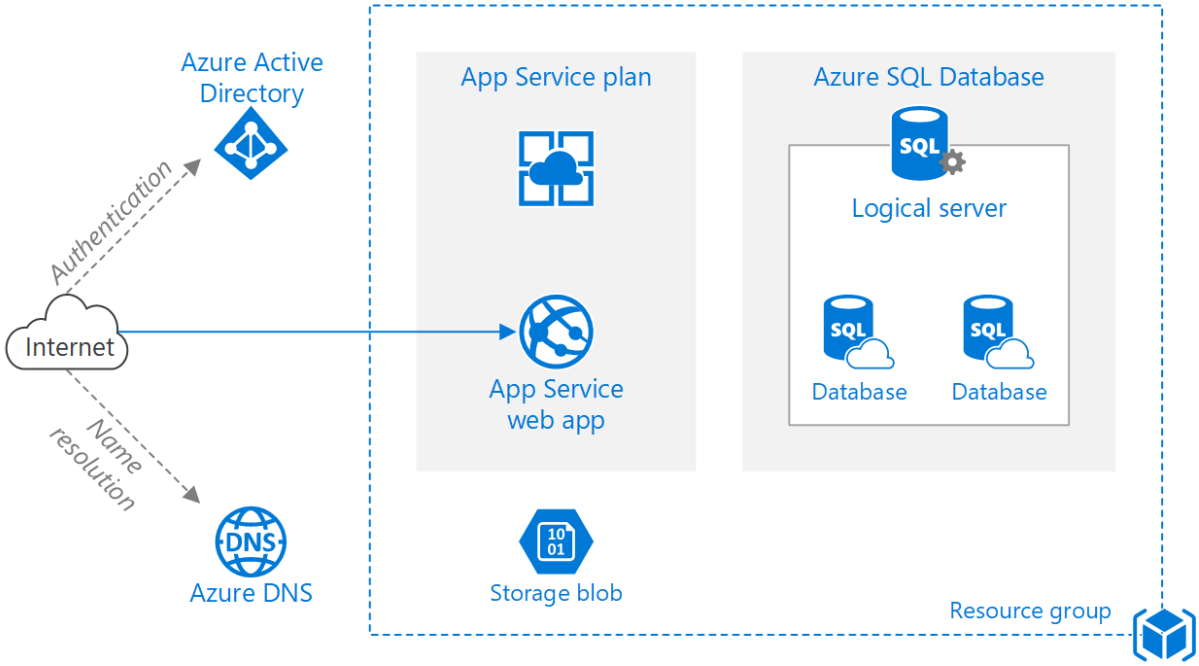
Run in multiple regions

Run a web application in multiple regions to achieve high availability.

Basic web application

4/11/2018 • 11 min to read • Edit Online

This reference architecture shows a set of proven practices for a web application that uses Azure App Service and Azure SQL Database. **Deploy this solution.**



Download a Visio file of this architecture.

Architecture

> **NOTE**
> This architecture does not focus on application development, and does not assume any particular application framework. The goal is to understand how various Azure services fit together.

The architecture has the following components:

- **Resource group**. A resource group is a logical container for Azure resources.
- **App Service app**. Azure App Service is a fully managed platform for creating and deploying cloud applications.
- **App Service plan**. An App Service plan provides the managed virtual machines (VMs) that host your app. All apps associated with a plan run on the same VM instances.
- **Deployment slots**. A deployment slot lets you stage a deployment and then swap it with the production deployment. That way, you avoid deploying directly into production. See the Manageability section for specific recommendations.
- **IP address**. The App Service app has a public IP address and a domain name. The domain name is a subdomain of `azurewebsites.net`, such as `contoso.azurewebsites.net`.
- **Azure DNS**. Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same

credentials, APIs, tools, and billing as your other Azure services. To use a custom domain name (such as `contoso.com`) create DNS records that map the custom domain name to the IP address. For more information, see Configure a custom domain name in Azure App Service.

- **Azure SQL Database**. SQL Database is a relational database-as-a-service in the cloud. SQL Database shares its code base with the Microsoft SQL Server database engine. Depending on your application requirements, you can also use Azure Database for MySQL or Azure Database for PostgreSQL. These are fully managed database services, based on the open source MySQL Server and Postgres database engines, respectively.
- **Logical server**. In Azure SQL Database, a logical server hosts your databases. You can create multiple databases per logical server.
- **Azure Storage**. Create an Azure storage account with a blob container to store diagnostic logs.
- **Azure Active Directory** (Azure AD). Use Azure AD or another identity provider for authentication.

Recommendations

Your requirements might differ from the architecture described here. Use the recommendations in this section as a starting point.

App Service plan

Use the Standard or Premium tiers, because they support scale out, autoscale, and secure sockets layer (SSL). Each tier supports several *instance sizes* that differ by number of cores and memory. You can change the tier or instance size after you create a plan. For more information about App Service plans, see App Service Pricing.

You are charged for the instances in the App Service plan, even if the app is stopped. Make sure to delete plans that you aren't using (for example, test deployments).

SQL Database

Use the V12 version of SQL Database. SQL Database supports Basic, Standard, and Premium service tiers, with multiple performance levels within each tier measured in Database Transaction Units (DTUs). Perform capacity planning and choose a tier and performance level that meets your requirements.

Region

Provision the App Service plan and the SQL Database in the same region to minimize network latency. Generally, choose the region closest to your users.

The resource group also has a region, which specifies where deployment metadata is stored. Put the resource group and its resources in the same region. This can improve availability during deployment.

Scalability considerations

A major benefit of Azure App Service is the ability to scale your application based on load. Here are some considerations to keep in mind when planning to scale your application.

Scaling the App Service app

There are two ways to scale an App Service app:

- *Scale up*, which means changing the instance size. The instance size determines the memory, number of cores, and storage on each VM instance. You can scale up manually by changing the instance size or the plan tier.
- *Scale out*, which means adding instances to handle increased load. Each pricing tier has a maximum number of instances.

You can scale out manually by changing the instance count, or use autoscaling to have Azure automatically add or remove instances based on a schedule and/or performance metrics. Each scale operation happens quickly—typically within seconds.

To enable autoscaling, create an autoscale *profile* that defines the minimum and maximum number of instances. Profiles can be scheduled. For example, you might create separate profiles for weekdays and weekends. Optionally, a profile contains rules for when to add or remove instances. (Example: Add two instances if CPU usage is above 70% for 5 minutes.)

Recommendations for scaling a web app:

- As much as possible, avoid scaling up and down, because it may trigger an application restart. Instead, select a tier and size that meet your performance requirements under typical load and then scale out the instances to handle changes in traffic volume.
- Enable autoscaling. If your application has a predictable, regular workload, create profiles to schedule the instance counts ahead of time. If the workload is not predictable, use rule-based autoscaling to react to changes in load as they occur. You can combine both approaches.
- CPU usage is generally a good metric for autoscale rules. However, you should load test your application, identify potential bottlenecks, and base your autoscale rules on that data.
- Autoscale rules include a *cool-down* period, which is the interval to wait after a scale action has completed before starting a new scale action. The cool-down period lets the system stabilize before scaling again. Set a shorter cool-down period for adding instances, and a longer cool-down period for removing instances. For example, set 5 minutes to add an instance, but 60 minutes to remove an instance. It's better to add new instances quickly under heavy load to handle the additional traffic, and then gradually scale back.

Scaling SQL Database

If you need a higher service tier or performance level for SQL Database, you can scale up individual databases with no application downtime. For more information, see SQL Database options and performance: Understand what's available in each service tier.

Availability considerations

At the time of writing, the service level agreement (SLA) for App Service is 99.95% and the SLA for SQL Database is 99.99% for Basic, Standard, and Premium tiers.

> **NOTE**
> The App Service SLA applies to both single and multiple instances.

Backups

In the event of data loss, SQL Database provides point-in-time restore and geo-restore. These features are available in all tiers and are automatically enabled. You don't need to schedule or manage the backups.

- Use point-in-time restore to recover from human error by returning the database to an earlier point in time.
- Use geo-restore to recover from a service outage by restoring a database from a geo-redundant backup.

For more information, see Cloud business continuity and database disaster recovery with SQL Database.

App Service provides a backup and restore feature for your application files. However, be aware that the backed-up files include app settings in plain text and these may include secrets, such as connection strings. Avoid using the App Service backup feature to back up your SQL databases because it exports the database to a SQL .bacpac file, consuming DTUs. Instead, use SQL Database point-in-time restore described above.

Manageability considerations

Create separate resource groups for production, development, and test environments. This makes it easier to manage deployments, delete test deployments, and assign access rights.

When assigning resources to resource groups, consider the following:

- Lifecycle. In general, put resources with the same lifecycle into the same resource group.
- Access. You can use role-based access control (RBAC) to apply access policies to the resources in a group.
- Billing. You can view the rolled-up costs for the resource group.

For more information, see Azure Resource Manager overview.

Deployment

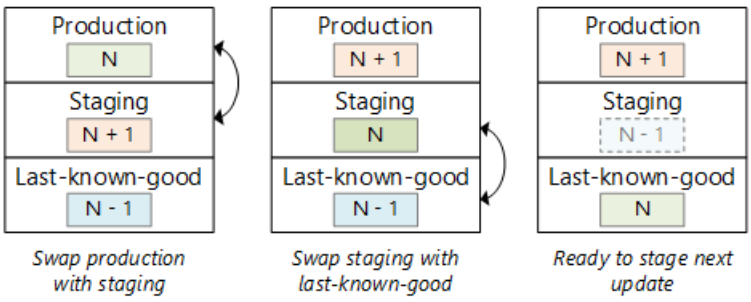
Deployment involves two steps:

1. Provisioning the Azure resources. We recommend that you use Azure Resoure Manager templates for this step. Templates make it easier to automate deployments via PowerShell or the Azure command line interface (CLI).
2. Deploying the application (code, binaries, and content files). You have several options, including deploying from a local Git repository, using Visual Studio, or continuous deployment from cloud-based source control. See Deploy your app to Azure App Service.

An App Service app always has one deployment slot named `production`, which represents the live production site. We recommend creating a staging slot for deploying updates. The benefits of using a staging slot include:

- You can verify the deployment succeeded, before swapping it into production.
- Deploying to a staging slot ensures that all instances are warmed up before being swapped into production. Many applications have a significant warmup and cold-start time.

We also recommend creating a third slot to hold the last-known-good deployment. After you swap staging and production, move the previous production deployment (which is now in staging) into the last-known-good slot. That way, if you discover a problem later, you can quickly revert to the last-known-good version.



If you revert to a previous version, make sure any database schema changes are backward compatible.

Don't use slots on your production deployment for testing because all apps within the same App Service plan share the same VM instances. For example, load tests might degrade the live production site. Instead, create separate App Service plans for production and test. By putting test deployments into a separate plan, you isolate them from the production version.

Configuration

Store configuration settings as app settings. Define the app settings in your Resource Manager templates, or using PowerShell. At runtime, app settings are available to the application as environment variables.

Never check passwords, access keys, or connection strings into source control. Instead, pass these as parameters to a deployment script that stores these values as app settings.

When you swap a deployment slot, the app settings are swapped by default. If you need different settings for production and staging, you can create app settings that stick to a slot and don't get swapped.

Diagnostics and monitoring

Enable diagnostics logging, including application logging and web server logging. Configure logging to use Blob storage. For performance reasons, create a separate storage account for diagnostic logs. Don't use the same storage account for logs and application data. For more detailed guidance on logging, see Monitoring and diagnostics guidance.

Use a service such as New Relic or Application Insights to monitor application performance and behavior under load. Be aware of the data rate limits for Application Insights.

Perform load testing, using a tool such as Visual Studio Team Services. For a general overview of performance analysis in cloud applications, see Performance Analysis Primer.

Tips for troubleshooting your application:

- Use the troubleshoot blade in the Azure portal to find solutions to common problems.
- Enable log streaming to see logging information in near-real time.
- The Kudu dashboard has several tools for monitoring and debugging your application. For more information, see Azure Websites online tools you should know about (blog post). You can reach the Kudu dashboard from the Azure portal. Open the blade for your app and click **Tools**, then click **Kudu**.
- If you use Visual Studio, see the article Troubleshoot a web app in Azure App Service using Visual Studio for debugging and troubleshooting tips.

Security considerations

This section lists security considerations that are specific to the Azure services described in this article. It's not a complete list of security best practices. For some additional security considerations, see Secure an app in Azure App Service.

SQL Database auditing

Auditing can help you maintain regulatory compliance and get insight into discrepancies and irregularities that could indicate business concerns or suspected security violations. See Get started with SQL database auditing.

Deployment slots

Each deployment slot has a public IP address. Secure the nonproduction slots using Azure Active Directory login so that only members of your development and DevOps teams can reach those endpoints.

Logging

Logs should never record users' passwords or other information that might be used to commit identity fraud. Scrub those details from the data before storing it.

SSL

An App Service app includes an SSL endpoint on a subdomain of `azurewebsites.net` at no additional cost. The SSL endpoint includes a wildcard certificate for the `*.azurewebsites.net` domain. If you use a custom domain name, you must provide a certificate that matches the custom domain. The simplest approach is to buy a certificate directly through the Azure portal. You can also import certificates from other certificate authorities. For more information, see Buy and Configure an SSL Certificate for your Azure App Service.

As a security best practice, your app should enforce HTTPS by redirecting HTTP requests. You can implement this inside your application or use a URL rewrite rule as described in Enable HTTPS for an app in Azure App Service.

Authentication

We recommend authenticating through an identity provider (IDP), such as Azure AD, Facebook, Google, or Twitter. Use OAuth 2 or OpenID Connect (OIDC) for the authentication flow. Azure AD provides functionality to manage users and groups, create application roles, integrate your on-premises identities, and consume backend services such as Office 365 and Skype for Business.

Avoid having the application manage user logins and credentials directly, as it creates a potential attack surface. At

a minimum, you would need to have email confirmation, password recovery, and multi-factor authentication; validate password strength; and store password hashes securely. The large identity providers handle all of those things for you, and are constantly monitoring and improving their security practices.

Consider using App Service authentication to implement the OAuth/OIDC authentication flow. The benefits of App Service authentication include:

- Easy to configure.
- No code is required for simple authentication scenarios.
- Supports delegated authorization using OAuth access tokens to consume resources on behalf of the user.
- Provides a built-in token cache.

Some limitations of App Service authentication:

- Limited customization options.
- Delegated authorization is restricted to one backend resource per login session.
- If you use more than one IDP, there is no built-in mechanism for home realm discovery.
- For multi-tenant scenarios, the application must implement the logic to validate the token issuer.

Deploy the solution

An example Resoure Manager template for this architecture is available on GitHub.

To deploy the template using PowerShell, run the following commands:

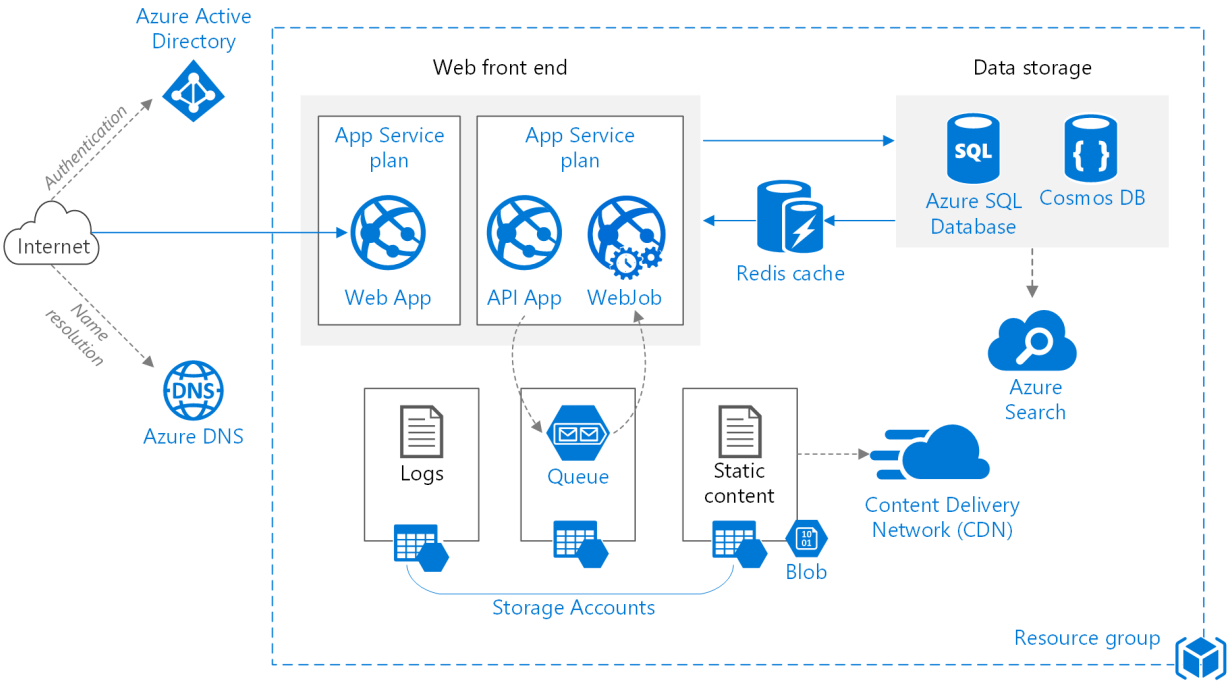
```
New-AzureRmResourceGroup -Name <resource-group-name> -Location "West US"

$parameters = @{"appName"="<app-name>";"environment"="dev";"locationShort"="uw";"databaseName"="app-
db";"administratorLogin"="<admin>";"administratorLoginPassword"="<password>"}

New-AzureRmResourceGroupDeployment -Name <deployment-name> -ResourceGroupName <resource-group-name> -
TemplateFile .\PaaS-Basic.json -TemplateParameterObject  $parameters
```

For more information, see Deploy resources with Azure Resource Manager templates.

Improve scalability in a web application

4/11/2018 • 6 min to read • Edit Online

This reference architecture shows proven practices for improving scalability and performance in an Azure App Service web application.



Download a Visio file of this architecture.

Architecture

This architecture builds on the one shown in Basic web application. It includes the following components:

- **Resource group**. A resource group is a logical container for Azure resources.
- **Web app** and **API app**. A typical modern application might include both a website and one or more RESTful web APIs. A web API might be consumed by browser clients through AJAX, by native client applications, or by server-side applications. For considerations on designing web APIs, see API design guidance.
- **WebJob**. Use Azure WebJobs to run long-running tasks in the background. WebJobs can run on a schedule, continously, or in response to a trigger, such as putting a message on a queue. A WebJob runs as a background process in the context of an App Service app.
- **Queue**. In the architecture shown here, the application queues background tasks by putting a message onto an Azure Queue storage queue. The message triggers a function in the WebJob. Alternatively, you can use Service Bus queues. For a comparison, see Azure Queues and Service Bus queues - compared and contrasted.
- **Cache**. Store semi-static data in Azure Redis Cache.
- **CDN**. Use Azure Content Delivery Network (CDN) to cache publicly available content for lower latency and faster delivery of content.
- **Data storage**. Use Azure SQL Database for relational data. For non-relational data, consider a NoSQL store, such as Cosmos DB.
- **Azure Search**. Use Azure Search to add search functionality such as search suggestions, fuzzy search, and language-specific search. Azure Search is typically used in conjunction with another data store, especially if the primary data store requires strict consistency. In this approach, store authoritative data in the other data store and the search index in Azure Search. Azure Search can also be used to consolidate a single search index from

multiple data stores.

- **Email/SMS**. Use a third-party service such as SendGrid or Twilio to send email or SMS messages instead of building this functionality directly into the application.
- **Azure DNS**. Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services.

Recommendations

Your requirements might differ from the architecture described here. Use the recommendations in this section as a starting point.

App Service apps

We recommend creating the web application and the web API as separate App Service apps. This design lets you run them in separate App Service plans so they can be scaled independently. If you don't need that level of scalability initially, you can deploy the apps into the same plan and move them into separate plans later if necessary.

> **NOTE**
> For the Basic, Standard, and Premium plans, you are billed for the VM instances in the plan, not per app. See App Service Pricing

If you intend to use the *Easy Tables* or *Easy APIs* features of App Service Mobile Apps, create a separate App Service app for this purpose. These features rely on a specific application framework to enable them.

WebJobs

Consider deploying resource intensive WebJobs to an empty App Service app within a separate App Service plan. This provides dedicated instances for the WebJob. See Background jobs guidance.

Cache

You can improve performance and scalability by using Azure Redis Cache to cache some data. Consider using Redis Cache for:

- Semi-static transaction data.
- Session state.
- HTML output. This can be useful in applications that render complex HTML output.

For more detailed guidance on designing a caching strategy, see Caching guidance.

CDN

Use Azure CDN to cache static content. The main benefit of a CDN is to reduce latency for users, because content is cached at an edge server that is geographically close to the user. CDN can also reduce load on the application, because that traffic is not being handled by the application.

If your app consists mostly of static pages, consider using CDN to cache the entire app. Otherwise, put static content such as images, CSS, and HTML files, into Azure Storage and use CDN to cache those files.

> **NOTE**
> Azure CDN cannot serve content that requires authentication.

For more detailed guidance, see Content Delivery Network (CDN) guidance.

Storage

Modern applications often process large amounts of data. In order to scale for the cloud, it's important to choose the right storage type. Here are some baseline recommendations.

WHAT YOU WANT TO STORE	EXAMPLE	RECOMMENDED STORAGE
Files	Images, documents, PDFs	Azure Blob Storage
Key/Value pairs	User profile data looked up by user ID	Azure Table storage
Short messages intended to trigger further processing	Order requests	Azure Queue storage, Service Bus queue, or Service Bus topic
Non-relational data with a flexible schema requiring basic querying	Product catalog	Document database, such as Azure Cosmos DB, MongoDB, or Apache CouchDB
Relational data requiring richer query support, strict schema, and/or strong consistency	Product inventory	Azure SQL Database

Scalability considerations

A major benefit of Azure App Service is the ability to scale your application based on load. Here are some considerations to keep in mind when planning to scale your application.

App Service app

If your solution includes several App Service apps, consider deploying them to separate App Service plans. This approach enables you to scale them independently because they run on separate instances.

Similarly, consider putting a WebJob into its own plan so that background tasks don't run on the same instances that handle HTTP requests.

SQL Database

Increase scalability of a SQL database by *sharding* the database. Sharding refers to partitioning the database horizontally. Sharding allows you to scale out the database horizontally using Elastic Database tools. Potential benefits of sharding include:

- Better transaction throughput.
- Queries can run faster over a subset of the data.

Azure Search

Azure Search removes the overhead of performing complex data searches from the primary data store, and it can scale to handle load. See Scale resource levels for query and indexing workloads in Azure Search.

Security considerations

This section lists security considerations that are specific to the Azure services described in this article. It's not a complete list of security best practices. For some additional security considerations, see Secure an app in Azure App Service.

Cross-Origin Resource Sharing (CORS)

If you create a website and web API as separate apps, the website cannot make client-side AJAX calls to the API unless you enable CORS.

> **NOTE**
>
> Browser security prevents a web page from making AJAX requests to another domain. This restriction is called the same-origin policy, and prevents a malicious site from reading sentitive data from another site. CORS is a W3C standard that allows a server to relax the same-origin policy and allow some cross-origin requests while rejecting others.

App Services has built-in support for CORS, without needing to write any application code. See Consume an API app from JavaScript using CORS. Add the website to the list of allowed origins for the API.

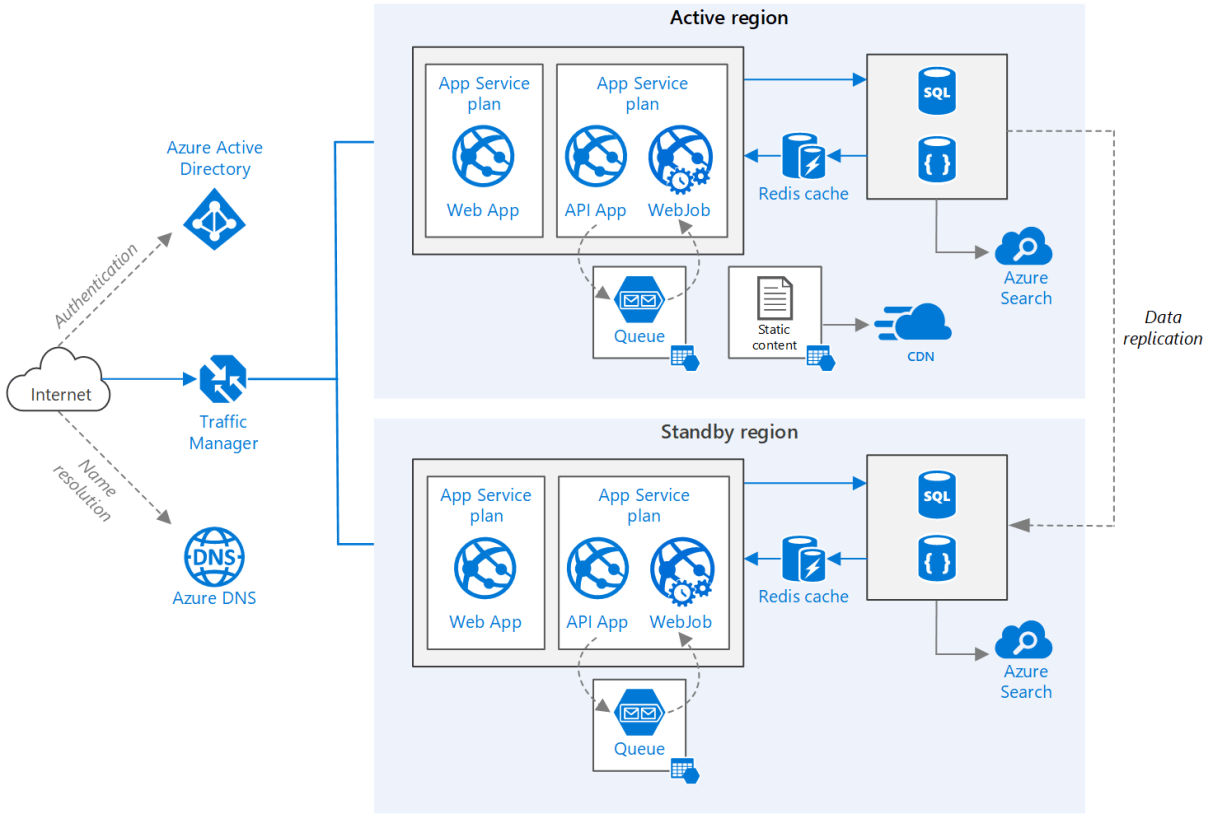
SQL Database encryption

Use Transparent Data Encryption if you need to encrypt data at rest in the database. This feature performs real-time encryption and decryption of an entire database (including backups and transaction log files) and requires no changes to the application. Encryption does add some latency, so it's a good practice to separate the data that must be secure into its own database and enable encryption only for that database.

Run a web application in multiple regions

4/11/2018 • 8 min to read • Edit Online

This reference architecture shows how to run an Azure App Service application in multiple regions to achieve high availability.



Download a Visio file of this architecture.

Architecture

This architecture builds on the one shown in Improve scalability in a web application. The main differences are:

- **Primary and secondary regions**. This architecture uses two regions to achieve higher availability. The application is deployed to each region. During normal operations, network traffic is routed to the primary region. If the primary region becomes unavailable, traffic is routed to the secondary region.
- **Azure DNS**. Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services.
- **Azure Traffic Manager**. Traffic Manager routes incoming requests to the primary region. If the application running that region becomes unavailable, Traffic Manager fails over to the secondary region.
- **Geo-replication** of SQL Database and Cosmos DB.

A multi-region architecture can provide higher availability than deploying to a single region. If a regional outage affects the primary region, you can use Traffic Manager to fail over to the secondary region. This architecture can also help if an individual subsystem of the application fails.

There are several general approaches to achieving high availability across regions:

- Active/passive with hot standby. Traffic goes to one region, while the other waits on hot standby. Hot standby

means the VMs in the secondary region are allocated and running at all times.

- Active/passive with cold standby. Traffic goes to one region, while the other waits on cold standby. Cold standby means the VMs in the secondary region are not allocated until needed for failover. This approach costs less to run, but will generally take longer to come online during a failure.
- Active/active. Both regions are active, and requests are load balanced between them. If one region becomes unavailable, it is taken out of rotation.

This reference architecture focuses on active/passive with hot standby, using Traffic Manager for failover.

Recommendations

Your requirements might differ from the architecture described here. Use the recommendations in this section as a starting point.

Regional pairing

Each Azure region is paired with another region within the same geography. In general, choose regions from the same regional pair (for example, East US 2 and Central US). Benefits of doing so include:

- If there is a broad outage, recovery of at least one region out of every pair is prioritized.
- Planned Azure system updates are rolled out to paired regions sequentially to minimize possible downtime.
- In most cases, regional pairs reside within the same geography to meet data residency requirements.

However, make sure that both regions support all of the Azure services needed for your application. See Services by region. For more information about regional pairs, see Business continuity and disaster recovery (BCDR): Azure Paired Regions.

Resource groups

Consider placing the primary region, secondary region, and Traffic Manager into separate resource groups. This lets you manage the resources deployed to each region as a single collection.

Traffic Manager configuration

Routing. Traffic Manager supports several routing algorithms. For the scenario described in this article, use *priority* routing (formerly called *failover* routing). With this setting, Traffic Manager sends all requests to the primary region unless the endpoint for that region becomes unreachable. At that point, it automatically fails over to the secondary region. See Configure Failover routing method.

Health probe. Traffic Manager uses an HTTP (or HTTPS) probe to monitor the availability of each endpoint. The probe gives Traffic Manager a pass/fail test for failing over to the secondary region. It works by sending a request to a specified URL path. If it gets a non-200 response within a timeout period, the probe fails. After four failed requests, Traffic Manager marks the endpoint as degraded and fails over to the other endpoint. For details, see Traffic Manager endpoint monitoring and failover.

As a best practice, create a health probe endpoint that reports the overall health of the application and use this endpoint for the health probe. The endpoint should check critical dependencies such as the App Service apps, storage queue, and SQL Database. Otherwise, the probe might report a healthy endpoint when critical parts of the application are actually failing.

On the other hand, don't use the health probe to check lower priority services. For example, if an email service goes down the application can switch to a second provider or just send emails later. This is not a high enough priority to cause the application to fail over. For more information, see Health Endpoint Monitoring Pattern.

SQL Database

Use Active Geo-Replication to create a readable secondary replica in a different region. You can have up to four readable secondary replicas. Fail over to a secondary database if your primary database fails or needs to be taken offline. Active Geo-Replication can be configured for any database in any elastic database pool.

Cosmos DB

Cosmos DB supports geo-replication across regions. One region is designated as writable and the others are read-only replicas.

If there is a regional outage, you can fail over by selecting another region to be the write region. The client SDK automatically sends write requests to the current write region, so you don't need to update the client configuration after a failover. For more information, see How to distribute data globally with Azure Cosmos DB.

> **NOTE**
> All of the replicas belong to the same resource group.

Storage

For Azure Storage, use read-access geo-redundant storage (RA-GRS). With RA-GRS storage, the data is replicated to a secondary region. You have read-only access to the data in the secondary region through a separate endpoint. If there is a regional outage or disaster, the Azure Storage team might decide to perform a geo-failover to the secondary region. There is no customer action required for this failover.

For Queue storage, create a backup queue in the secondary region. During failover, the app can use the backup queue until the primary region becomes available again. That way, the application can still process new requests.

Availability considerations

Traffic Manager

Traffic Manager automatically fails over if the primary region becomes unavailable. When Traffic Manager fails over, there is a period of time when clients cannot reach the application. The duration is affected by the following factors:

- The health probe must detect that the primary data center has become unreachable.
- Domain name service (DNS) servers must update the cached DNS records for the IP address, which depends on the DNS time-to-live (TTL). The default TTL is 300 seconds (5 minutes), but you can configure this value when you create the Traffic Manager profile.

For details, see About Traffic Manager Monitoring.

Traffic Manager is a possible failure point in the system. If the service fails, clients cannot access your application during the downtime. Review the Traffic Manager service level agreement (SLA) and determine whether using Traffic Manager alone meets your business requirements for high availability. If not, consider adding another traffic management solution as a failback. If the Azure Traffic Manager service fails, change your canonical name (CNAME) records in DNS to point to the other traffic management service. This step must be performed manually, and your application will be unavailable until the DNS changes are propagated.

SQL Database

The recovery point objective (RPO) and estimated recovery time (ERT) for SQL Database are documented in Overview of business continuity with Azure SQL Database.

Storage

RA-GRS storage provides durable storage, but it's important to understand what can happen during an outage:

- If a storage outage occurs, there will be a period of time when you don't have write-access to the data. You can still read from the secondary endpoint during the outage.
- If a regional outage or disaster affects the primary location and the data there cannot be recovered, the Azure Storage team may decide to perform a geo-failover to the secondary region.

- Data replication to the secondary region is performed asynchronously. Therefore, if a geo-failover is performed, some data loss is possible if the data can't be recovered from the primary region.
- Transient failures, such as a network outage, will not trigger a storage failover. Design your application to be resilient to transient failures. Possible mitigations:
  - Read from the secondary region.
  - Temporarily switch to another storage account for new write operations (for example, to queue messages).
  - Copy data from the secondary region to another storage account.
  - Provide reduced functionality until the system fails back.

For more information, see What to do if an Azure Storage outage occurs.

Manageability Considerations

Traffic Manager

If Traffic Manager fails over, we recommend performing a manual failback rather than implementing an automatic failback. Otherwise, you can create a situation where the application flips back and forth between regions. Verify that all application subsystems are healthy before failing back.

Note that Traffic Manager automatically fails back by default. To prevent this, manually lower the priority of the primary region after a failover event. For example, suppose the primary region is priority 1 and the secondary is priority 2. After a failover, set the primary region to priority 3, to prevent automatic failback. When you are ready to switch back, update the priority to 1.

The following commands update the priority.

PowerShell

```
$endpoint = Get-AzureRmTrafficManagerEndpoint -Name <endpoint> -ProfileName <profile> -ResourceGroupName
<resource-group> -Type AzureEndpoints
$endpoint.Priority = 3
Set-AzureRmTrafficManagerEndpoint -TrafficManagerEndpoint $endpoint
```

For more information, see Azure Traffic Manager Cmdlets.

Azure command line interface (CLI)

```
azure network traffic-manager endpoint set --name <endpoint> --profile-name <profile> --resource-group
<resource-group> --type AzureEndpoints --priority 3
```

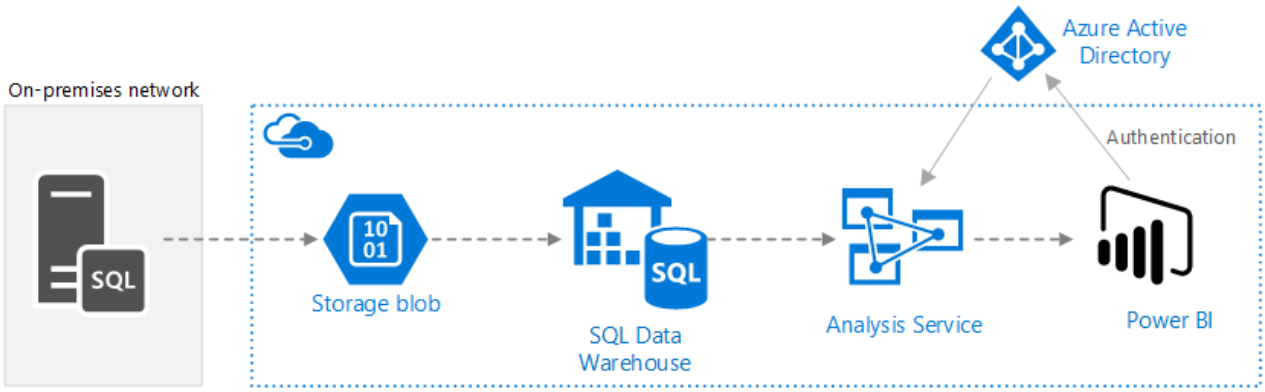
SQL Database

If the primary database fails, perform a manual failover to the secondary database. See Restore an Azure SQL Database or failover to a secondary. The secondary database remains read-only until you fail over.

Enterprise BI with SQL Data Warehouse

4/14/2018 • 18 min to read • Edit Online

This reference architecture implements an ELT (extract-load-transform) pipeline that moves data from an on-premises SQL Server database into SQL Data Warehouse and transforms the data for analysis. **Deploy this solution**.



Scenario: An organization has a large OLTP data set stored in a SQL Server database on premises. The organization wants to use SQL Data Warehouse to perform analysis using Power BI.

This reference architecture is designed for one-time or on-demand jobs. If you need to move data on a continuing basis (hourly or daily), we recommend using Azure Data Factory to define an automated workflow.

Architecture

The architecture consists of the following components.

SQL Server. The source data is located in a SQL Server database on premises. To simulate the on-premises environment, the deployment scripts for this architecture provision a virtual machine in Azure with SQL Server installed.

Blob Storage. Blob storage is used as a staging area to copy the data before loading it into SQL Data Warehouse.

Azure SQL Data Warehouse. SQL Data Warehouse is a distributed system designed to perform analytics on large data. It supports massive parallel processing (MPP), which makes it suitable for running high-performance analytics.

Azure Analysis Services. Analysis Services is a fully managed service that provides data modeling capabilities. Use Analysis Services to create a semantic model that users can query. Analysis Services is especially useful in a BI dashboard scenario. In this architecture, Analysis Services reads data from the data warehouse to process the semantic model, and efficiently serves dashboard queries. It also supports elastic concurrency, by scaling out replicas for faster query processing.

Currently, Azure Analysis Services supports tabular models but not multidimensional models. Tabular models use relational modeling constructs (tables and columns), whereas multidimensional models use OLAP modeling constructs (cubes, dimensions, and measures). If you require multidimensional models, use SQL Server Analysis Services (SSAS). For more information, see Comparing tabular and multidimensional solutions.

Power BI. Power BI is a suite of business analytics tools to analyze data for business insights. In this architecture, it queries the semantic model stored in Analysis Services.

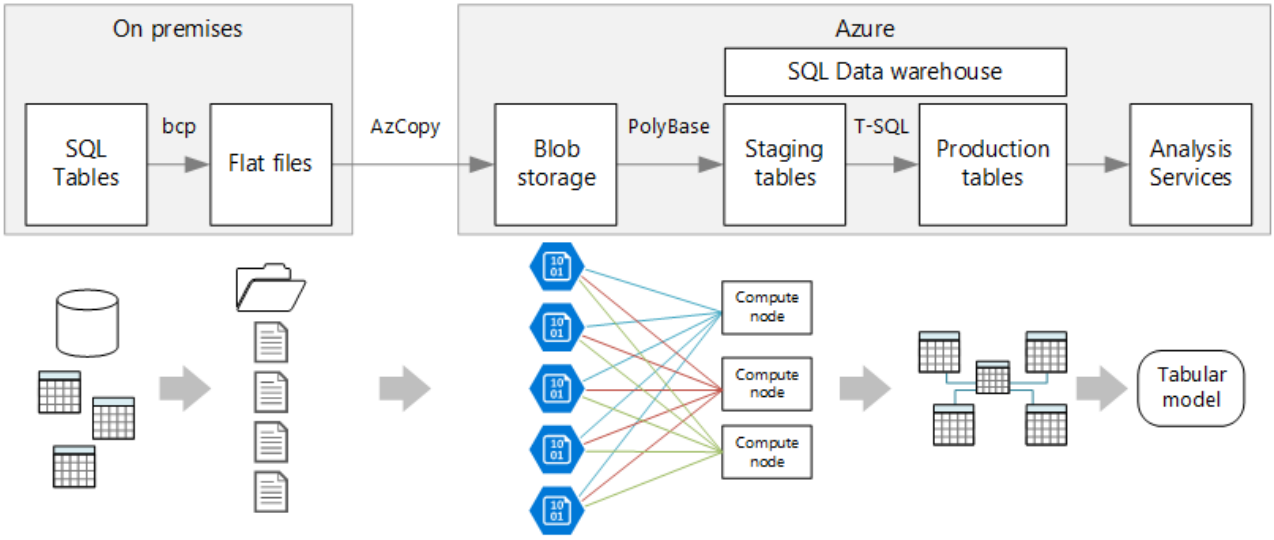
Azure Active Directory (Azure AD) authenticates users who connect to the Analysis Services server through

Power BI.

Data Pipeline

This reference architecture uses the WorldWideImporters sample database as data source. The data pipeline has the following stages:

1. Export the data from SQL Server to flat files (bcp utility).
2. Copy the flat files to Azure Blob Storage (AzCopy).
3. Load the data into SQL Data Warehouse (PolyBase).
4. Transform the data into a star schema (T-SQL).
5. Load a semantic model into Analysis Services (SQL Server Data Tools).



> **NOTE**
>
> For steps 1 – 3, consider using Redgate Data Platform Studio. Data Platform Studio applies the most appropriate compatibility fixes and optimizations, so it's the quickest way to get started with SQL Data Warehouse. For more information, see Load data with Redgate Data Platform Studio.

The next sections describe these stages in more detail.

Export data from SQL Server

The bcp (bulk copy program) utility is a fast way to create flat text files from SQL tables. In this step, you select the columns that you want to export, but don't transform the data. Any data transformations should happen in SQL Data Warehouse.

Recommendations

If possible, schedule data extraction during off-peak hours, to minimize resource contention in the production environment.

Avoid running bcp on the database server. Instead, run it from another machine. Write the files to a local drive. Ensure that you have sufficient I/O resources to handle the concurrent writes. For best performance, export the files to dedicated fast storage drives.

You can speed up the network transfer by saving the exported data in Gzip compressed format. However, loading compressed files into the warehouse is slower than loading uncompressed files, so there is a tradeoff between faster network transfer versus faster loading. If you decide to use Gzip compression, don't create a single Gzip file. Instead, split the data into multiple compressed files.

Copy flat files into blob storage

The AzCopy utility is designed for high-performance copying of data into Azure blob storage.

Recommendations

Create the storage account in a region near the location of the source data. Deploy the storage account and the SQL Data Warehouse instance in the same region.

Don't run AzCopy on the same machine that runs your production workloads, because the CPU and I/O consumption can interfere with the production workload.

Test the upload first to see what the upload speed is like. You can use the /NC option in AzCopy to specify the number of concurrent copy operations. Start with the default value, then experiment with this setting to tune the performance. In a low-bandwidth environment, too many concurrent operations can overwhelm the network connection and prevent the operations from completing successfully.

AzCopy moves data to storage over the public internet. If this isn't fast enough, consider setting up an ExpressRoute circuit. ExpressRoute is a service that routes your data through a dedicated private connection to Azure. Another option, if your network connection is too slow, is to physically ship the data on disk to an Azure datacenter. For more information, see Transferring data to and from Azure.

During a copy operation, AzCopy creates a temporary journal file, which enables AzCopy to restart the operation if it gets interrupted (for example, due to a network error). Make sure there is enough disk space to store the journal files. You can use the /Z option to specify where the journal files are written.

Load data into SQL Data Warehouse

Use PolyBase to load the files from blob storage into the data warehouse. PolyBase is designed to leverage the MPP (Massively Parallel Processing) architecture of SQL Data Warehouse, which makes it the fastest way to load data into SQL Data Warehouse.

Loading the data is a two-step process:

1. Create a set of external tables for the data. An external table is a table definition that points to data stored outside of the warehouse — in this case, the flat files in blob storage. This step does not move any data into the warehouse.
2. Create staging tables, and load the data into the staging tables. This step copies the data into the warehouse.

Recommendations

Consider SQL Data Warehouse when you have large amounts of data (more than 1 TB) and are running an analytics workload that will benefit from parallelism. SQL Data Warehouse is not a good fit for OLTP workloads or smaller data sets (< 250GB). For data sets less than 250GB, consider Azure SQL Database or SQL Server. For more information, see Data warehousing.

Create the staging tables as heap tables, which are not indexed. The queries that create the production tables will result in a full table scan, so there is no reason to index the staging tables.

PolyBase automatically takes advantage of parallelism in the warehouse. The load performance scales as you increase DWUs. For best performance, use a single load operation. There is no performance benefit to breaking the input data into chunks and running multiple concurrent loads.

PolyBase can read Gzip compressed files. However, only a single reader is used per compressed file, because uncompressing the file is a single-threaded operation. Therefore, avoid loading a single large compressed file. Instead, split the data into multiple compressed files, in order to take advantage of parallelism.

Be aware of the following limitations:

- PolyBase supports a maximum column size of `varchar(8000)`, `nvarchar(4000)`, or `varbinary(8000)`. If you

have data that exceeds these limits, one option is to break the data up into chunks when you export it, and then reassemble the chunks after import.

- PolyBase uses a fixed row terminator of \n or newline. This can cause problems if newline characters appear in the source data.
- Your source data schema might contain data types that are not supported in SQL Data Warehouse.

To work around these limitations, you can create a stored procedure that performs the necessary conversions. Reference this stored procedure when you run bcp. Alternatively, Redgate Data Platform Studio automatically converts data types that aren't supported in SQL Data Warehouse.

For more information, see the following articles:

- Best practices for loading data into Azure SQL Data Warehouse.
- Migrate your schemas to SQL Data Warehouse
- Guidance for defining data types for tables in SQL Data Warehouse

Transform the data

Transform the data and move it into production tables. In this step, the data is transformed into a star schema with dimension tables and fact tables, suitable for semantic modeling.

Create the production tables with clustered columnstore indexes, which offer the best overall query performance. Columnstore indexes are optimized for queries that scan many records. Columnstore indexes don't perform as well for singleton lookups (that is, looking up a single row). If you need to perform frequent singleton lookups, you can add a non-clustered index to a table. Singleton lookups can run significantly faster using a non-clustered index. However, singleton lookups are typically less common in data warehouse scenarios than OLTP workloads. For more information, see Indexing tables in SQL Data Warehouse.

> **NOTE**
> Clustered columnstore tables do not support `varchar(max)`, `nvarchar(max)`, or `varbinary(max)` data types. In that case, consider a heap or clustered index. You might put those columns into a separate table.

Because the sample database is not very large, we created replicated tables with no partitions. For production workloads, using distributed tables is likely to improve query performance. See Guidance for designing distributed tables in Azure SQL Data Warehouse. Our example scripts run the queries using a static resource class.

Load the semantic model

Load the data into a tabular model in Azure Analysis Services. In this step, you create a semantic data model by using SQL Server Data Tools (SSDT). You can also create a model by importing it from a Power BI Desktop file. Because SQL Data Warehouse does not support foreign keys, you must add the relationships to the semantic model, so that you can join across tables.

Use Power BI to visualize the data

Power BI supports two options for connecting to Azure Analysis Services:

- Import. The data is imported into the Power BI model.
- Live Connection. Data is pulled directly from Analysis Services.

We recommend Live Connection because it doesn't require copying data into the Power BI model. Also, using DirectQuery ensures that results are always consistent with the latest source data. For more information, see Connect with Power BI.

Recommendations

Avoid running BI dashboard queries directly against the data warehouse. BI dashboards require very low response times, which direct queries against the warehouse may be unable to satisfy. Also, refreshing the dashboard will count against the number of concurrent queries, which could impact performance.

Azure Analysis Services is designed to handle the query requirements of a BI dashboard, so the recommended practice is to query Analysis Services from Power BI.

Scalability Considerations

SQL Data Warehouse

With SQL Data Warehouse, you can scale out your compute resources on demand. The query engine optimizes queries for parallel processing based on the number of compute nodes, and moves data between nodes as necessary. For more information, see Manage compute in Azure SQL Data Warehouse.

Analysis Services

For production workloads, we recommend the Standard Tier for Azure Analysis Services, because it supports partitioning and DirectQuery. Within a tier, the instance size determines the memory and processing power. Processing power is measured in Query Processing Units (QPUs). Monitor your QPU usage to select the appropriate size. For more information, see Monitor server metrics.

Under high load, query performance can become degraded due to query concurrency. You can scale out Analysis Services by creating a pool of replicas to process queries, so that more queries can be performed concurrently. The work of processing the data model always happens on the primary server. By default, the primary server also handles queries. Optionally, you can designate the primary server to run processing exclusively, so that the query pool handles all queries. If you have high processing requirements, you should separate the processing from the query pool. If you have high query loads, and relatively light processing, you can include the primary server in the query pool. For more information, see Azure Analysis Services scale-out.

To reduce the amount of unnecessary processing, consider using partitions to divide the tabular model into logical parts. Each partition can be processed separately. For more information, see Partitions.

Security Considerations

IP whitelisting of Analysis Services clients

Consider using the Analysis Services firewall feature to whitelist client IP addresses. If enabled, the firewall blocks all client connections other than those specified in the firewall rules. The default rules whitelist the Power BI service, but you can disable this rule if desired. For more information, see Hardening Azure Analysis Services with the new firewall capability.

Authorization

Azure Analysis Services uses Azure Active Directory (Azure AD) to authenticate users who connect to an Analysis Services server. You can restrict what data a particular user is able to view, by creating roles and then assigning Azure AD users or groups to those roles. For each role, you can:

- Protect tables or individual columns.
- Protect individual rows based on filter expressions.

For more information, see Manage database roles and users.

Deploy the solution

A deployment for this reference architecture is available on GitHub. It deploys the following:

- A Windows VM to simulate an on-premises database server. It includes SQL Server 2017 and related tools, along with Power BI Desktop.

- An Azure storage account that provides Blob storage to hold data exported from the SQL Server database.
- An Azure SQL Data Warehouse instance.
- An Azure Analysis Services instance.

Prerequisites

1. Clone, fork, or download the zip file for the Azure reference architectures GitHub repository.
2. Install the Azure Building Blocks (azbb).
3. From a command prompt, bash prompt, or PowerShell prompt, login to your Azure account by using the command below and following the instructions.

```
az login
```

Deploy the simulated on-premises server

First you'll deploy a VM as a simulated on-premises server, which includes SQL Server 2017 and related tools. This step also loads the sample Wide World Importers OLTP database into SQL Server.

1. Navigate to the `data\enterprise-bi-sqldw\onprem\templates` folder of the repository you downloaded in the prerequisites above.
2. In the `onprem.parameters.json` file, replace the values for `adminUsername` and `adminPassword`. Also change the values in the `SqlUserCredentials` section to match the user name and password. Note the `.\\` prefix in the userName property.

   ```
   "SqlUserCredentials": {
     "userName": ".\\username",
     "password": "password"
   }
   ```

3. Run `azbb` as shown below to deploy the on-premises server.

   ```
   azbb -s <subscription_id> -g <resource_group_name> -l <location> -p onprem.parameters.json --deploy
   ```

4. The deployment may take 20 to 30 minutes to complete, which includes running the DSC script to install the tools and restore the database. Verify the deployment in the Azure portal by reviewing the resources in the resource group. You should see the `sql-vm1` virtual machine and its associated resources.

Deploy the Azure resources

This step provisions Azure SQL Data Warehouse and Azure Analysis Services, along with a Storage account. If you want, you can run this step in parallel with the previous step.

1. Navigate to the `data\enterprise-bi-sqldw\azure\templates` folder of the repository you downloaded in the prerequisites above.
2. Run the following Azure CLI command to create a resource group, replacing the bracketed parameters specified. Note that you can deploy to a different resource group than you used for the on-premises server in the previous step.

   ```
   az group create --name <resource_group_name> --location <location>
   ```

3. Run the following Azure CLI command to deploy the Azure resources, replacing the bracketed parameters specified. The `storageAccountName` parameter must follow the naming rules for Storage accounts. For the

`analysisServerAdmin` parameter, use your Azure Active Directory user principal name (UPN).

```
az group deployment create --resource-group <resource_group_name> --template-file azure-resources-deploy.json --parameters "dwServerName"="<server_name>" "dwAdminLogin"="<admin_username>" "dwAdminPassword"="<password>" "storageAccountName"="<storage_account_name>" "analysisServerName"="<analysis_server_name>" "analysisServerAdmin"="user@contoso.com"
```

4. Verify the deployment in the Azure portal by reviewing the resources in the resource group. You should see a storage account, Azure SQL Data Warehouse instance, and Analysis Services instance.

5. Use the Azure portal to get the access key for the storage account. Select the storage account to open it. Under **Settings**, select **Access keys**. Copy the primary key value. You will use it in the next step.

Export the source data to Azure Blob storage

In this step, you will run a PowerShell script that uses bcp to export the SQL database to flat files on the VM, and then uses AzCopy to copy those files into Azure Blob Storage.

1. Use Remote Desktop to connect to the simulated on-premises VM.

2. While logged into the VM, run the following commands from a PowerShell window.

```
cd 'C:\SampleDataFiles\reference-architectures\data\enterprise_bi_sqldw\onprem'

.\Load_SourceData_To_Blob.ps1 -File .\sql_scripts\db_objects.txt -Destination 'https://<storage_account_name>.blob.core.windows.net/wwi' -StorageAccountKey '<storage_account_key>'
```

   For the `Destination` parameter, replace `<storage_account_name>` with the name the Storage account that you created previously. For the `StorageAccountKey` parameter, use the access key for that Storage account.

3. In the Azure portal, verify that the source data was copied to Blob storage by navigating to the storage account, selecting the Blob service, and opening the `wwi` container. You should see a list of tables prefaced with `WorldWideImporters_Application_*`.

Execute the data warehouse scripts

1. From your Remote Desktop session, launch SQL Server Management Studio (SSMS).

2. Connect to SQL Data Warehouse

   - Server type: Database Engine
   - Server name: `<dwServerName>.database.windows.net`, where `<dwServerName>` is the name that you specified when you deployed the Azure resources. You can get this name from the Azure portal.
   - Authentication: SQL Server Authentication. Use the credentials that you specified when you deployed the Azure resources, in the `dwAdminLogin` and `dwAdminPassword` parameters.

3. Navigate to the `C:\SampleDataFiles\reference-architectures\data\enterprise_bi_sqldw\azure\sqldw_scripts` folder on the VM. You will execute the scripts in this folder in numerical order, `STEP_1` through `STEP_7`.

4. Select the `master` database in SSMS and open the `STEP_1` script. Change the value of the password in the following line, then execute the script.

```
CREATE LOGIN LoaderRC20 WITH PASSWORD = '<change this value>';
```

5. Select the `wwi` database in SSMS. Open the `STEP_2` script and execute the script. If you get an error, make sure you are running the script against the `wwi` database and not `master`.

6. Open a new connection to SQL Data Warehouse, using the `LoaderRC20` user and the password indicated in the `STEP_1` script.
7. Using this connection, open the `STEP_3` script. Set the following values in the script:
   - SECRET: Use the access key for your storage account.
   - LOCATION: Use the name of the storage account as follows: `wasbs://wwi@<storage_account_name>.blob.core.windows.net`.
8. Using the same connection, execute scripts `STEP_4` through `STEP_7` sequentially. Verify that each script completes successfully before running the next.

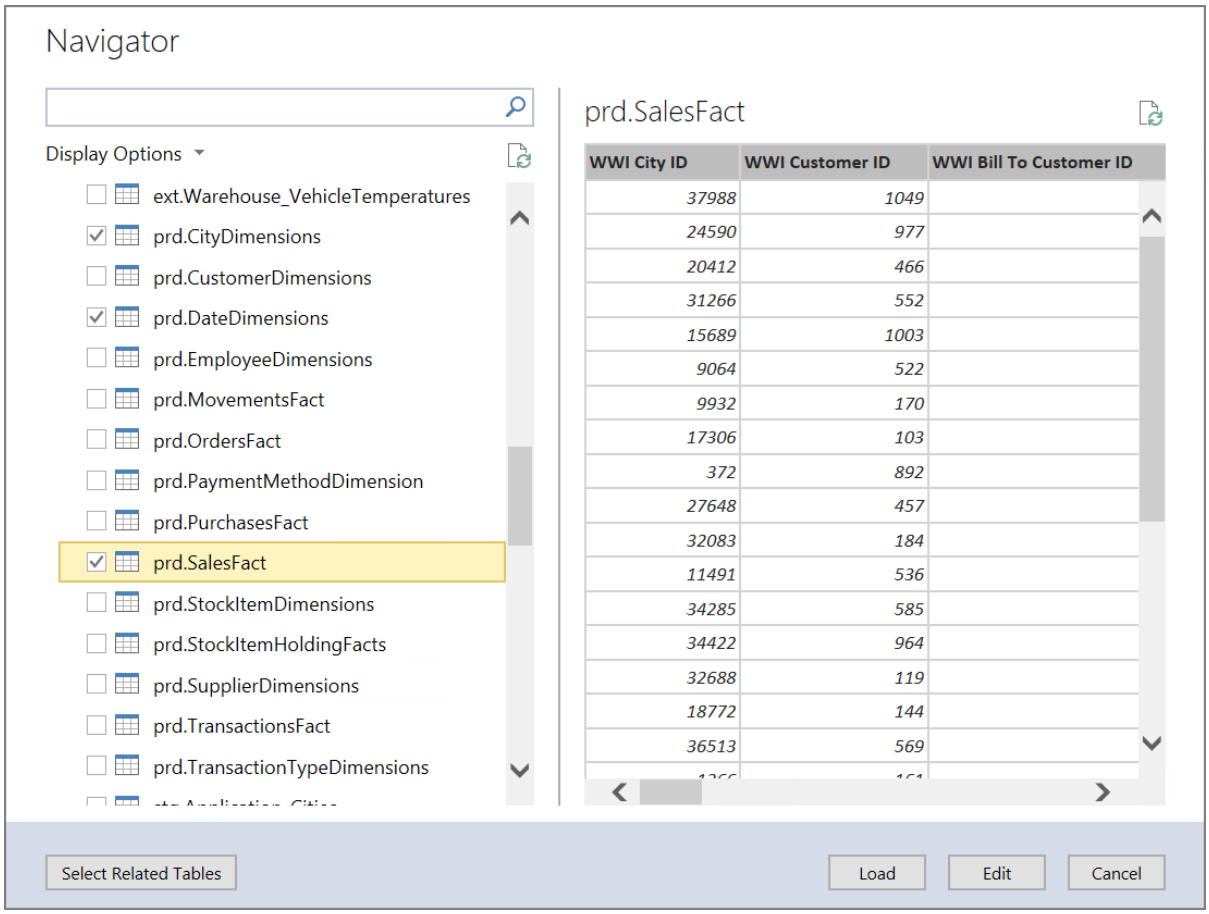
In SMSS, you should see a set of `prd.*` tables in the `wwi` database. To verify that the data was generated, run the following query:

```
SELECT TOP 10 * FROM prd.CityDimensions
```

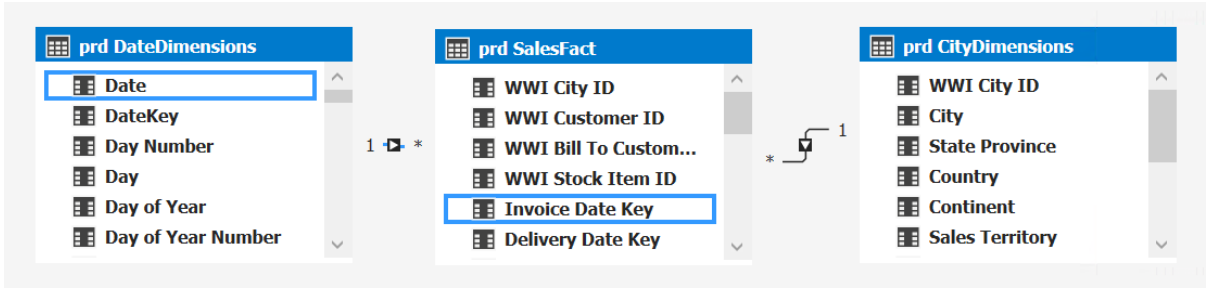
Build the Azure Analysis Services model

In this step, you will create a tabular model that imports data from the data warehouse. Then you will deploy the model to Azure Analysis Services.

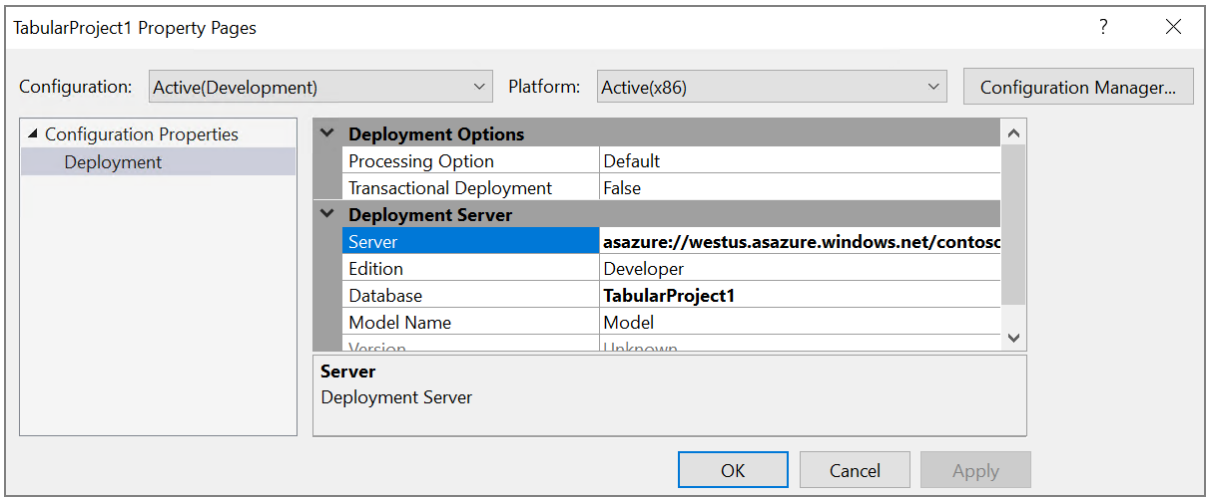
1. From your Remote Desktop session, launch SQL Server Data Tools 2015.
2. Select **File** > **New** > **Project**.
3. In the **New Project** dialog, under **Templates**, select **Business Intelligence** > **Analysis Services** > **Analysis Services Tabular Project**.
4. Name the project and click **OK**.
5. In the **Tabular model designer** dialog, select **Integrated workspace** and set **Compatibility level** to `SQL Server 2017 / Azure Analysis Services (1400)`. Click **OK**.
6. In the **Tabular Model Explorer** window, right-click the project and select **Import from Data Source**.
7. Select **Azure SQL Data Warehouse** and click **Connect**.
8. For **Server**, enter the fully qualified name of your Azure SQL Data Warehouse server. For **Database**, enter `wwi`. Click **OK**.
9. In the next dialog, choose **Database** authentication and enter your Azure SQL Data Warehouse user name and password, and click **OK**.
10. In the **Navigator** dialog, select the checkboxes for **prd.CityDimensions**, **prd.DateDimensions**, and **prd.SalesFact**.



11. Click **Load**. When processing is complete, click **Close**. You should now see a tabular view of the data.
12. In the **Tabular Model Explorer** window, right-click the project and select **Model View** > **Diagram View**.
13. Drag the **[prd.SalesFact].[WWI City ID]** field to the **[prd.CityDimensions].[WWI City ID]** field to create a relationship.
14. Drag the **[prd.SalesFact].[Invoice Date Key]** field to the **[prd.DateDimensions].[Date]** field.

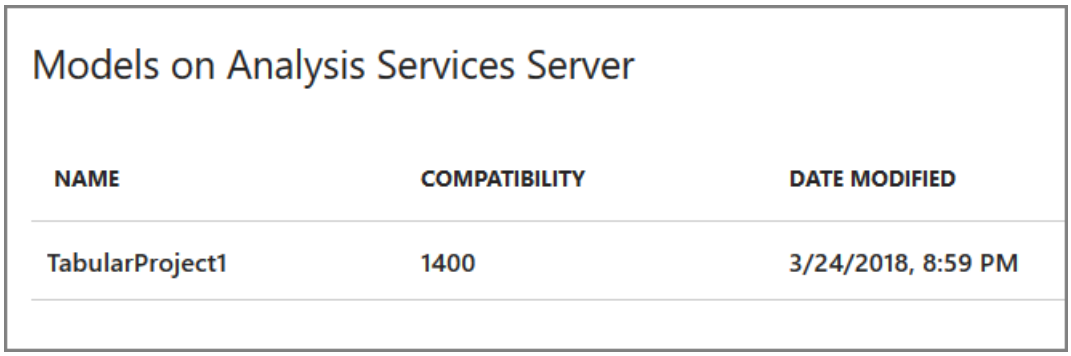


15. From the **File** menu, choose **Save All**.
16. In **Solution Explorer**, right-click the project and select **Properties**.
17. Under **Server**, enter the URL of your Azure Analysis Services instance. You can get this value from the Azure Portal. In the portal, select the Analysis Services resource, click the Overview pane, and look for the **Server Name** property. It will be similar to `asazure://westus.asazure.windows.net/contoso`. Click **OK**.



18. In **Solution Explorer**, right-click the project and select **Deploy**. Sign into Azure if prompted. When processing is complete, click **Close**.

19. In the Azure portal, view the details for your Azure Analysis Services instance. Verify that your model appears in the list of models.



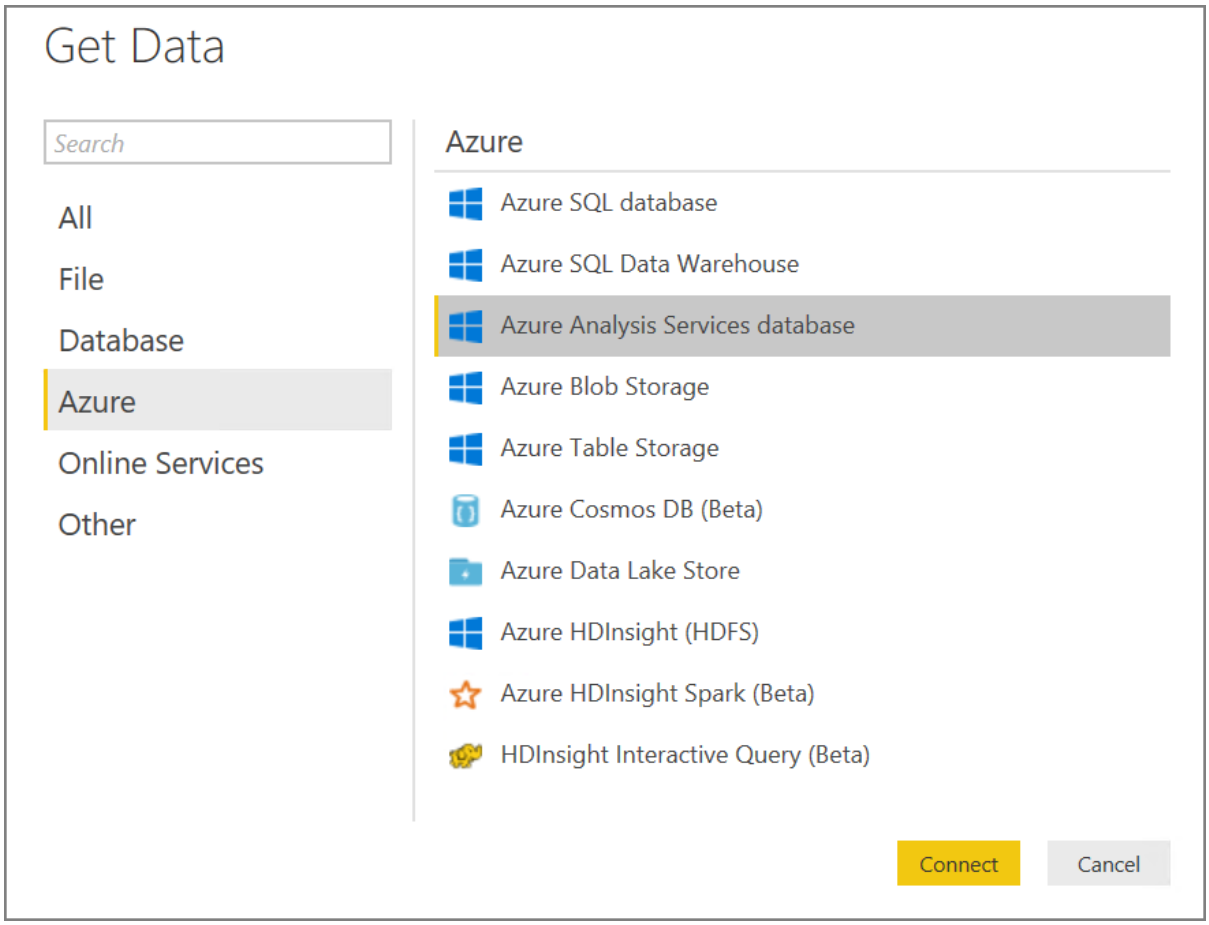
Analyze the data in Power BI Desktop

In this step, you will use Power BI to create a report from the data in Analysis Services.

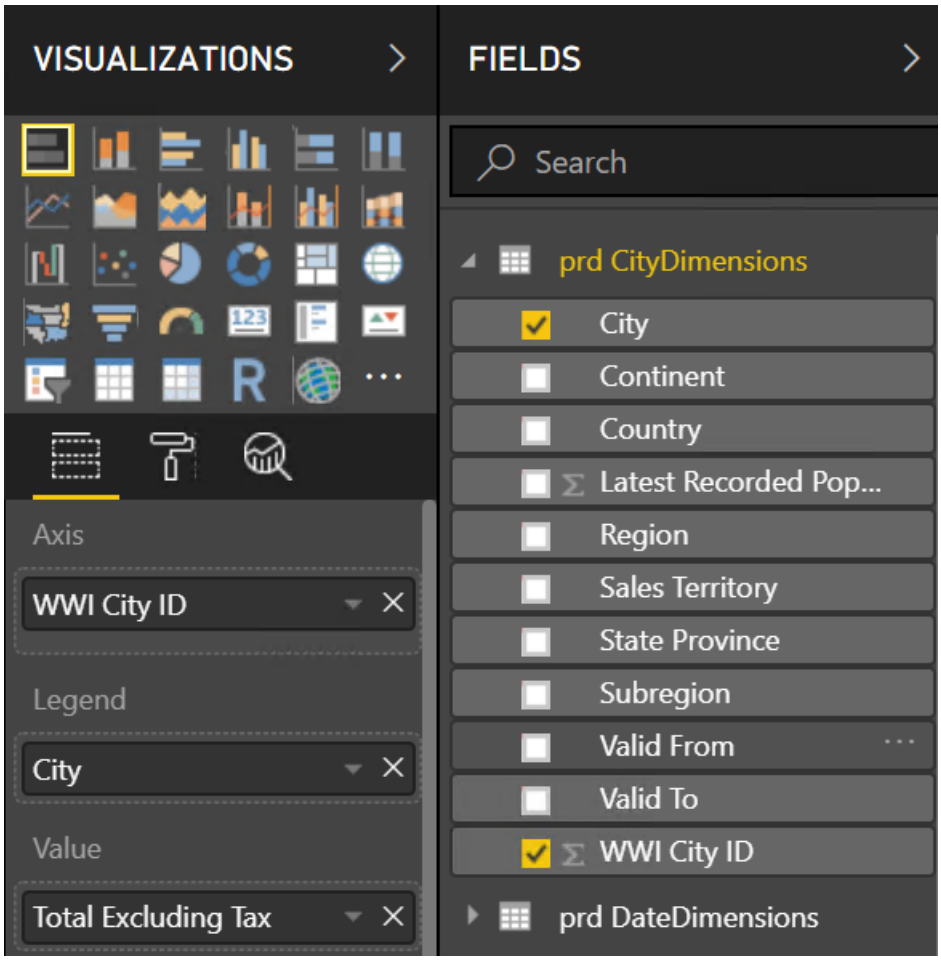
1. From your Remote Desktop session, launch Power BI Desktop.

2. In the Welcome Scren, click **Get Data**.

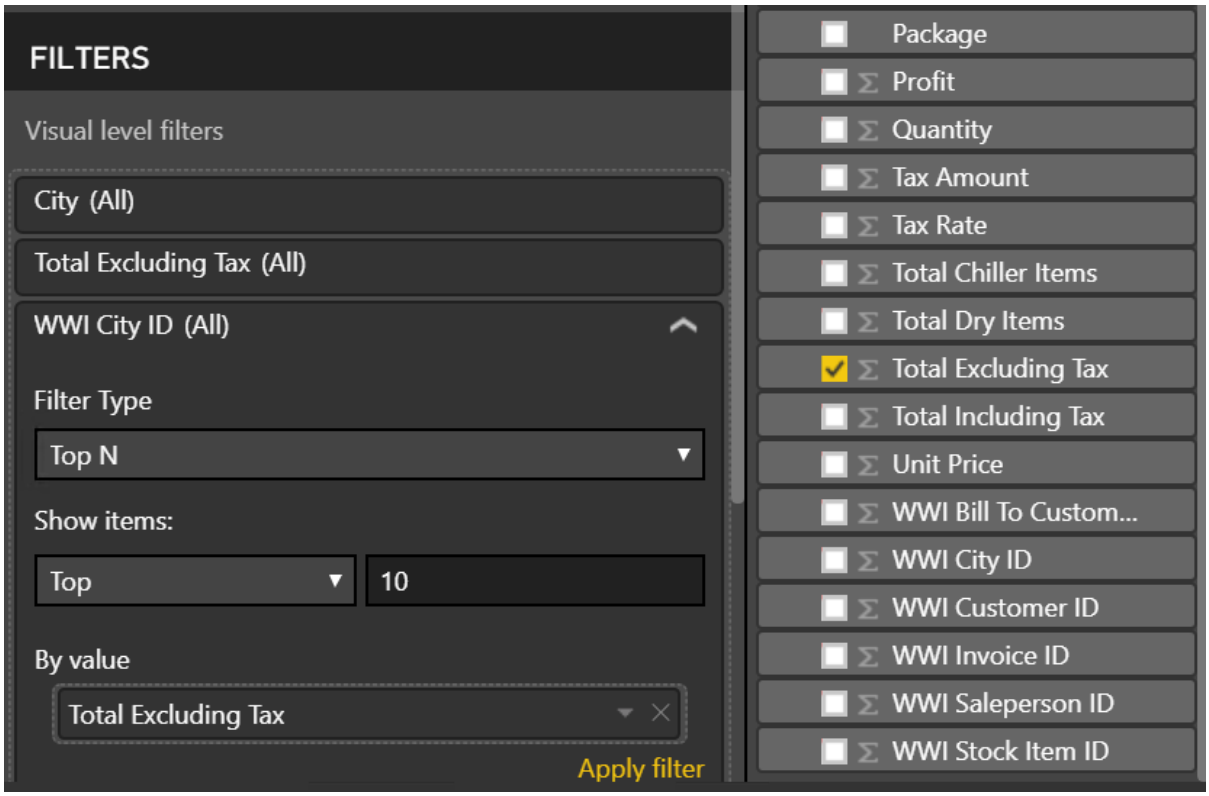
3. Select **Azure** > **Azure Analysis Services database**. Click **Connect**



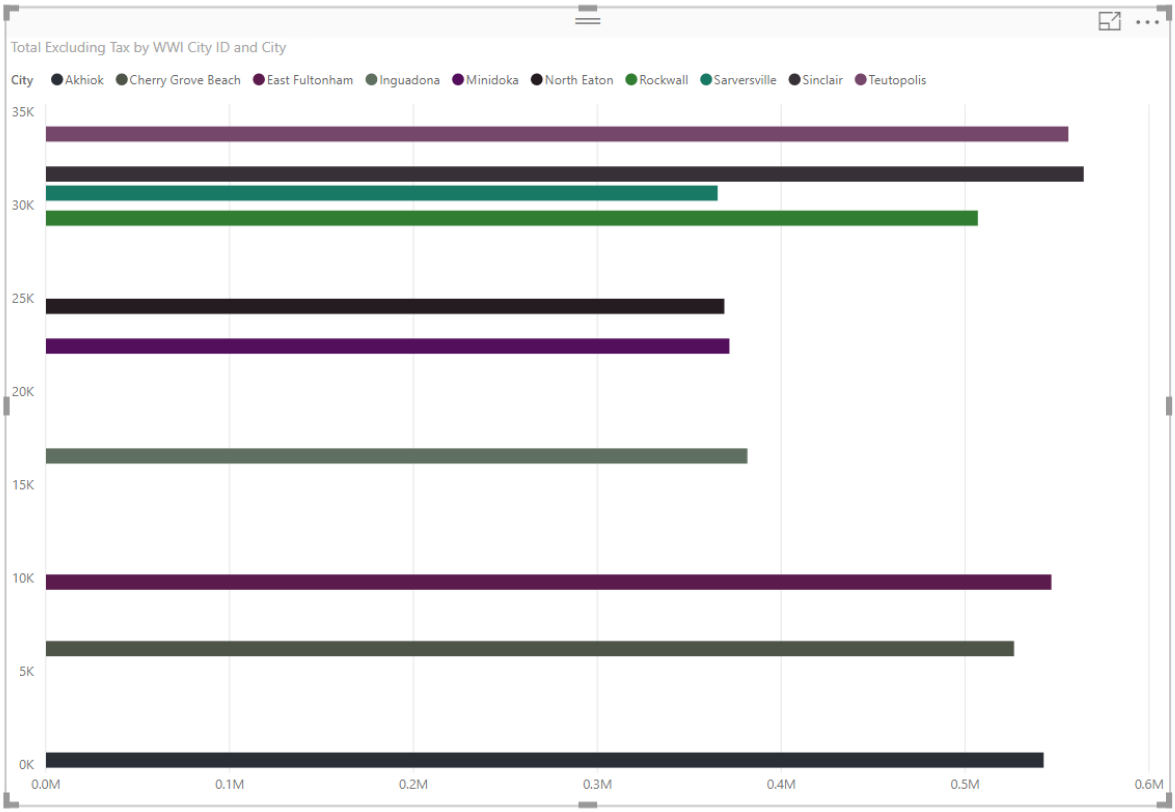
4. Enter the URL of your Analysis Services instance, then click **OK**. Sign into Azure if prompted.
5. In the **Navigator** dialog, expand the tabular project that you deployed, select the model that you created, and click **OK**.
6. In the **Visualizations** pane, select the **Stacked Bar Chart** icon. In the Report view, resize the visualization to make it larger.
7. In the **Fields** pane, expand **prd.CityDimensions**.
8. Drag **prd.CityDimensions** > **WWI City ID** to the **Axis well**.
9. Drag **prd.CityDimensions** > **City** to the **Legend** well.
10. In the Fields pane, expand **prd.SalesFact**.
11. Drag **prd.SalesFact** > **Total Excluding Tax** to the **Value** well.



12. Under **Visual Level Filters**, select **WWI City ID**.
13. Set the **Filter Type** to `Top N`, and set **Show Items** to `Top 10`.
14. Drag **prd.SalesFact** > **Total Excluding Tax** to the **By Value** well



15. Click **Apply Filter**. The visualization shows the top 10 total sales by city.

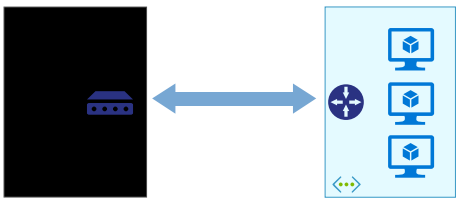


To learn more about Power BI Desktop, see Getting started with Power BI Desktop.

Next steps

- For more information about this reference architecture, visit our GitHub repository.
- Learn about the Azure Building Blocks.

These reference architectures show proven practices for creating a robust network connection between an on-premises network and Azure. Which should I choose?



VPN

Extend an on-premises network to Azure using a site-to-site virtual private network (VPN).

ExpressRoute

Extend an on-premises network to Azure using Azure ExpressRoute.

ExpressRoute with VPN failover

Extend an on-premises network to Azure using Azure ExpressRoute, with a VPN as a failover connection.

Hub-spoke topology

The hub is a central point of connectivity to your on-premises network. The spokes are VNets that peer with the hub, and can be used to isolate workloads.

Hub-spoke topology with shared services

Deploy a hub-spoke topology that includes shared services, including Active Directory services and a network virtual appliance (NVA). Shared services can be consumed by all spokes.

Choose a solution for connecting an on-premises network to Azure

8/14/2017 • 2 min to read • Edit Online

This article compares options for connecting an on-premises network to an Azure Virtual Network (VNet). We provide a reference architecture and a deployable solution for each option.

VPN connection

Use a virtual private network (VPN) to connect your on-premises network with an Azure VNet through an IPSec VPN tunnel.

This architecture is suitable for hybrid applications where the traffic between on-premises hardware and the cloud is likely to be light, or you are willing to trade slightly extended latency for the flexibility and processing power of the cloud.

Benefits

- Simple to configure.

Challenges

- Requires an on-premises VPN device.
- Although Microsoft guarantees 99.9% availability for each VPN Gateway, this SLA only covers the VPN gateway, and not your network connection to the gateway.
- A VPN connection over Azure VPN Gateway currently supports a maximum of 200 Mbps bandwidth. You may need to partition your Azure virtual network across multiple VPN connections if you expect to exceed this throughput.

Read more...

Azure ExpressRoute connection

ExpressRoute connections use a private, dedicated connection through a third-party connectivity provider. The private connection extends your on-premises network into Azure.

This architecture is suitable for hybrid applications running large-scale, mission-critical workloads that require a high degree of scalability.

Benefits

- Much higher bandwidth available; up to 10 Gbps depending on the connectivity provider.
- Supports dynamic scaling of bandwidth to help reduce costs during periods of lower demand. However, not all connectivity providers have this option.
- May allow your organization direct access to national clouds, depending on the connectivity provider.
- 99.9% availability SLA across the entire connection.

Challenges

- Can be complex to set up. Creating an ExpressRoute connection requires working with a third-party connectivity provider. The provider is responsible for provisioning the network connection.
- Requires high-bandwidth routers on-premises.

Read more...

ExpressRoute with VPN failover

This options combines the previous two, using ExpressRoute in normal conditions, but failing over to a VPN connection if there is a loss of connectivity in the ExpressRoute circuit.

This architecture is suitable for hybrid applications that need the higher bandwidth of ExpressRoute, and also require highly available network connectivity.

Benefits

- High availability if the ExpressRoute circuit fails, although the fallback connection is on a lower bandwidth network.

Challenges

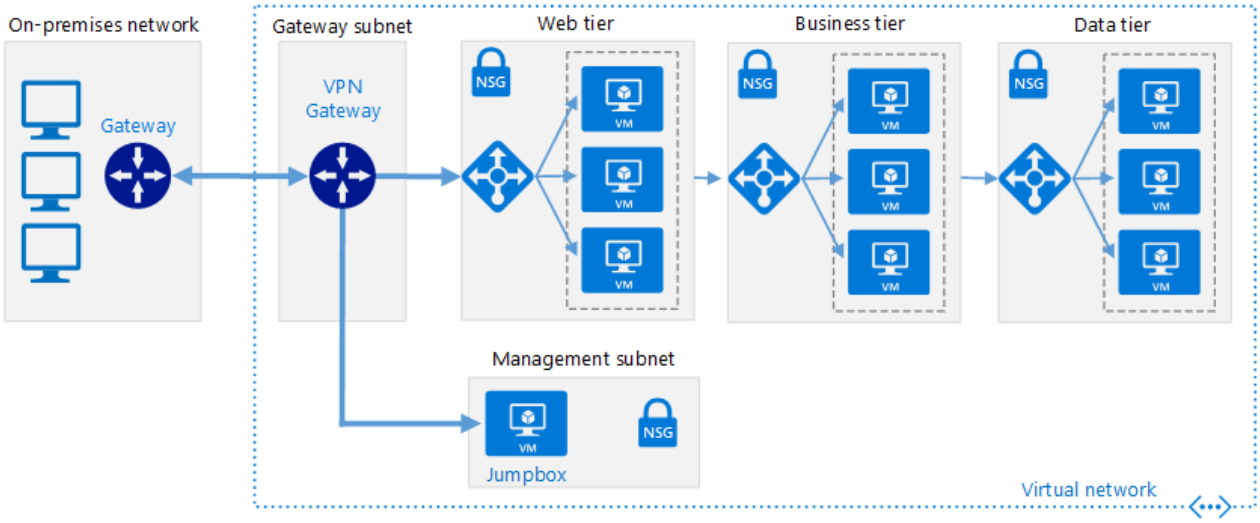
- Complex to configure. You need to set up both a VPN connection and an ExpressRoute circuit.
- Requires redundant hardware (VPN appliances), and a redundant Azure VPN Gateway connection for which you pay charges.

Read more...

Connect an on-premises network to Azure using a VPN gateway

4/11/2018 • 17 min to read • Edit Online

This reference architecture shows how to extend an on-premises network to Azure, using a site-to-site virtual private network (VPN). Traffic flows between the on-premises network and an Azure Virtual Network (VNet) through an IPSec VPN tunnel. **Deploy this solution**.



Download a Visio file of this architecture.

Architecture

The architecture consists of the following components.

- **On-premises network**. A private local-area network running within an organization.
- **VPN appliance**. A device or service that provides external connectivity to the on-premises network. The VPN appliance may be a hardware device, or it can be a software solution such as the Routing and Remote Access Service (RRAS) in Windows Server 2012. For a list of supported VPN appliances and information on configuring them to connect to an Azure VPN gateway, see the instructions for the selected device in the article About VPN devices for Site-to-Site VPN Gateway connections.
- **Virtual network (VNet)**. The cloud application and the components for the Azure VPN gateway reside in the same VNet.
- **Azure VPN gateway**. The VPN gateway service enables you to connect the VNet to the on-premises network through a VPN appliance. For more information, see Connect an on-premises network to a Microsoft Azure virtual network. The VPN gateway includes the following elements:
  - **Virtual network gateway**. A resource that provides a virtual VPN appliance for the VNet. It is responsible for routing traffic from the on-premises network to the VNet.
  - **Local network gateway**. An abstraction of the on-premises VPN appliance. Network traffic from the cloud application to the on-premises network is routed through this gateway.
  - **Connection**. The connection has properties that specify the connection type (IPSec) and the key shared with the on-premises VPN appliance to encrypt traffic.
  - **Gateway subnet**. The virtual network gateway is held in its own subnet, which is subject to various

requirements, described in the Recommendations section below.

- **Cloud application**. The application hosted in Azure. It might include multiple tiers, with multiple subnets connected through Azure load balancers. For more information about the application infrastructure, see Running Windows VM workloads and Running Linux VM workloads.
- **Internal load balancer**. Network traffic from the VPN gateway is routed to the cloud application through an internal load balancer. The load balancer is located in the front-end subnet of the application.

Recommendations

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

VNet and gateway subnet

Create an Azure VNet with an address space large enough for all of your required resources. Ensure that the VNet address space has sufficient room for growth if additional VMs are likely to be needed in the future. The address space of the VNet must not overlap with the on-premises network. For example, the diagram above uses the address space 10.20.0.0/16 for the VNet.

Create a subnet named *GatewaySubnet*, with an address range of /27. This subnet is required by the virtual network gateway. Allocating 32 addresses to this subnet will help to prevent reaching gateway size limitations in the future. Also, avoid placing this subnet in the middle of the address space. A good practice is to set the address space for the gateway subnet at the upper end of the VNet address space. The example shown in the diagram uses 10.20.255.224/27. Here is a quick procedure to calculate the CIDR:

1. Set the variable bits in the address space of the VNet to 1, up to the bits being used by the gateway subnet, then set the remaining bits to 0.
2. Convert the resulting bits to decimal and express it as an address space with the prefix length set to the size of the gateway subnet.

For example, for a VNet with an IP address range of 10.20.0.0/16, applying step #1 above becomes 10.20.0b11111111.0b11100000. Converting that to decimal and expressing it as an address space yields 10.20.255.224/27.

> **WARNING**
> Do not deploy any VMs to the gateway subnet. Also, do not assign an NSG to this subnet, as it will cause the gateway to stop functioning.

Virtual network gateway

Allocate a public IP address for the virtual network gateway.

Create the virtual network gateway in the gateway subnet and assign it the newly allocated public IP address. Use the gateway type that most closely matches your requirements and that is enabled by your VPN appliance:

- Create a policy-based gateway if you need to closely control how requests are routed based on policy criteria such as address prefixes. Policy-based gateways use static routing, and only work with site-to-site connections.
- Create a route-based gateway if you connect to the on-premises network using RRAS, support multi-site or cross-region connections, or implement VNet-to-VNet connections (including routes that traverse multiple VNets). Route-based gateways use dynamic routing to direct traffic between networks. They can tolerate failures in the network path better than static routes because they can try alternative routes. Route-based gateways can also reduce the management overhead because routes might not need to be updated manually when network addresses change.

For a list of supported VPN appliances, see About VPN devices for Site-to-Site VPN Gateway connections.

> **NOTE**
> After the gateway has been created, you cannot change between gateway types without deleting and re-creating the gateway.

Select the Azure VPN gateway SKU that most closely matches your throughput requirements. Azure VPN gateway is available in three SKUs shown in the following table.

SKU	VPN THROUGHPUT	MAX IPSEC TUNNELS
Basic	100 Mbps	10
Standard	100 Mbps	10
High Performance	200 Mbps	30

> **NOTE**
> The Basic SKU is not compatible with Azure ExpressRoute. You can change the SKU after the gateway has been created.

You are charged based on the amount of time that the gateway is provisioned and available. See VPN Gateway Pricing.

Create routing rules for the gateway subnet that direct incoming application traffic from the gateway to the internal load balancer, rather than allowing requests to pass directly to the application VMs.

On-premises network connection

Create a local network gateway. Specify the public IP address of the on-premises VPN appliance, and the address space of the on-premises network. Note that the on-premises VPN appliance must have a public IP address that can be accessed by the local network gateway in Azure VPN Gateway. The VPN device cannot be located behind a network address translation (NAT) device.

Create a site-to-site connection for the virtual network gateway and the local network gateway. Select the site-to-site (IPSec) connection type, and specify the shared key. Site-to-site encryption with the Azure VPN gateway is based on the IPSec protocol, using preshared keys for authentication. You specify the key when you create the Azure VPN gateway. You must configure the VPN appliance running on-premises with the same key. Other authentication mechanisms are not currently supported.

Ensure that the on-premises routing infrastructure is configured to forward requests intended for addresses in the Azure VNet to the VPN device.

Open any ports required by the cloud application in the on-premises network.

Test the connection to verify that:

- The on-premises VPN appliance correctly routes traffic to the cloud application through the Azure VPN gateway.
- The VNet correctly routes traffic back to the on-premises network.
- Prohibited traffic in both directions is blocked correctly.

Scalability considerations

You can achieve limited vertical scalability by moving from the Basic or Standard VPN Gateway SKUs to the High Performance VPN SKU.

For VNets that expect a large volume of VPN traffic, consider distributing the different workloads into separate smaller VNets and configuring a VPN gateway for each of them.

You can partition the VNet either horizontally or vertically. To partition horizontally, move some VM instances from each tier into subnets of the new VNet. The result is that each VNet has the same structure and functionality. To partition vertically, redesign each tier to divide the functionality into different logical areas (such as handling orders, invoicing, customer account management, and so on). Each functional area can then be placed in its own VNet.

Replicating an on-premises Active Directory domain controller in the VNet, and implementing DNS in the VNet, can help to reduce some of the security-related and administrative traffic flowing from on-premises to the cloud. For more information, see Extending Active Directory Domain Services (AD DS) to Azure.

Availability considerations

If you need to ensure that the on-premises network remains available to the Azure VPN gateway, implement a failover cluster for the on-premises VPN gateway.

If your organization has multiple on-premises sites, create multi-site connections to one or more Azure VNets. This approach requires dynamic (route-based) routing, so make sure that the on-premises VPN gateway supports this feature.

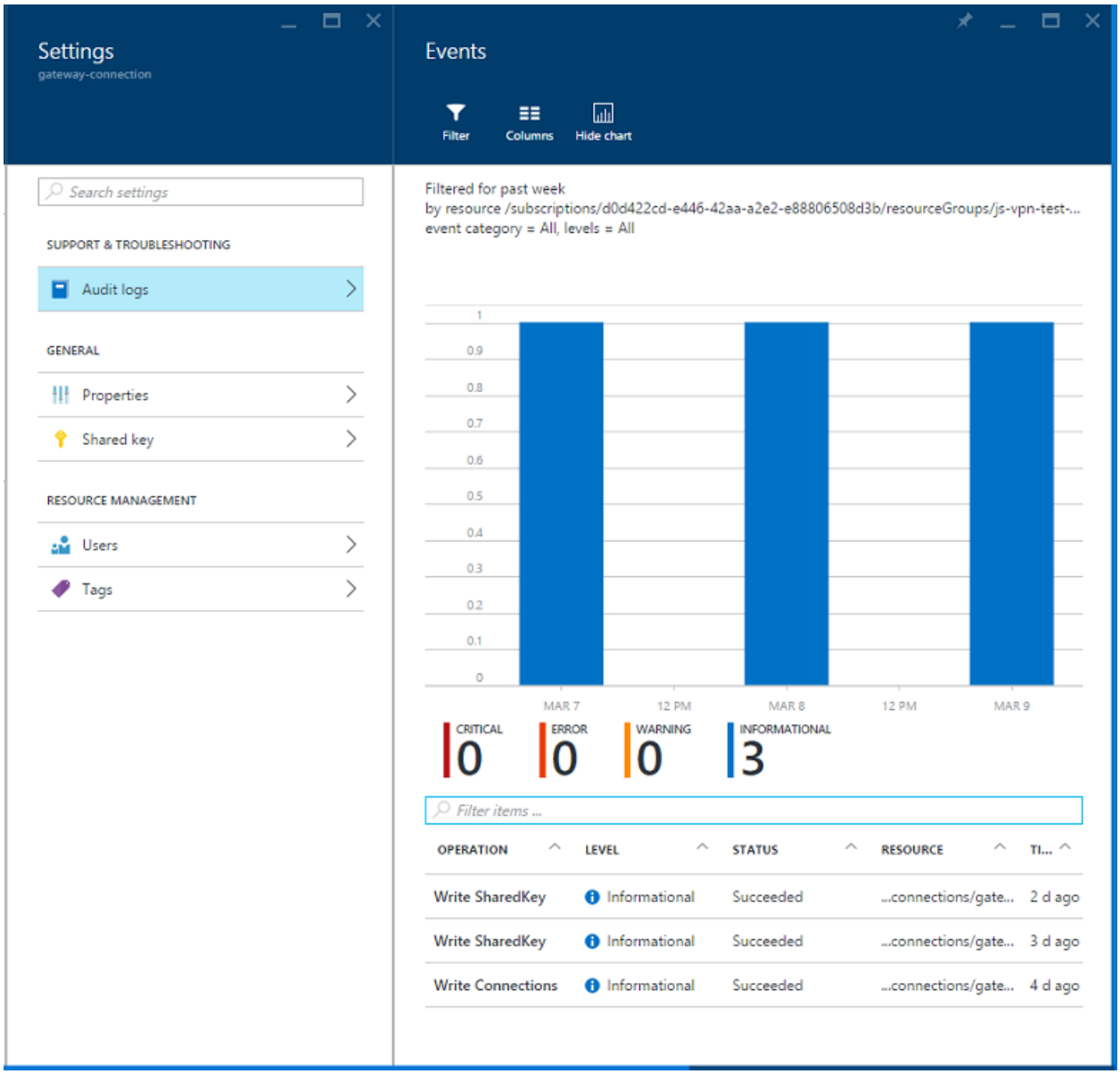
For details about service level agreements, see SLA for VPN Gateway.

Manageability considerations

Monitor diagnostic information from on-premises VPN appliances. This process depends on the features provided by the VPN appliance. For example, if you are using the Routing and Remote Access Service on Windows Server 2012, RRAS logging.

Use Azure VPN gateway diagnostics to capture information about connectivity issues. These logs can be used to track information such as the source and destinations of connection requests, which protocol was used, and how the connection was established (or why the attempt failed).

Monitor the operational logs of the Azure VPN gateway using the audit logs available in the Azure portal. Separate logs are available for the local network gateway, the Azure network gateway, and the connection. This information can be used to track any changes made to the gateway, and can be useful if a previously functioning gateway stops working for some reason.



Monitor connectivity, and track connectivity failure events. You can use a monitoring package such as Nagios to capture and report this information.

Security considerations

Generate a different shared key for each VPN gateway. Use a strong shared key to help resist brute-force attacks.

> **NOTE**
> Currently, you cannot use Azure Key Vault to preshare keys for the Azure VPN gateway.

Ensure that the on-premises VPN appliance uses an encryption method that is compatible with the Azure VPN gateway. For policy-based routing, the Azure VPN gateway supports the AES256, AES128, and 3DES encryption algorithms. Route-based gateways support AES256 and 3DES.

If your on-premises VPN appliance is on a perimeter network (DMZ) that has a firewall between the perimeter network and the Internet, you might have to configure additional firewall rules to allow the site-to-site VPN connection.

If the application in the VNet sends data to the Internet, consider implementing forced tunneling to route all Internet-bound traffic through the on-premises network. This approach enables you to audit outgoing requests made by the application from the on-premises infrastructure.

> **NOTE**
> Forced tunneling can impact connectivity to Azure services (the Storage Service, for example) and the Windows license manager.

Troubleshooting

For general information on troubleshooting common VPN-related errors, see Troubleshooting common VPN related errors.

The following recommendations are useful for determining if your on-premises VPN appliance is functioning correctly.

- **Check any log files generated by the VPN appliance for errors or failures.**

  This will help you determine if the VPN appliance is functioning correctly. The location of this information will vary according to your appliance. For example, if you are using RRAS on Windows Server 2012, you can use the following PowerShell command to display error event information for the RRAS service:

  ```
  Get-EventLog -LogName System -EntryType Error -Source RemoteAccess | Format-List -Property *
  ```

  The *Message* property of each entry provides a description of the error. Some common examples are:

  ```
    - Inability to connect, possibly due to an incorrect IP address specified for the Azure VPN gateway
  in the RRAS VPN network interface configuration.

    ```
    EventID   : 20111
    MachineName   : on-prem-vm
    Data   : {41, 3, 0, 0}
    Index   : 14231
    Category   : (0)
    CategoryNumber   : 0
    EntryType   : Error
    Message   : RoutingDomainID- {00000000-0000-0000-0000-000000000000}: A demand dial
  connection to the remote
                         interface AzureGateway on port VPN2-4 was successfully initiated but failed to
  complete
                         successfully because of the following error: The network connection between
  your computer and
                         the VPN server could not be established because the remote server is not
  responding. This could
                         be because one of the network devices (for example, firewalls, NAT, routers, and
  so on) between your computer
                         and the remote server is not configured to allow VPN connections. Please contact
  your
                         Administrator or your service provider to determine which device may be causing
  the problem.
    Source   : RemoteAccess
    ReplacementStrings : {{00000000-0000-0000-0000-000000000000}, AzureGateway, VPN2-4, The network
  connection between
                         your computer and the VPN server could not be established because the remote
  server is not
                         responding. This could be because one of the network devices (for example,
  firewalls, NAT, routers, and so on)
                         between your computer and the remote server is not configured to allow VPN
  connections. Please
                         contact your Administrator or your service provider to determine which device
  may be causing the
                         problem.}
    InstanceId   : 20111
    TimeGenerated   : 3/18/2016 1:26:02 PM
  ```
  ```

```
  TimeGenerated      : 3/18/2016 1:26:02 PM
  TimeWritten        : 3/18/2016 1:26:02 PM
  UserName           :
  Site               :
  Container          :
  ```

  - The wrong shared key being specified in the RRAS VPN network interface configuration.

  ```
  EventID            : 20111
  MachineName        : on-prem-vm
  Data               : {233, 53, 0, 0}
  Index              : 14245
  Category           : (0)
  CategoryNumber     : 0
  EntryType          : Error
  Message            : RoutingDomainID- {00000000-0000-0000-0000-000000000000}: A demand dial
connection to the remote
                       interface AzureGateway on port VPN2-4 was successfully initiated but failed to
complete
                       successfully because of the  following error: Internet key exchange (IKE)
authentication credentials are unacceptable.

  Source             : RemoteAccess
  ReplacementStrings : {{00000000-0000-0000-0000-000000000000}, AzureGateway, VPN2-4, IKE
authentication credentials are
                       unacceptable.
                       }
  InstanceId         : 20111
  TimeGenerated      : 3/18/2016 1:34:22 PM
  TimeWritten        : 3/18/2016 1:34:22 PM
  UserName           :
  Site               :
  Container          :
  ```
```

You can also obtain event log information about attempts to connect through the RRAS service using the following PowerShell command:

```
Get-EventLog -LogName Application -Source RasClient | Format-List -Property *
```

In the event of a failure to connect, this log will contain errors that look similar to the following:

```
EventID            : 20227
MachineName        : on-prem-vm
Data               : {}
Index              : 4203
Category           : (0)
CategoryNumber     : 0
EntryType          : Error
Message            : CoId={B4000371-A67F-452F-AA4C-3125AA9CFC78}: The user SYSTEM dialed a connection
named
                     AzureGateway that has failed. The error code returned on failure is 809.
Source             : RasClient
ReplacementStrings : {{B4000371-A67F-452F-AA4C-3125AA9CFC78}, SYSTEM, AzureGateway, 809}
InstanceId         : 20227
TimeGenerated      : 3/18/2016 1:29:21 PM
TimeWritten        : 3/18/2016 1:29:21 PM
UserName           :
Site               :
Container          :
```

- **Verify connectivity and routing across the VPN gateway.**
```

The VPN appliance may not be correctly routing traffic through the Azure VPN Gateway. Use a tool such as PsPing to verify connectivity and routing across the VPN gateway. For example, to test connectivity from an on-premises machine to a web server located on the VNet, run the following command (replacing `<<web-server-address>>` with the address of the web server):

```
PsPing -t <<web-server-address>>:80
```

If the on-premises machine can route traffic to the web server, you should see output similar to the following:

```
D:\PSTools>psping -t 10.20.0.5:80

PsPing v2.01 - PsPing - ping, latency, bandwidth measurement utility
Copyright (C) 2012-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

TCP connect to 10.20.0.5:80:
Infinite iterations (warmup 1) connecting test:
Connecting to 10.20.0.5:80 (warmup): 6.21ms
Connecting to 10.20.0.5:80: 3.79ms
Connecting to 10.20.0.5:80: 3.44ms
Connecting to 10.20.0.5:80: 4.81ms

  Sent = 3, Received = 3, Lost = 0 (0% loss),
  Minimum = 3.44ms, Maximum = 4.81ms, Average = 4.01ms
```

If the on-premises machine cannot communicate with the specified destination, you will see messages like this:

```
D:\PSTools>psping -t 10.20.1.6:80

PsPing v2.01 - PsPing - ping, latency, bandwidth measurement utility
Copyright (C) 2012-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

TCP connect to 10.20.1.6:80:
Infinite iterations (warmup 1) connecting test:
Connecting to 10.20.1.6:80 (warmup): This operation returned because the timeout period expired.
Connecting to 10.20.1.6:80: This operation returned because the timeout period expired.
Connecting to 10.20.1.6:80: This operation returned because the timeout period expired.
Connecting to 10.20.1.6:80: This operation returned because the timeout period expired.
Connecting to 10.20.1.6:80:
  Sent = 3, Received = 0, Lost = 3 (100% loss),
  Minimum = 0.00ms, Maximum = 0.00ms, Average = 0.00ms
```

- **Verify that the on-premises firewall allows VPN traffic to pass and that the correct ports are opened.**
- **Verify that the on-premises VPN appliance uses an encryption method that is compatible with the Azure VPN gateway.** For policy-based routing, the Azure VPN gateway supports the AES256, AES128, and 3DES encryption algorithms. Route-based gateways support AES256 and 3DES.

The following recommendations are useful for determining if there is a problem with the Azure VPN gateway:

- **Examine Azure VPN gateway diagnostic logs for potential issues.**
- **Verify that the Azure VPN gateway and on-premises VPN appliance are configured with the same shared authentication key.**

  You can view the shared key stored by the Azure VPN gateway using the following Azure CLI command:

```
azure network vpn-connection shared-key show <<resource-group>> <<vpn-connection-name>>
```

Use the command appropriate for your on-premises VPN appliance to show the shared key configured for that appliance.

Verify that the *GatewaySubnet* subnet holding the Azure VPN gateway is not associated with an NSG.

You can view the subnet details using the following Azure CLI command:

```
azure network vnet subnet show -g <<resource-group>> -e <<vnet-name>> -n GatewaySubnet
```

Ensure there is no data field named *Network Security Group id*. The following example shows the results for an instance of the *GatewaySubnet* that has an assigned NSG (*VPN-Gateway-Group*). This can prevent the gateway from working correctly if there are any rules defined for this NSG.

```
C:\>azure network vnet subnet show -g profx-prod-rg -e profx-vnet -n GatewaySubnet
    info:    Executing command network vnet subnet show
    + Looking up virtual network "profx-vnet"
    + Looking up the subnet "GatewaySubnet"
    data:    Id                              : /subscriptions/########-####-####-####-
############/resourceGroups/profx-prod-rg/providers/Microsoft.Network/virtualNetworks/profx-
vnet/subnets/GatewaySubnet
    data:    Name                            : GatewaySubnet
    data:    Provisioning state              : Succeeded
    data:    Address prefix                  : 10.20.3.0/27
    data:    Network Security Group id       : /subscriptions/########-####-####-####-
############/resourceGroups/profx-prod-rg/providers/Microsoft.Network/networkSecurityGroups/VPN-
Gateway-Group
    info:    network vnet subnet show command OK
```

- **Verify that the virtual machines in the Azure VNet are configured to permit traffic coming in from outside the VNet.**

  Check any NSG rules associated with subnets containing these virtual machines. You can view all NSG rules using the following Azure CLI command:

  ```
  azure network nsg show -g <<resource-group>> -n <<nsg-name>>
  ```

- **Verify that the Azure VPN gateway is connected.**

  You can use the following Azure PowerShell command to check the current status of the Azure VPN connection. The `<<connection-name>>` parameter is the name of the Azure VPN connection that links the virtual network gateway and the local gateway.

  ```
  Get-AzureRmVirtualNetworkGatewayConnection -Name <<connection-name>> - ResourceGroupName <<resource-
  group>>
  ```

  The following snippets highlight the output generated if the gateway is connected (the first example), and disconnected (the second example):

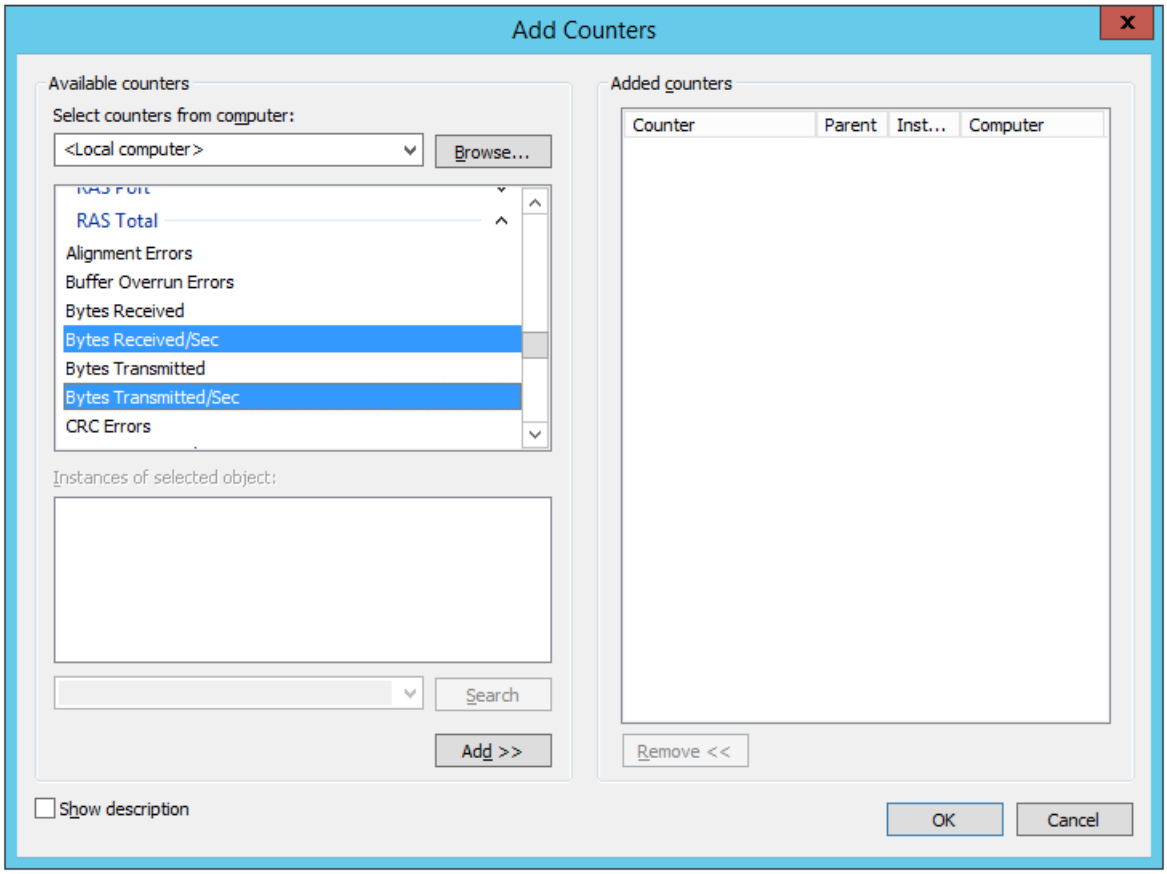
```
PS C:\> Get-AzureRmVirtualNetworkGatewayConnection -Name profx-gateway-connection -ResourceGroupName
profx-prod-rg

AuthorizationKey           :
VirtualNetworkGateway1     : Microsoft.Azure.Commands.Network.Models.PSVirtualNetworkGateway
VirtualNetworkGateway2     :
LocalNetworkGateway2       : Microsoft.Azure.Commands.Network.Models.PSLocalNetworkGateway
Peer                       :
ConnectionType             : IPsec
RoutingWeight              : 0
SharedKey                  : ####################################
ConnectionStatus           : Connected
EgressBytesTransferred     : 55254803
IngressBytesTransferred    : 32227221
ProvisioningState          : Succeeded
...
```

```
PS C:\> Get-AzureRmVirtualNetworkGatewayConnection -Name profx-gateway-connection2 -ResourceGroupName
profx-prod-rg

AuthorizationKey           :
VirtualNetworkGateway1     : Microsoft.Azure.Commands.Network.Models.PSVirtualNetworkGateway
VirtualNetworkGateway2     :
LocalNetworkGateway2       : Microsoft.Azure.Commands.Network.Models.PSLocalNetworkGateway
Peer                       :
ConnectionType             : IPsec
RoutingWeight              : 0
SharedKey                  : ####################################
ConnectionStatus           : NotConnected
EgressBytesTransferred     : 0
IngressBytesTransferred    : 0
ProvisioningState          : Succeeded
...
```

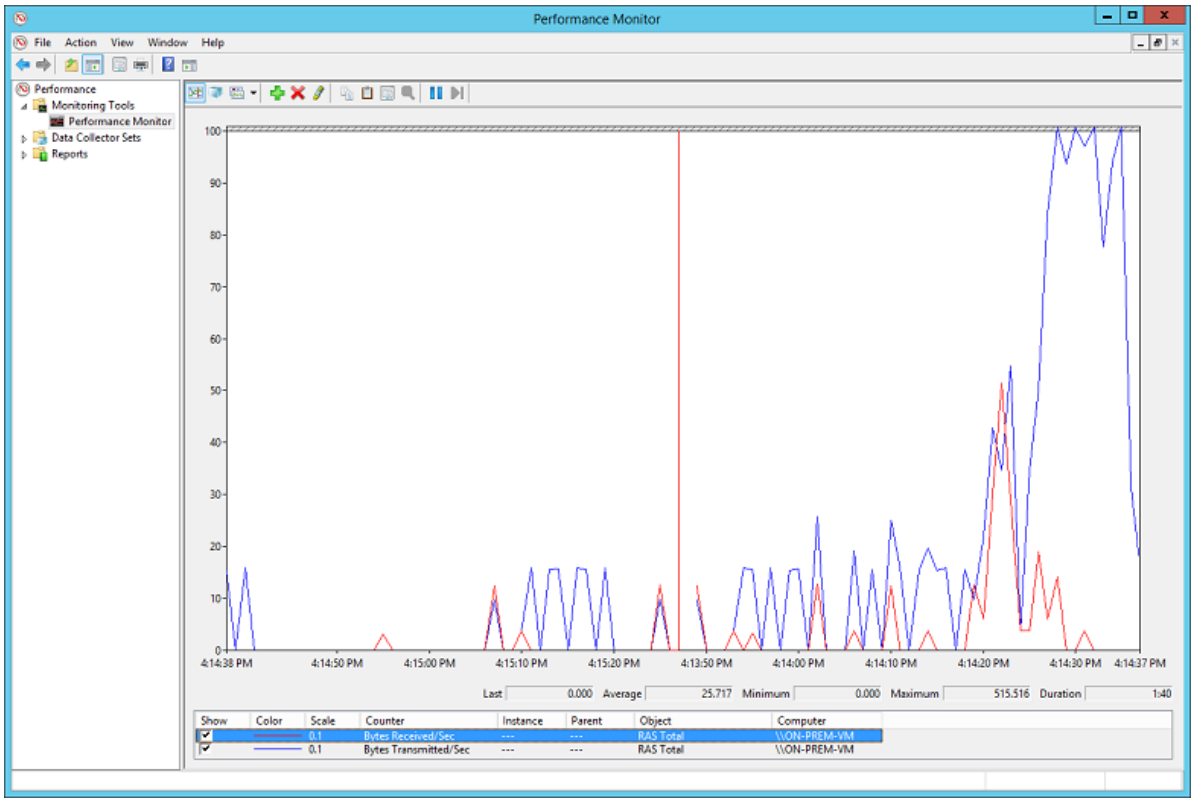
The following recommendations are useful for determining if there is an issue with Host VM configuration, network bandwidth utilization, or application performance:

- **Verify that the firewall in the guest operating system running on the Azure VMs in the subnet is configured correctly to allow permitted traffic from the on-premises IP ranges.**
- **Verify that the volume of traffic is not close to the limit of the bandwidth available to the Azure VPN gateway.**

  How to verify this depends on the VPN appliance running on-premises. For example, if you are using RRAS on Windows Server 2012, you can use Performance Monitor to track the volume of data being received and transmitted over the VPN connection. Using the *RAS Total* object, select the *Bytes Received/Sec* and *Bytes Transmitted/Sec* counters:



You should compare the results with the bandwidth available to the VPN gateway (100 Mbps for the Basic and Standard SKUs, and 200 Mbps for the High Performance SKU):



- **Verify that you have deployed the right number and size of VMs for your application load.**

  Determine if any of the virtual machines in the Azure VNet are running slowly. If so, they may be overloaded, there may be too few to handle the load, or the load-balancers may not be configured correctly. To determine this, capture and analyze diagnostic information. You can examine the results using the Azure portal, but many third-party tools are also available that can provide detailed insights into the performance data.

- **Verify that the application is making efficient use of cloud resources.**

  Instrument application code running on each VM to determine whether applications are making the best use of resources. You can use tools such as Application Insights.

Deploy the solution

Prequisites. You must have an existing on-premises infrastructure already configured with a suitable network appliance.

To deploy the solution, perform the following steps.

1. Click the button below:

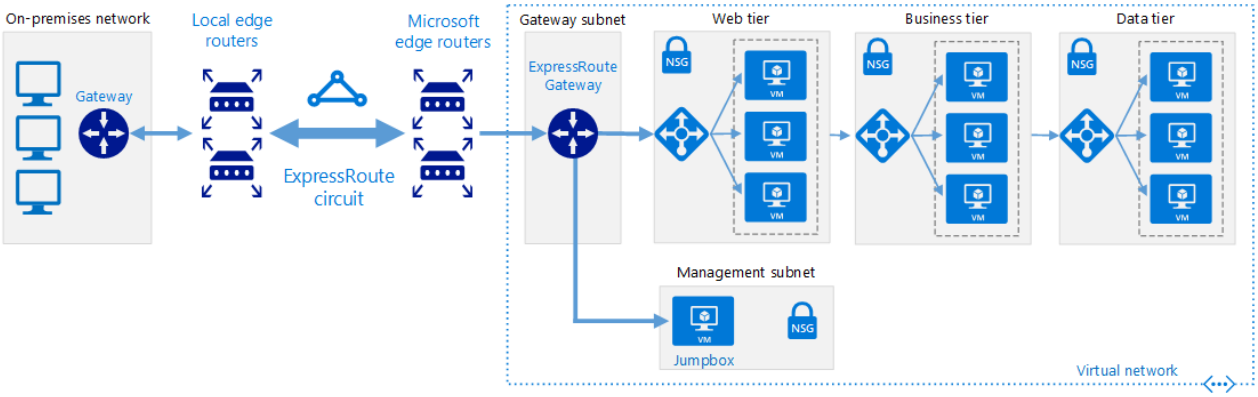
   

2. Wait for the link to open in the Azure portal, then follow these steps:
   - The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-hybrid-vpn-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
3. Wait for the deployment to complete.

Connect an on-premises network to Azure using ExpressRoute

4/11/2018 • 12 min to read • Edit Online

This reference architecture shows how to connect an on-premises network to virtual networks on Azure, using Azure ExpressRoute. ExpressRoute connections use a private, dedicated connection through a third-party connectivity provider. The private connection extends your on-premises network into Azure. **Deploy this solution**.



Download a Visio file of this architecture.

Architecture

The architecture consists of the following components.

- **On-premises corporate network**. A private local-area network running within an organization.
- **ExpressRoute circuit**. A layer 2 or layer 3 circuit supplied by the connectivity provider that joins the on-premises network with Azure through the edge routers. The circuit uses the hardware infrastructure managed by the connectivity provider.
- **Local edge routers**. Routers that connect the on-premises network to the circuit managed by the provider. Depending on how your connection is provisioned, you may need to provide the public IP addresses used by the routers.
- **Microsoft edge routers**. Two routers in an active-active highly available configuration. These routers enable a connectivity provider to connect their circuits directly to their datacenter. Depending on how your connection is provisioned, you may need to provide the public IP addresses used by the routers.
- **Azure virtual networks (VNets)**. Each VNet resides in a single Azure region, and can host multiple application tiers. Application tiers can be segmented using subnets in each VNet.
- **Azure public services**. Azure services that can be used within a hybrid application. These services are also available over the Internet, but accessing them using an ExpressRoute circuit provides low latency and more predictable performance, because traffic does not go through the Internet. Connections are performed using public peering, with addresses that are either owned by your organization or supplied by your connectivity provider.
- **Office 365 services**. The publicly available Office 365 applications and services provided by Microsoft. Connections are performed using Microsoft peering, with addresses that are either owned by your organization or supplied by your connectivity provider. You can also connect directly to Microsoft CRM

Online through Microsoft peering.

- **Connectivity providers** (not shown). Companies that provide a connection either using layer 2 or layer 3 connectivity between your datacenter and an Azure datacenter.

Recommendations

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

Connectivity providers

Select a suitable ExpressRoute connectivity provider for your location. To get a list of connectivity providers available at your location, use the following Azure PowerShell command:

```
Get-AzureRmExpressRouteServiceProvider
```

ExpressRoute connectivity providers connect your datacenter to Microsoft in the following ways:

- **Co-located at a cloud exchange**. If you're co-located in a facility with a cloud exchange, you can order virtual cross-connections to Azure through the co-location provider's Ethernet exchange. Co-location providers can offer either layer 2 cross-connections, or managed layer 3 cross-connections between your infrastructure in the co-location facility and Azure.
- **Point-to-point Ethernet connections**. You can connect your on-premises datacenters/offices to Azure through point-to-point Ethernet links. Point-to-point Ethernet providers can offer layer 2 connections, or managed layer 3 connections between your site and Azure.
- **Any-to-any (IPVPN) networks**. You can integrate your wide area network (WAN) with Azure. Internet protocol virtual private network (IPVPN) providers (typically a multiprotocol label switching VPN) offer any-to-any connectivity between your branch offices and datacenters. Azure can be interconnected to your WAN to make it look just like any other branch office. WAN providers typically offer managed layer 3 connectivity.

For more information about connectivity providers, see the ExpressRoute introduction.

ExpressRoute circuit

Ensure that your organization has met the ExpressRoute prerequisite requirements for connecting to Azure.

If you haven't already done so, add a subnet named `GatewaySubnet` to your Azure VNet and create an ExpressRoute virtual network gateway using the Azure VPN gateway service. For more information about this process, see ExpressRoute workflows for circuit provisioning and circuit states.

Create an ExpressRoute circuit as follows:

1. Run the following PowerShell command:

   ```
   New-AzureRmExpressRouteCircuit -Name <<circuit-name>> -ResourceGroupName <<resource-group>> -Location
   <<location>> -SkuTier <<sku-tier>> -SkuFamily <<sku-family>> -ServiceProviderName <<service-provider-
   name>> -PeeringLocation <<peering-location>> -BandwidthInMbps <<bandwidth-in-mbps>>
   ```

2. Send the `ServiceKey` for the new circuit to the service provider.

3. Wait for the provider to provision the circuit. To verify the provisioning state of a circuit, run the following PowerShell command:

   ```
   Get-AzureRmExpressRouteCircuit -Name <<circuit-name>> -ResourceGroupName <<resource-group>>
   ```

   The `Provisioning state` field in the `Service Provider` section of the output will change from

`NotProvisioned` to `Provisioned` when the circuit is ready.

> **NOTE**
> If you're using a layer 3 connection, the provider should configure and manage routing for you. You provide the information necessary to enable the provider to implement the appropriate routes.

4. If you're using a layer 2 connection:

   a. Reserve two /30 subnets composed of valid public IP addresses for each type of peering you want to implement. These /30 subnets will be used to provide IP addresses for the routers used for the circuit. If you are implementing private, public, and Microsoft peering, you'll need 6 /30 subnets with valid public IP addresses.

   b. Configure routing for the ExpressRoute circuit. Run the following PowerShell commands for each type of peering you want to configure (private, public, and Microsoft). For more information, see Create and modify routing for an ExpressRoute circuit.

   ```
   Set-AzureRmExpressRouteCircuitPeeringConfig -Name <<peering-name>> -Circuit <<circuit-name>> -
   PeeringType <<peering-type>> -PeerASN <<peer-asn>> -PrimaryPeerAddressPrefix <<primary-peer-
   address-prefix>> -SecondaryPeerAddressPrefix <<secondary-peer-address-prefix>> -VlanId <<vlan-
   id>>

   Set-AzureRmExpressRouteCircuit -ExpressRouteCircuit <<circuit-name>>
   ```

   c. Reserve another pool of valid public IP addresses to use for network address translation (NAT) for public and Microsoft peering. It is recommended to have a different pool for each peering. Specify the pool to your connectivity provider, so they can configure border gateway protocol (BGP) advertisements for those ranges.

5. Run the following PowerShell commands to link your private VNet(s) to the ExpressRoute circuit. For more information,see Link a virtual network to an ExpressRoute circuit.

   ```
   $circuit = Get-AzureRmExpressRouteCircuit -Name <<circuit-name>> -ResourceGroupName <<resource-group>>
   $gw = Get-AzureRmVirtualNetworkGateway -Name <<gateway-name>> -ResourceGroupName <<resource-group>>
   New-AzureRmVirtualNetworkGatewayConnection -Name <<connection-name>> -ResourceGroupName <<resource-
   group>> -Location <<location> -VirtualNetworkGateway1 $gw -PeerId $circuit.Id -ConnectionType
   ExpressRoute
   ```

You can connect multiple VNets located in different regions to the same ExpressRoute circuit, as long as all VNets and the ExpressRoute circuit are located within the same geopolitical region.

Troubleshooting

If a previously functioning ExpressRoute circuit now fails to connect, in the absence of any configuration changes on-premises or within your private VNet, you may need to contact the connectivity provider and work with them to correct the issue. Use the following Powershell commands to verify that the ExpressRoute circuit has been provisioned:

```
Get-AzureRmExpressRouteCircuit -Name <<circuit-name>> -ResourceGroupName <<resource-group>>
```

The output of this command shows several properties for your circuit, including `ProvisioningState`, `CircuitProvisioningState`, and `ServiceProviderProvisioningState` as shown below.

```
ProvisioningState                : Succeeded
Sku                              : {
                                     "Name": "Standard_MeteredData",
                                     "Tier": "Standard",
                                     "Family": "MeteredData"
                                   }
CircuitProvisioningState         : Enabled
ServiceProviderProvisioningState : NotProvisioned
```

If the `ProvisioningState` is not set to `Succeeded` after you tried to create a new circuit, remove the circuit by using the command below and try to create it again.

```
Remove-AzureRmExpressRouteCircuit -Name <<circuit-name>> -ResourceGroupName <<resource-group>>
```

If your provider had already provisioned the circuit, and the `ProvisioningState` is set to `Failed`, or the `CircuitProvisioningState` is not `Enabled`, contact your provider for further assistance.

Scalability considerations

ExpressRoute circuits provide a high bandwidth path between networks. Generally, the higher the bandwidth the greater the cost.

ExpressRoute offers two pricing plans to customers, a metered plan and an unlimited data plan. Charges vary according to circuit bandwidth. Available bandwidth will likely vary from provider to provider. Use the `Get-AzureRmExpressRouteServiceProvider` cmdlet to see the providers available in your region and the bandwidths that they offer.

A single ExpressRoute circuit can support a certain number of peerings and VNet links. See ExpressRoute limits for more information.

For an extra charge, the ExpressRoute Premium add-on provides some additional capability:

- Increased route limits for public and private peering.
- Increased number of VNet links per ExpressRoute circuit.
- Global connectivity for services.

See ExpressRoute pricing for details.

ExpressRoute circuits are designed to allow temporary network bursts up to two times the bandwidth limit that you procured for no additional cost. This is achieved by using redundant links. However, not all connectivity providers support this feature. Verify that your connectivity provider enables this feature before depending on it.

Although some providers allow you to change your bandwidth, make sure you pick an initial bandwidth that surpasses your needs and provides room for growth. If you need to increase bandwidth in the future, you are left with two options:

- Increase the bandwidth. You should avoid this option as much as possible, and not all providers allow you to increase bandwidth dynamically. But if a bandwidth increase is needed, check with your provider to verify they support changing ExpressRoute bandwidth properties via Powershell commands. If they do, run the commands below.

  ```
  $ckt = Get-AzureRmExpressRouteCircuit -Name <<circuit-name>> -ResourceGroupName <<resource-group>>
  $ckt.ServiceProviderProperties.BandwidthInMbps = <<bandwidth-in-mbps>>
  Set-AzureRmExpressRouteCircuit -ExpressRouteCircuit $ckt
  ```

  You can increase the bandwidth without loss of connectivity. Downgrading the bandwidth will result in

disruption in connectivity, because you must delete the circuit and recreate it with the new configuration.

- Change your pricing plan and/or upgrade to Premium. To do so, run the following commands. The `Sku.Tier` property can be `Standard` or `Premium`; the `Sku.Name` property can be `MeteredData` or `UnlimitedData`.

```
$ckt = Get-AzureRmExpressRouteCircuit -Name <<circuit-name>> -ResourceGroupName <<resource-group>>

$ckt.Sku.Tier = "Premium"
$ckt.Sku.Family = "MeteredData"
$ckt.Sku.Name = "Premium_MeteredData"

Set-AzureRmExpressRouteCircuit -ExpressRouteCircuit $ckt
```

> **IMPORTANT**
> Make sure the `Sku.Name` property matches the `Sku.Tier` and `Sku.Family`. If you change the family and tier, but not the name, your connection will be disabled.

  You can upgrade the SKU without disruption, but you cannot switch from the unlimited pricing plan to metered. When downgrading the SKU, your bandwidth consumption must remain within the default limit of the standard SKU.

Availability considerations

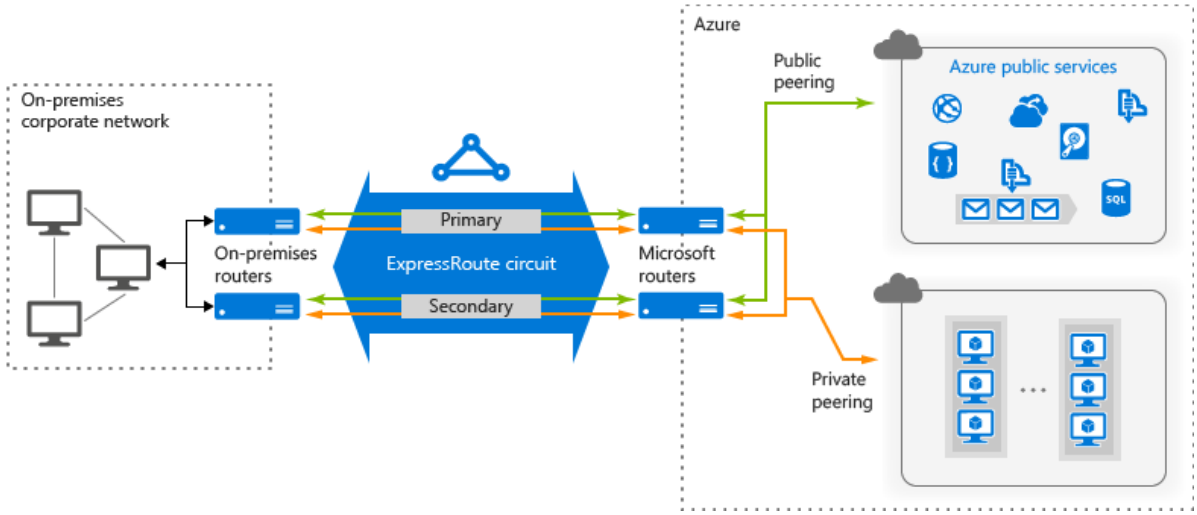
ExpressRoute does not support router redundancy protocols such as hot standby routing protocol (HSRP) and virtual router redundancy protocol (VRRP) to implement high availability. Instead, it uses a redundant pair of BGP sessions per peering. To facilitate highly-available connections to your network, Azure provisions you with two redundant ports on two routers (part of the Microsoft edge) in an active-active configuration.

By default, BGP sessions use an idle timeout value of 60 seconds. If a session times out three times (180 seconds total), the router is marked as unavailable, and all traffic is redirected to the remaining router. This 180-second timeout might be too long for critical applications. If so, you can change your BGP time-out settings on the on-premises router to a smaller value.

You can configure high availability for your Azure connection in different ways, depending on the type of provider you use, and the number of ExpressRoute circuits and virtual network gateway connections you're willing to configure. The following summarizes your availability options:

- If you're using a layer 2 connection, deploy redundant routers in your on-premises network in an active-active configuration. Connect the primary circuit to one router, and the secondary circuit to the other. This will give you a highly available connection at both ends of the connection. This is necessary if you require the ExpressRoute service level agreement (SLA). See SLA for Azure ExpressRoute for details.

  The following diagram shows a configuration with redundant on-premises routers connected to the primary and secondary circuits. Each circuit handles the traffic for a public peering and a private peering (each peering is designated a pair of /30 address spaces, as described in the previous section).



- If you're using a layer 3 connection, verify that it provides redundant BGP sessions that handle availability for you.
- Connect the VNet to multiple ExpressRoute circuits, supplied by different service providers. This strategy provides additional high-availability and disaster recovery capabilities.
- Configure a site-to-site VPN as a failover path for ExpressRoute. For more about this option, see Connect an on-premises network to Azure using ExpressRoute with VPN failover. This option only applies to private peering. For Azure and Office 365 services, the Internet is the only failover path.

Manageability considerations

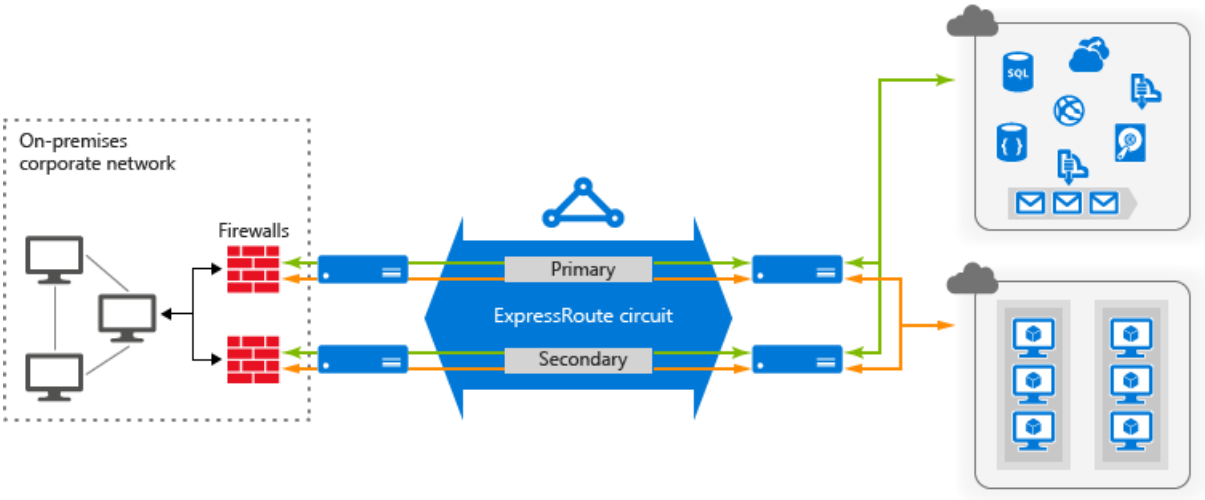
You can use the Azure Connectivity Toolkit (AzureCT) to monitor connectivity between your on-premises datacenter and Azure.

Security considerations

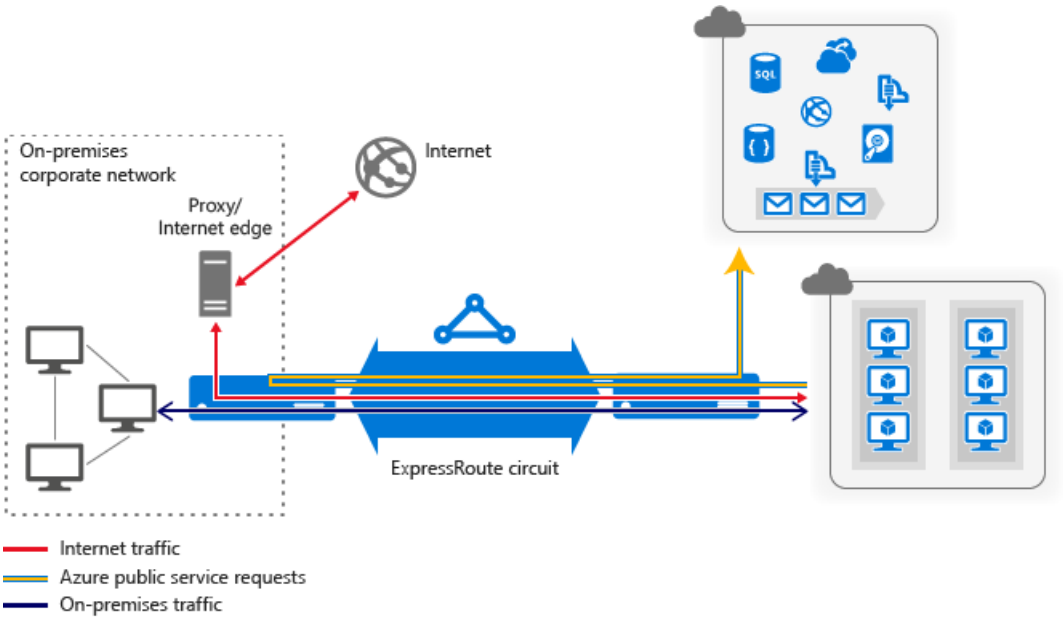
You can configure security options for your Azure connection in different ways, depending on your security concerns and compliance needs.

ExpressRoute operates in layer 3. Threats in the application layer can be prevented by using a network security appliance that restricts traffic to legitimate resources. Additionally, ExpressRoute connections using public peering can only be initiated from on-premises. This prevents a rogue service from accessing and compromising on-premises data from the Internet.

To maximize security, add network security appliances between the on-premises network and the provider edge routers. This will help to restrict the inflow of unauthorized traffic from the VNet:



For auditing or compliance purposes, it may be necessary to prohibit direct access from components running in the VNet to the Internet and implement forced tunneling. In this situation, Internet traffic should be redirected back through a proxy running on-premises where it can be audited. The proxy can be configured to block unauthorized traffic flowing out, and filter potentially malicious inbound traffic.



To maximize security, do not enable a public IP address for your VMs, and use NSGs to ensure that these VMs aren't publicly accessible. VMs should only be available using the internal IP address. These addresses can be made accessible through the ExpressRoute network, enabling on-premises DevOps staff to perform configuration or maintenance.

If you must expose management endpoints for VMs to an external network, use NSGs or access control lists to restrict the visibility of these ports to a whitelist of IP addresses or networks.

> **NOTE**
>
> By default, Azure VMs deployed through the Azure portal include a public IP address that provides login access.

Deploy the solution

Prequisites. You must have an existing on-premises infrastructure already configured with a suitable network appliance.

To deploy the solution, perform the following steps.

1. Click the button below:

   Deploy to Azure

2. Wait for the link to open in the Azure portal, then follow these steps:
   - The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-hybrid-er-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
3. Wait for the deployment to complete.
4. Click the button below:

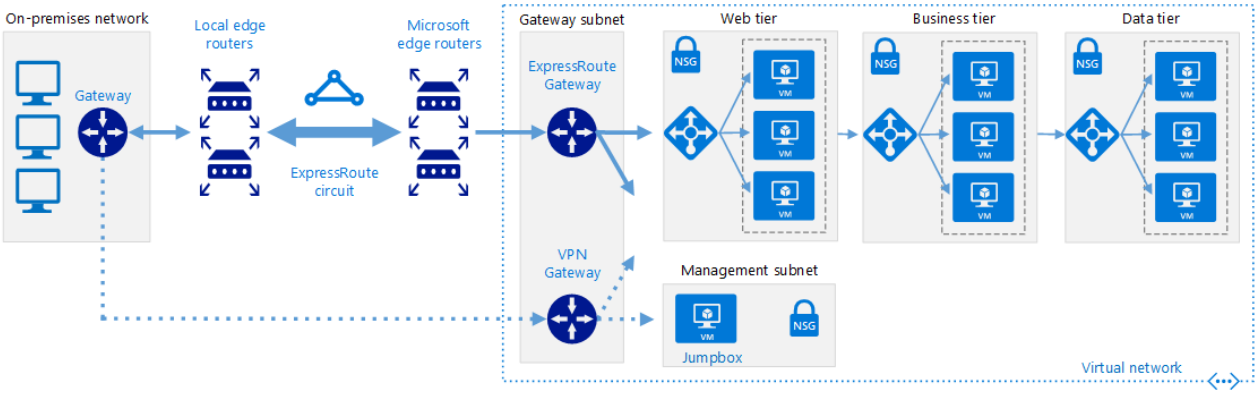
5. Wait for the link to open in the Azure portal, then follow these steps:
   - Select **Use existing** in the **Resource group** section and enter `ra-hybrid-er-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
6. Wait for the deployment to complete.

Connect an on-premises network to Azure using ExpressRoute with VPN failover

4/11/2018 • 4 min to read • Edit Online

This reference architecture shows how to connect an on-premises network to an Azure virtual network (VNet) using ExpressRoute, with a site-to-site virtual private network (VPN) as a failover connection. Traffic flows between the on-premises network and the Azure VNet through an ExpressRoute connection. If there is a loss of connectivity in the ExpressRoute circuit, traffic is routed through an IPSec VPN tunnel. **Deploy this solution**.

Note that if the ExpressRoute circuit is unavailable, the VPN route will only handle private peering connections. Public peering and Microsoft peering connections will pass over the Internet.



Download a Visio file of this architecture.

Architecture

The architecture consists of the following components.

- **On-premises network**. A private local-area network running within an organization.
- **VPN appliance**. A device or service that provides external connectivity to the on-premises network. The VPN appliance may be a hardware device, or it can be a software solution such as the Routing and Remote Access Service (RRAS) in Windows Server 2012. For a list of supported VPN appliances and information on configuring selected VPN appliances for connecting to Azure, see About VPN devices for Site-to-Site VPN Gateway connections.
- **ExpressRoute circuit**. A layer 2 or layer 3 circuit supplied by the connectivity provider that joins the on-premises network with Azure through the edge routers. The circuit uses the hardware infrastructure managed by the connectivity provider.
- **ExpressRoute virtual network gateway**. The ExpressRoute virtual network gateway enables the VNet to connect to the ExpressRoute circuit used for connectivity with your on-premises network.
- **VPN virtual network gateway**. The VPN virtual network gateway enables the VNet to connect to the VPN appliance in the on-premises network. The VPN virtual network gateway is configured to accept requests from the on-premises network only through the VPN appliance. For more information, see Connect an on-premises network to a Microsoft Azure virtual network.
- **VPN connection**. The connection has properties that specify the connection type (IPSec) and the key shared with the on-premises VPN appliance to encrypt traffic.

- **Azure Virtual Network (VNet)**. Each VNet resides in a single Azure region, and can host multiple application tiers. Application tiers can be segmented using subnets in each VNet.
- **Gateway subnet**. The virtual network gateways are held in the same subnet.
- **Cloud application**. The application hosted in Azure. It might include multiple tiers, with multiple subnets connected through Azure load balancers. For more information about the application infrastructure, see Running Windows VM workloads and Running Linux VM workloads.

Recommendations

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

VNet and GatewaySubnet

Create the ExpressRoute virtual network gateway and the VPN virtual network gateway in the same VNet. This means that they should share the same subnet named *GatewaySubnet*.

If the VNet already includes a subnet named *GatewaySubnet*, ensure that it has a /27 or larger address space. If the existing subnet is too small, use the following PowerShell command to remove the subnet:

```
$vnet = Get-AzureRmVirtualNetworkGateway -Name <yourvnetname> -ResourceGroupName <yourresourcegroup>
Remove-AzureRmVirtualNetworkSubnetConfig -Name GatewaySubnet -VirtualNetwork $vnet
```

If the VNet does not contain a subnet named **GatewaySubnet**, create a new one using the following Powershell command:

```
$vnet = Get-AzureRmVirtualNetworkGateway -Name <yourvnetname> -ResourceGroupName <yourresourcegroup>
Add-AzureRmVirtualNetworkSubnetConfig -Name "GatewaySubnet" -VirtualNetwork $vnet -AddressPrefix
"10.200.255.224/27"
$vnet = Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

VPN and ExpressRoute gateways

Verify that your organization meets the ExpressRoute prerequisite requirements for connecting to Azure.

If you already have a VPN virtual network gateway in your Azure VNet, use the following Powershell command to remove it:

```
Remove-AzureRmVirtualNetworkGateway -Name <yourgatewayname> -ResourceGroupName <yourresourcegroup>
```

Follow the instructions in Implementing a hybrid network architecture with Azure ExpressRoute to establish your ExpressRoute connection.

Follow the instructions in Implementing a hybrid network architecture with Azure and On-premises VPN to establish your VPN virtual network gateway connection.

After you have established the virtual network gateway connections, test the environment as follows:

1. Make sure you can connect from your on-premises network to your Azure VNet.
2. Contact your provider to stop ExpressRoute connectivity for testing.
3. Verify that you can still connect from your on-premises network to your Azure VNet using the VPN virtual network gateway connection.
4. Contact your provider to reestablish ExpressRoute connectivity.

Considerations

For ExpressRoute considerations, see the Implementing a Hybrid Network Architecture with Azure ExpressRoute guidance.

For site-to-site VPN considerations, see the Implementing a Hybrid Network Architecture with Azure and On-premises VPN guidance.

For general Azure security considerations, see Microsoft cloud services and network security.

Deploy the solution

Prequisites. You must have an existing on-premises infrastructure already configured with a suitable network appliance.

To deploy the solution, perform the following steps.

1. Click the button below:

2. Wait for the link to open in the Azure portal, then follow these steps:
   - The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-hybrid-vpn-er-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
3. Wait for the deployment to complete.
4. Click the button below:

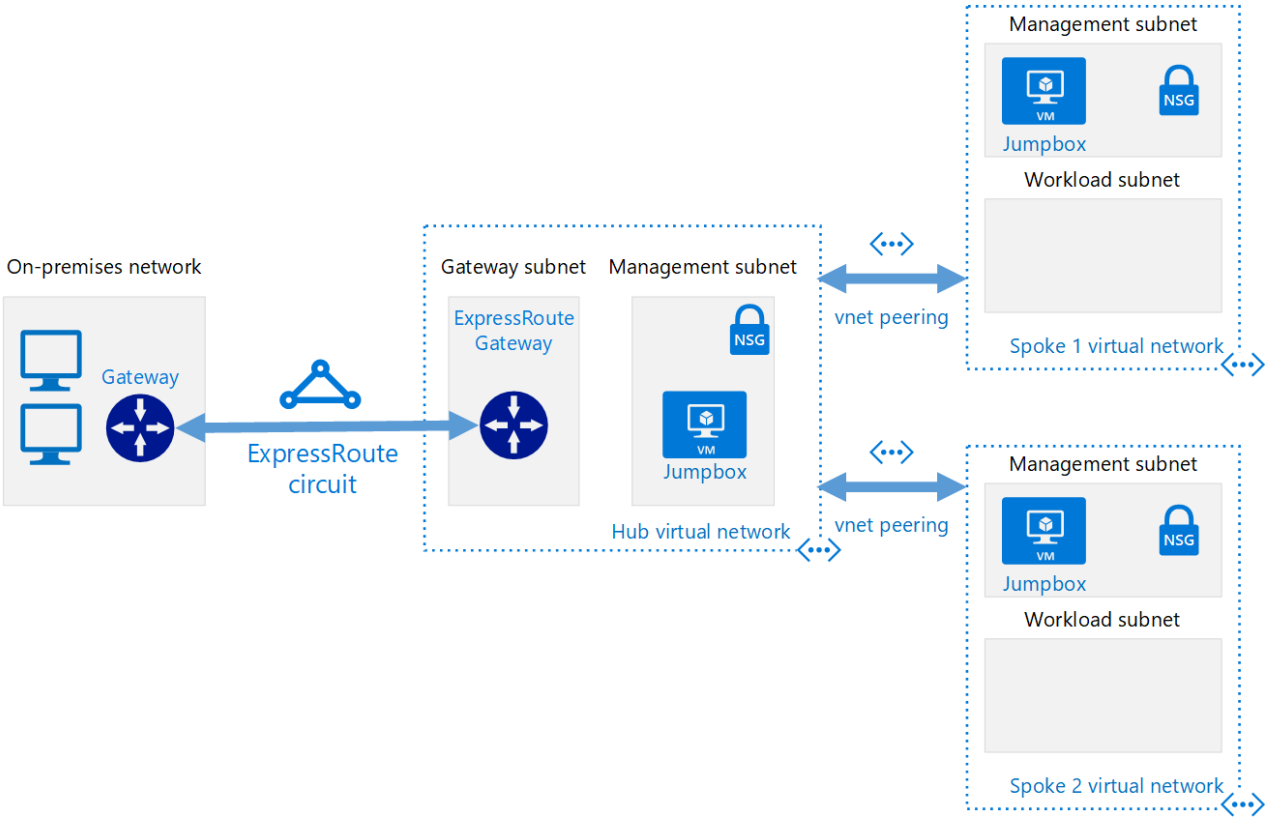
   

5. Wait for the link to open in the Azure portal, then enter then follow these steps:
   - Select **Use existing** in the **Resource group** section and enter `ra-hybrid-vpn-er-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.

Implement a hub-spoke network topology in Azure

4/28/2018 • 9 min to read • Edit Online

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



Download a Visio file of this architecture

The benefits of this toplogy include:

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

Typical uses for this architecture include:

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

Architecture

The architecture consists of the following components.

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

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

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

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

Recommendations

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

Resource groups

The hub VNet, and each spoke VNet, can be implemented in different resource groups, and even different subscriptions, as long as they belong to the same Azure Active Directory (Azure AD) tenant in the same Azure region. This allows for a decentralized management of each workload, while sharing services maintained in the hub VNet.

VNet and GatewaySubnet

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

the future.

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

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

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

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

VNet peering

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

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

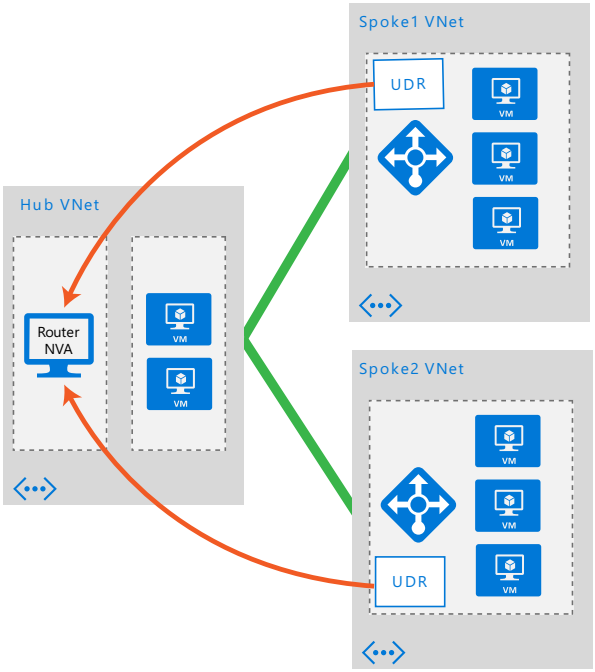
You can also configure spokes to use the hub VNet gateway to communicate with remote networks. To allow gateway traffic to flow from spoke to hub, and connect to remote networks, you must:

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

Considerations

Spoke connectivity

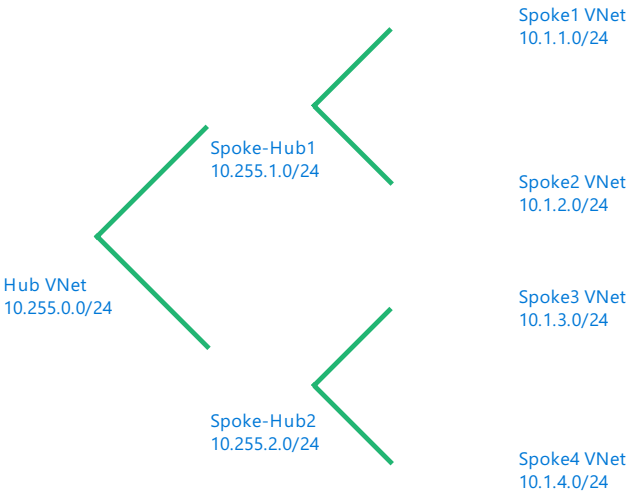
If you require connectivity between spokes, consider implementing an NVA for routing in the hub, and using UDRs in the spoke to forward traffic to the hub.



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

Overcoming VNet peering limits

Make sure you consider the limitation on number of VNets peerings per VNet in Azure. If you decide you need more spokes than the limit will allow, consider creating a hub-spoke-hub-spoke topology, where the first level of spokes also act as hubs. The following diagram shows this approach.



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

Deploy the solution

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

The deployment creates the following resource groups in your subscription:

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

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

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Install Azure CLI 2.0.
3. Install the Azure building blocks npm package.
4. From a command prompt, bash prompt, or PowerShell prompt, log into your Azure account by using the command below.

```
az login
```

Deploy the simulated on-premises datacenter

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

1. Navigate to the `hybrid-networking/hub-spoke` folder of the reference architectures repository.

2. Open the `onprem.json` file. Replace the values for `adminUsername` and `adminPassword`.

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

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

4. Run the following command:

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

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

Deploy the hub VNet

To deploy the hub VNet, perform the following steps.

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

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

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

3. For `sharedKey`, enter a shared key for the VPN connection.

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

4. Run the following command:

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

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

Test connectivity with the hub

Test conectivity from the simulated on-premises environment to the hub VNet.

Windows deployment

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

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

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

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

The output should look similar to the following:

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

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

Linux deployment

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

```
ping 10.0.0.68
```

Deploy the spoke VNets

To deploy the spoke VNets, perform the following steps.

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

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

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

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

4. Repeat steps 1-2 for the `spoke2.json` file.
5. Run the following command:

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

6. Run the following command:

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

Test connectivity

Test conectivity from the simulated on-premises environment to the spoke VNets.

Windows deployment

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

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

Linux deployment

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

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

```
ping 10.1.0.68
ping 10.2.0.68
```

Add connectivity between spokes

This step is optional. If you want to allow spokes to connect to each other, you must use a newtwork virtual appliance (NVA) as a router in the hub VNet, and force traffic from spokes to the router when trying to connect to another spoke. To deploy a basic sample NVA as a single VM, along with user-defined routes (UDRs) to allow the two spoke VNets to connect, perform the following steps:

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

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

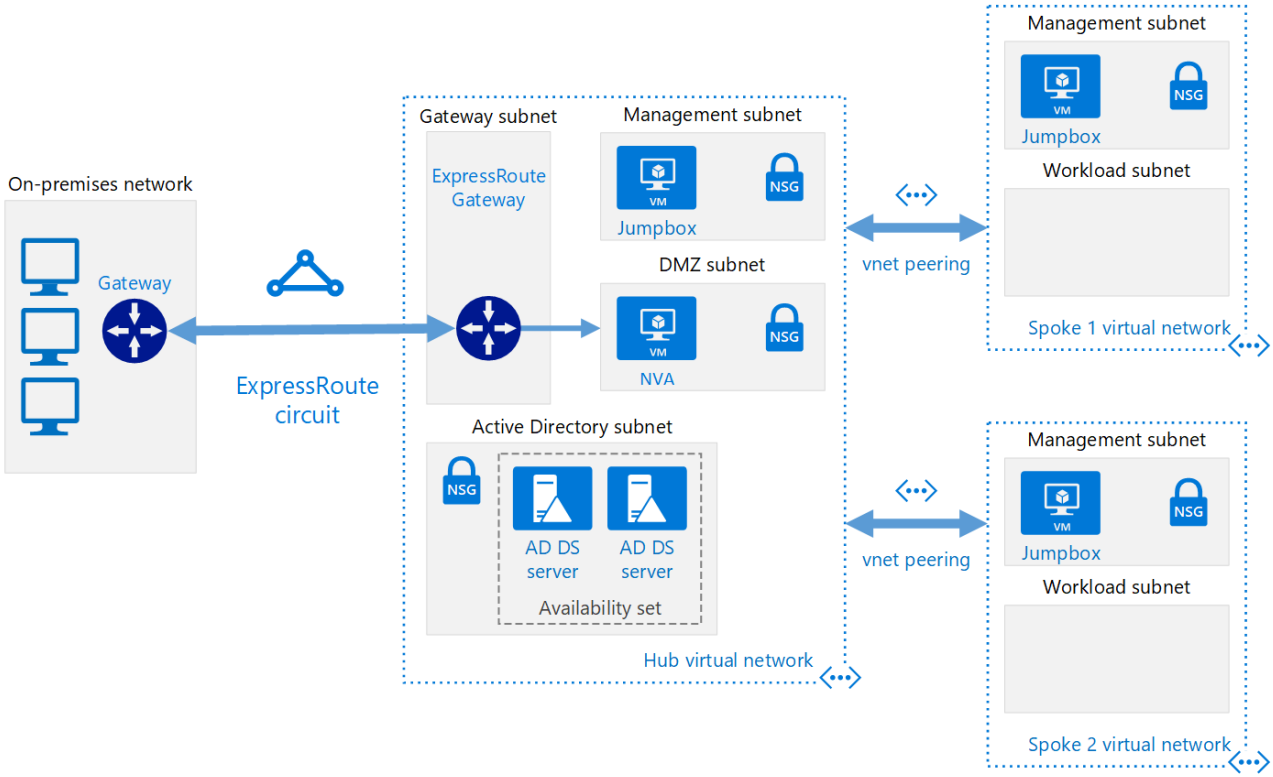
2. Run the following command:

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

Implement a hub-spoke network topology with shared services in Azure

4/14/2018 • 9 min to read • Edit Online

This reference architecture builds on the hub-spoke reference architecture to include shared services in the hub that can be consumed by all spokes. As a first step toward migrating a datacenter to the cloud, and building a virtual datacenter, the first services you need to share are identity and security. This reference architecture shows you how to extend your Active Directory services from your on-premises datacenter to Azure, and how to add a network virtual appliance (NVA) that can act as a firewall, in a hub-spoke topology. **Deploy this solution**.



Download a Visio file of this architecture

The benefits of this topology include:

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

Typical uses for this architecture include:

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

Architecture

The architecture consists of the following components.

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

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

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

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

Recommendations

All the recommendations for the hub-spoke reference architecture also apply to the shared services reference architecture.

ALso, the following recommendations apply for most scenarios under shared services. Follow these recommendations unless you have a specific requirement that overrides them.

Identity

Most enterprise organizations have an Active Directory Directory Services (ADDS) environment in their on-premises datacenter. To facilitate management of assets moved to Azure from your on-premises network that depend on ADDS, it is recommended to host ADDS domain controllers in Azure.

If you make use of Group Policy Objects, that you want to control separately for Azure and your on-premises environment, use a different AD site for each Azure region. Place your domain controllers in a central VNet (hub) that dependent workloads can access.

Security

As you move workloads from your on-premises environment to Azure, some of these workloads will require to be hosted in VMs. For compliance reasons, you may need to enforce restrictions on traffic traversing those workloads.

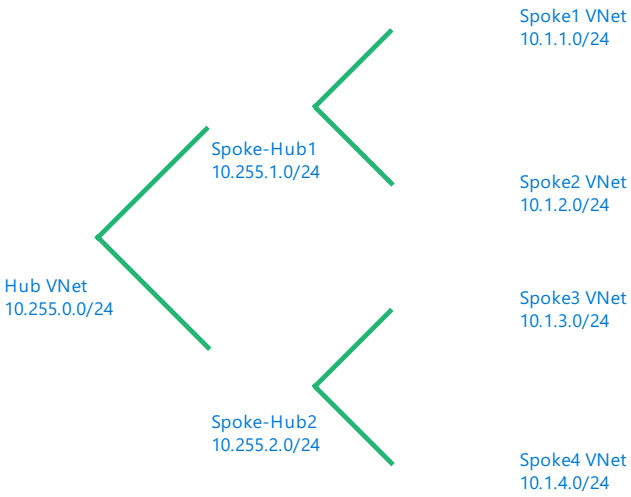
You can use network virtual appliances (NVAs) in Azure to host different types of security and performance services. If you are familiar with a given set of appliances on-premises today, it is recommended to use the same virtualized appliances in Azure, where applicable.

> **NOTE**
> The deployment scripts for this reference architecture use an Ubuntu VM with IP forwarding enabled to mimic a network virtual appliance.

Considerations

Overcoming VNet peering limits

Make sure you consider the limitation on number of VNets peerings per VNet in Azure. If you decide you need more spokes than the limit will allow, consider creating a hub-spoke-hub-spoke topology, where the first level of spokes also act as hubs. The following diagram shows this approach.



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

Deploy the solution

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

Prerequisites

Before you can deploy the reference architecture to your own subscription, you must perform the following steps.

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Make sure you have the Azure CLI 2.0 installed on your computer. For CLI installation instructions, see Install Azure CLI 2.0.
3. Install the Azure building blocks npm package.
4. From a command prompt, bash prompt, or PowerShell prompt, login to your Azure account by using the command below, and follow the prompts.

```
az login
```

Deploy the simulated on-premises datacenter using azbb

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

1. Navigate to the `hybrid-networking\shared-services-stack\` folder for the repository you downloaded in the pre-requisites step above.
2. Open the `onprem.json` file and enter a username and password between the quotes in line 45 and 46, as shown below, then save the file.

```
"adminUsername": "XXX",
"adminPassword": "YYY",
```

3. Run `azbb` to deploy the simulated onprem environment as shown below.

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

> **NOTE**
> If you decide to use a different resource group name (other than `onprem-vnet-rg`), make sure to search for all parameter files that use that name and edit them to use your own resource group name.

4. Wait for the deployment to finish. This deployment creates a virtual network, a virtual machine running Windows, and a VPN gateway. The VPN gateway creation can take more than 40 minutes to complete.

Azure hub VNet

To deploy the hub VNet, and connect to the simulated on-premises VNet created above, perform the following steps.

1. Open the `hub-vnet.json` file and enter a username and password between the quotes in line 50 and 51, as shown below.

```
"adminUsername": "XXX",
"adminPassword": "YYY",
```

2. On line 52, for `osType`, type `Windows` or `Linux` to install either Windows Server 2016 Datacenter, or Ubuntu 16.04 as the operating system for the jumpbox.
3. Enter a shared key between the quotes in line 83, as shown below, then save the file.

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

4. Run `azbb` to deploy the simulated onprem environment as shown below.

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

> **NOTE**
> If you decide to use a different resource group name (other than `hub-vnet-rg`), make sure to search for all parameter files that use that name and edit them to use your own resource group name.

5. Wait for the deployment to finish. This deployment creates a virtual network, a virtual machine, a VPN gateway, and a connection to the gateway created in the previous section. The VPN gateway creation can take more than 40 minutes to complete.

ADDS in Azure

To deploy the ADDS domain controllers in Azure, perform the following steps.

1. Open the `hub-adds.json` file and enter a username and password between the quotes in lines 14 and 15, as shown below, then save the file.

```
"adminUsername": "XXX",
"adminPassword": "YYY",
```

2. Run `azbb` to deploy the ADDS domain controllers as shown below.

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

> **NOTE**
> If you decide to use a different resource group name (other than `hub-adds-rg`), make sure to search for all parameter files that use that name and edit them to use your own resource group name.

> **NOTE**
> This part of the deployment may take several minutes, since it requires joining the two VMs to the domain hosted int he simulated on-premises datacenter, then installing AD DS on them.

NVA

To deploy an NVA in the `dmz` subnet, perform the following steps:

1. Open the `hub-nva.json` file and enter a username and password between the quotes in lines 13 and 14, as shown below, then save the file.

```
"adminUsername": "XXX",
"adminPassword": "YYY",
```

2. Run `azbb` to deploy the NVA VM and user defined routes.

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

> **NOTE**
>
> If you decide to use a different resource group name (other than `hub-nva-rg`), make sure to search for all parameter files that use that name and edit them to use your own resource group name.

Azure spoke VNets

To deploy the spoke VNets, perform the following steps.

1. Open the `spoke1.json` file and enter a username and password between the quotes in lines 52 and 53, as shown below, then save the file.

```
"adminUsername": "XXX",
"adminPassword": "YYY",
```

2. On line 54, for `osType`, type `Windows` or `Linux` to install either Windows Server 2016 Datacenter, or Ubuntu 16.04 as the operating system for the jumpbox.

3. Run `azbb` to deploy the first spoke VNet environment as shown below.

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

> **NOTE**
>
> If you decide to use a different resource group name (other than `spoke1-vnet-rg`), make sure to search for all parameter files that use that name and edit them to use your own resource group name.

4. Repeat step 1 above for file `spoke2.json`.

5. Run `azbb` to deploy the second spoke VNet environment as shown below.

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

> **NOTE**
>
> If you decide to use a different resource group name (other than `spoke2-vnet-rg`), make sure to search for all parameter files that use that name and edit them to use your own resource group name.

Azure hub VNet peering to spoke VNets

To create a peering connection from the hub VNet to the spoke VNets, perform the following steps.

1. Open the `hub-vnet-peering.json` file and verify that the resource group name, and virtual network name for each of the virtual network peerings starting in line 29 are correct.

2. Run `azbb` to deploy the first spoke VNet environment as shown below.

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

NOTE

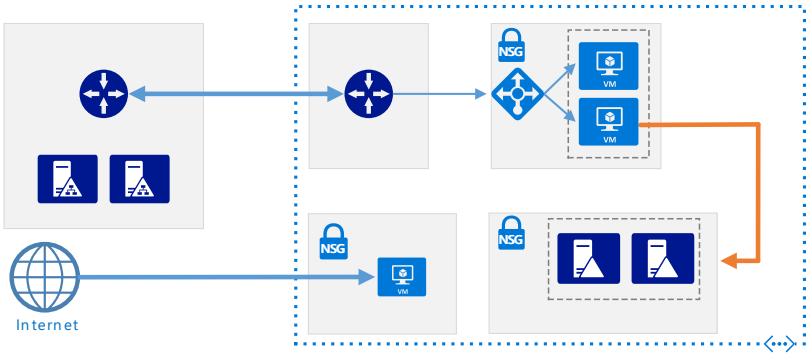
If you decide to use a different resource group name (other than `hub-vnet-rg`), make sure to search for all parameter files that use that name and edit them to use your own resource group name.

These reference architectures show options for integrating your on-premises Active Directory (AD) environment with an Azure network.

Which should I choose?

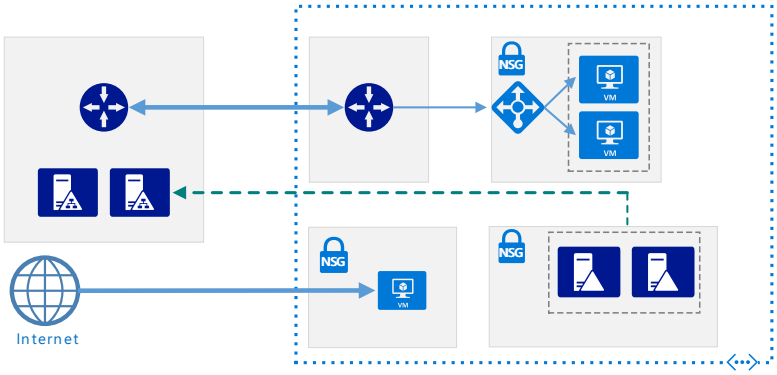
Integrate with Azure Active Directory

Integrate on-premises Active Directory domains and forests with Azure AD.



Extend AD DS to Azure

Extend your Active Directory environment to Azure using Active Directory Domain Services.



Create an AD DS forest in Azure

Create a separate AD domain in Azure that is trusted by domains in your on-premises forest.

Extend AD FS to Azure

Use Active Directory Federation Services for federated authentication and authorization in Azure.

Choose a solution for integrating on-premises Active Directory with Azure

10/28/2017 • 4 min to read • Edit Online

This article compares options for integrating your on-premises Active Directory (AD) environment with an Azure network. We provide a reference architecture and a deployable solution for each option.

Many organizations use Active Directory Domain Services (AD DS) to authenticate identities associated with users, computers, applications, or other resources that are included in a security boundary. Directory and identity services are typically hosted on-premises, but if your application is hosted partly on-premises and partly in Azure, there may be latency sending authentication requests from Azure back to on-premises. Implementing directory and identity services in Azure can reduce this latency.

Azure provides two solutions for implementing directory and identity services in Azure:

- Use Azure AD to create an Active Directory domain in the cloud and connect it to your on-premises Active Directory domain. Azure AD Connect integrates your on-premises directories with Azure AD.
- Extend your existing on-premises Active Directory infrastructure to Azure, by deploying a VM in Azure that runs AD DS as a domain controller. This architecture is more common when the on-premises network and the Azure virtual network (VNet) are connected by a VPN or ExpressRoute connection. Several variations of this architecture are possible:
  - Create a domain in Azure and join it to your on-premises AD forest.
  - Create a separate forest in Azure that is trusted by domains in your on-premises forest.
  - Replicate an Active Directory Federation Services (AD FS) deployment to Azure.

The next sections describe each of these options in more detail.

Integrate your on-premises domains with Azure AD

Use Azure Active Directory (Azure AD) to create a domain in Azure and link it to an on-premises AD domain.

The Azure AD directory is not an extension of an on-premises directory. Rather, it's a copy that contains the same objects and identities. Changes made to these items on-premises are copied to Azure AD, but changes made in Azure AD are not replicated back to the on-premises domain.

You can also use Azure AD without using an on-premises directory. In this case, Azure AD acts as the primary source of all identity information, rather than containing data replicated from an on-premises directory.

Benefits

- You don't need to maintain an AD infrastructure in the cloud. Azure AD is entirely managed and maintained by Microsoft.
- Azure AD provides the same identity information that is available on-premises.
- Authentication can happen in Azure, reducing the need for external applications and users to contact the on-premises domain.

Challenges

- Identity services are limited to users and groups. There is no ability to authenticate service and computer accounts.

- You must configure connectivity with your on-premises domain to keep the Azure AD directory synchronized.
- Applications may need to be rewritten to enable authentication through Azure AD.

Read more...

AD DS in Azure joined to an on-premises forest

Deploy AD Domain Services (AD DS) servers to Azure. Create a domain in Azure and join it to your on-premises AD forest.

Consider this option if you need to use AD DS features that are not currently implemented by Azure AD.

Benefits

- Provides access to the same identity information that is available on-premises.
- You can authenticate user, service, and computer accounts on-premises and in Azure.
- You don't need to manage a separate AD forest. The domain in Azure can belong to the on-premises forest.
- You can apply group policy defined by on-premises Group Policy Objects to the domain in Azure.

Challenges

- You must deploy and manage your own AD DS servers and domain in the cloud.
- There may be some synchronization latency between the domain servers in the cloud and the servers running on-premises.

Read more...

AD DS in Azure with a separate forest

Deploy AD Domain Services (AD DS) servers to Azure, but create a separate Active Directory forest that is separate from the on-premises forest. This forest is trusted by domains in your on-premises forest.

Typical uses for this architecture include maintaining security separation for objects and identities held in the cloud, and migrating individual domains from on-premises to the cloud.

Benefits

- You can implement on-premises identities and separate Azure-only identities.
- You don't need to replicate from the on-premises AD forest to Azure.

Challenges

- Authentication within Azure for on-premises identities requires extra network hops to the on-premises AD servers.
- You must deploy your own AD DS servers and forest in the cloud, and establish the appropriate trust relationships between forests.

Read more...

Extend AD FS to Azure

Replicate an Active Directory Federation Services (AD FS) deployment to Azure, to perform federated authentication and authorization for components running in Azure.

Typical uses for this architecture:

- Authenticate and authorize users from partner organizations.
- Allow users to authenticate from web browsers running outside of the organizational firewall.

- Allow users to connect from authorized external devices such as mobile devices.

Benefits

- You can leverage claims-aware applications.
- Provides the ability to trust external partners for authentication.
- Compatibility with large set of authentication protocols.

Challenges

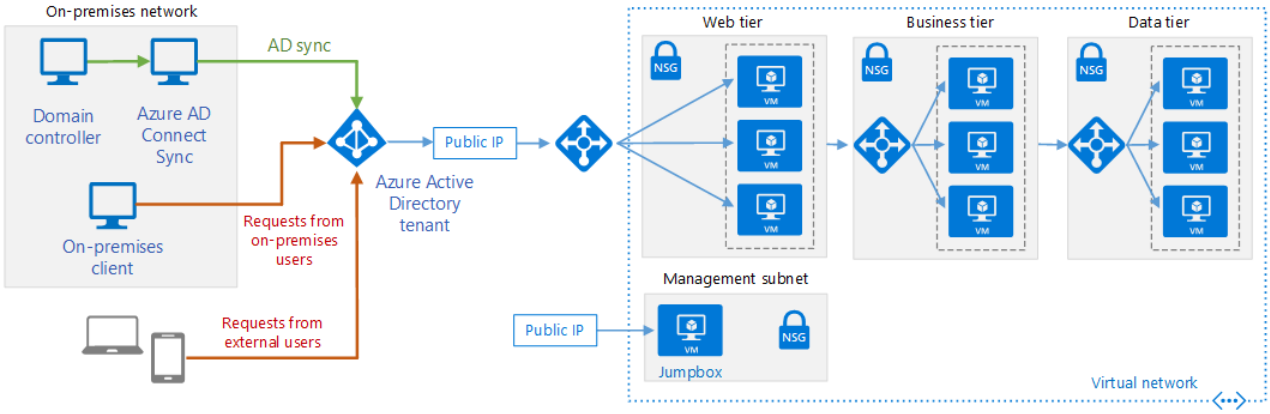
- You must deploy your own AD DS, AD FS, and AD FS Web Application Proxy servers in Azure.
- This architecture can be complex to configure.

Read more...

Integrate on-premises Active Directory domains with Azure Active Directory

4/21/2018 • 16 min to read • Edit Online

Azure Active Directory (Azure AD) is a cloud based multi-tenant directory and identity service. This reference architecture shows best practices for integrating on-premises Active Directory domains with Azure AD to provide cloud-based identity authentication. **Deploy this solution**.



Download a Visio file of this architecture.

> **NOTE**
> For simplicity, this diagram only shows the connections directly related to Azure AD, and not protocol-related traffic that may occur as part of authentication and identity federation. For example, a web application may redirect the web browser to authenticate the request through Azure AD. Once authenticated, the request can be passed back to the web application, with the appropriate identity information.

Typical uses for this reference architecture include:

- Web applications deployed in Azure that provide access to remote users who belong to your organization.
- Implementing self-service capabilities for end-users, such as resetting their passwords, and delegating group management. Note that this requires Azure AD Premium edition.
- Architectures in which the on-premises network and the application's Azure VNet are not connected using a VPN tunnel or ExpressRoute circuit.

> **NOTE**
> Azure AD currently supports user authentication only. Some applications and services, such as SQL Server, may require computer authentication, in which case this solution is not appropriate.

For additional considerations, see Choose a solution for integrating on-premises Active Directory with Azure.

Architecture

The architecture has the following components.

- **Azure AD tenant**. An instance of Azure AD created by your organization. It acts as a directory service for cloud applications by storing objects copied from the on-premises Active Directory and provides identity

services.

- **Web tier subnet**. This subnet holds VMs that run a web application. Azure AD can act as an identity broker for this application.
- **On-premises AD DS server**. An on-premise directory and identity service. The AD DS directory can be synchronized with Azure AD to enable it to authenticate on-premise users.
- **Azure AD Connect sync server**. An on-premises computer that runs the Azure AD Connect sync service. This service synchronizes information held in the on-premises Active Directory to Azure AD. For example, if you provision or deprovision groups and users on-premises, these changes propagate to Azure AD.

  > **NOTE**
  > For security reasons, Azure AD stores user's passwords as a hash. If a user requires a password reset, this must be performed on-premises and the new hash must be sent to Azure AD. Azure AD Premium editions include features that can automate this task to enable users to reset their own passwords.

- **VMs for N-tier application**. The deployment includes infrastructure for an N-tier application. For more information about these resources, see Run VMs for an N-tier architecture.

Recommendations

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

Azure AD Connect sync service

The Azure AD Connect sync service ensures that identity information stored in the cloud is consistent with that held on-premises. You install this service using the Azure AD Connect software.

Before implementing Azure AD Connect sync, determine the synchronization requirements of your organization. For example, what to synchronize, from which domains, and how frequently. For more information, see Determine directory synchronization requirements.

You can run the Azure AD Connect sync service on a VM or a computer hosted on-premises. Depending on the volatility of the information in your Active Directory directory, the load on the Azure AD Connect sync service is unlikely to be high after the initial synchronization with Azure AD. Running the service on a VM makes it easier to scale the server if needed. Monitor the activity on the VM as described in the Monitoring considerations section to determine whether scaling is necessary.

If you have multiple on-premises domains in a forest, we recommend storing and synchronizing information for the entire forest to a single Azure AD tenant. Filter information for identities that occur in more than one domain, so that each identity appears only once in Azure AD, rather than being duplicated. Duplication can lead to inconsistencies when data is synchronized. For more information, see the Topology section below.

Use filtering so that only necessary data is stored in Azure AD. For example, your organization might not want to store information about inactive accounts in Azure AD. Filtering can be group-based, domain-based, organization unit (OU)-based, or attribute-based. You can combine filters to generate more complex rules. For example, you could synchronize objects held in a domain that have a specific value in a selected attribute. For detailed information, see Azure AD Connect sync: Configure Filtering.

To implement high availability for the AD Connect sync service, run a secondary staging server. For more information, see the Topology recommendations section.

Security recommendations

User password management. The Azure AD Premium editions support password writeback, enabling your on-

premises users to perform self-service password resets from within the Azure portal. This feature should only be enabled after reviewing your organization's password security policy. For example, you can restrict which users can change their passwords, and you can tailor the password management experience. For more information, see Customizing Password Management to fit your organization's needs.

Protect on-premises applications that can be accessed externally. Use the Azure AD Application Proxy to provide controlled access to on-premises web applications for external users through Azure AD. Only users that have valid credentials in your Azure directory have permission to use the application. For more information, see the article Enable Application Proxy in the Azure portal.

Actively monitor Azure AD for signs of suspicious activity. Consider using Azure AD Premium P2 edition, which includes Azure AD Identity Protection. Identity Protection uses adaptive machine learning algorithms and heuristics to detect anomalies and risk events that may indicate that an identity has been compromised. For example, it can detect potentially unusual activity such as irregular sign-in activities, sign-ins from unknown sources or from IP addresses with suspicious activity, or sign-ins from devices that may be infected. Using this data, Identity Protection generates reports and alerts that enables you to investigate these risk events and take appropriate action. For more information, see Azure Active Directory Identity Protection.

You can use the reporting feature of Azure AD in the Azure portal to monitor security-related activities occurring in your system. For more information about using these reports, see Azure Active Directory Reporting Guide.

Topology recommendations

Configure Azure AD Connect to implement a topology that most closely matches the requirements of your organization. Topologies that Azure AD Connect supports include the following:

- **Single forest, single Azure AD directory**. In this topology, Azure AD Connect synchronizes objects and identity information from one or more domains in a single on-premises forest into a single Azure AD tenant. This is the default topology implemented by the express installation of Azure AD Connect.

  > **NOTE**
  > Don't use multiple Azure AD Connect sync servers to connect different domains in the same on-premises forest to the same Azure AD tenant, unless you are running a server in staging mode, described below.

- **Multiple forests, single Azure AD directory**. In this topology, Azure AD Connect synchronizes objects and identity information from multiple forests into a single Azure AD tenant. Use this topology if your organization has more than one on-premises forest. You can consolidate identity information so that each unique user is represented once in the Azure AD directory, even if the same user exists in more than one forest. All forests use the same Azure AD Connect sync server. The Azure AD Connect sync server does not have to be part of any domain, but it must be reachable from all forests.

  > **NOTE**
  > In this topology, don't use separate Azure AD Connect sync servers to connect each on-premises forest to a single Azure AD tenant. This can result in duplicated identity information in Azure AD if users are present in more than one forest.

- **Multiple forests, separate topologies**. This topology merges identity information from separate forests into a single Azure AD tenant, treating all forests as separate entities. This topology is useful if you are combining forests from different organizations and the identity information for each user is held in only one forest.

> **NOTE**
> If the global address lists (GAL) in each forest are synchronized, a user in one forest may be present in another as a contact. This can occur if your organization has implemented GALSync with Forefront Identity manager 2010 or Microsoft Identity Manager 2016. In this scenario, you can specify that users should be identified by their *Mail* attribute. You can also match identities using the *ObjectSID* and *msExchMasterAccountSID* attributes. This is useful if you have one or more resource forests with disabled accounts.

- **Staging server**. In this configuration, you run a second instance of the Azure AD Connect sync server in parallel with the first. This structure supports scenarios such as:
  - High availability.
  - Testing and deploying a new configuration of the Azure AD Connect sync server.
  - Introducing a new server and decommissioning an old configuration.

  In these scenarios, the second instance runs in *staging mode*. The server records imported objects and synchronization data in its database, but does not pass the data to Azure AD. If you disable staging mode, the server starts writing data to Azure AD, and also starts performing password write-backs into the on-premises directories where appropriate. For more information, see Azure AD Connect sync: Operational tasks and considerations.
- **Multiple Azure AD directories**. It is recommended that you create a single Azure AD directory for an organization, but there may be situations where you need to partition information across separate Azure AD directories. In this case, avoid synchronization and password write-back issues by ensuring that each object from the on-premises forest appears in only one Azure AD directory. To implement this scenario, configure separate Azure AD Connect sync servers for each Azure AD directory, and use filtering so that each Azure AD Connect sync server operates on a mutually exclusive set of objects.

For more information about these topologies, see Topologies for Azure AD Connect.

User authentication

By default, the Azure AD Connect sync server configures password hash synchronization between the on-premises domain and Azure AD, and the Azure AD service assumes that users authenticate by providing the same password that they use on-premises. For many organizations, this is appropriate, but you should consider your organization's existing policies and infrastructure. For example:

- The security policy of your organization may prohibit synchronizing password hashes to the cloud. In this case your organization should consider pass-through authentication.
- You might require that users experience seamless single sign-on (SSO) when accessing cloud resources from domain-joined machines on the corporate network.
- Your organization might already have Active Directory Federation Services (AD FS) or a third party federation provider deployed. You can configure Azure AD to use this infrastructure to implement authentication and SSO rather than by using password information held in the cloud.

For more information, see Azure AD Connect User Sign on options.

Azure AD application proxy

Use Azure AD to provide access to on-premises applications.

Expose your on-premises web applications using application proxy connectors managed by the Azure AD application proxy component. The application proxy connector opens an outbound network connection to the Azure AD application proxy, and remote users' requests are routed back from Azure AD through this connection to the web apps. This removes the need to open inbound ports in the on-premises firewall and reduces the attack surface exposed by your organization.

For more information, see Publish applications using Azure AD Application proxy.

Object synchronization

Azure AD Connect's default configuration synchronizes objects from your local Active Directory directory based on the rules specified in the article Azure AD Connect sync: Understanding the default configuration. Objects that satisfy these rules are synchronized while all other objects are ignored. Some example rules:

- User objects must have a unique *sourceAnchor* attribute and the *accountEnabled* attribute must be populated.
- User objects must have a *sAMAccountName* attribute and cannot start with the text *Azure AD_* or *MSOL_*.

Azure AD Connect applies several rules to User, Contact, Group, ForeignSecurityPrincipal, and Computer objects. Use the Synchronization Rules Editor installed with Azure AD Connect if you need to modify the default set of rules. For more information, see Azure AD Connect sync: Understanding the default configuration).

You can also define your own filters to limit the objects to be synchronized by domain or OU. Alternatively, you can implement more complex custom filtering such as that described in Azure AD Connect sync: Configure Filtering.

Monitoring

Health monitoring is performed by the following agents installed on-premises:

- Azure AD Connect installs an agent that captures information about synchronization operations. Use the Azure AD Connect Health blade in the Azure portal to monitor its health and performance. For more information, see Using Azure AD Connect Health for sync.
- To monitor the health of the AD DS domains and directories from Azure, install the Azure AD Connect Health for AD DS agent on a machine within the on-premises domain. Use the Azure Active Directory Connect Health blade in the Azure portal for health monitoring. For more information, see Using Azure AD Connect Health with AD DS
- Install the Azure AD Connect Health for AD FS agent to monitor the health of services running on on-premises, and use the Azure Active Directory Connect Health blade in the Azure portal to monitor AD FS. For more information, see Using Azure AD Connect Health with AD FS

For more information on installing the AD Connect Health agents and their requirements, see Azure AD Connect Health Agent Installation.

Scalability considerations

The Azure AD service supports scalability based on replicas, with a single primary replica that handles write operations plus multiple read-only secondary replicas. Azure AD transparently redirects attempted writes made against secondary replicas to the primary replica and provides eventual consistency. All changes made to the primary replica are propagated to the secondary replicas. This architecture scales well because most operations against Azure AD are reads rather than writes. For more information, see Azure AD: Under the hood of our geo-redundant, highly available, distributed cloud directory.

For the Azure AD Connect sync server, determine how many objects you are likely to synchronize from your local directory. If you have less than 100,000 objects, you can use the default SQL Server Express LocalDB software provided with Azure AD Connect. If you have a larger number of objects, you should install a production version of SQL Server and perform a custom installation of Azure AD Connect, specifying that it should use an existing instance of SQL Server.

Availability considerations

The Azure AD service is geo-distributed and runs in multiple data centers spread around the world with automated failover. If a data center becomes unavailable, Azure AD ensures that your directory data is available for instance access in at least two more regionally dispersed data centers.

> **NOTE**
> The service level agreement (SLA) for Azure AD Basic and Premium services guarantees at least 99.9% availability. There is no SLA for the Free tier of Azure AD. For more information, see SLA for Azure Active Directory.

Consider provisioning a second instance of Azure AD Connect sync server in staging mode to increase availability, as discussed in the topology recommendations section.

If you are not using the SQL Server Express LocalDB instance that comes with Azure AD Connect, consider using SQL clustering to achieve high availability. Solutions such as mirroring and Always On are not supported by Azure AD Connect.

For additional considerations about achieving high availability of the Azure AD Connect sync server and also how to recover after a failure, see Azure AD Connect sync: Operational tasks and considerations - Disaster Recovery.

Manageability considerations

There are two aspects to managing Azure AD:

- Administering Azure AD in the cloud.
- Maintaining the Azure AD Connect sync servers.

Azure AD provides the following options for managing domains and directories in the cloud:

- **Azure Active Directory PowerShell Module**. Use this module if you need to script common Azure AD administrative tasks such as user management, domain management, and configuring single sign-on.
- **Azure AD management blade in the Azure portal**. This blade provides an interactive management view of the directory, and enables you to control and configure most aspects of Azure AD.

Azure AD Connect installs the following tools to maintain Azure AD Connect sync services from your on-premises machines:

- **Microsoft Azure Active Directory Connect console**. This tool enables you to modify the configuration of the Azure AD Sync server, customize how synchronization occurs, enable or disable staging mode, and switch the user sign-in mode. Note that you can enable Active Directory FS sign-in using your on-premises infrastructure.
- **Synchronization Service Manager**. Use the *Operations* tab in this tool to manage the synchronization process and detect whether any parts of the process have failed. You can trigger synchronizations manually using this tool. The *Connectors* tab enables you to control the connections for the domains that the synchronization engine is attached to.
- **Synchronization Rules Editor**. Use this tool to customize the way objects are transformed when they are copied between an on-premises directory and Azure AD. This tool enables you to specify additional attributes and objects for synchronization, then executes filters to determine which objects should or should not be synchronized. For more information, see the Synchronization Rule Editor section in the document Azure AD Connect sync: Understanding the default configuration.

For more information and tips for managing Azure AD Connect, see Azure AD Connect sync: Best practices for changing the default configuration.

Security considerations

Use conditional access control to deny authentication requests from unexpected sources:

- Trigger Azure Multi-Factor Authentication (MFA) if a user attempts to connect from a nontrusted location such as across the Internet instead of a trusted network.

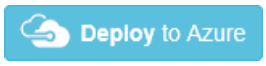
- Use the device platform type of the user (iOS, Android, Windows Mobile, Windows) to determine access policy to applications and features.
- Record the enabled/disabled state of users' devices, and incorporate this information into the access policy checks. For example, if a user's phone is lost or stolen it should be recorded as disabled to prevent it from being used to gain access.
- Control user access to resources based on group membership. Use Azure AD dynamic membership rules to simplify group administration. For a brief overview of how this works, see Introduction to Dynamic Memberships for Groups.
- Use conditional access risk policies with Azure AD Identity Protection to provide advanced protection based on unusual sign-in activities or other events.

For more information, see Azure Active Directory conditional access.

Deploy the solution

A deployment for a reference architecture that implements these recommendations and considerations is available on GitHub. This reference architecture deploys a simulated on-premise network in Azure that you can use to test and experiment. The reference architecture can be deployed with either with Windows or Linux VMs by following the directions below:

1. Click the button below:

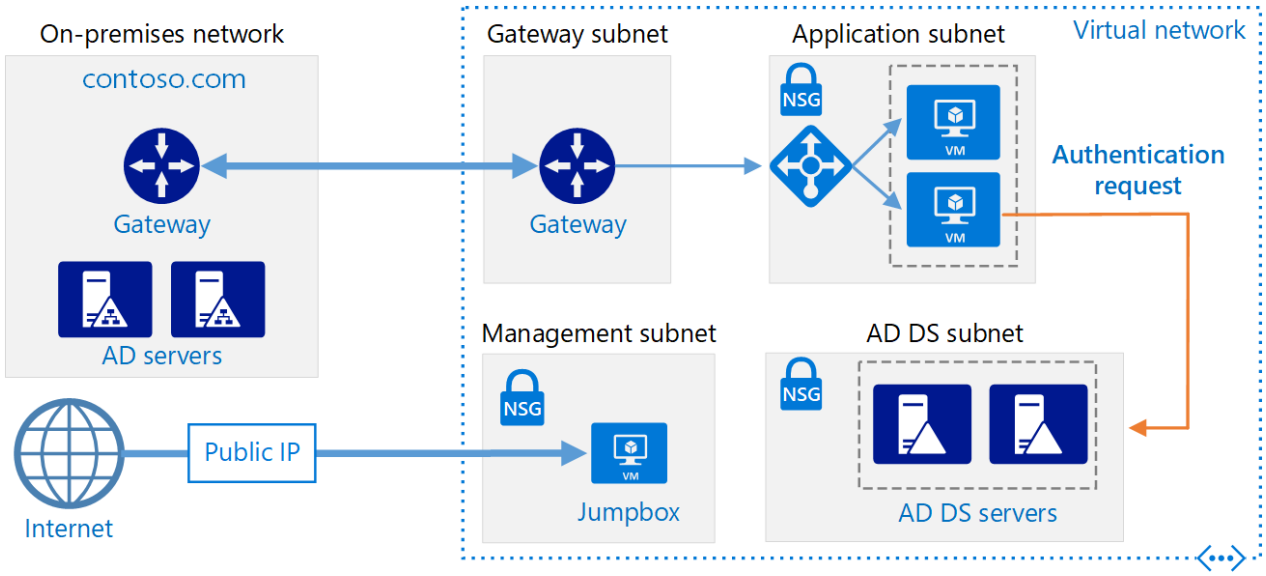
   

2. Once the link has opened in the Azure portal, you must enter values for some of the settings:
   - The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-aad-onpremise-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Select **windows** or **linux** in the **Os Type** the drop down box.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
3. Wait for the deployment to complete.
4. The parameter files include a hard-coded administrator user names and passwords, and it is strongly recommended that you immediately change both on all the VMs. Click each VM in the Azure Portal then click on **Reset password** in the **Support + troubleshooting** blade. Select **Reset password** in the **Mode** drop down box, then select a new **User name** and **Password**. Click the **Update** button to persist the new user name and password.

Extend Active Directory Domain Services (AD DS) to Azure

5/2/2018 • 8 min to read • Edit Online

This reference architecture shows how to extend your Active Directory environment to Azure to provide distributed authentication services using Active Directory Domain Services (AD DS). **Deploy this solution**.



Download a Visio file of this architecture.

AD DS is used to authenticate user, computer, application, or other identities that are included in a security domain. It can be hosted on-premises, but if your application is hosted partly on-premises and partly in Azure, it may be more efficient to replicate this functionality in Azure. This can reduce the latency caused by sending authentication and local authorization requests from the cloud back to AD DS running on-premises.

This architecture is commonly used when the on-premises network and the Azure virtual network are connected by a VPN or ExpressRoute connection. This architecture also supports bidirectional replication, meaning changes can be made either on-premises or in the cloud, and both sources will be kept consistent. Typical uses for this architecture include hybrid applications in which functionality is distributed between on-premises and Azure, and applications and services that perform authentication using Active Directory.

For additional considerations, see Choose a solution for integrating on-premises Active Directory with Azure.

Architecture

This architecture extends the architecture shown in DMZ between Azure and the Internet. It has the following components.

- **On-premises network**. The on-premises network includes local Active Directory servers that can perform authentication and authorization for components located on-premises.
- **Active Directory servers**. These are domain controllers implementing directory services (AD DS) running as VMs in the cloud. These servers can provide authentication of components running in your Azure virtual network.
- **Active Directory subnet**. The AD DS servers are hosted in a separate subnet. Network security group (NSG) rules protect the AD DS servers and provide a firewall against traffic from unexpected sources.
- **Azure Gateway and Active Directory synchronization**. The Azure gateway provides a connection

between the on-premises network and the Azure VNet. This can be a VPN connection or Azure ExpressRoute. All synchronization requests between the Active Directory servers in the cloud and on-premises pass through the gateway. User-defined routes (UDRs) handle routing for on-premises traffic that passes to Azure. Traffic to and from the Active Directory servers does not pass through the network virtual appliances (NVAs) used in this scenario.

For more information about configuring UDRs and the NVAs, see Implementing a secure hybrid network architecture in Azure.

Recommendations

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

VM recommendations

Determine your VM size requirements based on the expected volume of authentication requests. Use the specifications of the machines hosting AD DS on premises as a starting point, and match them with the Azure VM sizes. Once deployed, monitor utilization and scale up or down based on the actual load on the VMs. For more information about sizing AD DS domain controllers, see Capacity Planning for Active Directory Domain Services.

Create a separate virtual data disk for storing the database, logs, and SYSVOL for Active Directory. Do not store these items on the same disk as the operating system. Note that by default, data disks that are attached to a VM use write-through caching. However, this form of caching can conflict with the requirements of AD DS. For this reason, set the *Host Cache Preference* setting on the data disk to *None*. For more information, see Placement of the Windows Server AD DS database and SYSVOL.

Deploy at least two VMs running AD DS as domain controllers and add them to an availability set.

Networking recommendations

Configure the VM network interface (NIC) for each AD DS server with a static private IP address for full domain name service (DNS) support. For more information, see How to set a static private IP address in the Azure portal.

> **NOTE**
> Do not configure the VM NIC for any AD DS with a public IP address. See Security considerations for more details.

The Active Directory subnet NSG requires rules to permit incoming traffic from on-premises. For detailed information on the ports used by AD DS, see Active Directory and Active Directory Domain Services Port Requirements. Also, ensure the UDR tables do not route AD DS traffic through the NVAs used in this architecture.

Active Directory site

In AD DS, a site represents a physical location, network, or collection of devices. AD DS sites are used to manage AD DS database replication by grouping together AD DS objects that are located close to one another and are connected by a high speed network. AD DS includes logic to select the best strategy for replacating the AD DS database between sites.

We recommend that you create an AD DS site including the subnets defined for your application in Azure. Then, configure a site link between your on-premises AD DS sites, and AD DS will automatically perform the most efficient database replication possible. Note that this database replication requires little beyond the initial configuration.

Active Directory operations masters

The operations masters role can be assigned to AD DS domain controllers to support consistency checking

between instances of replicated AD DS databases. There are five operations master roles: schema master, domain naming master, relative identifier master, primary domain controller master emulator, and infrastructure master. For more information about these roles, see What are Operations Masters?.

We recommend you do not assign operations masters roles to the domain controllers deployed in Azure.

Monitoring

Monitor the resources of the domain controller VMs as well as the AD DS Services and create a plan to quickly correct any problems. For more information, see Monitoring Active Directory. You can also install tools such as Microsoft Systems Center on the monitoring server (see the architecture diagram) to help perform these tasks.

Scalability considerations

AD DS is designed for scalability. You don't need to configure a load balancer or traffic controller to direct requests to AD DS domain controllers. The only scalability consideration is to configure the VMs running AD DS with the correct size for your network load requirements, monitor the load on the VMs, and scale up or down as necessary.

Availability considerations

Deploy the VMs running AD DS into an availability set. Also, consider assigning the role of standby operations master to at least one server, and possibly more depending on your requirements. A standby operations master is an active copy of the operations master that can be used in place of the primary operations masters server during fail over.

Manageability considerations

Perform regular AD DS backups. Don't simply copy the VHD files of domain controllers instead of performing regular backups, because the AD DS database file on the VHD may not be in a consistent state when it's copied, making it impossible to restart the database.

Do not shut down a domain controller VM using Azure portal. Instead, shut down and restart from the guest operating system. Shutting down through the portal causes the VM to be deallocated, which resets both the `VM-GenerationID` and the `invocationID` of the Active Directory repository. This discards the AD DS relative identifier (RID) pool and marks SYSVOL as nonauthoritative, and may require reconfiguration of the domain controller.

Security considerations

AD DS servers provide authentication services and are an attractive target for attacks. To secure them, prevent direct Internet connectivity by placing the AD DS servers in a separate subnet with an NSG acting as a firewall. Close all ports on the AD DS servers except those necessary for authentication, authorization, and server synchronization. For more information, see Active Directory and Active Directory Domain Services Port Requirements.

Consider implementing an additional security perimeter around servers with a pair of subnets and NVAs, as described in Implementing a secure hybrid network architecture with Internet access in Azure.

Use either BitLocker or Azure disk encryption to encrypt the disk hosting the AD DS database.

Deploy the solution

A deployment for this architecture is available on GitHub. Note that the entire deployment can take up to two hours, which includes creating the VPN gateway and running the scripts that configure AD DS.

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Install Azure CLI 2.0.
3. Install the Azure building blocks npm package.
4. From a command prompt, bash prompt, or PowerShell prompt, log into your Azure account by using the command below.

```
az login
```

Deploy the simulated on-premises datacenter

1. Navigate to the `identity/adds-extend-domain` folder of the GitHub repository.
2. Open the `onprem.json` file. Search for instances of `adminPassword` and `Password` and add values for the passwords.
3. Run the following command and wait for the deployment to finish:

```
azbb -s <subscription_id> -g <resource group> -l <location> -p onprem.json --deploy
```

Deploy the Azure VNet

1. Open the `azure.json` file. Search for instances of `adminPassword` and `Password` and add values for the passwords.
2. In the same file, search for instances of `sharedKey` and enter shared keys for the VPN connection.

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

3. Run the following command and wait for the deployment to finish.

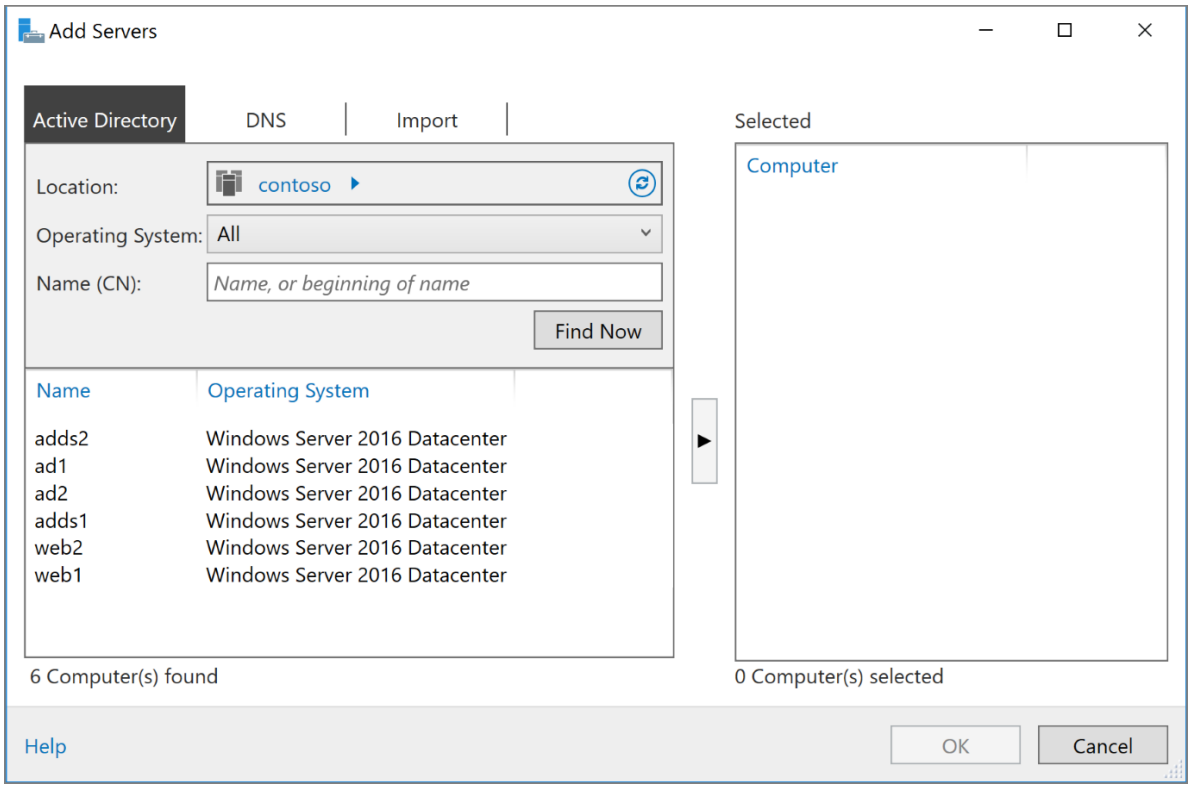
```
azbb -s <subscription_id> -g <resource group> -l <location> -p onoprem.json --deploy
```

   Deploy to the same resource group as the on-premises VNet.

Test connectivity with the Azure VNet

After deployment completes, you can test conectivity from the simulated on-premises environment to the Azure VNet.

1. Use the Azure portal, navigate to the resource group that you created.
2. Find the VM named `ra-onpremise-mgmt-vm1`.
3. Click `Connect` to open a remote desktop session to the VM. The username is `contoso\testuser`, and the password is the one that you specified in the `onprem.json` parameter file.
4. From inside your remote desktop session, open another remote desktop session to 10.0.4.4, which is the IP address of the VM named `adds-vm1`. The username is `contoso\testuser`, and the password is the one that you specified in the `azure.json` parameter file.
5. From inside the remote desktop session for `adds-vm1`, go to **Server Manager** and click **Add other servers to manage.**
6. In the **Active Directory** tab, click **Find now**. You should see a list of the AD, AD DS, and Web VMs.



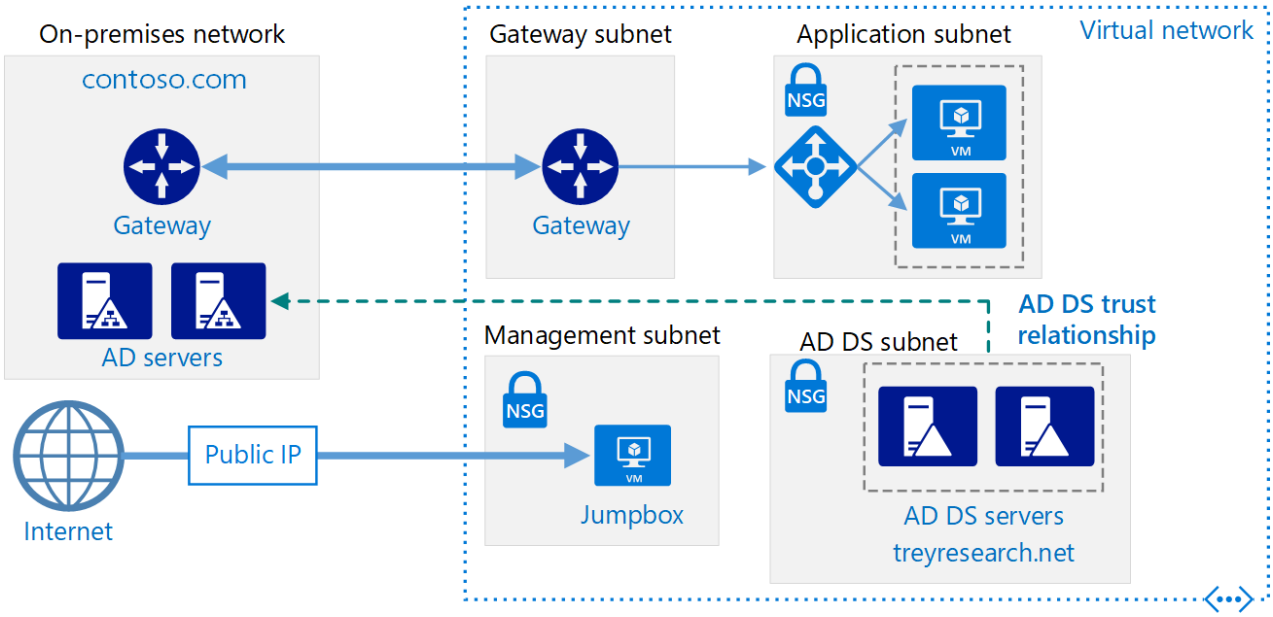
Next steps

- Learn the best practices for creating an AD DS resource forest in Azure.
- Learn the best practices for creating an Active Directory Federation Services (AD FS) infrastructure in Azure.

Create an Active Directory Domain Services (AD DS) resource forest in Azure

5/2/2018 • 6 min to read • Edit Online

This reference architecture shows how to create a separate Active Directory domain in Azure that is trusted by domains in your on-premises AD forest. **Deploy this solution**.



Download a Visio file of this architecture.

Active Directory Domain Services (AD DS) stores identity information in a hierarchical structure. The top node in the hierarchical structure is known as a forest. A forest contains domains, and domains contain other types of objects. This reference architecture creates an AD DS forest in Azure with a one-way outgoing trust relationship with an on-premises domain. The forest in Azure contains a domain that does not exist on-premises. Because of the trust relationship, logons made against on-premises domains can be trusted for access to resources in the separate Azure domain.

Typical uses for this architecture include maintaining security separation for objects and identities held in the cloud, and migrating individual domains from on-premises to the cloud.

For additional considerations, see Choose a solution for integrating on-premises Active Directory with Azure.

Architecture

The architecture has the following components.

- **On-premises network**. The on-premises network contains its own Active Directory forest and domains.
- **Active Directory servers**. These are domain controllers implementing domain services running as VMs in the cloud. These servers host a forest containing one or more domains, separate from those located on-premises.
- **One-way trust relationship**. The example in the diagram shows a one-way trust from the domain in Azure to the on-premises domain. This relationship enables on-premises users to access resources in the domain in Azure, but not the other way around. It is possible to create a two-way trust if cloud users also require access to on-premises resources.
- **Active Directory subnet**. The AD DS servers are hosted in a separate subnet. Network security group (NSG) rules protect the AD DS servers and provide a firewall against traffic from unexpected sources.

- **Azure gateway**. The Azure gateway provides a connection between the on-premises network and the Azure VNet. This can be a VPN connection or Azure ExpressRoute. For more information, see Implementing a secure hybrid network architecture in Azure.

Recommendations

For specific recommendations on implementing Active Directory in Azure, see the following articles:

- Extending Active Directory Domain Services (AD DS) to Azure.
- Guidelines for Deploying Windows Server Active Directory on Azure Virtual Machines.

Trust

The on-premises domains are contained within a different forest from the domains in the cloud. To enable authentication of on-premises users in the cloud, the domains in Azure must trust the logon domain in the on-premises forest. Similarly, if the cloud provides a logon domain for external users, it may be necessary for the on-premises forest to trust the cloud domain.

You can establish trusts at the forest level by creating forest trusts, or at the domain level by creating external trusts. A forest level trust creates a relationship between all domains in two forests. An external domain level trust only creates a relationship between two specified domains. You should only create external domain level trusts between domains in different forests.

Trusts can be unidirectional (one-way) or bidirectional (two-way):

- A one-way trust enables users in one domain or forest (known as the *incoming* domain or forest) to access the resources held in another (the *outgoing* domain or forest).
- A two-way trust enables users in either domain or forest to access resources held in the other.

The following table summarizes trust configurations for some simple scenarios:

SCENARIO	ON-PREMISES TRUST	CLOUD TRUST
On-premises users require access to resources in the cloud, but not vice versa	One-way, incoming	One-way, outgoing
Users in the cloud require access to resources located on-premises, but not vice versa	One-way, outgoing	One-way, incoming
Users in the cloud and on-premises both requires access to resources held in the cloud and on-premises	Two-way, incoming and outgoing	Two-way, incoming and outgoing

Scalability considerations

Active Directory is automatically scalable for domain controllers that are part of the same domain. Requests are distributed across all controllers within a domain. You can add another domain controller, and it synchronizes automatically with the domain. Do not configure a separate load balancer to direct traffic to controllers within the domain. Ensure that all domain controllers have sufficient memory and storage resources to handle the domain database. Make all domain controller VMs the same size.

Availability considerations

Provision at least two domain controllers for each domain. This enables automatic replication between servers. Create an availability set for the VMs acting as Active Directory servers handling each domain. Put at least two

servers in this availability set.

Also, consider designating one or more servers in each domain as standby operations masters in case connectivity to a server acting as a flexible single master operation (FSMO) role fails.

Manageability considerations

For information about management and monitoring considerations, see Extending Active Directory to Azure.

For additional information, see Monitoring Active Directory. You can install tools such as Microsoft Systems Center on a monitoring server in the management subnet to help perform these tasks.

Security considerations

Forest level trusts are transitive. If you establish a forest level trust between an on-premises forest and a forest in the cloud, this trust is extended to other new domains created in either forest. If you use domains to provide separation for security purposes, consider creating trusts at the domain level only. Domain level trusts are non-transitive.

For Active Directory-specific security considerations, see the security considerations section in Extending Active Directory to Azure.

Deploy the solution

A deployment for this architecture is available on GitHub. Note that the entire deployment can take up to two hours, which includes creating the VPN gateway and running the scripts that configure AD DS.

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Install Azure CLI 2.0.
3. Install the Azure building blocks npm package.
4. From a command prompt, bash prompt, or PowerShell prompt, log into your Azure account by using the command below.

```
az login
```

Deploy the simulated on-premises datacenter

1. Navigate to the `identity/adds-forest` folder of the GitHub repository.
2. Open the `onprem.json` file. Search for instances of `adminPassword` and `Password` and add values for the passwords.
3. Run the following command and wait for the deployment to finish:

```
azbb -s <subscription_id> -g <resource group> -l <location> -p onprem.json --deploy
```

Deploy the Azure VNet

1. Open the `azure.json` file. Search for instances of `adminPassword` and `Password` and add values for the passwords.
2. In the same file, search for instances of `sharedKey` and enter shared keys for the VPN connection.

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

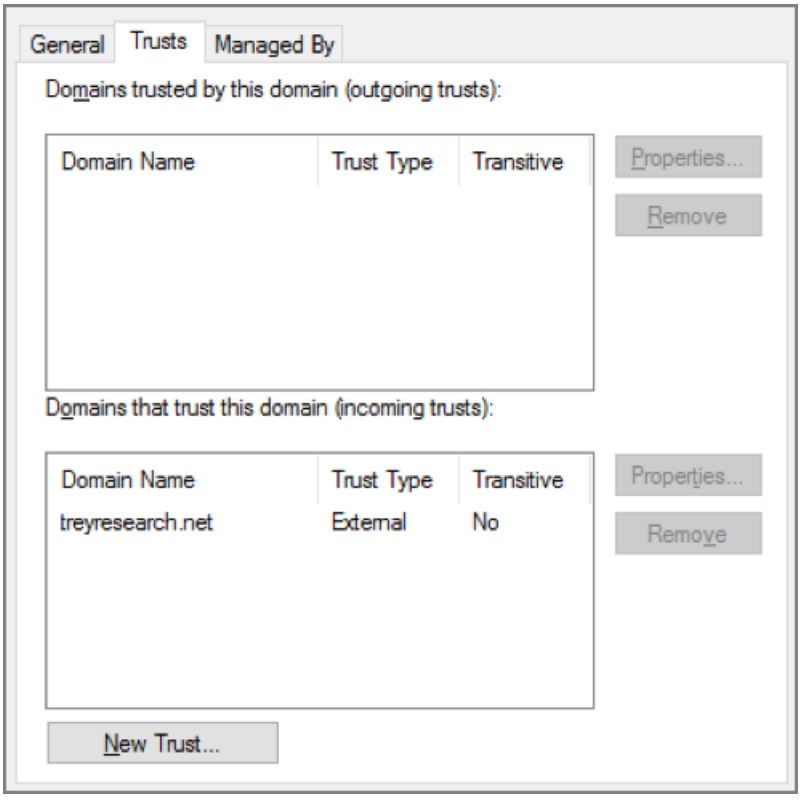
3. Run the following command and wait for the deployment to finish.

```
azbb -s <subscription_id> -g <resource group> -l <location> -p onoprem.json --deploy
```

   Deploy to the same resource group as the on-premises VNet.

Test the AD trust relation

1. Use the Azure portal, navigate to the resource group that you created.
2. Use the Azure portal to find the VM named `ra-adt-mgmt-vm1`.
3. Click `Connect` to open a remote desktop session to the VM. The username is `contoso\testuser`, and the password is the one that you specified in the `onprem.json` parameter file.
4. From inside your remote desktop session, open another remote desktop session to 192.168.0.4, which is the IP address of the VM named `ra-adtrust-onpremise-ad-vm1`. The username is `contoso\testuser`, and the password is the one that you specified in the `azure.json` parameter file.
5. From inside the remote desktop session for `ra-adtrust-onpremise-ad-vm1`, go to **Server Manager** and click **Tools** > **Active Directory Domains and Trusts**.
6. In the left pane, right-click on the contoso.com and select **Properties**.
7. Click the **Trusts** tab. You should see treyresearch.net listed as an incoming trust.



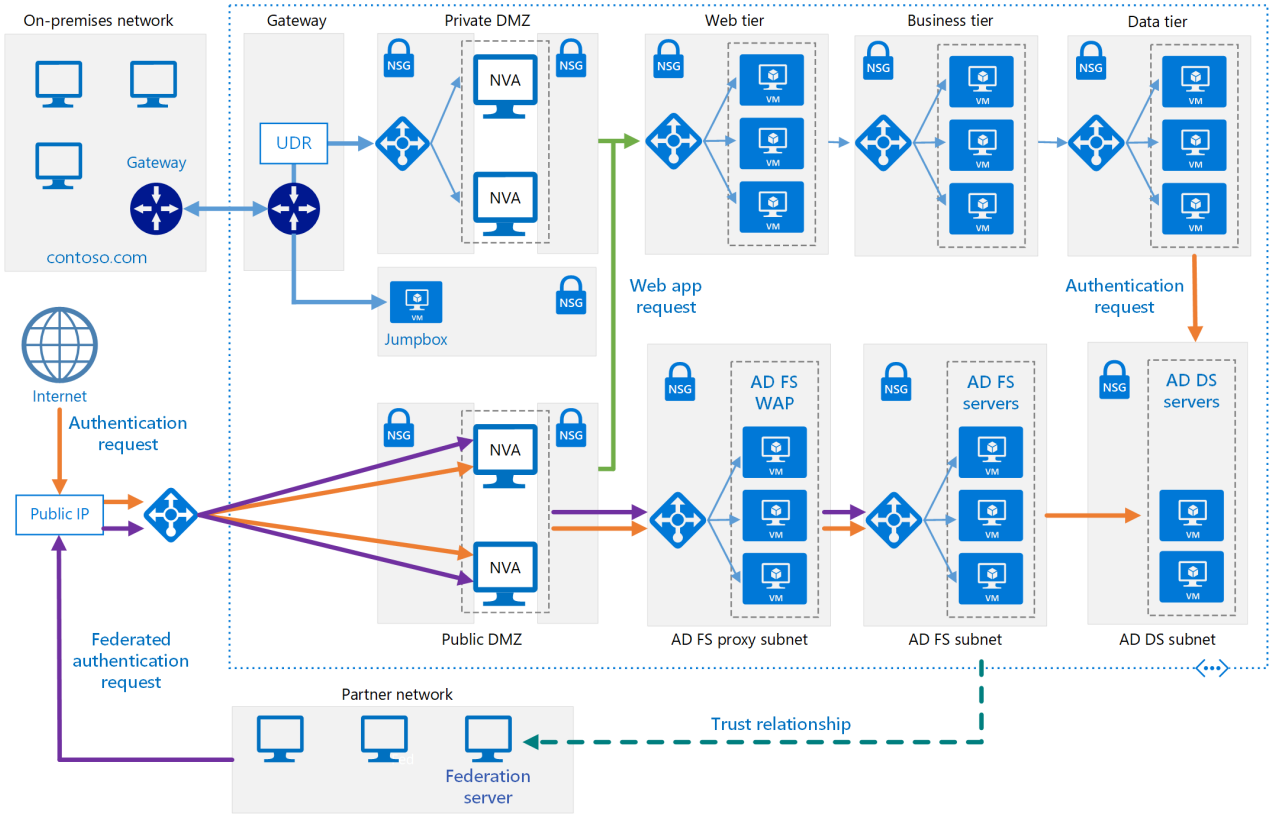
Next steps

- Learn the best practices for extending your on-premises AD DS domain to Azure
- Learn the best practices for creating an AD FS infrastructure in Azure.

Extend Active Directory Federation Services (AD FS) to Azure

4/11/2018 • 15 min to read • Edit Online

This reference architecture implements a secure hybrid network that extends your on-premises network to Azure and uses Active Directory Federation Services (AD FS) to perform federated authentication and authorization for components running in Azure. **Deploy this solution**.



Download a Visio file of this architecture.

AD FS can be hosted on-premises, but if your application is a hybrid in which some parts are implemented in Azure, it may be more efficient to replicate AD FS in the cloud.

The diagram shows the following scenarios:

- Application code from a partner organization accesses a web application hosted inside your Azure VNet.
- An external, registered user with credentials stored inside Active Directory Domain Services (DS) accesses a web application hosted inside your Azure VNet.
- A user connected to your VNet using an authorized device executes a web application hosted inside your Azure VNet.

Typical uses for this architecture include:

- Hybrid applications where workloads run partly on-premises and partly in Azure.
- Solutions that use federated authorization to expose web applications to partner organizations.
- Systems that support access from web browsers running outside of the organizational firewall.
- Systems that enable users to access to web applications by connecting from authorized external devices such as remote computers, notebooks, and other mobile devices.

This reference architecture focuses on *passive federation*, in which the federation servers decide how and when to authenticate a user. The user provides sign in information when the application is started. This mechanism is most commonly used by web browsers and involves a protocol that redirects the browser to a site where the user authenticates. AD FS also supports *active federation*, where an application takes on responsibility for supplying credentials without further user interaction, but that scenario is outside the scope of this architecture.

For additional considerations, see Choose a solution for integrating on-premises Active Directory with Azure.

Architecture

This architecture extends the implementation described in Extending AD DS to Azure. It contains the followign components.

- **AD DS subnet**. The AD DS servers are contained in their own subnet with network security group (NSG) rules acting as a firewall.
- **AD DS servers**. Domain controllers running as VMs in Azure. These servers provide authentication of local identities within the domain.
- **AD FS subnet**. The AD FS servers are located within their own subnet with NSG rules acting as a firewall.
- **AD FS servers**. The AD FS servers provide federated authorization and authentication. In this architecture, they perform the following tasks:
  - Receiving security tokens containing claims made by a partner federation server on behalf of a partner user. AD FS verifies that the tokens are valid before passing the claims to the web application running in Azure to authorize requests.

    The web application running in Azure is the *relying party*. The partner federation server must issue claims that are understood by the web application. The partner federation servers are referred to as *account partners*, because they submit access requests on behalf of authenticated accounts in the partner organization. The AD FS servers are called *resource partners* because they provide access to resources (the web application).
  - Authenticating and authorizing incoming requests from external users running a web browser or device that needs access to web applications, by using AD DS and the Active Directory Device Registration Service.

  The AD FS servers are configured as a farm accessed through an Azure load balancer. This implementation improves availability and scalability. The AD FS servers are not exposed directly to the Internet. All Internet traffic is filtered through AD FS web application proxy servers and a DMZ (also referred to as a perimeter network).

  For more information about how AD FS works, see Active Directory Federation Services Overview. Also, the article AD FS deployment in Azure contains a detailed step-by-step introduction to implementation.
- **AD FS proxy subnet**. The AD FS proxy servers can be contained within their own subnet, with NSG rules providing protection. The servers in this subnet are exposed to the Internet through a set of network virtual appliances that provide a firewall between your Azure virtual network and the Internet.
- **AD FS web application proxy (WAP) servers**. These VMs act as AD FS servers for incoming requests from partner organizations and external devices. The WAP servers act as a filter, shielding the AD FS servers from direct access from the Internet. As with the AD FS servers, deploying the WAP servers in a farm with load balancing gives you greater availability and scalability than deploying a collection of stand-alone servers.

> **NOTE**
> For detailed information about installing WAP servers, see Install and Configure the Web Application Proxy Server

- **Partner organization**. A partner organization running a web application that requests access to a web application running in Azure. The federation server at the partner organization authenticates requests locally, and submits security tokens containing claims to AD FS running in Azure. AD FS in Azure validates the security tokens, and if valid can pass the claims to the web application running in Azure to authorize them.

> **NOTE**
> You can also configure a VPN tunnel using Azure gateway to provide direct access to AD FS for trusted partners. Requests received from these partners do not pass through the WAP servers.

For more information about the parts of the architecture that are not related to AD FS, see the following:

- Implementing a secure hybrid network architecture in Azure
- Implementing a secure hybrid network architecture with Internet access in Azure
- Implementing a secure hybrid network architecture with Active Directory identities in Azure.

Recommendations

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

VM recommendations

Create VMs with sufficient resources to handle the expected volume of traffic. Use the size of the existing machines hosting AD FS on premises as a starting point. Monitor the resource utilization. You can resize the VMs and scale down if they are too large.

Follow the recommendations listed in Running a Windows VM on Azure.

Networking recommendations

Configure the network interface for each of the VMs hosting AD FS and WAP servers with static private IP addresses.

Do not give the AD FS VMs public IP addresses. For more information, see the Security considerations section.

Set the IP address of the preferred and secondary domain name service (DNS) servers for the network interfaces for each AD FS and WAP VM to reference the Active Directory DS VMs. The Active Directory DS VMS should be running DNS. This step is necessary to enable each VM to join the domain.

AD FS availability

Create an AD FS farm with at least two servers to increase availability of the service. Use different storage accounts for each AD FS VM in the farm. This approach helps to ensure that a failure in a single storage account does not make the entire farm inaccessible.

> **IMPORTANT**
> We recommend the use of managed disks. Managed disks do not require a storage account. You simply specify the size and type of disk and it is deployed in a highly available way. Our reference architectures do not currently deploy managed disks but the template building blocks will be updated to deploy managed disks in version 2.

Create separate Azure availability sets for the AD FS and WAP VMs. Ensure that there are at least two VMs in each set. Each availability set must have at least two update domains and two fault domains.

Configure the load balancers for the AD FS VMs and WAP VMs as follows:

- Use an Azure load balancer to provide external access to the WAP VMs, and an internal load balancer to distribute the load across the AD FS servers in the farm.
- Only pass traffic appearing on port 443 (HTTPS) to the AD FS/WAP servers.
- Give the load balancer a static IP address.
- Create a health probe using the TCP protocol rather than HTTPS. You can ping port 443 to verify that an AD FS server is functioning.

  > **NOTE**
  > AD FS servers use the Server Name Indication (SNI) protocol, so attempting to probe using an HTTPS endpoint from the load balancer fails.

- Add a DNS *A* record to the domain for the AD FS load balancer. Specify the IP address of the load balancer, and give it a name in the domain (such as adfs.contoso.com). This is the name clients and the WAP servers use to access the AD FS server farm.

AD FS security

Prevent direct exposure of the AD FS servers to the Internet. AD FS servers are domain-joined computers that have full authorization to grant security tokens. If a server is compromised, a malicious user can issue full access tokens to all web applications and to all federation servers that are protected by AD FS. If your system must handle requests from external users not connecting from trusted partner sites, use WAP servers to handle these requests. For more information, see Where to Place a Federation Server Proxy.

Place AD FS servers and WAP servers in separate subnets with their own firewalls. You can use NSG rules to define firewall rules. If you require more comprehensive protection you can implement an additional security perimeter around servers by using a pair of subnets and network virtual appliances (NVAs), as described in the document Implementing a secure hybrid network architecture with Internet access in Azure. All firewalls should allow traffic on port 443 (HTTPS).

Restrict direct sign in access to the AD FS and WAP servers. Only DevOps staff should be able to connect.

Do not join the WAP servers to the domain.

AD FS installation

The article Deploying a Federation Server Farm provides detailed instructions for installing and configuring AD FS. Perform the following tasks before configuring the first AD FS server in the farm:

1. Obtain a publicly trusted certificate for performing server authentication. The *subject name* must contain the name clients use to access the federation service. This can be the DNS name registered for the load balancer, for example, *adfs.contoso.com* (avoid using wildcard names such as **.contoso.com*, for security reasons). Use the same certificate on all AD FS server VMs. You can purchase a certificate from a trusted certification authority, but if your organization uses Active Directory Certificate Services you can create your own.

   The *subject alternative name* is used by the device registration service (DRS) to enable access from external devices. This should be of the form *enterpriseregistration.contoso.com*.

   For more information, see Obtain and Configure a Secure Sockets Layer (SSL) Certificate for AD FS.

2. On the domain controller, generate a new root key for the Key Distribution Service. Set the effective time to

the current time minus 10 hours (this configuration reduces the delay that can occur in distributing and synchronizing keys across the domain). This step is necessary to support creating the group service account that is used to run the AD FS service. The following PowerShell command shows an example of how to do this:

```
Add-KdsRootKey -EffectiveTime (Get-Date).AddHours(-10)
```

3. Add each AD FS server VM to the domain.

> **NOTE**
> To install AD FS, the domain controller running the primary domain controller (PDC) emulator flexible single master operation (FSMO) role for the domain must be running and accessible from the AD FS VMs. <<RBC: Is there a way to make this less repetitive?>>

AD FS trust

Establish federation trust between your AD FS installation, and the federation servers of any partner organizations. Configure any claims filtering and mapping required.

- DevOps staff at each partner organization must add a relying party trust for the web applications accessible through your AD FS servers.
- DevOps staff in your organization must configure claims-provider trust to enable your AD FS servers to trust the claims that partner organizations provide.
- DevOps staff in your organization must also configure AD FS to pass claims on to your organization's web applications.

For more information, see Establishing Federation Trust.

Publish your organization's web applications and make them available to external partners by using preauthentication through the WAP servers. For more information, see Publish Applications using AD FS Preauthentication

AD FS supports token transformation and augmentation. Azure Active Directory does not provide this feature. With AD FS, when you set up the trust relationships, you can:

- Configure claim transformations for authorization rules. For example, you can map group security from a representation used by a non-Microsoft partner organization to something that that Active Directory DS can authorize in your organization.
- Transform claims from one format to another. For example, you can map from SAML 2.0 to SAML 1.1 if your application only supports SAML 1.1 claims.

AD FS monitoring

The Microsoft System Center Management Pack for Active Directory Federation Services 2012 R2 provides both proactive and reactive monitoring of your AD FS deployment for the federation server. This management pack monitors:

- Events that the AD FS service records in its event logs.
- The performance data that the AD FS performance counters collect.
- The overall health of the AD FS system and web applications (relying parties), and provides alerts for critical issues and warnings.

Scalability considerations

The following considerations, summarized from the article Plan your AD FS deployment, give a starting point for

sizing AD FS farms:

- If you have fewer than 1000 users, do not create dedicated servers, but instead install AD FS on each of the Active Directory DS servers in the cloud. Make sure that you have at least two Active Directory DS servers to maintain availability. Create a single WAP server.
- If you have between 1000 and 15000 users, create two dedicated AD FS servers and two dedicated WAP servers.
- If you have between 15000 and 60000 users, create between three and five dedicated AD FS servers and at least two dedicated WAP servers.

These considerations assume that you are using dual quad-core VM (Standard D4_v2, or better) sizes in Azure.

If you are using the Windows Internal Database to store AD FS configuration data, you are limited to eight AD FS servers in the farm. If you anticipate that you will need more in the future, use SQL Server. For more information, see The Role of the AD FS Configuration Database.

Availability considerations

You can use either SQL Server or the Windows Internal Database to hold AD FS configuration information. The Windows Internal Database provides basic redundancy. Changes are written directly to only one of the AD FS databases in the AD FS cluster, while the other servers use pull replication to keep their databases up to date. Using SQL Server can provide full database redundancy and high availability using failover clustering or mirroring.

Manageability considerations

DevOps staff should be prepared to perform the following tasks:

- Managing the federation servers, including managing the AD FS farm, managing trust policy on the federation servers, and managing the certificates used by the federation services.
- Managing the WAP servers including managing the WAP farm and certificates.
- Managing web applications including configuring relying parties, authentication methods, and claims mappings.
- Backing up AD FS components.

Security considerations

AD FS utilizes the HTTPS protocol, so make sure that the NSG rules for the subnet containing the web tier VMs permit HTTPS requests. These requests can originate from the on-premises network, the subnets containing the web tier, business tier, data tier, private DMZ, public DMZ, and the subnet containing the AD FS servers.

Consider using a set of network virtual appliances that logs detailed information on traffic traversing the edge of your virtual network for auditing purposes.

Deploy the solution

A solution is available on GitHub to deploy this reference architecture. You will need the latest version of the Azure CLI to run the Powershell script that deploys the solution. To deploy the reference architecture, follow these steps:

1. Download or clone the solution folder from GitHub to your local machine.
2. Open the Azure CLI and navigate to the local solution folder.
3. Run the following command:

```
.\Deploy-ReferenceArchitecture.ps1 <subscription id> <location> <mode>
```

Replace `<subscription id>` with your Azure subscription ID.

For `<location>`, specify an Azure region, such as `eastus` or `westus`.

The `<mode>` parameter controls the granularity of the deployment, and can be one of the following values:

- `Onpremise`: Deploys a simulated on-premises environment. You can use this deployment to test and experiment if you do not have an existing on-premises network, or if you want to test this reference architecture without changing the configuration of your existing on-premises network.
- `Infrastructure`: deploys the VNet infrastructure and jump box.
- `CreateVpn`: deploys an Azure virtual network gateway and connects it to the simulated on-premises network.
- `AzureADDS`: deploys the VMs acting as Active Directory DS servers, deploys Active Directory to these VMs, and creates the domain in Azure.
- `AdfsVm`: deploys the AD FS VMs and joins them to the domain in Azure.
- `PublicDMZ`: deploys the public DMZ in Azure.
- `ProxyVm`: deploys the AD FS proxy VMs and joins them to the domain in Azure.
- `Prepare`: deploys all of the preceding deployments. **This is the recommended option if you are building an entirely new deployment and you don't have an existing on-premises infrastructure.**
- `Workload`: optionally deploys web, business, and data tier VMs and supporting network. Not included in the `Prepare` deployment mode.
- `PrivateDMZ`: optionally deploys the private DMZ in Azure in front of the `Workload` VMs deployed above. Not included in the `Prepare` deployment mode.

4. Wait for the deployment to complete. If you used the `Prepare` option, the deployment takes several hours to complete, and finishes with the message
`Preparation is completed. Please install certificate to all AD FS and proxy VMs.`

5. Restart the jump box (*ra-adfs-mgmt-vm1* in the *ra-adfs-security-rg* group) to allow its DNS settings to take effect.

6. Obtain an SSL Certificate for AD FS and install this certificate on the AD FS VMs. Note that you can connect to them through the jump box. The IP addresses are *10.0.5.4* and *10.0.5.5*. The default username is *contoso\testuser* with password *AweSome@PW*.

   > **NOTE**
   > The comments in the Deploy-ReferenceArchitecture.ps1 script at this point provides detailed instructions for creating a self-signed test certificate and authority using the `makecert` command. However, perform these steps as a **test** only and do not use the certificates generated by makecert in a production environment.

7. Run the following PowerShell command to deploy the AD FS server farm:

```
.\Deploy-ReferenceArchitecture.ps1 <subscription id> <location> Adfs
```

8. On the jump box, browse to `https://adfs.contoso.com/adfs/ls/idpinitiatedsignon.htm` to test the AD FS installation (you may receive a certificate warning that you can ignore for this test). Verify that the Contoso Corporation sign-in page appears. Sign in as *contoso\testuser* with password *AweS0me@PW*.

9. Install the SSL certificate on the AD FS proxy VMs. The IP addresses are *10.0.6.4* and *10.0.6.5*.
10. Run the following PowerShell command to deploy the first AD FS proxy server:

```
.\Deploy-ReferenceArchitecture.ps1 <subscription id> <location> Proxy1
```

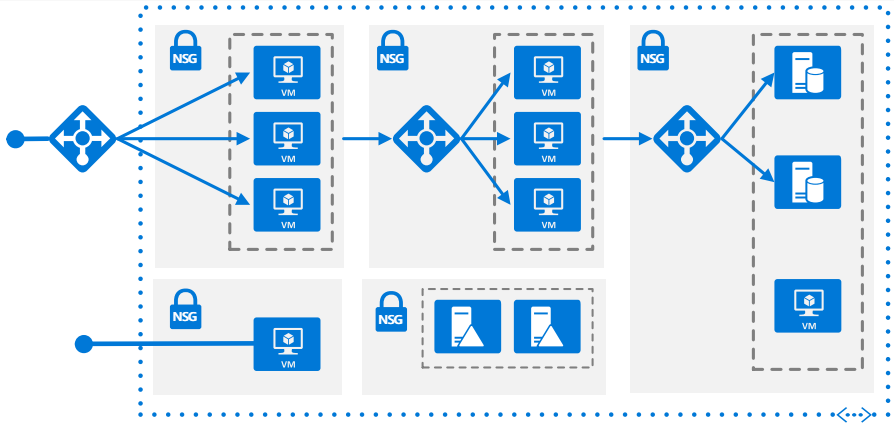
11. Follow the instructions displayed by the script to test the installation of the first proxy server.
12. Run the following PowerShell command to deploy the second proxy server:

```
.\Deploy-ReferenceArchitecture.ps1 <subscription id> <location> Proxy2
```

13. Follow the instructions displayed by the script to test the complete proxy configuration.

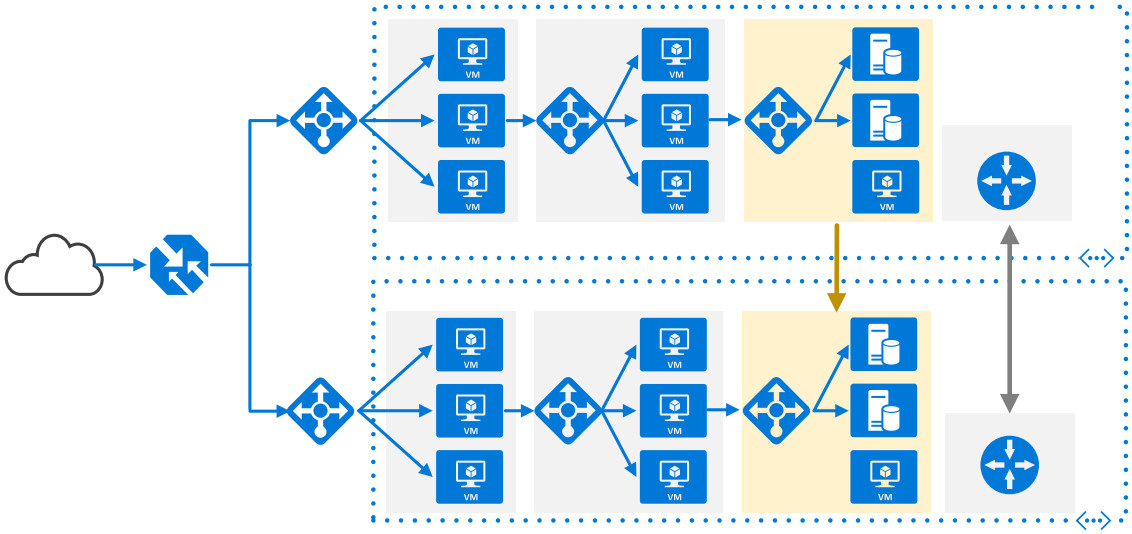
Next steps

- Learn about Azure Active Directory.
- Learn about Azure Active Directory B2C.



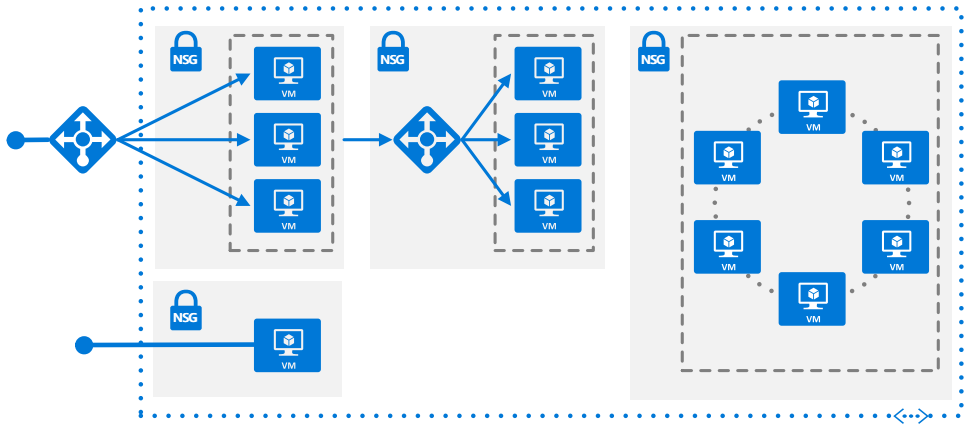
N-tier application with SQL Server

Deploy VMs and a virtual network configured for an N-tier application using SQL Server on Windows.



Multi-region N-tier application with SQL Server

Deploy an N-tier application to two regions for high availability, using SQL Server Always On Availability Groups.



N-tier application with Cassandra

Deploy Linux VMs and a virtual network configured for an N-tier application using Apache Cassandra.

Windows VM

Baseline recommendations for running a Windows VM in Azure.

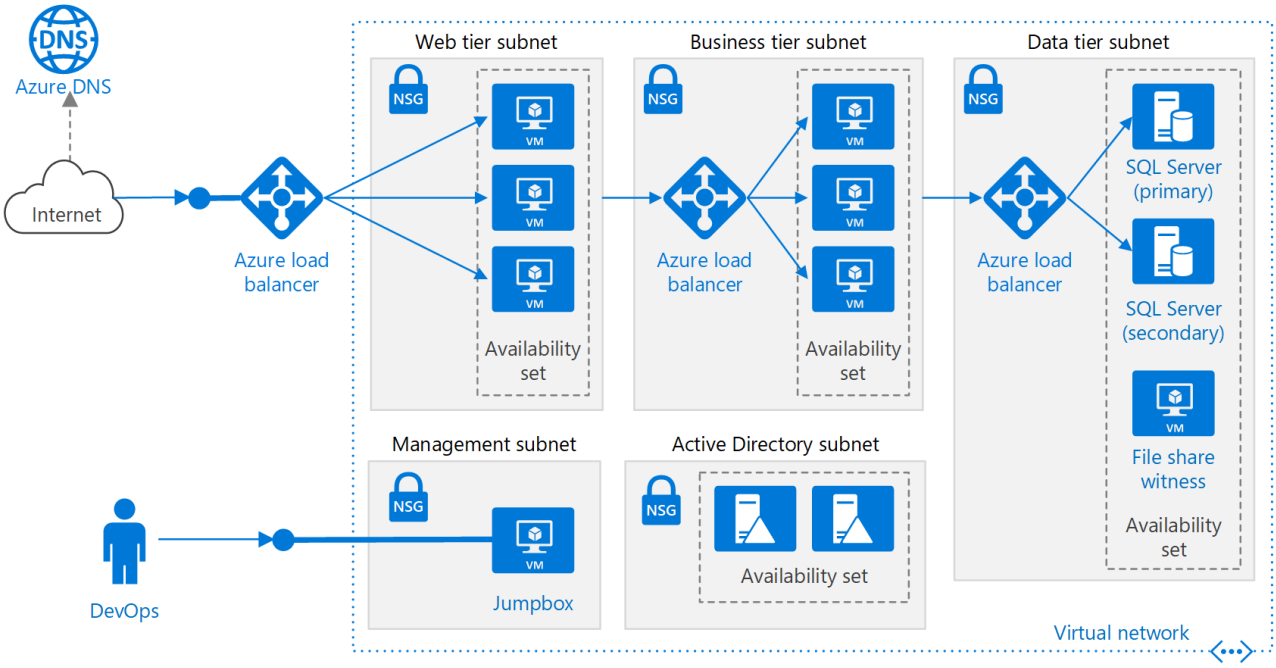
Linux VM

Baseline recommendations for running a Linux VM in Azure.

N-tier application with SQL Server

5/4/2018 • 11 min to read • Edit Online

This reference architecture shows how to deploy VMs and a virtual network configured for an N-tier application, using SQL Server on Windows for the data tier. **Deploy this solution**.



Download a Visio file of this architecture.

Architecture

The architecture has the following components:

- **Resource group.** Resource groups are used to group resources so they can be managed by lifetime, owner, or other criteria.
- **Virtual network (VNet) and subnets.** Every Azure VM is deployed into a VNet that can be segmented into multiple subnets. Create a separate subnet for each tier.
- **NSGs.** Use network security groups (NSGs) to restrict network traffic within the VNet. For example, in the 3-tier architecture shown here, the database tier does not accept traffic from the web front end, only from the business tier and the management subnet.
- **Virtual machines**. For recommendations on configuring VMs, see Run a Windows VM on Azure and Run a Linux VM on Azure.
- **Availability sets.** Create an availability set for each tier, and provision at least two VMs in each tier. This makes the VMs eligible for a higher service level agreement (SLA) for VMs.
- **VM scale set** (not shown). A VM scale set is an alternative to using an availability set. A scale sets makes it easy to scale out the VMs in a tier, either manually or automatically based on predefined rules.
- **Azure Load balancers.** The load balancers distribute incoming Internet requests to the VM instances. Use a public load balancer to distribute incoming Internet traffic to the web tier, and an internal load balancer to distribute network traffic from the web tier to the business tier.

- **Public IP address**. A public IP address is needed for the public load balancer to receive Internet traffic.
- **Jumpbox.** Also called a bastion host. A secure VM on the network that administrators use to connect to the other VMs. The jumpbox has an NSG that allows remote traffic only from public IP addresses on a safe list. The NSG should permit remote desktop (RDP) traffic.
- **SQL Server Always On Availability Group.** Provides high availability at the data tier, by enabling replication and failover.
- **Active Directory Domain Services (AD DS) Servers**. The SQL Server Always On Availability Groups are joined to a domain, to enable the Windows Server Failover Cluster (WSFC) technology for failover.
- **Azure DNS**. Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services.

Recommendations

Your requirements might differ from the architecture described here. Use these recommendations as a starting point.

VNet / Subnets

When you create the VNet, determine how many IP addresses your resources in each subnet require. Specify a subnet mask and a VNet address range large enough for the required IP addresses, using CIDR notation. Use an address space that falls within the standard private IP address blocks, which are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.

Choose an address range that does not overlap with your on-premises network, in case you need to set up a gateway between the VNet and your on-premise network later. Once you create the VNet, you can't change the address range.

Design subnets with functionality and security requirements in mind. All VMs within the same tier or role should go into the same subnet, which can be a security boundary. For more information about designing VNets and subnets, see Plan and design Azure Virtual Networks.

Load balancers

Do not expose the VMs directly to the Internet, but instead give each VM a private IP address. Clients connect using the IP address of the public load balancer.

Define load balancer rules to direct network traffic to the VMs. For example, to enable HTTP traffic, create a rule that maps port 80 from the front-end configuration to port 80 on the back-end address pool. When a client sends an HTTP request to port 80, the load balancer selects a back-end IP address by using a hashing algorithm that includes the source IP address. In that way, client requests are distributed across all the VMs.

Network security groups

Use NSG rules to restrict traffic between tiers. For example, in the 3-tier architecture shown above, the web tier does not communicate directly with the database tier. To enforce this, the database tier should block incoming traffic from the web tier subnet.

1. Deny all inbound traffic from the VNet. (Use the `VIRTUAL_NETWORK` tag in the rule.)
2. Allow inbound traffic from the business tier subnet.
3. Allow inbound traffic from the database tier subnet itself. This rule allows communication between the database VMs, which is needed for database replication and failover.
4. Allow RDP traffic (port 3389) from the jumpbox subnet. This rule lets administrators connect to the database tier from the jumpbox.

Create rules 2 – 4 with higher priority than the first rule, so they override it.

SQL Server Always On Availability Groups

We recommend Always On Availability Groups for SQL Server high availability. Prior to Windows Server 2016, Always On Availability Groups require a domain controller, and all nodes in the availability group must be in the same AD domain.

Other tiers connect to the database through an availability group listener. The listener enables a SQL client to connect without knowing the name of the physical instance of SQL Server. VMs that access the database must be joined to the domain. The client (in this case, another tier) uses DNS to resolve the listener's virtual network name into IP addresses.

Configure the SQL Server Always On Availability Group as follows:

1. Create a Windows Server Failover Clustering (WSFC) cluster, a SQL Server Always On Availability Group, and a primary replica. For more information, see Getting Started with Always On Availability Groups.
2. Create an internal load balancer with a static private IP address.
3. Create an availability group listener, and map the listener's DNS name to the IP address of an internal load balancer.
4. Create a load balancer rule for the SQL Server listening port (TCP port 1433 by default). The load balancer rule must enable *floating IP*, also called Direct Server Return. This causes the VM to reply directly to the client, which enables a direct connection to the primary replica.

   > **NOTE**
   > When floating IP is enabled, the front-end port number must be the same as the back-end port number in the load balancer rule.

When a SQL client tries to connect, the load balancer routes the connection request to the primary replica. If there is a failover to another replica, the load balancer automatically routes subsequent requests to a new primary replica. For more information, see Configure an ILB listener for SQL Server Always On Availability Groups.

During a failover, existing client connections are closed. After the failover completes, new connections will be routed to the new primary replica.

If your application makes significantly more reads than writes, you can offload some of the read-only queries to a secondary replica. See Using a Listener to Connect to a Read-Only Secondary Replica (Read-Only Routing).

Test your deployment by forcing a manual failover of the availability group.

Jumpbox

Do not allow RDP access from the public Internet to the VMs that run the application workload. Instead, all RDP access to these VMs must come through the jumpbox. An administrator logs into the jumpbox, and then logs into the other VM from the jumpbox. The jumpbox allows RDP traffic from the Internet, but only from known, safe IP addresses.

The jumpbox has minimal performance requirements, so select a small VM size. Create a public IP address for the jumpbox. Place the jumpbox in the same VNet as the other VMs, but in a separate management subnet.

To secure the jumpbox, add an NSG rule that allows RDP connections only from a safe set of public IP addresses. Configure the NSGs for the other subnets to allow RDP traffic from the management subnet.

Scalability considerations

VM scale sets help you to deploy and manage a set of identical VMs. Scale sets support autoscaling based on performance metrics. As the load on the VMs increases, additional VMs are automatically added to the load balancer. Consider scale sets if you need to quickly scale out VMs, or need to autoscale.

There are two basic ways to configure VMs deployed in a scale set:

- Use extensions to configure the VM after it is provisioned. With this approach, new VM instances may take longer to start up than a VM with no extensions.
- Deploy a managed disk with a custom disk image. This option may be quicker to deploy. However, it requires you to keep the image up to date.

For additional considerations, see Design considerations for scale sets.

> **TIP**
> When using any autoscale solution, test it with production-level workloads well in advance.

Each Azure subscription has default limits in place, including a maximum number of VMs per region. You can increase the limit by filing a support request. For more information, see Azure subscription and service limits, quotas, and constraints.

Availability considerations

If you are not using VM scale sets, put VMs in the same tier into an availability set. Create at least two VMs in the availability set to support the availability SLA for Azure VMs. For more information, see Manage the availability of virtual machines.

The load balancer uses health probes to monitor the availability of VM instances. If a probe cannot reach an instance within a timeout period, the load balancer stops sending traffic to that VM. However, the load balancer will continue to probe, and if the VM becomes available again, the load balancer resumes sending traffic to that VM.

Here are some recommendations on load balancer health probes:

- Probes can test either HTTP or TCP. If your VMs run an HTTP server, create an HTTP probe. Otherwise create a TCP probe.
- For an HTTP probe, specify the path to an HTTP endpoint. The probe checks for an HTTP 200 response from this path. This can be the root path ("/"), or a health-monitoring endpoint that implements some custom logic to check the health of the application. The endpoint must allow anonymous HTTP requests.
- The probe is sent from a known IP address, 168.63.129.16. Make sure you don't block traffic to or from this IP address in any firewall policies or network security group (NSG) rules.
- Use health probe logs to view the status of the health probes. Enable logging in the Azure portal for each load balancer. Logs are written to Azure Blob storage. The logs show how many VMs on the back end are not receiving network traffic due to failed probe responses.

If you need higher availability than the Azure SLA for VMs provides, consider replication the application across two regions, using Azure Traffic Manager for failover. For more information, see Multi-region N-tier application for high availability.

Security considerations

Virtual networks are a traffic isolation boundary in Azure. VMs in one VNet cannot communicate directly with VMs in a different VNet. VMs within the same VNet can communicate, unless you create network security groups (NSGs) to restrict traffic. For more information, see Microsoft cloud services and network security.

For incoming Internet traffic, the load balancer rules define which traffic can reach the back end. However, load balancer rules don't support IP safe lists, so if you want to add certain public IP addresses to a safe list, add an NSG to the subnet.

Consider adding a network virtual appliance (NVA) to create a DMZ between the Internet and the Azure virtual network. NVA is a generic term for a virtual appliance that can perform network-related tasks, such as firewall, packet inspection, auditing, and custom routing. For more information, see Implementing a DMZ between Azure and the Internet.

Encrypt sensitive data at rest and use Azure Key Vault to manage the database encryption keys. Key Vault can store encryption keys in hardware security modules (HSMs). For more information, see Configure Azure Key Vault Integration for SQL Server on Azure VMs. It's also recommended to store application secrets, such as database connection strings, in Key Vault.

Deploy the solution

A deployment for this reference architecture is available on GitHub.

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Make sure you have the Azure CLI 2.0 installed on your computer. To install the CLI, follow the instructions in Install Azure CLI 2.0.
3. Install the Azure building blocks npm package.

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

4. From a command prompt, bash prompt, or PowerShell prompt, login to your Azure account by using one of the commands below, and follow the prompts.

```
az login
```

Deploy the solution using azbb

To deploy the Windows VMs for an N-tier application reference architecture, follow these steps:

1. Navigate to the `virtual-machines\n-tier-windows` folder for the repository you cloned in step 1 of the pre-requisites above.
2. The parameter file specifies a default adminstrator user name and password for each VM in the deployment. You must change these before you deploy the reference architecture. Open the `n-tier-windows.json` file and replace each **adminUsername** and **adminPassword** field with your new settings.

> **NOTE**
> There are multiple scripts that run during this deployment both in the **VirtualMachineExtension** objects and in the **extensions** settings for some of the **VirtualMachine** objects. Some of these scripts require the administrator user name and password that you have just changed. It's recommended that you review these scripts to ensure that you specified the correct credentials. The deployment may fail if you have not specified the correct credentials.

Save the file.

3. Deploy the reference architecture using the **azbb** command line tool as shown below.

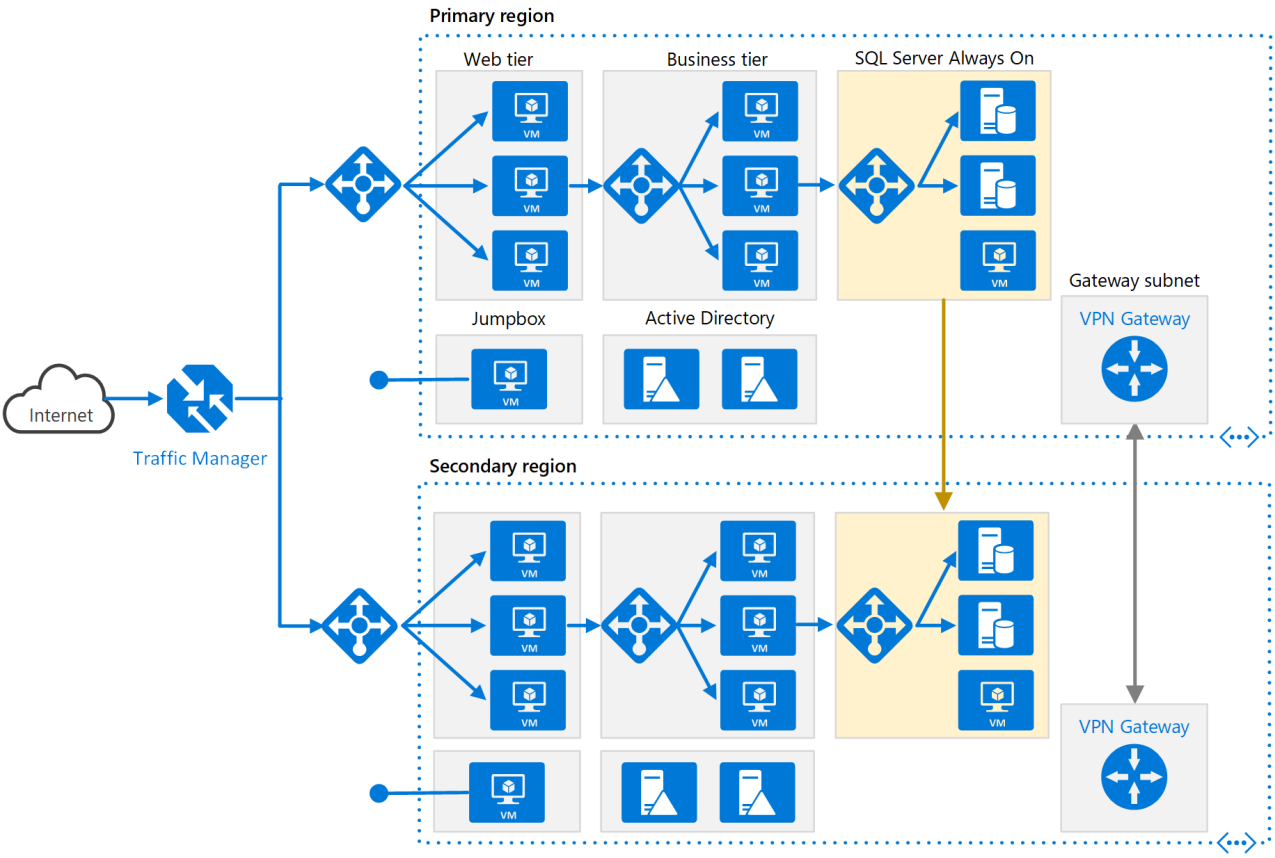
```
azbb -s <your subscription_id> -g <your resource_group_name> -l <azure region> -p n-tier-windows.json --deploy
```

For more information on deploying this sample reference architecture using Azure Building Blocks, visit the GitHub repository.

Multi-region N-tier application for high availability

5/4/2018 • 8 min to read • Edit Online

This reference architecture shows a set of proven practices for running an N-tier application in multiple Azure regions, in order to achieve availability and a robust disaster recovery infrastructure.



Download a Visio file of this architecture.

Architecture

This architecture builds on the one shown in N-tier application with SQL Server.

- **Primary and secondary regions**. Use two regions to achieve higher availability. One is the primary region. The other region is for failover.
- **Azure Traffic Manager**. Traffic Manager routes incoming requests to one of the regions. During normal operations, it routes requests to the primary region. If that region becomes unavailable, Traffic Manager fails over to the secondary region. For more information, see the section Traffic Manager configuration.
- **Resource groups**. Create separate resource groups for the primary region, the secondary region, and for Traffic Manager. This gives you the flexibility to manage each region as a single collection of resources. For example, you could redeploy one region, without taking down the other one. Link the resource groups, so that you can run a query to list all the resources for the application.
- **VNets**. Create a separate VNet for each region. Make sure the address spaces do not overlap.
- **SQL Server Always On Availability Group**. If you are using SQL Server, we recommend SQL Always On Availability Groups for high availability. Create a single availability group that includes the SQL Server instances in both regions.

> **NOTE**
>
> Also consider Azure SQL Database, which provides a relational database as a cloud service. With SQL Database, you don't need to configure an availability group or manage failover.

- **VPN Gateways**. Create a VPN gateway in each VNet, and configure a VNet-to-VNet connection, to enable network traffic between the two VNets. This is required for the SQL Always On Availability Group.

Recommendations

A multi-region architecture can provide higher availability than deploying to a single region. If a regional outage affects the primary region, you can use Traffic Manager to fail over to the secondary region. This architecture can also help if an individual subsystem of the application fails.

There are several general approaches to achieving high availability across regions:

- Active/passive with hot standby. Traffic goes to one region, while the other waits on hot standby. Hot standby means the VMs in the secondary region are allocated and running at all times.
- Active/passive with cold standby. Traffic goes to one region, while the other waits on cold standby. Cold standby means the VMs in the secondary region are not allocated until needed for failover. This approach costs less to run, but will generally take longer to come online during a failure.
- Active/active. Both regions are active, and requests are load balanced between them. If one region becomes unavailable, it is taken out of rotation.

This reference architecture focuses on active/passive with hot standby, using Traffic Manager for failover. Note that you could deploy a small number of VMs for hot standby and then scale out as needed.

Regional pairing

Each Azure region is paired with another region within the same geography. In general, choose regions from the same regional pair (for example, East US 2 and US Central). Benefits of doing so include:

- If there is a broad outage, recovery of at least one region out of every pair is prioritized.
- Planned Azure system updates are rolled out to paired regions sequentially, to minimize possible downtime.
- Pairs reside within the same geography, to meet data residency requirements.

However, make sure that both regions support all of the Azure services needed for your application (see Services by region). For more information about regional pairs, see Business continuity and disaster recovery (BCDR): Azure Paired Regions.

Traffic Manager configuration

Consider the following points when configuring Traffic Manager:

- **Routing**. Traffic Manager supports several routing algorithms. For the scenario described in this article, use *priority* routing (formerly called *failover* routing). With this setting, Traffic Manager sends all requests to the primary region, unless the primary region becomes unreachable. At that point, it automatically fails over to the secondary region. See Configure Failover routing method.
- **Health probe**. Traffic Manager uses an HTTP (or HTTPS) probe to monitor the availability of each region. The probe checks for an HTTP 200 response for a specified URL path. As a best practice, create an endpoint that reports the overall health of the application, and use this endpoint for the health probe. Otherwise, the probe might report a healthy endpoint when critical parts of the application are actually failing. For more information, see Health Endpoint Monitoring Pattern.

When Traffic Manager fails over there is a period of time when clients cannot reach the application. The duration is affected by the following factors:

- The health probe must detect that the primary region has become unreachable.
- DNS servers must update the cached DNS records for the IP address, which depends on the DNS time-to-live (TTL). The default TTL is 300 seconds (5 minutes), but you can configure this value when you create the Traffic Manager profile.

For details, see About Traffic Manager Monitoring.

If Traffic Manager fails over, we recommend performing a manual failback rather than implementing an automatic failback. Otherwise, you can create a situation where the application flips back and forth between regions. Verify that all application subsystems are healthy before failing back.

Note that Traffic Manager automatically fails back by default. To prevent this, manually lower the priority of the primary region after a failover event. For example, suppose the primary region is priority 1 and the secondary is priority 2. After a failover, set the primary region to priority 3, to prevent automatic failback. When you are ready to switch back, update the priority to 1.

The following Azure CLI command updates the priority:

```
azure network traffic-manager  endpoint set --resource-group <resource-group> --profile-name <profile>
    --name <traffic-manager-name> --type AzureEndpoints --priority 3
```

Another approach is to temporarily disable the endpoint until you are ready to fail back:

```
azure network traffic-manager  endpoint set --resource-group <resource-group> --profile-name <profile>
    --name <traffic-manager-name> --type AzureEndpoints --status Disabled
```

Depending on the cause of a failover, you might need to redeploy the resources within a region. Before failing back, perform an operational readiness test. The test should verify things like:

- VMs are configured correctly. (All required software is installed, IIS is running, and so on.)
- Application subsystems are healthy.
- Functional testing. (For example, the database tier is reachable from the web tier.)

Configure SQL Server Always On Availability Groups

Prior to Windows Server 2016, SQL Server Always On Availability Groups require a domain controller, and all nodes in the availability group must be in the same Active Directory (AD) domain.

To configure the availability group:

- At a minimum, place two domain controllers in each region.
- Give each domain controller a static IP address.
- Create a VNet-to-VNet connection to enable communication between the VNets.
- For each VNet, add the IP addresses of the domain controllers (from both regions) to the DNS server list. You can use the following CLI command. For more information, see Manage DNS servers used by a virtual network (VNet).

  ```
  azure network vnet set --resource-group dc01-rg --name dc01-vnet --dns-servers
  "10.0.0.4,10.0.0.6,172.16.0.4,172.16.0.6"
  ```

- Create a Windows Server Failover Clustering (WSFC) cluster that includes the SQL Server instances in both regions.
- Create a SQL Server Always On Availability Group that includes the SQL Server instances in both the

primary and secondary regions. See Extending Always On Availability Group to Remote Azure Datacenter (PowerShell) for the steps.

- Put the primary replica in the primary region.
- Put one or more secondary replicas in the primary region. Configure these to use synchronous commit with automatic failover.
- Put one or more secondary replicas in the secondary region. Configure these to use *asynchronous* commit, for performance reasons. (Otherwise, all T-SQL transactions have to wait on a round trip over the network to the secondary region.)

> **NOTE**
> Asynchronous commit replicas do not support automatic failover.

Availability considerations

With a complex N-tier app, you may not need to replicate the entire application in the secondary region. Instead, you might just replicate a critical subsystem that is needed to support business continuity.

Traffic Manager is a possible failure point in the system. If the Traffic Manager service fails, clients cannot access your application during the downtime. Review the Traffic Manager SLA, and determine whether using Traffic Manager alone meets your business requirements for high availability. If not, consider adding another traffic management solution as a failback. If the Azure Traffic Manager service fails, change your CNAME records in DNS to point to the other traffic management service. (This step must be performed manually, and your application will be unavailable until the DNS changes are propagated.)

For the SQL Server cluster, there are two failover scenarios to consider:

- All of the SQL Server database replicas in the primary region fail. For example, this could happen during a regional outage. In that case, you must manually fail over the availability group, even though Traffic Manager automatically fails over on the front end. Follow the steps in Perform a Forced Manual Failover of a SQL Server Availability Group, which describes how to perform a forced failover by using SQL Server Management Studio, Transact-SQL, or PowerShell in SQL Server 2016.

  > **WARNING**
  > With forced failover, there is a risk of data loss. Once the primary region is back online, take a snapshot of the database and use tablediff to find the differences.

- Traffic Manager fails over to the secondary region, but the primary SQL Server database replica is still available. For example, the front-end tier might fail, without affecting the SQL Server VMs. In that case, Internet traffic is routed to the secondary region, and that region can still connect to the primary replica. However, there will be increased latency, because the SQL Server connections are going across regions. In this situation, you should perform a manual failover as follows:

  1. Temporarily switch a SQL Server database replica in the secondary region to *synchronous* commit. This ensures there won't be data loss during the failover.
  2. Fail over to that replica.
  3. When you fail back to the primary region, restore the asynchronous commit setting.

Manageability considerations

When you update your deployment, update one region at a time to reduce the chance of a global failure from an

incorrect configuration or an error in the application.

Test the resiliency of the system to failures. Here are some common failure scenarios to test:

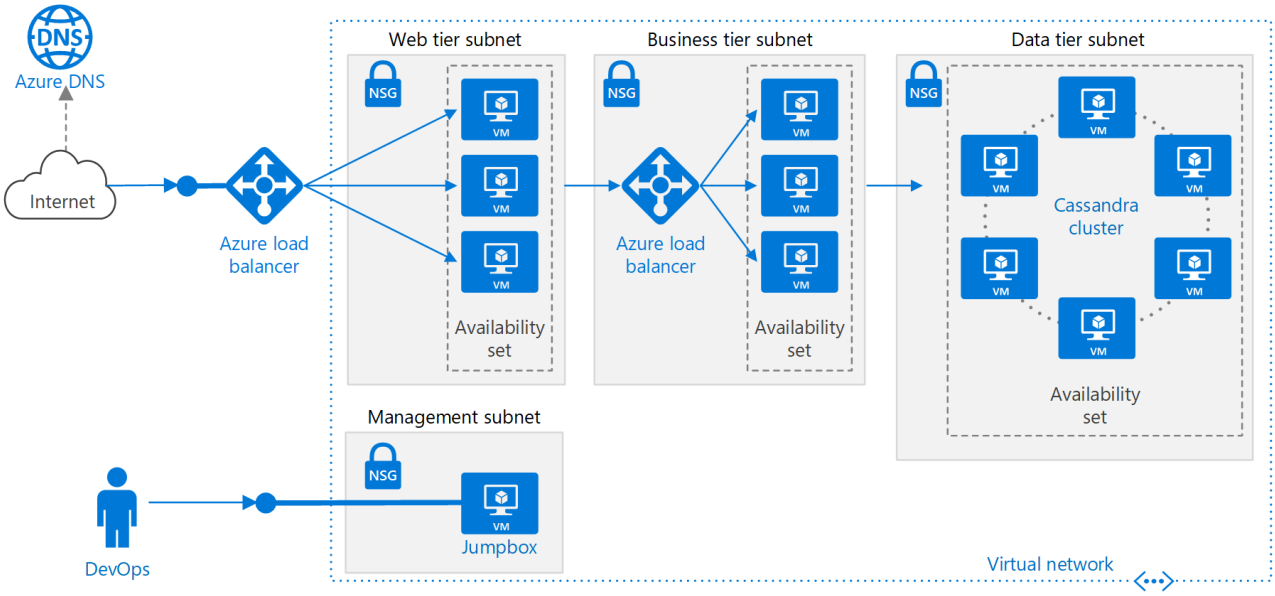
- Shut down VM instances.
- Pressure resources such as CPU and memory.
- Disconnect/delay network.
- Crash processes.
- Expire certificates.
- Simulate hardware faults.
- Shut down the DNS service on the domain controllers.

Measure the recovery times and verify they meet your business requirements. Test combinations of failure modes, as well.

N-tier application with Apache Cassandra

5/4/2018 • 10 min to read • Edit Online

This reference architecture shows how to deploy VMs and a virtual network configured for an N-tier application, using Apache Cassandra on Linux for the data tier. **Deploy this solution**.



Download a Visio file of this architecture.

Architecture

The architecture has the following components:

- **Resource group.** Resource groups are used to group resources so they can be managed by lifetime, owner, or other criteria.
- **Virtual network (VNet) and subnets.** Every Azure VM is deployed into a VNet that can be segmented into multiple subnets. Create a separate subnet for each tier.
- **NSGs.** Use network security groups (NSGs) to restrict network traffic within the VNet. For example, in the 3-tier architecture shown here, the database tier does not accept traffic from the web front end, only from the business tier and the management subnet.
- **Virtual machines**. For recommendations on configuring VMs, see Run a Windows VM on Azure and Run a Linux VM on Azure.
- **Availability sets.** Create an availability set for each tier, and provision at least two VMs in each tier. This makes the VMs eligible for a higher service level agreement (SLA) for VMs.
- **VM scale set** (not shown). A VM scale set is an alternative to using an availability set. A scale sets makes it easy to scale out the VMs in a tier, either manually or automatically based on predefined rules.
- **Azure Load balancers.** The load balancers distribute incoming Internet requests to the VM instances. Use a public load balancer to distribute incoming Internet traffic to the web tier, and an internal load balancer to distribute network traffic from the web tier to the business tier.
- **Public IP address**. A public IP address is needed for the public load balancer to receive Internet traffic.

- **Jumpbox.** Also called a bastion host. A secure VM on the network that administrators use to connect to the other VMs. The jumpbox has an NSG that allows remote traffic only from public IP addresses on a safe list. The NSG should permit ssh traffic.
- **Apache Cassandra database**. Provides high availability at the data tier, by enabling replication and failover.
- **Azure DNS**. Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services.

Recommendations

Your requirements might differ from the architecture described here. Use these recommendations as a starting point.

VNet / Subnets

When you create the VNet, determine how many IP addresses your resources in each subnet require. Specify a subnet mask and a VNet address range large enough for the required IP addresses, using CIDR notation. Use an address space that falls within the standard private IP address blocks, which are 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16.

Choose an address range that does not overlap with your on-premises network, in case you need to set up a gateway between the VNet and your on-premise network later. Once you create the VNet, you can't change the address range.

Design subnets with functionality and security requirements in mind. All VMs within the same tier or role should go into the same subnet, which can be a security boundary. For more information about designing VNets and subnets, see Plan and design Azure Virtual Networks.

Load balancers

Do not expose the VMs directly to the Internet, but instead give each VM a private IP address. Clients connect using the IP address of the public load balancer.

Define load balancer rules to direct network traffic to the VMs. For example, to enable HTTP traffic, create a rule that maps port 80 from the front-end configuration to port 80 on the back-end address pool. When a client sends an HTTP request to port 80, the load balancer selects a back-end IP address by using a hashing algorithm that includes the source IP address. In that way, client requests are distributed across all the VMs.

Network security groups

Use NSG rules to restrict traffic between tiers. For example, in the 3-tier architecture shown above, the web tier does not communicate directly with the database tier. To enforce this, the database tier should block incoming traffic from the web tier subnet.

1. Deny all inbound traffic from the VNet. (Use the `VIRTUAL_NETWORK` tag in the rule.)
2. Allow inbound traffic from the business tier subnet.
3. Allow inbound traffic from the database tier subnet itself. This rule allows communication between the database VMs, which is needed for database replication and failover.
4. Allow ssh traffic (port 22) from the jumpbox subnet. This rule lets administrators connect to the database tier from the jumpbox.

Create rules 2 – 4 with higher priority than the first rule, so they override it.

Cassandra

We recommend DataStax Enterprise for production use, but these recommendations apply to any Cassandra edition. For more information on running DataStax in Azure, see DataStax Enterprise Deployment Guide for Azure.

Put the VMs for a Cassandra cluster in an availability set to ensure that the Cassandra replicas are distributed across multiple fault domains and upgrade domains. For more information about fault domains and upgrade domains, see Manage the availability of virtual machines.

Configure three fault domains (the maximum) per availability set and 18 upgrade domains per availability set. This provides the maximum number of upgrade domains that can still be distributed evenly across the fault domains.

Configure nodes in rack-aware mode. Map fault domains to racks in the `cassandra-rackdc.properties` file.

You don't need a load balancer in front of the cluster. The client connects directly to a node in the cluster.

For high availability, deploy Cassandra in more than one Azure region. Within each region, nodes are configured in rack-aware mode with fault and upgrade domains, for resiliency inside the region.

Jumpbox

Do not allow ssh access from the public Internet to the VMs that run the application workload. Instead, all ssh access to these VMs must come through the jumpbox. An administrator logs into the jumpbox, and then logs into the other VM from the jumpbox. The jumpbox allows ssh traffic from the Internet, but only from known, safe IP addresses.

The jumpbox has minimal performance requirements, so select a small VM size. Create a public IP address for the jumpbox. Place the jumpbox in the same VNet as the other VMs, but in a separate management subnet.

To secure the jumpbox, add an NSG rule that allows ssh connections only from a safe set of public IP addresses. Configure the NSGs for the other subnets to allow ssh traffic from the management subnet.

Scalability considerations

VM scale sets help you to deploy and manage a set of identical VMs. Scale sets support autoscaling based on performance metrics. As the load on the VMs increases, additional VMs are automatically added to the load balancer. Consider scale sets if you need to quickly scale out VMs, or need to autoscale.

There are two basic ways to configure VMs deployed in a scale set:

- Use extensions to configure the VM after it is provisioned. With this approach, new VM instances may take longer to start up than a VM with no extensions.
- Deploy a managed disk with a custom disk image. This option may be quicker to deploy. However, it requires you to keep the image up to date.

For additional considerations, see Design considerations for scale sets.

> **TIP**
> When using any autoscale solution, test it with production-level workloads well in advance.

Each Azure subscription has default limits in place, including a maximum number of VMs per region. You can increase the limit by filing a support request. For more information, see Azure subscription and service limits, quotas, and constraints.

Availability considerations

If you are not using VM scale sets, put VMs in the same tier into an availability set. Create at least two VMs in the availability set to support the availability SLA for Azure VMs. For more information, see Manage the availability of virtual machines.

The load balancer uses health probes to monitor the availability of VM instances. If a probe cannot reach an

instance within a timeout period, the load balancer stops sending traffic to that VM. However, the load balancer will continue to probe, and if the VM becomes available again, the load balancer resumes sending traffic to that VM.

Here are some recommendations on load balancer health probes:

- Probes can test either HTTP or TCP. If your VMs run an HTTP server, create an HTTP probe. Otherwise create a TCP probe.
- For an HTTP probe, specify the path to an HTTP endpoint. The probe checks for an HTTP 200 response from this path. This can be the root path ("/"), or a health-monitoring endpoint that implements some custom logic to check the health of the application. The endpoint must allow anonymous HTTP requests.
- The probe is sent from a known IP address, 168.63.129.16. Make sure you don't block traffic to or from this IP address in any firewall policies or network security group (NSG) rules.
- Use health probe logs to view the status of the health probes. Enable logging in the Azure portal for each load balancer. Logs are written to Azure Blob storage. The logs show how many VMs on the back end are not receiving network traffic due to failed probe responses.

For the Cassandra cluster, the failover scenarios to consider depend on the consistency levels used by the application, as well as the number of replicas used. For consistency levels and usage in Cassandra, see Configuring data consistency and Cassandra: How many nodes are talked to with Quorum? Data availability in Cassandra is determined by the consistency level used by the application and the replication mechanism. For replication in Cassandra, see Data Replication in NoSQL Databases Explained.

Security considerations

Virtual networks are a traffic isolation boundary in Azure. VMs in one VNet cannot communicate directly with VMs in a different VNet. VMs within the same VNet can communicate, unless you create network security groups (NSGs) to restrict traffic. For more information, see Microsoft cloud services and network security.

For incoming Internet traffic, the load balancer rules define which traffic can reach the back end. However, load balancer rules don't support IP safe lists, so if you want to add certain public IP addresses to a safe list, add an NSG to the subnet.

Consider adding a network virtual appliance (NVA) to create a DMZ between the Internet and the Azure virtual network. NVA is a generic term for a virtual appliance that can perform network-related tasks, such as firewall, packet inspection, auditing, and custom routing. For more information, see Implementing a DMZ between Azure and the Internet.

Encrypt sensitive data at rest and use Azure Key Vault to manage the database encryption keys. Key Vault can store encryption keys in hardware security modules (HSMs). It's also recommended to store application secrets, such as database connection strings, in Key Vault.

Deploy the solution

A deployment for this reference architecture is available on GitHub.

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Make sure you have the Azure CLI 2.0 installed on your computer. To install the CLI, follow the instructions in Install Azure CLI 2.0.
3. Install the Azure building blocks npm package.

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

4. From a command prompt, bash prompt, or PowerShell prompt, login to your Azure account by using one of the commands below, and follow the prompts.

```
az login
```

Deploy the solution using azbb

To deploy the Linux VMs for an N-tier application reference architecture, follow these steps:

1. Navigate to the `virtual-machines\n-tier-linux` folder for the repository you cloned in step 1 of the pre-requisites above.
2. The parameter file specifies a default adminstrator user name and password for each VM in the deployment. You must change these before you deploy the reference architecture. Open the `n-tier-linux.json` file and replace each **adminUsername** and **adminPassword** field with your new settings. Save the file.
3. Deploy the reference architecture using the **azbb** command line tool as shown below.

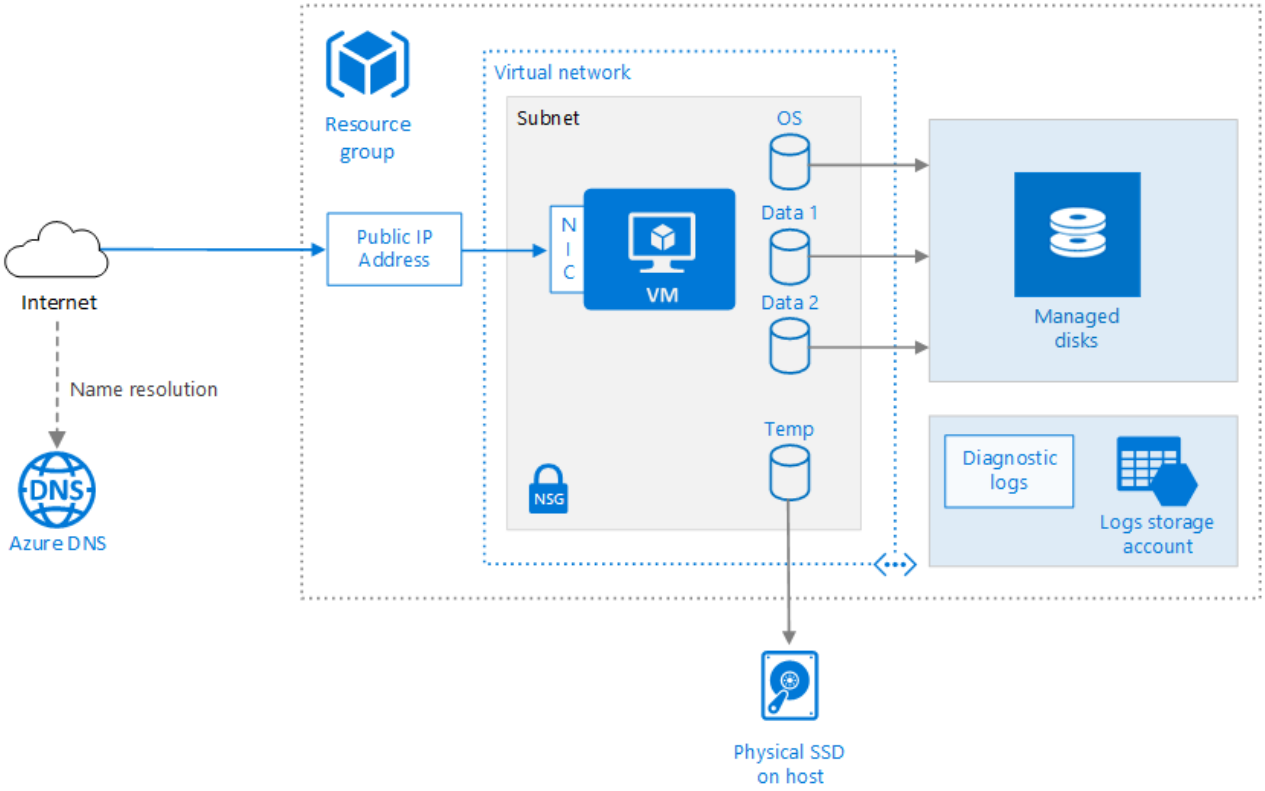
```
azbb -s <your subscription_id> -g <your resource_group_name> -l <azure region> -p n-tier-linux.json --
deploy
```

For more information on deploying this sample reference architecture using Azure Building Blocks, visit the GitHub repository.

Run a Linux VM on Azure

5/4/2018 • 10 min to read • Edit Online

This article describes a set of proven practices for running a Linux virtual machine (VM) on Azure. It includes recommendations for provisioning the VM along with networking and storage components. **Deploy this solution.**



Components

Provisioning an Azure VM requires some additional components besides the VM itself, including networking and storage resources.

- **Resource group.** A resource group is a logical container that holds related Azure resources. In general, group resources based on their lifetime and who will manage them.
- **VM**. You can provision a VM from a list of published images, or from a custom managed image or virtual hard disk (VHD) file uploaded to Azure Blob storage. Azure supports running various popular Linux distributions, including CentOS, Debian, Red Hat Enterprise, Ubuntu, and FreeBSD. For more information, see Azure and Linux.
- **Managed Disks**. Azure Managed Disks simplify disk management by handling the storage for you. The OS disk is a VHD stored in Azure Storage, so it persists even when the host machine is down. For Linux VMs, the OS disk is `/dev/sda1`. We also recommend creating one or more data disks, which are persistent VHDs used for application data.
- **Temporary disk.** The VM is created with a temporary disk. This disk is stored on a physical drive on the host machine. It is *not* saved in Azure Storage and may be deleted during reboots and other VM lifecycle events. Use this disk only for temporary data, such as page or swap files. For Linux VMs, the temporary disk is `/dev/sdb1` and is mounted at `/mnt/resource` or `/mnt`.

- **Virtual network (VNet).** Every Azure VM is deployed into a VNet that can be segmented into multiple subnets.
- **Network interface (NIC)**. The NIC enables the VM to communicate with the virtual network.
- **Public IP address.** A public IP address is needed to communicate with the VM — for example, via SSH.
- **Azure DNS**. Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services.
- **Network security group (NSG)**. Network security groups are used to allow or deny network traffic to VMs. NSGs can be associated either with subnets or with individual VM instances.
- **Diagnostics.** Diagnostic logging is crucial for managing and troubleshooting the VM.

VM recommendations

Azure offers many different virtual machine sizes. For more information, see Sizes for virtual machines in Azure. If you are moving an existing workload to Azure, start with the VM size that's the closest match to your on-premises servers. Then measure the performance of your actual workload with respect to CPU, memory, and disk input/output operations per second (IOPS), and adjust the size as needed. If you require multiple NICs for your VM, be aware that a maximum number of NICs is defined for each VM size.

Generally, choose an Azure region that is closest to your internal users or customers. However, not all VM sizes are available in all regions. For more information, see Services by region. For a list of the VM sizes available in a specific region, run the following command from the Azure command-line interface (CLI):

```
az vm list-sizes --location <location>
```

For information about choosing a published VM image, see Find Linux VM images.

Enable monitoring and diagnostics, including basic health metrics, diagnostics infrastructure logs, and boot diagnostics. Boot diagnostics can help you diagnose boot failure if your VM gets into a non-bootable state. For more information, see Enable monitoring and diagnostics.

Disk and storage recommendations

For best disk I/O performance, we recommend Premium Storage, which stores data on solid-state drives (SSDs). Cost is based on the capacity of the provisioned disk. IOPS and throughput (that is, data transfer rate) also depend on disk size, so when you provision a disk, consider all three factors (capacity, IOPS, and throughput).

We also recommend using Managed Disks. Managed disks do not require a storage account. You simply specify the size and type of disk and it is deployed as a highly available resource.

Add one or more data disks. When you create a VHD, it is unformatted. Log into the VM to format the disk. In the Linux shell, data disks are displayed as `/dev/sdc`, `/dev/sdd`, and so on. You can run `lsblk` to list the block devices, including the disks. To use a data disk, create a partition and file system, and mount the disk. For example:

```
# Create a partition.
sudo fdisk /dev/sdc     # Enter 'n' to partition, 'w' to write the change.

# Create a file system.
sudo mkfs -t ext3 /dev/sdc1

# Mount the drive.
sudo mkdir /data1
sudo mount /dev/sdc1 /data1
```

When you add a data disk, a logical unit number (LUN) ID is assigned to the disk. Optionally, you can specify the LUN ID — for example, if you're replacing a disk and want to retain the same LUN ID, or you have an application that looks for a specific LUN ID. However, remember that LUN IDs must be unique for each disk.

You may want to change the I/O scheduler to optimize for performance on SSDs because the disks for VMs with premium storage accounts are SSDs. A common recommendation is to use the NOOP scheduler for SSDs, but you should use a tool such as iostat to monitor disk I/O performance for your workload.

Create a storage account to hold diagnostic logs. A standard locally redundant storage (LRS) account is sufficient for diagnostic logs.

> **NOTE**
>
> If you aren't using Managed Disks, create separate Azure storage accounts for each VM to hold the virtual hard disks (VHDs), in order to avoid hitting the (IOPS) limits for storage accounts. Be aware of the total I/O limits of the storage account. For more information, see virtual machine disk limits.

Network recommendations

The public IP address can be dynamic or static. The default is dynamic.

- Reserve a static IP address if you need a fixed IP address that won't change — for example, if you need to create an A record in DNS, or need the IP address to be added to a safe list.
- You can also create a fully qualified domain name (FQDN) for the IP address. You can then register a CNAME record in DNS that points to the FQDN. For more information, see Create a fully qualified domain name in the Azure portal. You can use Azure DNS or another DNS service.

All NSGs contain a set of default rules, including a rule that blocks all inbound Internet traffic. The default rules cannot be deleted, but other rules can override them. To enable Internet traffic, create rules that allow inbound traffic to specific ports — for example, port 80 for HTTP.

To enable SSH, add an NSG rule that allows inbound traffic to TCP port 22.

Scalability considerations

You can scale a VM up or down by changing the VM size. To scale out horizontally, put two or more VMs behind a load balancer. For more information, see the N-tier reference architecture.

Availability considerations

For higher availability, deploy multiple VMs in an availability set. This also provides a higher service level agreement (SLA).

Your VM may be affected by planned maintenance or unplanned maintenance. You can use VM reboot logs to determine whether a VM reboot was caused by planned maintenance.

To protect against accidental data loss during normal operations (for example, because of user error), you should also implement point-in-time backups, using blob snapshots or another tool.

Manageability considerations

Resource groups. Put closely associated resources that share the same lifecycle into the same resource group. Resource groups allow you to deploy and monitor resources as a group and track billing costs by resource group. You can also delete resources as a set, which is very useful for test deployments. Assign meaningful resource names to simplify locating a specific resource and understanding its role. For more information, see Recommended naming conventions for Azure resources.

SSH. Before you create a Linux VM, generate a 2048-bit RSA public-private key pair. Use the public key file when you create the VM. For more information, see How to Use SSH with Linux and Mac on Azure.

Stopping a VM. Azure makes a distinction between "stopped" and "deallocated" states. You are charged when the VM status is stopped, but not when the VM is deallocated. In the Azure portal, the **Stop** button deallocates the VM. If you shut down through the OS while logged in, the VM is stopped but **not** deallocated, so you will still be charged.

Deleting a VM. If you delete a VM, the VHDs are not deleted. That means you can safely delete the VM without losing data. However, you will still be charged for storage. To delete the VHD, delete the file from Blob storage. To prevent accidental deletion, use a resource lock to lock the entire resource group or lock individual resources, such as a VM.

Security considerations

Use Azure Security Center to get a central view of the security state of your Azure resources. Security Center monitors potential security issues and provides a comprehensive picture of the security health of your deployment. Security Center is configured per Azure subscription. Enable security data collection as described in the Azure Security Center quick start guide. When data collection is enabled, Security Center automatically scans any VMs created under that subscription.

Patch management. If enabled, Security Center checks whether any security and critical updates are missing.

Antimalware. If enabled, Security Center checks whether antimalware software is installed. You can also use Security Center to install antimalware software from inside the Azure portal.

Operations. Use role-based access control (RBAC) to control access to the Azure resources that you deploy. RBAC lets you assign authorization roles to members of your DevOps team. For example, the Reader role can view Azure resources but not create, manage, or delete them. Some roles are specific to particular Azure resource types. For example, the Virtual Machine Contributor role can restart or deallocate a VM, reset the administrator password, create a new VM, and so on. Other built-in RBAC roles that may be useful for this architecture include DevTest Labs User and Network Contributor. A user can be assigned to multiple roles, and you can create custom roles for even more fine-grained permissions.

> **NOTE**
> RBAC does not limit the actions that a user logged into a VM can perform. Those permissions are determined by the account type on the guest OS.

Use audit logs to see provisioning actions and other VM events.

Data encryption. Consider Azure Disk Encryption if you need to encrypt the OS and data disks.

Deploy the solution

A deployment is available on GitHub. It deploys the following:

- A virtual network with a single subnet named **web** used to host the VM.
- An NSG with two incoming rules to allow SSH and HTTP traffic to the VM.
- A VM running the latest version of Ubuntu 16.04.3 LTS.
- A sample custom script extension that formats the two data disks and deploys Apache HTTP Server to the Ubuntu VM.

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Make sure you have the Azure CLI 2.0 installed on your computer. For CLI installation instructions, see Install Azure CLI 2.0.
3. Install the Azure building blocks npm package.
4. From a command prompt, bash prompt, or PowerShell prompt, enter the following command to log into your Azure account.

```
az login
```

5. Create an SSH key pair. For more information, see How to create and use an SSH public and private key pair for Linux VMs in Azure.

Deploy the solution using azbb

1. Navigate to the `virtual-machines/single-vm/parameters/linux` folder for the repository you downloaded in the prerequisites step above.
2. Open the `single-vm-v2.json` file and enter a username and your SSH public key between the quotes, then save the file.

```
"adminUsername": "<your username>",
"sshPublicKey": "ssh-rsa AAAAB3NzaC1...",
```

3. Run `azbb` to deploy the sample VM as shown below.

```
azbb -s <subscription_id> -g <resource_group_name> -l <location> -p single-vm-v2.json --deploy
```

To verify the deployment, run the following Azure CLI command to find the public IP address of the VM:

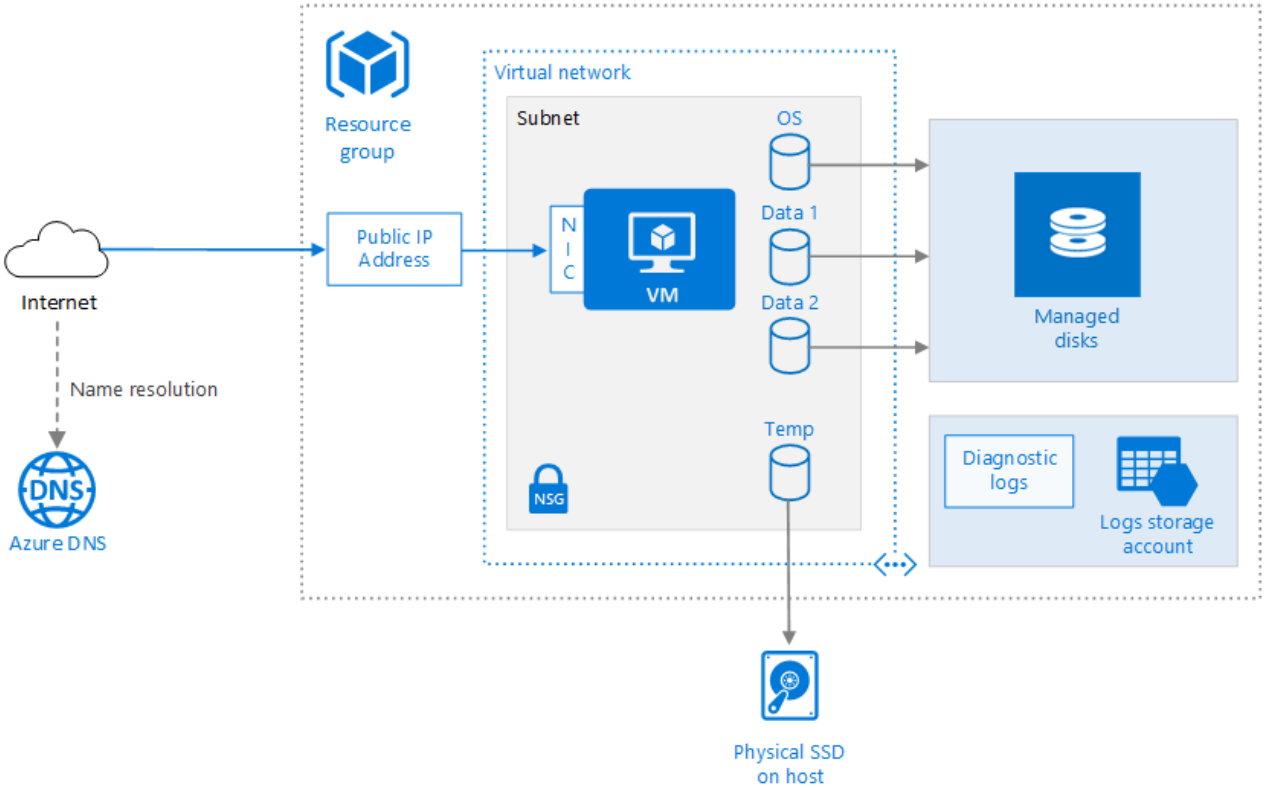
```
az vm show -n ra-single-linux-vm1 -g <resource-group-name> -d -o table
```

If you navigate to this address in a web browser, you should see the default Apache2 homepage.

Run a Windows VM on Azure

5/4/2018 • 9 min to read • Edit Online

This article describes a set of proven practices for running a Windows virtual machine (VM) on Azure. It includes recommendations for provisioning the VM along with networking and storage components. **Deploy this solution.**



Components

Provisioning an Azure VM requires some additional components besides the VM itself, including networking and storage resources.

- **Resource group.** A resource group is a logical container that holds related Azure resources. In general, group resources based on their lifetime and who will manage them.
- **VM**. You can provision a VM from a list of published images, or from a custom managed image or virtual hard disk (VHD) file uploaded to Azure Blob storage.
- **Managed Disks**. Azure Managed Disks simplify disk management by handling the storage for you. The OS disk is a VHD stored in Azure Storage, so it persists even when the host machine is down. We also recommend creating one or more data disks, which are persistent VHDs used for application data.
- **Temporary disk.** The VM is created with a temporary disk (the `D:` drive on Windows). This disk is stored on a physical drive on the host machine. It is *not* saved in Azure Storage and may be deleted during reboots and other VM lifecycle events. Use this disk only for temporary data, such as page or swap files.
- **Virtual network (VNet).** Every Azure VM is deployed into a VNet that can be segmented into multiple subnets.
- **Network interface (NIC)**. The NIC enables the VM to communicate with the virtual network.

- **Public IP address.** A public IP address is needed to communicate with the VM — for example, via remote desktop (RDP).
- **Azure DNS**. Azure DNS is a hosting service for DNS domains, providing name resolution using Microsoft Azure infrastructure. By hosting your domains in Azure, you can manage your DNS records using the same credentials, APIs, tools, and billing as your other Azure services.
- **Network security group (NSG)**. Network security groups are used to allow or deny network traffic to VMs. NSGs can be associated either with subnets or with individual VM instances.
- **Diagnostics.** Diagnostic logging is crucial for managing and troubleshooting the VM.

VM recommendations

Azure offers many different virtual machine sizes. For more information, see Sizes for virtual machines in Azure. If you are moving an existing workload to Azure, start with the VM size that's the closest match to your on-premises servers. Then measure the performance of your actual workload with respect to CPU, memory, and disk input/output operations per second (IOPS), and adjust the size as needed. If you require multiple NICs for your VM, be aware that a maximum number of NICs is defined for each VM size.

Generally, choose an Azure region that is closest to your internal users or customers. However, not all VM sizes are available in all regions. For more information, see Services by region. For a list of the VM sizes available in a specific region, run the following command from the Azure command-line interface (CLI):

```
az vm list-sizes --location <location>
```

For information about choosing a published VM image, see Find Windows VM images.

Enable monitoring and diagnostics, including basic health metrics, diagnostics infrastructure logs, and boot diagnostics. Boot diagnostics can help you diagnose boot failure if your VM gets into a non-bootable state. For more information, see Enable monitoring and diagnostics.

Disk and storage recommendations

For best disk I/O performance, we recommend Premium Storage, which stores data on solid-state drives (SSDs). Cost is based on the capacity of the provisioned disk. IOPS and throughput (that is, data transfer rate) also depend on disk size, so when you provision a disk, consider all three factors (capacity, IOPS, and throughput).

We also recommend using Managed Disks. Managed disks do not require a storage account. You simply specify the size and type of disk and it is deployed as a highly available resource.

Add one or more data disks. When you create a VHD, it is unformatted. Log into the VM to format the disk. When possible, install applications on a data disk, not the OS disk. Some legacy applications might need to install components on the C: drive; in that case, you can resize the OS disk using PowerShell.

Create a storage account to hold diagnostic logs. A standard locally redundant storage (LRS) account is sufficient for diagnostic logs.

> **NOTE**
>
> If you aren't using Managed Disks, create separate Azure storage accounts for each VM to hold the virtual hard disks (VHDs), in order to avoid hitting the (IOPS) limits for storage accounts. Be aware of the total I/O limits of the storage account. For more information, see virtual machine disk limits.

Network recommendations

The public IP address can be dynamic or static. The default is dynamic.

- Reserve a static IP address if you need a fixed IP address that won't change — for example, if you need to create a DNS 'A' record or add the IP address to a safe list.
- You can also create a fully qualified domain name (FQDN) for the IP address. You can then register a CNAME record in DNS that points to the FQDN. For more information, see Create a fully qualified domain name in the Azure portal.

All NSGs contain a set of default rules, including a rule that blocks all inbound Internet traffic. The default rules cannot be deleted, but other rules can override them. To enable Internet traffic, create rules that allow inbound traffic to specific ports — for example, port 80 for HTTP.

To enable RDP, add an NSG rule that allows inbound traffic to TCP port 3389.

Scalability considerations

You can scale a VM up or down by changing the VM size. To scale out horizontally, put two or more VMs behind a load balancer. For more information, see the N-tier reference architecture.

Availability considerations

For higher availability, deploy multiple VMs in an availability set. This also provides a higher service level agreement (SLA).

Your VM may be affected by planned maintenance or unplanned maintenance. You can use VM reboot logs to determine whether a VM reboot was caused by planned maintenance.

To protect against accidental data loss during normal operations (for example, because of user error), you should also implement point-in-time backups, using blob snapshots or another tool.

Manageability considerations

Resource groups. Put closely associated resources that share the same lifecycle into the same resource group. Resource groups allow you to deploy and monitor resources as a group and track billing costs by resource group. You can also delete resources as a set, which is very useful for test deployments. Assign meaningful resource names to simplify locating a specific resource and understanding its role. For more information, see Recommended Naming Conventions for Azure Resources.

Stopping a VM. Azure makes a distinction between "stopped" and "deallocated" states. You are charged when the VM status is stopped, but not when the VM is deallocated. In the Azure portal, the **Stop** button deallocates the VM. If you shut down through the OS while logged in, the VM is stopped but **not** deallocated, so you will still be charged.

Deleting a VM. If you delete a VM, the VHDs are not deleted. That means you can safely delete the VM without losing data. However, you will still be charged for storage. To delete the VHD, delete the file from Blob storage. To prevent accidental deletion, use a resource lock to lock the entire resource group or lock individual resources, such as a VM.

Security considerations

Use Azure Security Center to get a central view of the security state of your Azure resources. Security Center monitors potential security issues and provides a comprehensive picture of the security health of your deployment. Security Center is configured per Azure subscription. Enable security data collection as described in the Azure Security Center quick start guide. When data collection is enabled, Security Center automatically scans any VMs created under that subscription.

Patch management. If enabled, Security Center checks whether any security and critical updates are missing. Use Group Policy settings on the VM to enable automatic system updates.

Antimalware. If enabled, Security Center checks whether antimalware software is installed. You can also use Security Center to install antimalware software from inside the Azure portal.

Operations. Use role-based access control (RBAC) to control access to the Azure resources that you deploy. RBAC lets you assign authorization roles to members of your DevOps team. For example, the Reader role can view Azure resources but not create, manage, or delete them. Some roles are specific to particular Azure resource types. For example, the Virtual Machine Contributor role can restart or deallocate a VM, reset the administrator password, create a new VM, and so on. Other built-in RBAC roles that may be useful for this architecture include DevTest Labs User and Network Contributor. A user can be assigned to multiple roles, and you can create custom roles for even more fine-grained permissions.

> **NOTE**
> RBAC does not limit the actions that a user logged into a VM can perform. Those permissions are determined by the account type on the guest OS.

Use audit logs to see provisioning actions and other VM events.

Data encryption. Consider Azure Disk Encryption if you need to encrypt the OS and data disks.

Deploy the solution

A deployment for this architecture is available on GitHub. It deploys the following:

- A virtual network with a single subnet named **web** used to host the VM.
- An NSG with two incoming rules to allow RDP and HTTP traffic to the VM.
- A VM running the latest version of Windows Server 2016 Datacenter Edition.
- A sample custom script extension that formats the two data disks, and a PowerShell DSC script that deploys Internet Information Services (IIS).

Prerequisites

1. Clone, fork, or download the zip file for the reference architectures GitHub repository.
2. Make sure you have the Azure CLI 2.0 installed on your computer. For CLI installation instructions, see Install Azure CLI 2.0.
3. Install the Azure building blocks npm package.
4. From a command prompt, bash prompt, or PowerShell prompt, enter the following command to log into your Azure account.

```
az login
```

Deploy the solution using azbb

To deploy this reference architecture, follow these steps:

1. Navigate to the `virtual-machines\single-vm\parameters\windows` folder for the repository you downloaded in the prerequisites step above.
2. Open the `single-vm-v2.json` file and enter a username and password between the quotes, then save the file.

```
"adminUsername": "",
"adminPassword": "",
```

3. Run `azbb` to deploy the sample VM as shown below.

```
azbb -s <subscription_id> -g <resource_group_name> -l <location> -p single-vm-v2.json --deploy
```

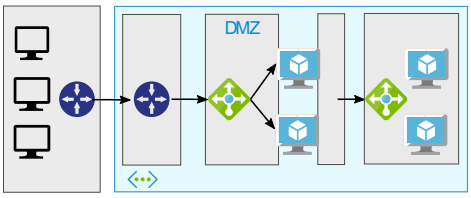
To verify the deployment, run the following Azure CLI command to find the public IP address of the VM:

```
az vm show -n ra-single-windows-vm1 -g <resource-group-name> -d -o table
```

If you navigate to this address in a web browser, you should see the default IIS homepage.

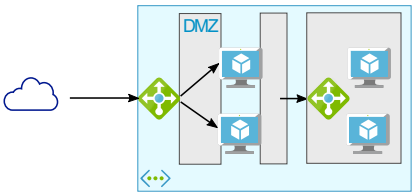
For information about customizing this deployment, visit our GitHub repository.

These reference architectures show proven practices for creating a network DMZ that protects the boundary between an Azure virtual network and an on-premises network or the Internet.



DMZ between Azure and on-premises

Implements a secure hybrid network that extends an on-premises network to Azure.



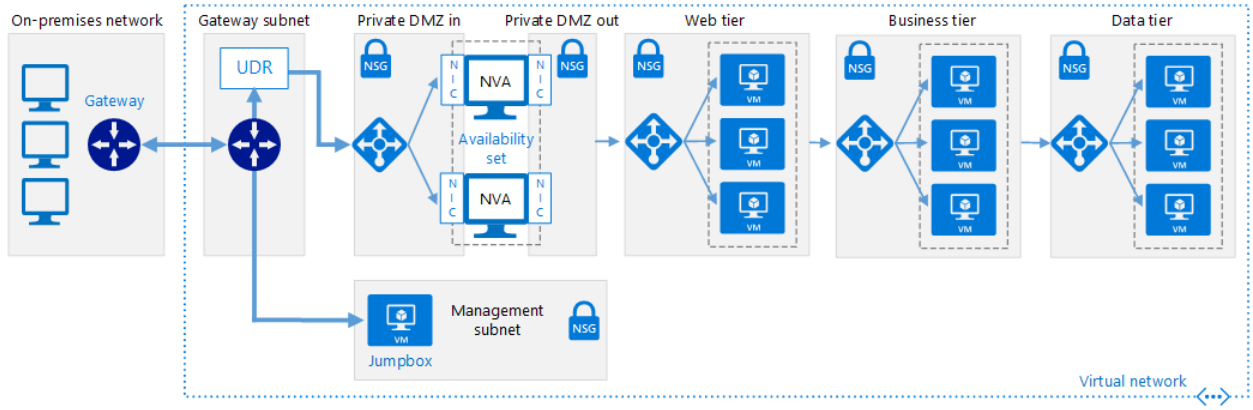
DMZ between Azure and the Internet

Implements a secure network that accepts Internet traffic to Azure.

DMZ between Azure and your on-premises datacenter

4/11/2018 • 11 min to read • Edit Online

This reference architecture shows a secure hybrid network that extends an on-premises network to Azure. The architecture implements a DMZ, also called a *perimeter network*, between the on-premises network and an Azure virtual network (VNet). The DMZ includes network virtual appliances (NVAs) that implement security functionality such as firewalls and packet inspection. All outgoing traffic from the VNet is force-tunneled to the Internet through the on-premises network, so that it can be audited.



Download a Visio file of this architecture.

This architecture requires a connection to your on-premises datacenter, using either a VPN gateway or an ExpressRoute connection. Typical uses for this architecture include:

- Hybrid applications where workloads run partly on-premises and partly in Azure.
- Infrastructure that requires granular control over traffic entering an Azure VNet from an on-premises datacenter.
- Applications that must audit outgoing traffic. This is often a regulatory requirement of many commercial systems and can help to prevent public disclosure of private information.

Architecture

The architecture consists of the following components.

- **On-premises network**. A private local-area network implemented in an organization.
- **Azure virtual network (VNet)**. The VNet hosts the application and other resources running in Azure.
- **Gateway**. The gateway provides connectivity between the routers in the on-premises network and the VNet.
- **Network virtual appliance (NVA)**. NVA is a generic term that describes a VM performing tasks such as allowing or denying access as a firewall, optimizing wide area network (WAN) operations (including network compression), custom routing, or other network functionality.
- **Web tier, business tier, and data tier subnets**. Subnets hosting the VMs and services that implement an example 3-tier application running in the cloud. See Running Windows VMs for an N-tier architecture on Azure for more information.
- **User defined routes (UDR)**. User defined routes define the flow of IP traffic within Azure VNets.

> **NOTE**
> Depending on the requirements of your VPN connection, you can configure Border Gateway Protocol (BGP) routes instead of using UDRs to implement the forwarding rules that direct traffic back through the on-premises network.

- **Management subnet.** This subnet contains VMs that implement management and monitoring capabilities for the components running in the VNet.

Recommendations

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

Access control recommendations

Use Role-Based Access Control (RBAC) to manage the resources in your application. Consider creating the following custom roles:

- A DevOps role with permissions to administer the infrastructure for the application, deploy the application components, and monitor and restart VMs.
- A centralized IT administrator role to manage and monitor network resources.
- A security IT administrator role to manage secure network resources such as the NVAs.

The DevOps and IT administrator roles should not have access to the NVA resources. This should be restricted to the security IT administrator role.

Resource group recommendations

Azure resources such as VMs, VNets, and load balancers can be easily managed by grouping them together into resource groups. Assign RBAC roles to each resource group to restrict access.

We recommend creating the following resource groups:

- A resource group containing the VNet (excluding the VMs), NSGs, and the gateway resources for connecting to the on-premises network. Assign the centralized IT administrator role to this resource group.
- A resource group containing the VMs for the NVAs (including the load balancer), the jumpbox and other management VMs, and the UDR for the gateway subnet that forces all traffic through the NVAs. Assign the security IT administrator role to this resource group.
- Separate resource groups for each application tier that contain the load balancer and VMs. Note that this resource group shouldn't include the subnets for each tier. Assign the DevOps role to this resource group.

Virtual network gateway recommendations

On-premises traffic passes to the VNet through a virtual network gateway. We recommend an Azure VPN gateway or an Azure ExpressRoute gateway.

NVA recommendations

NVAs provide different services for managing and monitoring network traffic. The Azure Marketplace offers several third-party vendor NVAs that you can use. If none of these third-party NVAs meet your requirements, you can create a custom NVA using VMs.

For example, the solution deployment for this reference architecture implements an NVA with the following functionality on a VM:

- Traffic is routed using IP forwarding on the NVA network interfaces (NICs).
- Traffic is permitted to pass through the NVA only if it is appropriate to do so. Each NVA VM in the reference architecture is a simple Linux router. Inbound traffic arrives on network interface *eth0*, and outbound traffic

matches rules defined by custom scripts dispatched through network interface *eth1*.

- The NVAs can only be configured from the management subnet.
- Traffic routed to the management subnet does not pass through the NVAs. Otherwise, if the NVAs fail, there would be no route to the management subnet to fix them.
- The VMs for the NVA are placed in an availability set behind a load balancer. The UDR in the gateway subnet directs NVA requests to the load balancer.

Include a layer-7 NVA to terminate application connections at the NVA level and maintain affinity with the backend tiers. This guarantees symmetric connectivity, in which response traffic from the backend tiers returns through the NVA.

Another option to consider is connecting multiple NVAs in series, with each NVA performing a specialized security task. This allows each security function to be managed on a per-NVA basis. For example, an NVA implementing a firewall could be placed in series with an NVA running identity services. The tradeoff for ease of management is the addition of extra network hops that may increase latency, so ensure that this doesn't affect your application's performance.

NSG recommendations

The VPN gateway exposes a public IP address for the connection to the on-premises network. We recommend creating a network security group (NSG) for the inbound NVA subnet, with rules to block all traffic not originating from the on-premises network.

We also recommend NSGs for each subnet to provide a second level of protection against inbound traffic bypassing an incorrectly configured or disabled NVA. For example, the web tier subnet in the reference architecture implements an NSG with a rule to ignore all requests other than those received from the on-premises network (192.168.0.0/16) or the VNet, and another rule that ignores all requests not made on port 80.

Internet access recommendations

Force-tunnel all outbound Internet traffic through your on-premises network using the site-to-site VPN tunnel, and route to the Internet using network address translation (NAT). This prevents accidental leakage of any confidential information stored in your data tier and allows inspection and auditing of all outgoing traffic.

> **NOTE**
> Don't completely block Internet traffic from the application tiers, as this will prevent these tiers from using Azure PaaS services that rely on public IP addresses, such as VM diagnostics logging, downloading of VM extensions, and other functionality. Azure diagnostics also requires that components can read and write to an Azure Storage account.

Verify that outbound internet traffic is force-tunneled correctly. If you're using a VPN connection with the routing and remote access service on an on-premises server, use a tool such as WireShark or Microsoft Message Analyzer.

Management subnet recommendations

The management subnet contains a jumpbox that performs management and monitoring functionality. Restrict execution of all secure management tasks to the jumpbox.

Do not create a public IP address for the jumpbox. Instead, create one route to access the jumpbox through the incoming gateway. Create NSG rules so the management subnet only responds to requests from the allowed route.

Scalability considerations

The reference architecture uses a load balancer to direct on-premises network traffic to a pool of NVA devices, which route the traffic. The NVAs are placed in an availability set. This design allows you to monitor the

throughput of the NVAs over time and add NVA devices in response to increases in load.

The standard SKU VPN gateway supports sustained throughput of up to 100 Mbps. The High Performance SKU provides up to 200 Mbps. For higher bandwidths, consider upgrading to an ExpressRoute gateway. ExpressRoute provides up to 10 Gbps bandwidth with lower latency than a VPN connection.

For more information about the scalability of Azure gateways, see the scalability consideration section in Implementing a hybrid network architecture with Azure and on-premises VPN and Implementing a hybrid network architecture with Azure ExpressRoute.

Availability considerations

As mentioned, the reference architecture uses a pool of NVA devices behind a load balancer. The load balancer uses a health probe to monitor each NVA and will remove any unresponsive NVAs from the pool.

If you're using Azure ExpressRoute to provide connectivity between the VNet and on-premises network, configure a VPN gateway to provide failover if the ExpressRoute connection becomes unavailable.

For specific information on maintaining availability for VPN and ExpressRoute connections, see the availability considerations in Implementing a hybrid network architecture with Azure and on-premises VPN and Implementing a hybrid network architecture with Azure ExpressRoute.

Manageability considerations

All application and resource monitoring should be performed by the jumpbox in the management subnet. Depending on your application requirements, you may need additional monitoring resources in the management subnet. If so, these resources should be accessed through the jumpbox.

If gateway connectivity from your on-premises network to Azure is down, you can still reach the jumpbox by deploying a public IP address, adding it to the jumpbox, and remoting in from the internet.

Each tier's subnet in the reference architecture is protected by NSG rules. You may need to create a rule to open port 3389 for remote desktop protocol (RDP) access on Windows VMs or port 22 for secure shell (SSH) access on Linux VMs. Other management and monitoring tools may require rules to open additional ports.

If you're using ExpressRoute to provide the connectivity between your on-premises datacenter and Azure, use the Azure Connectivity Toolkit (AzureCT) to monitor and troubleshoot connection issues.

You can find additional information specifically aimed at monitoring and managing VPN and ExpressRoute connections in the articles Implementing a hybrid network architecture with Azure and on-premises VPN and Implementing a hybrid network architecture with Azure ExpressRoute.

Security considerations

This reference architecture implements multiple levels of security.

Routing all on-premises user requests through the NVA

The UDR in the gateway subnet blocks all user requests other than those received from on-premises. The UDR passes allowed requests to the NVAs in the private DMZ subnet, and these requests are passed on to the application if they are allowed by the NVA rules. You can add other routes to the UDR, but make sure they don't inadvertently bypass the NVAs or block administrative traffic intended for the management subnet.

The load balancer in front of the NVAs also acts as a security device by ignoring traffic on ports that are not open in the load balancing rules. The load balancers in the reference architecture only listen for HTTP requests on port 80 and HTTPS requests on port 443. Document any additional rules that you add to the load balancers, and monitor traffic to ensure there are no security issues.

Using NSGs to block/pass traffic between application tiers

Traffic between tiers is restricted by using NSGs. The business tier blocks all traffic that doesn't originate in the web tier, and the data tier blocks all traffic that doesn't originate in the business tier. If you have a requirement to expand the NSG rules to allow broader access to these tiers, weigh these requirements against the security risks. Each new inbound pathway represents an opportunity for accidental or purposeful data leakage or application damage.

DevOps access

Use RBAC to restrict the operations that DevOps can perform on each tier. When granting permissions, use the principle of least privilege. Log all administrative operations and perform regular audits to ensure any configuration changes were planned.

Solution deployment

A deployment for a reference architecture that implements these recommendations is available on GitHub. The reference architecture can be deployed by following the directions below:

1. Click the button below:

2. Once the link has opened in the Azure portal, you must enter values for some of the settings:
   - The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-private-dmz-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
3. Wait for the deployment to complete.
4. The parameter files include hard-coded administrator user name and password for all VMs, and it is strongly recommended that you immediately change both. For each VM in the deployment, select it in the Azure portal and then click **Reset password** in the **Support + troubleshooting** blade. Select **Reset password** in the **Mode** drop down box, then select a new **User name** and **Password**. Click the **Update** button to save.

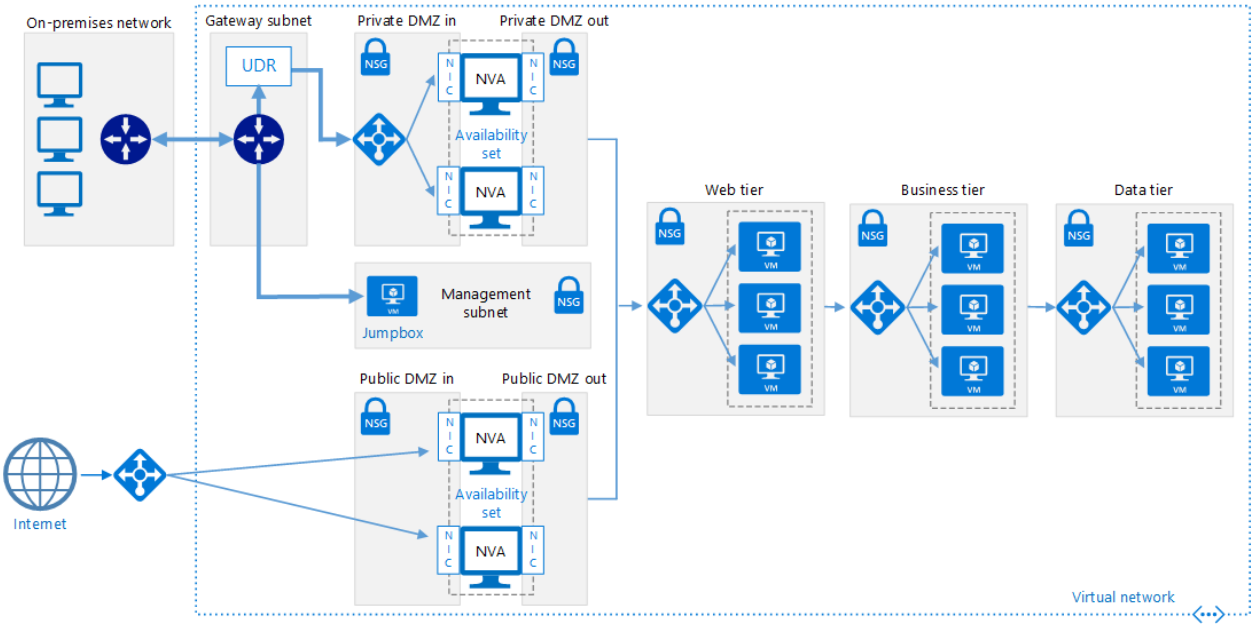
Next steps

- Learn how to implement a DMZ between Azure and the Internet.
- Learn how to implement a highly available hybrid network architecture.
- For more information about managing network security with Azure, see Microsoft cloud services and network security.
- For detailed information about protecting resources in Azure, see Getting started with Microsoft Azure security.
- For additional details on addressing security concerns across an Azure gateway connection, see Implementing a hybrid network architecture with Azure and on-premises VPN and Implementing a hybrid network architecture with Azure ExpressRoute.

DMZ between Azure and the Internet

4/11/2018 • 5 min to read • Edit Online

This reference architecture shows a secure hybrid network that extends an on-premises network to Azure and also accepts Internet traffic.



Download a Visio file of this architecture.

This reference architecture extends the architecture described in Implementing a DMZ between Azure and your on-premises datacenter. It adds a public DMZ that handles Internet traffic, in addition to the private DMZ that handles traffic from the on-premises network

Typical uses for this architecture include:

- Hybrid applications where workloads run partly on-premises and partly in Azure.
- Azure infrastructure that routes incoming traffic from on-premises and the Internet.

Architecture

The architecture consists of the following components.

- **Public IP address (PIP)**. The IP address of the public endpoint. External users connected to the Internet can access the system through this address.
- **Network virtual appliance (NVA)**. This architecture includes a separate pool of NVAs for traffic originating on the Internet.
- **Azure load balancer**. All incoming requests from the Internet pass through the load balancer and are distributed to the NVAs in the public DMZ.
- **Public DMZ inbound subnet**. This subnet accepts requests from the Azure load balancer. Incoming requests are passed to one of the NVAs in the public DMZ.
- **Public DMZ outbound subnet**. Requests that are approved by the NVA pass through this subnet to the internal load balancer for the web tier.

Recommendations

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

NVA recommendations

Use one set of NVAs for traffic originating on the Internet, and another for traffic originating on-premises. Using only one set of NVAs for both is a security risk, because it provides no security perimeter between the two sets of network traffic. Using separate NVAs reduces the complexity of checking security rules, and makes it clear which rules correspond to each incoming network request. One set of NVAs implements rules for Internet traffic only, while another set of NVAs implement rules for on-premises traffic only.

Include a layer-7 NVA to terminate application connections at the NVA level and maintain compatibility with the backend tiers. This guarantees symmetric connectivity where response traffic from the backend tiers returns through the NVA.

Public load balancer recommendations

For scalability and availability, deploy the public DMZ NVAs in an availability set and use an Internet facing load balancer to distribute Internet requests across the NVAs in the availability set.

Configure the load balancer to accept requests only on the ports necessary for Internet traffic. For example, restrict inbound HTTP requests to port 80 and inbound HTTPS requests to port 443.

Scalability considerations

Even if your architecture initially requires a single NVA in the public DMZ, we recommend putting a load balancer in front of the public DMZ from the beginning. That will make it easier to scale to multiple NVAs in the future, if needed.

Availability considerations

The Internet facing load balancer requires each NVA in the public DMZ inbound subnet to implement a health probe. A health probe that fails to respond on this endpoint is considered to be unavailable, and the load balancer will direct requests to other NVAs in the same availability set. Note that if all NVAs fail to respond, your application will fail, so it's important to have monitoring configured to alert DevOps when the number of healthy NVA instances falls below a defined threshold.

Manageability considerations

All monitoring and management for the NVAs in the public DMZ should be performed by the jumpbox in the management subnet. As discussed in Implementing a DMZ between Azure and your on-premises datacenter, define a single network route from the on-premises network through the gateway to the jumpbox, in order to restrict access.

If gateway connectivity from your on-premises network to Azure is down, you can still reach the jumpbox by deploying a public IP address, adding it to the jumpbox, and logging in from the Internet.

Security considerations

This reference architecture implements multiple levels of security:

- The Internet facing load balancer directs requests to the NVAs in the inbound public DMZ subnet, and only on the ports necessary for the application.
- The NSG rules for the inbound and outbound public DMZ subnets prevent the NVAs from being compromised, by blocking requests that fall outside of the NSG rules.
- The NAT routing configuration for the NVAs directs incoming requests on port 80 and port 443 to the web tier load balancer, but ignores requests on all other ports.

You should log all incoming requests on all ports. Regularly audit the logs, paying attention to requests that fall outside of expected parameters, as these may indicate intrusion attempts.

Solution deployment

A deployment for a reference architecture that implements these recommendations is available on GitHub. The reference architecture can be deployed either with Windows or Linux VMs by following the directions below:

1. Click the button below:

2. Once the link has opened in the Azure portal, you must enter values for some of the settings:
   - The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-public-dmz-network-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Select the **Os Type** from the drop down box, **windows** or **linux**.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
3. Wait for the deployment to complete.
4. Click the button below:

5. Once the link has opened in the Azure portal, you must enter values for some of the settings:
   - The **Resource group** name is already defined in the parameter file, so select **Create New** and enter `ra-public-dmz-wl-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
6. Wait for the deployment to complete.
7. Click the button below:

8. Once the link has opened in the Azure portal, you must enter values for some of the settings:
   - The **Resource group** name is already defined in the parameter file, so select **Use Existing** and enter `ra-public-dmz-network-rg` in the text box.
   - Select the region from the **Location** drop down box.
   - Do not edit the **Template Root Uri** or the **Parameter Root Uri** text boxes.
   - Review the terms and conditions, then click the **I agree to the terms and conditions stated above** checkbox.
   - Click the **Purchase** button.
9. Wait for the deployment to complete.
10. The parameter files include hard-coded administrator user name and password for all VMs, and it is strongly recommended that you immediately change both. For each VM in the deployment, select it in the Azure portal and then click **Reset password** in the **Support + troubleshooting** blade. Select **Reset password** in the **Mode** drop down box, then select a new **User name** and **Password**. Click the **Update** button to save.

Deploy highly available network virtual appliances

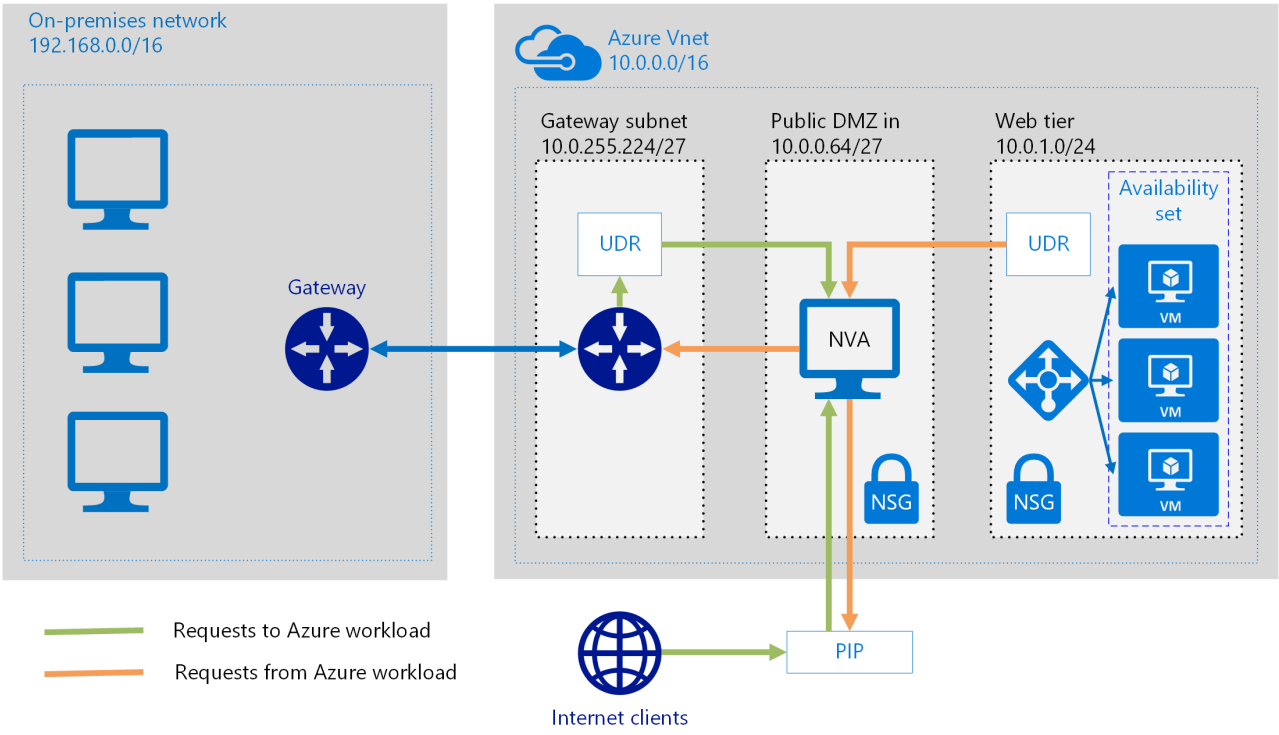
4/11/2018 • 6 min to read • Edit Online

This article shows how to deploy a set of network virtual appliances (NVAs) for high availability in Azure. An NVA is typically used to control the flow of network traffic from a perimeter network, also known as a DMZ, to other networks or subnets. To learn about implementing a DMZ in Azure, see Microsoft cloud services and network security. The article includes example architectures for ingress only, egress only, and both ingress and egress.

Prerequisites: This article assumes a basic understanding of Azure networking, Azure load balancers, and user-defined routes (UDRs).

Architecture Diagrams

An NVA can be deployed to a DMZ in many different architectures. For example, the following figure illustrates the use of a single NVA for ingress.



In this architecture, the NVA provides a secure network boundary by checking all inbound and outbound network traffic and passing only the traffic that meets network security rules. However, the fact that all network traffic must pass through the NVA means that the NVA is a single point of failure in the network. If the NVA fails, there is no other path for network traffic and all the back-end subnets are unavailable.

To make an NVA highly available, deploy more than one NVA into an availability set.

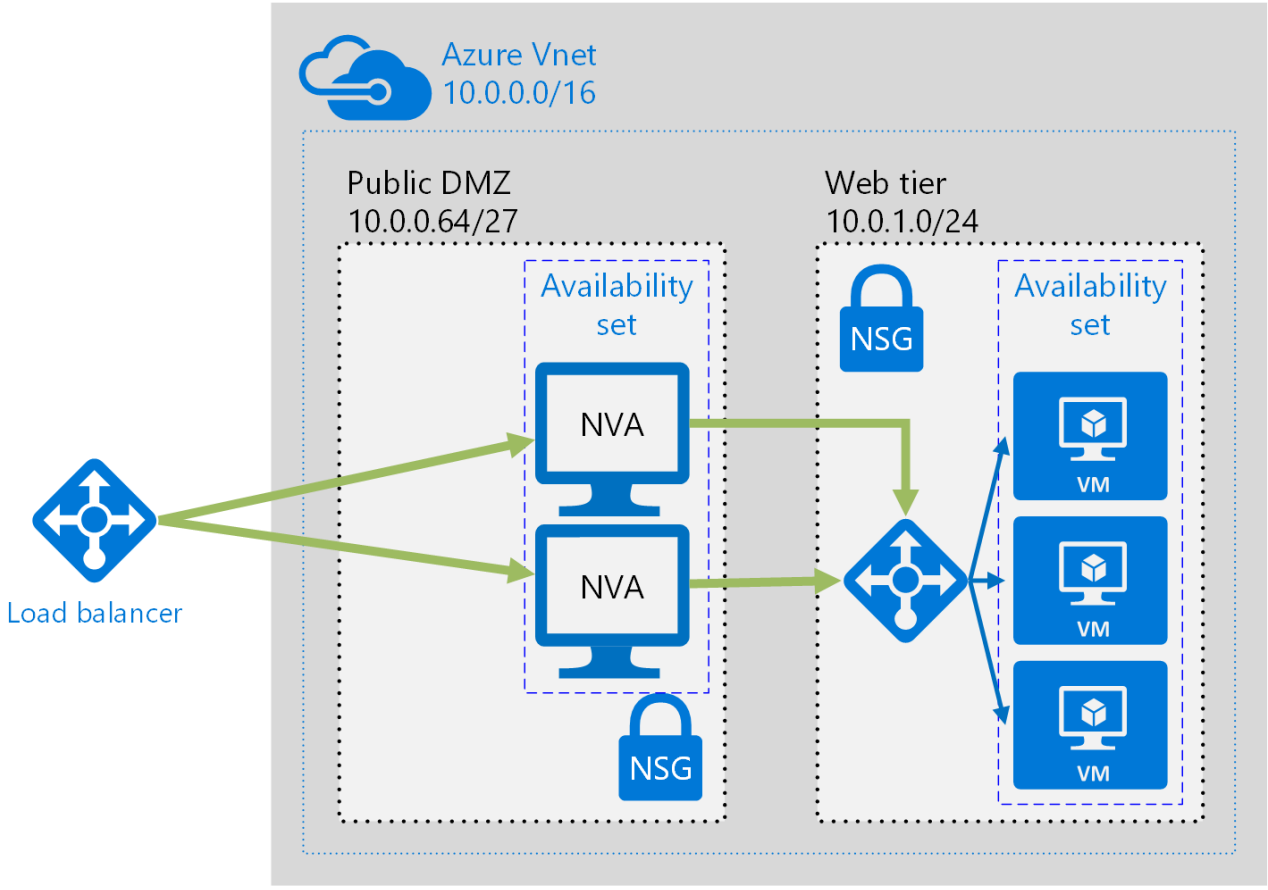
The following architectures describe the resources and configuration necessary for highly available NVAs:

SOLUTION	BENEFITS	CONSIDERATIONS

SOLUTION	BENEFITS	CONSIDERATIONS
Ingress with layer 7 NVAs	All NVA nodes are active	Requires an NVA that can terminate connections and use SNAT Requires a separate set of NVAs for traffic coming from the Internet and from Azure Can only be used for traffic originating outside Azure
Egress with layer 7 NVAs	All NVA nodes are active	Requires an NVA that can terminate connections and implements source network address translation (SNAT)
Ingress-Egress with layer 7 NVAs	All nodes are active Able to handle traffic originated in Azure	Requires an NVA that can terminate connections and use SNAT Requires a separate set of NVAs for traffic coming from the Internet and from Azure
PIP-UDR switch	Single set of NVAs for all traffic Can handle all traffic (no limit on port rules)	Active-passive Requires a failover process

Ingress with layer 7 NVAs

The following figure shows a high availability architecture that implements an ingress DMZ behind an internet-facing load balancer. This architecture is designed to provide connectivity to Azure workloads for layer 7 traffic, such as HTTP or HTTPS:



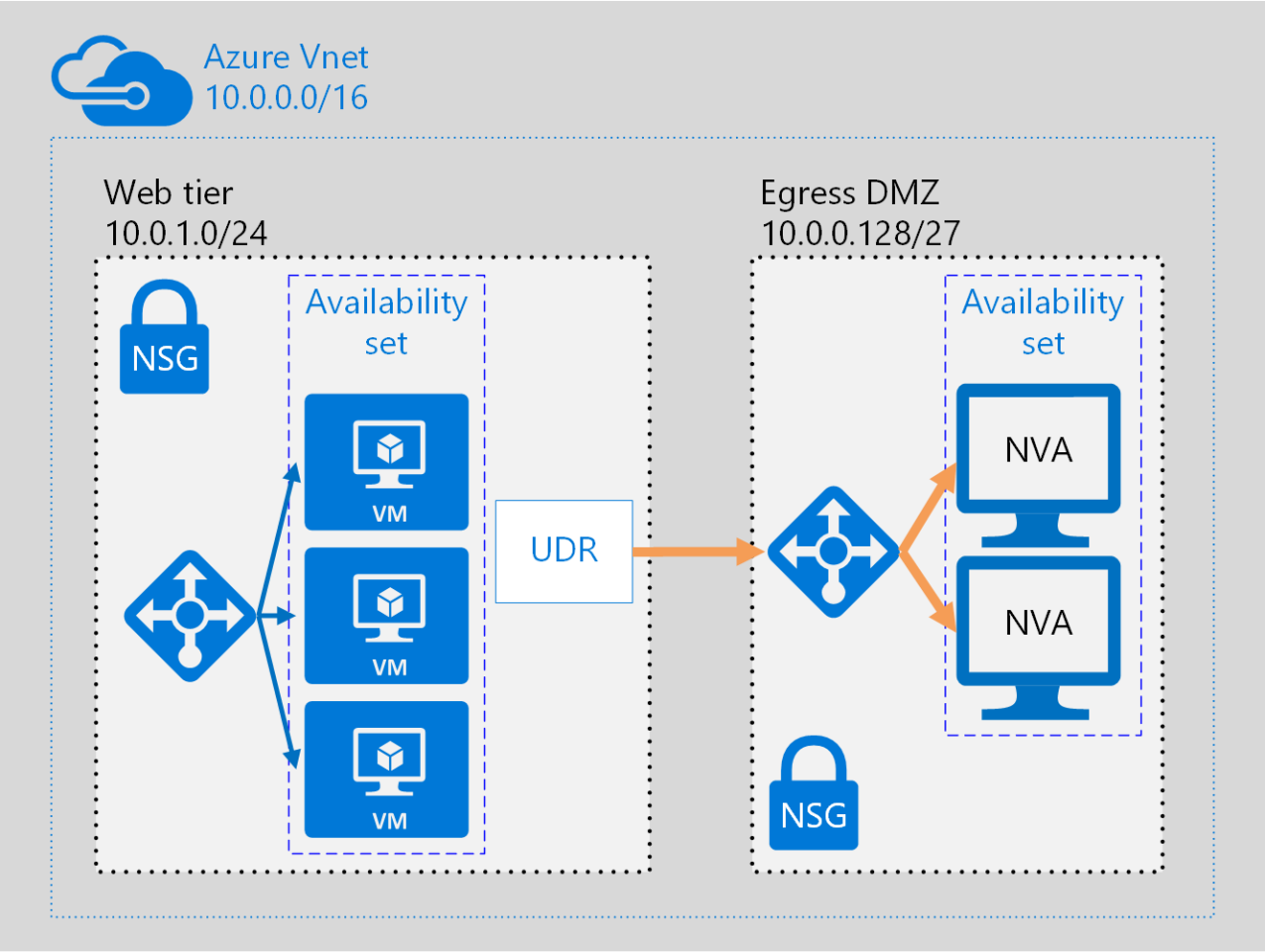
The benefit of this architecture is that all NVAs are active, and if one fails the load balancer directs network traffic to the other NVA. Both NVAs route traffic to the internal load balancer so as long as one NVA is active, traffic

continues to flow. The NVAs are required to terminate SSL traffic intended for the web tier VMs. These NVAs cannot be extended to handle on-premises traffic because on-premises traffic requires another dedicated set of NVAs with their own network routes.

> **NOTE**
>
> This architecture is used in the DMZ between Azure and your on-premises datacenter reference architecture and the DMZ between Azure and the Internet reference architecture. Each of these reference architectures includes a deployment solution that you can use. Follow the links for more information.

Egress with layer 7 NVAs

The previous architecture can be expanded to provide an egress DMZ for requests originating in the Azure workload. The following architecture is designed to provide high availability of the NVAs in the DMZ for layer 7 traffic, such as HTTP or HTTPS:

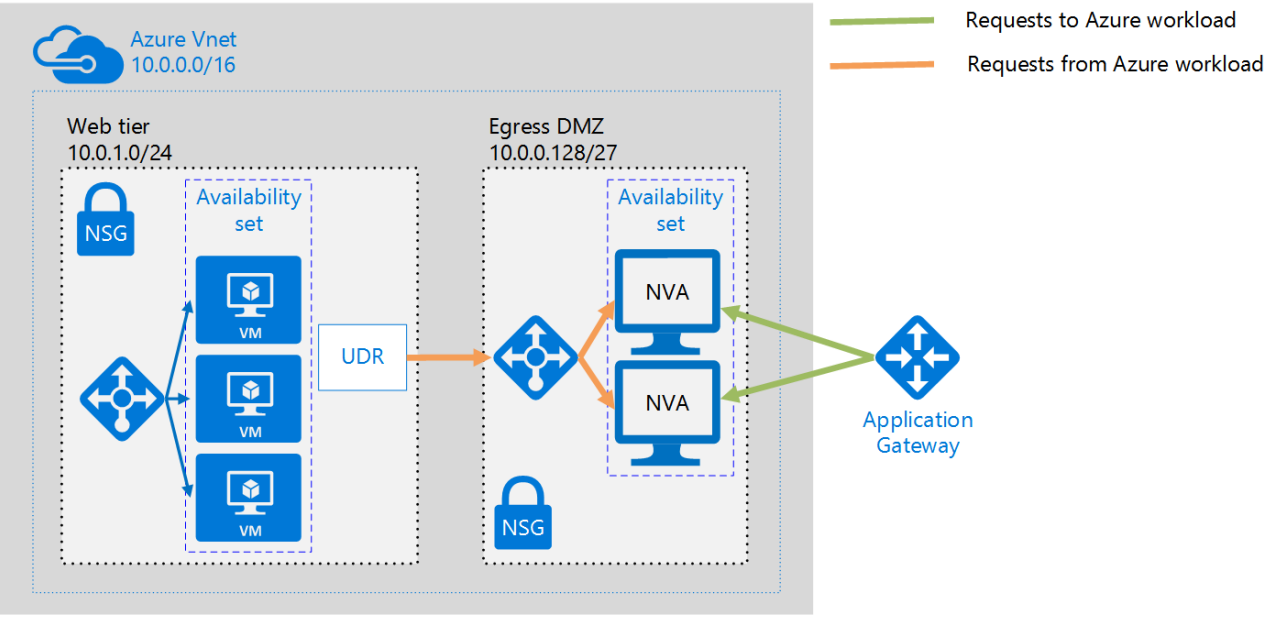


In this architecture, all traffic originating in Azure is routed to an internal load balancer. The load balancer distributes outgoing requests between a set of NVAs. These NVAs direct traffic to the Internet using their individual public IP addresses.

> **NOTE**
>
> This architecture is used in the DMZ between Azure and your on-premises datacenter reference architecture and the DMZ between Azure and the Internet reference architecture. Each of these reference architectures includes a deployment solution that you can use. Follow the links for more information.

Ingress-egress with layer 7 NVAs

In the two previous architectures, there was a separate DMZ for ingress and egress. The following architecture demonstrates how to create a DMZ that can be used for both ingress and egress for layer 7 traffic, such as HTTP or HTTPS:

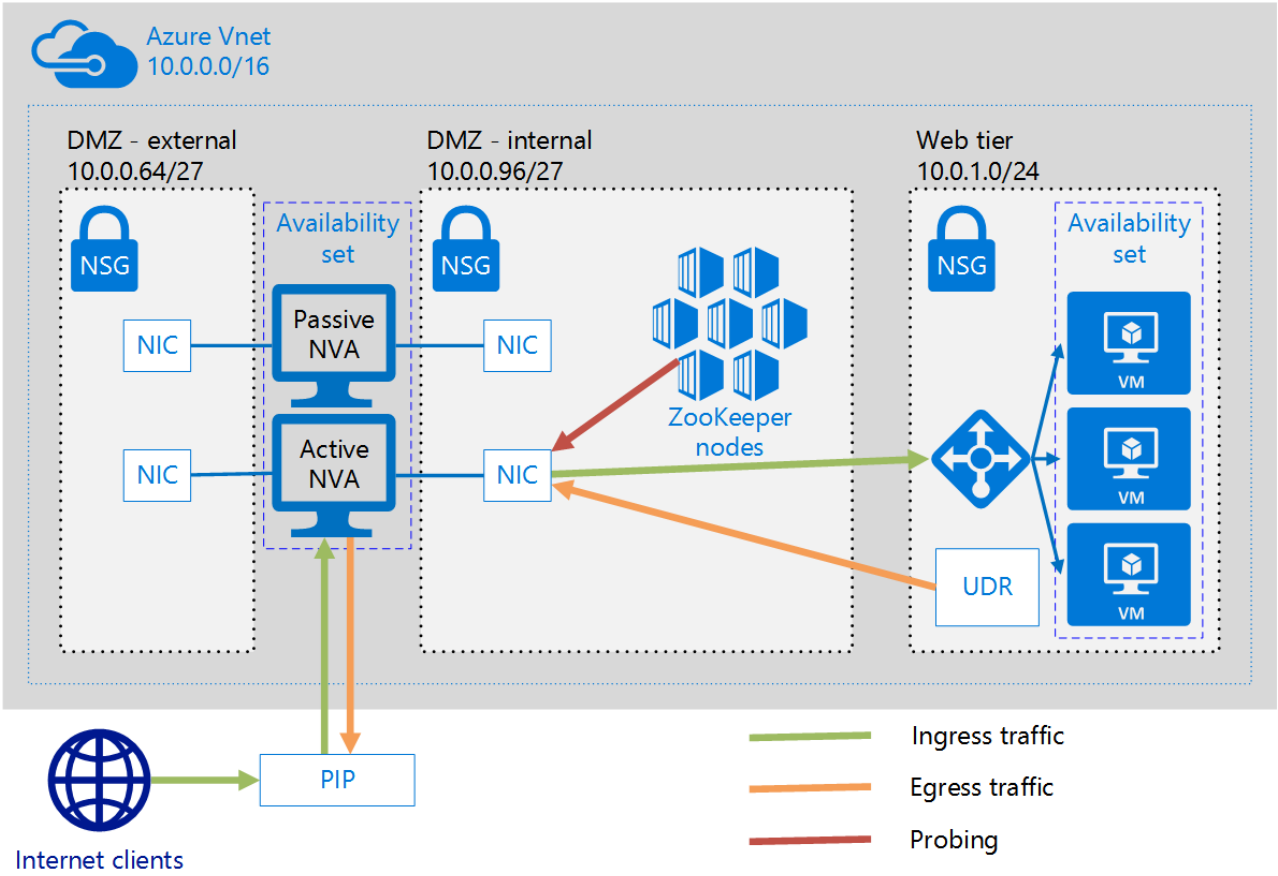


In this architecture, the NVAs process incoming requests from the application gateway. The NVAs also process outgoing requests from the workload VMs in the back-end pool of the load balancer. Because incoming traffic is routed with an application gateway and outgoing traffic is routed with a load balancer, the NVAs are responsible for maintaining session affinity. That is, the application gateway maintains a mapping of inbound and outbound requests so it can forward the correct response to the original requestor. However, the internal load balancer does not have access to the application gateway mappings, and uses its own logic to send responses to the NVAs. It's possible the load balancer could send a response to an NVA that did not initially receive the request from the application gateway. In this case, the NVAs must communicate and transfer the response between them so the correct NVA can forward the response to the application gateway.

> **NOTE**
>
> You can also solve the asymmetric routing issue by ensuring the NVAs perform inbound source network address translation (SNAT). This would replace the original source IP of the requestor to one of the IP addresses of the NVA used on the inbound flow. This ensures that you can use multiple NVAs at a time, while preserving the route symmetry.

PIP-UDR switch with layer 4 NVAs

The following architecture demonstrates an architecture with one active and one passive NVA. This architecture handles both ingress and egress for layer 4 traffic:



This architecture is similar to the first architecture discussed in this article. That architecture included a single NVA accepting and filtering incoming layer 4 requests. This architecture adds a second passive NVA to provide high availability. If the active NVA fails, the passive NVA is made active and the UDR and PIP are changed to point to the NICs on the now active NVA. These changes to the UDR and PIP can either be done manually or using an automated process. The automated process is typically daemon or other monitoring service running in Azure. It queries a health probe on the active NVA and performs the UDR and PIP switch when it detects a failure of the NVA.

The preceding figure shows an example ZooKeeper cluster providing a high availability daemon. Within the ZooKeeper cluster, a quorum of nodes elects a leader. If the leader fails, the remaining nodes hold an election to elect a new leader. For this architecture, the leader node executes the daemon that queries the health endpoint on the NVA. If the NVA fails to respond to the health probe, the daemon activates the passive NVA. The daemon then calls the Azure REST API to remove the PIP from the failed NVA and attaches it to newly activated NVA. The daemon then modifies the UDR to point to the newly activated NVA's internal IP address.

> **NOTE**
>
> Do not include the ZooKeeper nodes in a subnet that is only accessible using a route that includes the NVA. Otherwise, the ZooKeeper nodes are inaccessible if the NVA fails. Should the daemon fail for any reason, you won't be able to access any of the ZooKeeper nodes to diagnose the problem.

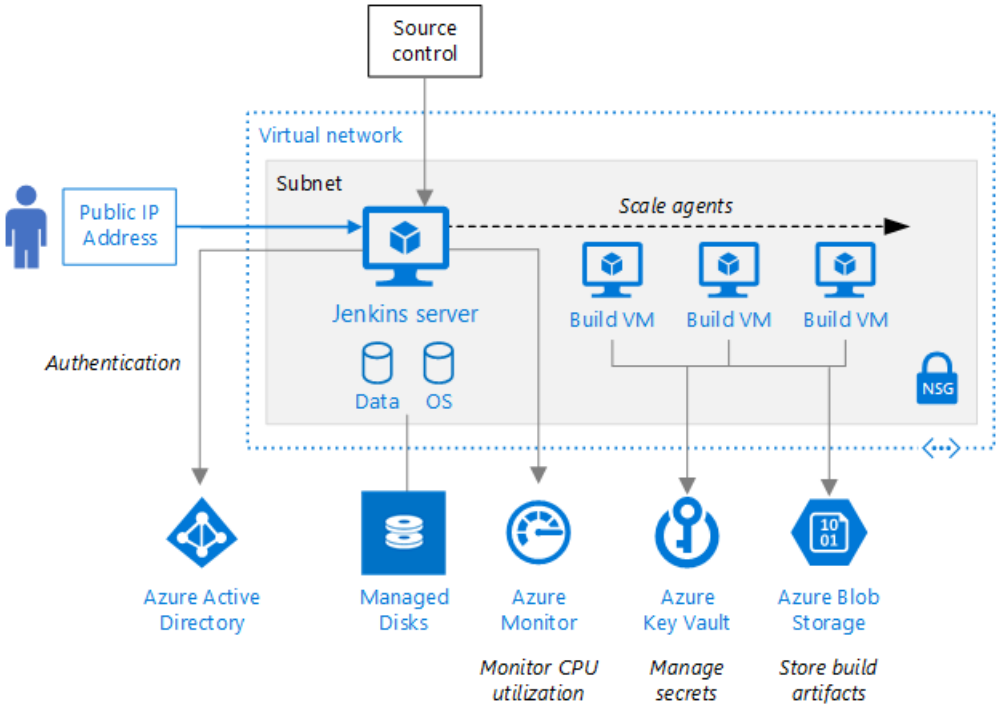
Next steps

- Learn how to implement a DMZ between Azure and your on-premises datacenter using layer-7 NVAs.
- Learn how to implement a DMZ between Azure and the Internet using layer-7 NVAs.

Run a Jenkins server on Azure

5/2/2018 • 12 min to read • Edit Online

This reference architecture shows how to deploy and operate a scalable, enterprise-grade Jenkins server on Azure secured with single sign-on (SSO). The architecture also uses Azure Monitor to monitor the state of the Jenkins server. **Deploy this solution**.



Download a Visio file that contains this architecture diagram.

This architecture supports disaster recovery with Azure services but does not cover more advanced scale-out scenarios involving multiple masters or high availability (HA) with no downtime. For general insights about the various Azure components, including a step-by-step tutorial about building out a CI/CD pipeline on Azure, see Jenkins on Azure.

The focus of this document is on the core Azure operations needed to support Jenkins, including the use of Azure Storage to maintain build artifacts, the security items needed for SSO, other services that can be integrated, and scalability for the pipeline. The architecture is designed to work with an existing source control repository. For example, a common scenario is to start Jenkins jobs based on GitHub commits.

Architecture

The architecture consists of the following components:

- **Resource group.** A resource group is used to group Azure assets so they can be managed by lifetime, owner, and other criteria. Use resource groups to deploy and monitor Azure assets as a group and track billing costs by resource group. You can also delete resources as a set, which is very useful for test deployments.
- **Jenkins server**. A virtual machine is deployed to run Jenkins as an automation server and serve as Jenkins Master. This reference architecture uses the solution template for Jenkins on Azure, installed on a Linux (Ubuntu 16.04 LTS) virtual machine on Azure. Other Jenkins offerings are available in the Azure Marketplace.

> **NOTE**
>
> Nginx is installed on the VM to act as a reverse proxy to Jenkins. You can configure Nginx to enable SSL for the Jenkins server.

- **Virtual network**. A virtual network connects Azure resources to each other and provides logical isolation. In this architecture, the Jenkins server runs in a virtual network.
- **Subnets**. The Jenkins server is isolated in a subnet to make it easier to manage and segregate network traffic without impacting performance.
- **NSGs**. Use network security groups (NSGs) to restrict network traffic from the Internet to the subnet of a virtual network.
- **Managed disks**. A managed disk is a persistent virtual hard disk (VHD) used for application storage and also to maintain the state of the Jenkins server and provide disaster recovery. Data disks are stored in Azure Storage. For high performance, premium storage is recommended.
- **Azure Blob Storage**. The Windows Azure Storage plugin uses Azure Blob Storage to store the build artifacts that are created and shared with other Jenkins builds.
- **Azure Active Directory (Azure AD)**. Azure AD supports user authentication, allowing you to set up SSO. Azure AD service principals define the policy and permissions for each role authorization in the workflow, using role-based access control (RBAC). Each service principal is associated with a Jenkins job.
- **Azure Key Vault.** To manage secrets and cryptographic keys used to provision Azure resources when secrets are required, this architecture uses Key Vault. For added help storing secrets associated with the application in the pipeline, see also the Azure Credentials plugin for Jenkins.
- **Azure monitoring services**. This service monitors the Azure virtual machine hosting Jenkins. This deployment monitors the virtual machine status and CPU utilization and sends alerts.

Recommendations

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

Azure AD

The Azure AD tenant for your Azure subscription is used to enable SSO for Jenkins users and set up service principals that enable Jenkins jobs to access Azure resources.

SSO authentication and authorization are implemented by the Azure AD plugin installed on the Jenkins server. SSO allows you to authenticate using your organization credentials from Azure AD when logging on to the Jenkins server. When configuring the Azure AD plugin, you can specify the level of a user's authorized access to the Jenkins server.

To provide Jenkins jobs with access to Azure resources, an Azure AD administrator creates service principals. These grant applications—in this case, the Jenkins jobs—authenticated, authorized access to Azure resources.

RBAC further defines and controls access to Azure resources for users or service principals through their assigned role. Both built-in and custom roles are supported. Roles also help secure the pipeline and ensure that a user's or agent's responsibilities are assigned and authorized correctly. In addition, RBAC can be set up to limit access to Azure assets. For example, a user can be limited to working with only the assets in a particular resource group.

Storage

Use the Jenkins Windows Azure Storage plugin, which is installed from the Azure Marketplace, to store build

artifacts that can be shared with other builds and tests. An Azure Storage account must be configured before this plugin can be used by the Jenkins jobs.

Jenkins Azure plugins

The solution template for Jenkins on Azure installs several Azure plugins. The Azure DevOps Team builds and maintains the solution template and the following plugins, which work with other Jenkins offerings in Azure Marketplace as well as any Jenkins master set up on premises:

- Azure AD plugin allows the Jenkins server to support SSO for users based on Azure AD.
- Azure VM Agents plugin uses an Azure Resource Manager template to create Jenkins agents in Azure virtual machines.
- Azure Credentials plugin allows you to store Azure service principal credentials in Jenkins.
- Windows Azure Storage plugin uploads build artifacts to, or downloads build dependencies from, Azure Blob storage.

We also recommend reviewing the growing list of all available Azure plugins that work with Azure resources. To see all the latest list, visit Jenkins Plugin Index and search for Azure. For example, the following plugins are available for deployment:

- Azure Container Agents helps you to run a container as an agent in Jenkins.
- Kubernetes Continuous Deploy deploys resource configurations to a Kubernetes cluster.
- Azure Container Service deploys configurations to Azure Container Service with Kubernetes, DC/OS with Marathon, or Docker Swarm.
- Azure Functions deploys your project to Azure Function.
- Azure App Service deploys to Azure App Service.

Scalability considerations

Jenkins can scale to support very large workloads. For elastic builds, do not run builds on the Jenkins master server. Instead, offload build tasks to Jenkins agents, which can be elastically scaled in and out as need. Consider two options for scaling agents:

- Use the Azure VM Agents plugin to create Jenkins agents that run in Azure VMs. This plugin enables elastic scale-out for agents and can use distinct types of virtual machines. You can select a different base image from Azure Marketplace or use a custom image. For details about how the Jenkins agents scale, see Architecting for Scale in the Jenkins documentation.
- Use the Azure Container Agents plugin to run a container as an agent in either Azure Container Service with Kubernetes, or Azure Container Instances.

Virtual machines generally cost more to scale than containers. To use containers for scaling, however, your build process must run with containers.

Also, use Azure Storage to share build artifacts that may be used in the next stage of the pipeline by other build agents.

Scaling the Jenkins server

You can scale the Jenkins server VM up or down by changing the VM size. The solution template for Jenkins on Azure specifies the DS2 v2 size (with two CPUs, 7 GB) by default. This size handles a small to medium team workload. Change the VM size by choosing a different option when building out the server.

Selecting the correct server size depends on the size of the expected workload. The Jenkins community maintains a

selection guide to help identify the configuration that best meets your requirements. Azure offers many sizes for Linux VMs to meet any requirements. For more information about scaling the Jenkins master, refer to the Jenkins community of best practices, which also includes details about scaling Jenkins master.

Availability considerations

Availability in the context of a Jenkins server means being able to recover any state information associated with your workflow, such as test results, libraries you have created, or other artifacts. Critical workflow state or artifacts must be maintained to recover the workflow if the Jenkins server goes down. To assess your availability requirements, consider two common metrics:

- Recovery Time Objective (RTO) specifies how long you can go without Jenkins.
- Recovery Point Objective (RPO) indicates how much data you can afford to lose if a disruption in service affects Jenkins.

In practice, RTO and RPO imply redundancy and backup. Availability is not a question of hardware recovery—that is part of Azure—but rather ensuring you maintain the state of your Jenkins server. Microsoft offers a service level agreement (SLA) for single VM instances. If this SLA doesn't meet your uptime requirements, make sure you have a plan for disaster recovery, or consider using a multi-master Jenkins server deployment (not covered in this document).

Consider using the disaster recovery scripts in step 7 of the deployment to create an Azure Storage account with managed disks to store the Jenkins server state. If Jenkins goes down, it can be restored to the state stored in this separate storage account.

Security considerations

Use the following approaches to help lock down security on a basic Jenkins server, since in its basic state, it is not secure.

- Set up a secure way to log into the Jenkins server. This architecture uses HTTP and has a public IP, but HTTP is not secure by default. Consider setting up HTTPS on the Nginx server being used for a secure logon.

  > **NOTE**
  > When adding SSL to your server, create an NSG rule for the Jenkins subnet to open port 443. For more information, see How to open ports to a virtual machine with the Azure portal.

- Ensure that the Jenkins configuration prevents cross site request forgery (Manage Jenkins > Configure Global Security). This is the default for Microsoft Jenkins Server.
- Configure read-only access to the Jenkins dashboard by using the Matrix Authorization Strategy Plugin.
- Install the Azure Credentials plugin to use Key Vault to handle secrets for the Azure assets, the agents in the pipeline, and third-party components.
- Use RBAC to restrict the access of the service principal to the minimum required to run the jobs. This helps limit the scope of damage from a rogue job.

Jenkins jobs often require secrets to access Azure services that require authorization, such as Azure Container Service. Use Key Vault along with the Azure Credential plugin to manage these secrets securely. Use Key Vault to store service principal credentials, passwords, tokens, and other secrets.

To get a central view of the security state of your Azure resources, use Azure Security Center. Security Center monitors potential security issues and provides a comprehensive picture of the security health of your deployment.

Security Center is configured per Azure subscription. Enable security data collection as described in the Azure Security Center quick start guide. When data collection is enabled, Security Center automatically scans any virtual machines created under that subscription.

The Jenkins server has its own user management system, and the Jenkins community provides best practices for securing a Jenkins instance on Azure. The solution template for Jenkins on Azure implements these best practices.

Manageability considerations

Use resource groups to organize the Azure resources that are deployed. Deploy production environments and development/test environments in separate resource groups, so that you can monitor each environment's resources and roll up billing costs by resource group. You can also delete resources as a set, which is very useful for test deployments.

Azure provides several features for monitoring and diagnostics of the overall infrastructure. To monitor CPU usage, this architecture deploys Azure Monitor. For example, you can use Azure Monitor to monitor CPU utilization, and send a notification if CPU usage exceeds 80 percent. (High CPU usage indicates that you might want to scale up the Jenkins server VM.) You can also notify a designated user if the VM fails or becomes unavailable.

Communities

Communities can answer questions and help you set up a successful deployment. Consider the following:

- Jenkins Community Blog
- Azure Forum
- Stack Overflow Jenkins

For more best practices from the Jenkins community, visit Jenkins best practices.

Deploy the solution

To deploy this architecture, follow the steps below to install the solution template for Jenkins on Azure, then install the scripts that set up monitoring and disaster recovery in the steps below.

Prerequisites

- This reference architecture requires an Azure subscription.
- To create an Azure service principal, you must have admin rights to the Azure AD tenant that is associated with the deployed Jenkins server.
- These instructions assume that the Jenkins administrator is also an Azure user with at least Contributor privileges.

Step 1: Deploy the Jenkins server

1. Open the Azure Marketplace image for Jenkins in your web browser and select **GET IT NOW** from the left side of the page.
2. Review the pricing details and select **Continue**, then select **Create** to configure the Jenkins server in the Azure portal.

For detailed instructions, see Create a Jenkins server on an Azure Linux VM from the Azure portal. For this reference architecture, it is sufficient to get the server up and running with the admin logon. Then you can provision it to use various other services.

Step 2: Set up SSO

The step is run by the Jenkins administrator, who must also have a user account in the subscription's Azure AD

directory and must be assigned the Contributor role.

Use the Azure AD Plugin from the Jenkins Update Center in the Jenkins server and follow the instructions to set up SSO.

Step 3: Provision Jenkins server with Azure VM Agent plugin

The step is run by the Jenkins administrator to set up the Azure VM Agent plugin, which is already installed.

Follow these steps to configure the plugin. For a tutorial about setting up service principals for the plugin, see Scale your Jenkins deployments to meet demand with Azure VM agents.

Step 4: Provision Jenkins server with Azure Storage

The step is run by the Jenkins administrator, who sets up the Windows Azure Storage Plugin, which is already installed.

Follow these steps to configure the plugin.

Step 5: Provision Jenkins server with Azure Credential plugin

The step is run by the Jenkins administrator to set up the Azure Credential plugin, which is already installed.

Follow these steps to configure the plugin.

Step 6: Provision Jenkins server for monitoring by the Azure Monitor Service

To set up monitoring for your Jenkins server, follow the instructions in Create metric alerts in Azure Monitor for Azure services.

Step 7: Provision Jenkins server with Managed Disks for disaster recovery

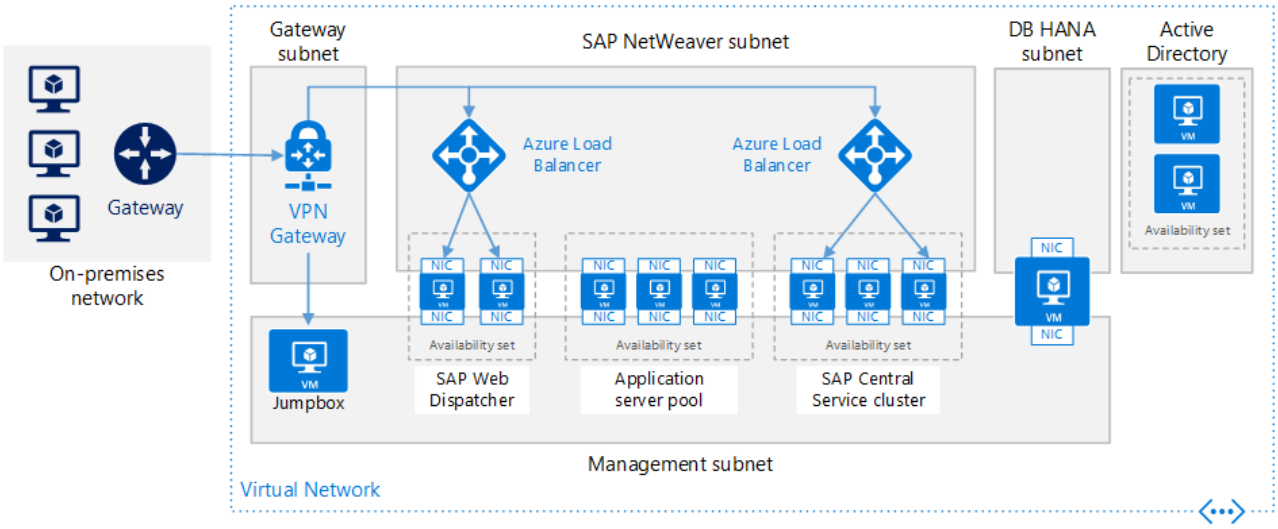
The Microsoft Jenkins product group has created disaster recovery scripts that build a managed disk used to save the Jenkins state. If the server goes down, it can be restored to its latest state.

Download and run the disaster recovery scripts from GitHub.

Deploy SAP NetWeaver and SAP HANA on Azure

4/11/2018 • 11 min to read • Edit Online

This reference architecture shows a set of proven practices for running SAP HANA in a high availability environment on Azure. **Deploy this solution**.



Download a Visio file of this architecture.

> **NOTE**
> Deploying this reference architecture requires appropriate licensing of SAP products and other non-Microsoft technologies. For information about the partnership between Microsoft and SAP, see SAP HANA on Azure.

Architecture

The architecture consists of the following components.

- **Virtual network (VNet)**. A VNet is a representation of a logically isolated network in Azure. All of the VMs in this reference architecture are deployed to the same VNet. The VNet is further subdivided into subnets. Create a separate subnet for each tier, including application (SAP NetWeaver), database (SAP HANA), management (the jumpbox), and Active Directory.
- **Virtual machines (VMs)**. The VMs for this architecture are grouped into several distinct tiers.
  - **SAP NetWeaver**. Includes SAP ASCS, SAP Web Dispatcher, and the SAP application servers.
  - **SAP HANA**. This reference architecture uses SAP HANA for the database tier, running on a single GS5 instance. SAP HANA is certified for production OLAP workloads on GS5 or SAP HANA on Azure Large Instances. This reference architecture is for Azure virtual machines in the G-series and M-series. For information about SAP HANA on Azure Large Instances, see SAP HANA (large instances) overview and architecture on Azure.
  - **Jumpbox**. Also called a bastion host. This is a secure VM on the network that administrators use to connect to the other VMs.
  - **Windows Server Active Directory (AD) domain controllers.** The domain controllers are used to configure the Windows Server Failover Cluster (see below).

- **Availability sets**. Place the VMs for the SAP Web Dispatcher, SAP application server, and SAP ACSC roles into separate availability sets, and provision at least two VMs for each role. This makes the VMs eligible for a higher service level agreement (SLA).
- **NICs.** The VMs that run SAP NetWeaver and SAP HANA require two network interfaces (NICs). Each NIC is assigned to a different subnet, to segregate different types of traffic. See Recommendations below for more information.
- **Windows Server Failover Cluster**. The VMs that run SAP ACSC are configured as a failover cluster for high availability. To support the failover cluster, SIOS DataKeeper Cluster Edition performs the cluster shared volume (CSV) function by replicating independent disks owned by the cluster nodes. For details, see Running SAP applications on the Microsoft platform.
- **Load balancers.** Two Azure Load Balancer instances are used. The first, shown on the left of the diagram, distributes traffic to the SAP Web Dispatcher VMs. This configuration implements the parallel web dispatcher option described in High Availability of the SAP Web Dispatcher. The second load balancer, shown on the right, enables failover in the Windows Server Failover Cluster, by directing incoming connections to the active/healthy node.
- **VPN Gateway.** The VPN Gateway extends your on-premises network to the Azure VNet. You can also use ExpressRoute, which uses a dedicated private connection that does not go over the public Internet. The example solution does not deploy the gateway. For more information, see Connect an on-premises network to Azure.

Recommendations

Your requirements might differ from the architecture described here. Use these recommendations as a starting point.

Load balancers

SAP Web Dispatcher handles load balancing of HTTP(S) traffic to dual-stack servers (ABAP and Java). SAP has advocated single-stack application servers for years, so very few applications run on a dual-stack deployment model nowadays. The Azure load balancer shown in the architecture diagram implements the high availability cluster for the SAP Web Dispatcher.

Load balancing of traffic to the application servers is handled within SAP. For traffic from SAPGUI clients connecting to a SAP server via DIAG and Remote Function Calls (RFC), the SCS message server balances the load by creating SAP App Server Logon Groups.

SMLG is an SAP ABAP transaction used to manage the logon load balancing capability of SAP Central Services. The backend pool of the logon group has more than one ABAP application server. Clients accessing ASCS cluster services connect to the Azure load balancer through a frontend IP address. The ASCS cluster virtual network name also has an IP address. Optionally, this address can be associated with an additional IP address on the Azure load balancer, so that the cluster can be managed remotely.

NICs

SAP landscape management functions require segregation of server traffic on different NICs. For example, business data should be separated from administrative traffic and backup traffic. Assigning multiple NICs to different subnets enables this data segregation. For more information, see "Network" in Building High Availability for SAP NetWeaver and SAP HANA (PDF).

Assign the administration NIC to the management subnet, and assign the data communication NIC to a separate subnet. For configuration details, see Create and manage a Windows virtual machine that has multiple NICs.

Azure Storage

With all database server VMs, we recommend using Azure Premium Storage for consistent read/write latency. For

SAP application servers, including the (A)SCS virtual machines, you can use Azure Standard Storage, because application execution takes place in memory and uses disks for logging only.

For best reliability, we recommend using Azure Managed Disks. Managed disks ensure that the disks for VMs within an availability set are isolated to avoid single points of failure.

> **NOTE**
> Currently the Resource Manager template for this reference architecture does not use managed disks. We are planning to update the template to use managed disks.

To achieve high IOPS and disk bandwidth throughput, the common practices in storage volume performance optimization apply to Azure storage layout. For example, striping multiple disks together to create a larger disk volume improves IO performance. Enabling the read cache on storage content that changes infrequently enhances the speed of data retrieval. For details about performance requirements, see SAP note 1943937 - Hardware Configuration Check Tool.

For the backup data store, we recommend using the cool storage tier of Azure Blob storage. The cool storage tier is a cost-effective way to store data that is less frequently accessed and long-lived.

Scalability considerations

At the SAP application layer, Azure offers a wide range of virtual machine sizes for scaling up. For an inclusive list, see SAP note 1928533 - SAP Applications on Azure: Supported Products and Azure VM Types. Scale out by adding more VMs to the availability set.

For SAP HANA on Azure virtual machines with both OLTP and OLAP SAP applications, the SAP-certified virtual machine size is GS5 with a single VM instance. For larger workloads, Microsoft also offers Azure Large Instances for SAP HANA on physical servers co-located in a Microsoft Azure certified datacenter, which provides up to 4 TB of memory capacity for a single instance at this time. Multi-node configuration is also possible with a total memory capacity of up to 32 TB.

Availability considerations

In this distributed installation of the SAP application on a centralized database, the base installation is replicated to achieve high availability. For each layer of the architecture, the high availability design varies:

- **Web Dispatcher.** High availability is achieved with redundant SAP Web Dispatcher instances with SAP application traffic. See SAP Web Dispatcher in the SAP Documentation.
- **ASCS.** For high availability of ASCS on Azure Windows virtual machines, Windows Sever Failover Clustering is used with SIOS DataKeeper to implement the cluster shared volume. For implementation details, see Clustering SAP ASCS on Azure.
- **Application servers.** High availability is achieved by load balancing traffic within a pool of application servers.
- **Database tier.** This reference architecture deploys a single SAP HANA database instance. For high availability, deploy more than one instance and use HANA System Replication (HSR) to implement manual failover. To enable automatic failover, an HA extension for the specific Linux distribution is required.

Disaster recovery considerations

Each tier uses a different strategy to provide disaster recovery (DR) protection.

- **Application servers.** SAP application servers don't contain business data. On Azure, a simple DR strategy is to create SAP application servers in another region. Upon any configuration changes or kernel updates

on the primary application server, the same changes must be copied to VMs in the DR region. For example, the kernel executables copied to the DR VMs.

- **SAP Central Services.** This component of the SAP application stack also doesn't persist business data. You can build a VM in the DR region to run the SCS role. The only content from the primary SCS node to synchronize is the **/sapmnt** share content. Also, if configuration changes or kernel updates take place on the primary SCS servers, they must be repeated on the DR SCS. To synchronize the two servers, simply use a regularly scheduled copy job to copy **/sapmnt** to the DR side. For details about the build, copy, and test failover process, download SAP NetWeaver: Building a Hyper-V and Microsoft Azure–based Disaster Recovery Solution, and refer to "4.3. SAP SPOF layer (ASCS)."
- **Database tier.** Use HANA-supported replication solutions such as HSR or Storage Replication.

Manageability considerations

SAP HANA has a backup feature that uses the underlying Azure infrastructure. To back up the SAP HANA database running on Azure virtual machines, both the SAP HANA snapshot and Azure storage snapshot are used to ensure the consistency of backup files. For details, see Backup guide for SAP HANA on Azure Virtual Machines and the Azure Backup service FAQ.

Azure provides several functions for monitoring and diagnostics of the overall infrastructure. Also, enhanced monitoring of Azure virtual machines (Linux or Windows) is handled by Azure Operations Management Suite (OMS).

To provide SAP-based monitoring of resources and service performance of the SAP infrastructure, use the Azure SAP Enhanced Monitoring extension. This extension feeds Azure monitoring statistics into the SAP application for operating system monitoring and DBA Cockpit functions.

Security considerations

SAP has its own Users Management Engine (UME) to control role-based access and authorization within the SAP application. For details, see SAP HANA Security - An Overview. (A SAP Service Marketplace account is required for access.)

For infrastructure security, data is safeguarded in transit and at rest. The “Security considerations” section of the SAP NetWeaver on Azure Virtual Machines (VMs) – Planning and Implementation Guide begins to address network security. The guide also specifies the network ports you must open on the firewalls to allow application communication.

To encrypt Windows and Linux IaaS virtual machine disks, you can use Azure Disk Encryption. Azure Disk Encryption uses the BitLocker feature of Windows and the DM-Crypt feature of Linux to provide volume encryption for the operating system and the data disks. The solution also works with Azure Key Vault to help you control and manage the disk-encryption keys and secrets in your Key Vault subscription. Data on the virtual machine disks are encrypted at rest in your Azure storage.

For SAP HANA data-at-rest encryption, we recommend using the SAP HANA native encryption technology.

> **NOTE**
> Don't use the HANA data-at-rest encryption with Azure disk encryption on the same server.

Consider using network security groups (NSGs) to restrict traffic between the various subnets in the VNet.

Deploy the solution

The deployment scripts for this reference architecture are available on GitHub.

Prerequisites

- You must have access to the SAP Software Download Center to complete the installation.
- Install the latest version of Azure PowerShell.
- This deployment requires 51 cores. Before deploying, verify that your subscription has sufficient quota for VM cores. If not, use the Azure portal to submit a support request for more quota.
- This deployment uses a GS-series VM. Check the availability of the GS series per region here.
- To estimate the cost of this deployment, see the Azure Pricing Calculator.

This reference architecture deploys the following VMs:

RESOURCE NAME	VM SIZE	PURPOSE
`ra-sapApps-scs-vm1` ... `ra-sapApps-scs-vmN`	DS11v2	SAP Central Services
`ra-sapApps-vm1` ... `ra-sapApps-vmN`	DS11v2	SAP NetWeaver application
`ra-sap-wdp-vm1` ... `ra-sap-wdp-vmN`	DS11v2	SAP Web Dispatcher
`ra-sap-data-vm1`	GS5	SAP HANA database instance
`ra-sap-jumpbox-vm1`	DS1V2	Jumpbox

A single SAP HANA instance is deployed. For the application VMs, the number of instances to deploy is specified in the template parameters.

Deploy SAP infrastructure

You can deploy this architecture incrementally or all at once. The first time, we recommend an incremental deployment, so that you can see what each deployment step does. Specify the increment using one of the following *mode* parameters

MODE	WHAT IT DOES
infrastructure	Deploys the network infrastructure in Azure.
workload	Deploys the SAP servers to the network.
all	Deploys all the preceding deployments.

To deploy the solution, perform the following steps:

1. Download or clone the GitHub repo to your local computer.
2. Open a PowerShell window and navigate to the `/sap/sap-hana/` folder.
3. Run the following PowerShell cmdlet. For `<subscription id>`, use your Azure subscription ID. For `<location>`, specify an Azure region, such as `eastus` or `westus`. For `<mode>`, specify one of the modes listed above.

```
.\Deploy-ReferenceArchitecture -SubscriptionId <subscription id> -Location <location> -
ResourceGroupName <resource group> <mode>
```

4. When prompted, log on to your Azure account.

The deployment scripts can take up to several hours to complete, depending on the mode you selected.

> **WARNING**
> The parameter files include a hard-coded password (`AweS0me@PW`) in various places. Change these values before you deploy.

Configure SAP applications and database

After deploying the SAP infrastructure, install and configure your SAP applications and HANA database on the virtual machines as follows.

> **NOTE**
> For SAP installation instructions, you must have a SAP Support Portal username and password to download the SAP installation guides.

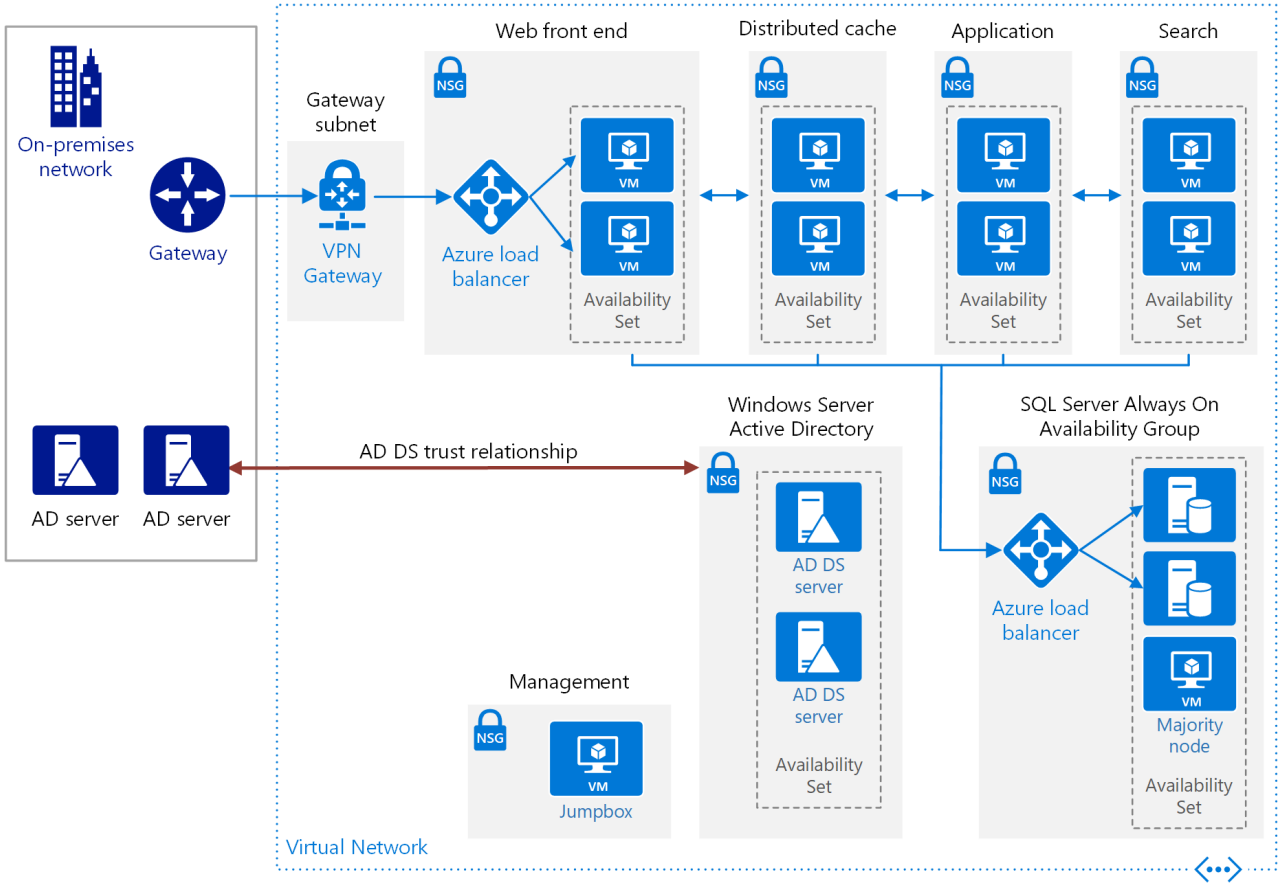
1. Log into the jumpbox (`ra-sap-jumpbox-vm1`). You will use the jumpbox to log into the other VMs.
2. For each VM named `ra-sap-wdp-vm1` ... `ra-sap-wdp-vmN` , log into the VM, and install and configure the SAP Web Dispatcher instance using the steps described in the Web Dispatcher Installation wiki.
3. Log into the VM named `ra-sap-data-vm1` . Install and configure the SAP Hana Database instance using the SAP HANA Server Installation and Update Guide.
4. For each VM named `ra-sapApps-scs-vm1` ... `ra-sapApps-scs-vmN` , log into the VM, and install and configure the SAP Central Services (SCS) using the SAP installation guides.
5. For each VM named `ra-sapApps-vm1` ... `ra-sapApps-vmN` , log into the VM, and install and configure the SAP NetWeaver application using the SAP installation guides.

Contributors to this reference architecture — Rick Rainey, Ross Sponholtz, Ben Trinh

Run a high availability SharePoint Server 2016 farm in Azure

4/11/2018 • 14 min to read • Edit Online

This reference architecture shows a set of proven practices for setting up a high availability SharePoint Server 2016 farm on Azure, using MinRole topology and SQL Server Always On availability groups. The SharePoint farm is deployed in a secured virtual network with no Internet-facing endpoint or presence. **Deploy this solution**.



Download a Visio file of this architecture.

Architecture

This architecture builds on the one shown in Run Windows VMs for an N-tier application. It deploys a SharePoint Server 2016 farm with high availability inside an Azure virtual network (VNet). This architecture is suitable for a test or production environment, a SharePoint hybrid infrastructure with Office 365, or as the basis for a disaster recovery scenario.

The architecture consists of the following components:

- **Resource groups**. A resource group is a container that holds related Azure resources. One resource group is used for the SharePoint servers, and another resource group is used for infrastructure components that are independent of VMs, such as the virtual network and load balancers.
- **Virtual network (VNet)**. The VMs are deployed in a VNet with a unique intranet address space. The VNet is further subdivided into subnets.
- **Virtual machines (VMs)**. The VMs are deployed into the VNet, and private static IP addresses are

assigned to all of the VMs. Static IP addresses are recommended for the VMs running SQL Server and SharePoint Server 2016, to avoid issues with IP address caching and changes of addresses after a restart.

- **Availability sets**. Place the VMs for each SharePoint role into separate availability sets, and provision at least two virtual machines (VMs) for each role. This makes the VMs eligible for a higher service level agreement (SLA).
- **Internal load balancer**. The load balancer distributes SharePoint request traffic from the on-premises network to the front-end web servers of the SharePoint farm.
- **Network security groups (NSGs)**. For each subnet that contains virtual machines, a network security group is created. Use NSGs to restrict network traffic within the VNet, in order to isolate subnets.
- **Gateway**. The gateway provides a connection between your on-premises network and the Azure virtual network. Your connection can use ExpressRoute or site-to-site VPN. For more information, see Connect an on-premises network to Azure.
- **Windows Server Active Directory (AD) domain controllers**. Because SharePoint Server 2016 does not support using Azure Active Directory Domain Services, you must deploy Windows Server AD domain controllers. These domain controllers run in the Azure VNet and have a trust relationship with the on-premises Windows Server AD forest. Client web requests for SharePoint farm resources are authenticated in the VNet rather than sending that authentication traffic across the gateway connection to the on-premises network. In DNS, intranet A or CNAME records are created so that intranet users can resolve the name of the SharePoint farm to the private IP address of the internal load balancer.
- **SQL Server Always On Availability Group**. For high availability of the SQL Server database, we recommend SQL Server Always On Availability Groups. Two virtual machines are used for SQL Server. One contains the primary database replica and the other contains the secondary replica.
- **Majority node VM**. This VM allows the failover cluster to establish quorum. For more information, see Understanding Quorum Configurations in a Failover Cluster.
- **SharePoint servers**. The SharePoint servers perform the web front-end, caching, application, and search roles.
- **Jumpbox**. Also called a bastion host. This is a secure VM on the network that administrators use to connect to the other VMs. The jumpbox has an NSG that allows remote traffic only from public IP addresses on a safe list. The NSG should permit remote desktop (RDP) traffic.

Recommendations

Your requirements might differ from the architecture described here. Use these recommendations as a starting point.

Resource group recommendations

We recommend separating resource groups according to the server role, and having a separate resource group for infrastructure components that are global resources. In this architecture, the SharePoint resources form one group, while the SQL Server and other utility assets form another.

Virtual network and subnet recommendations

Use one subnet for each SharePoint role, plus a subnet for the gateway and one for the jumpbox.

The gateway subnet must be named *GatewaySubnet*. Assign the gateway subnet address space from the last part of the virtual network address space. For more information, see Connect an on-premises network to Azure using a VPN gateway.

VM recommendations

Based on Standard DSv2 virtual machine sizes, this architecture requires a minimum of 38 cores:

- 8 SharePoint servers on Standard_DS3_v2 (4 cores each) = 32 cores
- 2 Active Directory domain controllers on Standard_DS1_v2 (1 core each) = 2 cores
- 2 SQL Server VMs on Standard_DS1_v2 = 2 cores
- 1 majority node on Standard_DS1_v2 = 1 core
- 1 management server on Standard_DS1_v2 = 1 core

The total number of cores will depend on the VM sizes that you select. For more information, see SharePoint Server recommendations below.

Make sure your Azure subscription has enough VM core quota for the deployment, or the deployment will fail. See Azure subscription and service limits, quotas, and constraints.

NSG recommendations

We recommend having one NSG for each subnet that contains VMs, to enable subnet isolation. If you want to configure subnet isolation, add NSG rules that define the allowed or denied inbound or outbound traffic for each subnet. For more information, see Filter network traffic with network security groups.

Do not assign an NSG to the gateway subnet, or the gateway will stop functioning.

Storage recommendations

The storage configuration of the VMs in the farm should match the appropriate best practices used for on-premises deployments. SharePoint servers should have a separate disk for logs. SharePoint servers hosting search index roles require additional disk space for the search index to be stored. For SQL Server, the standard practice is to separate data and logs. Add more disks for database backup storage, and use a separate disk for tempdb.

For best reliability, we recommend using Azure Managed Disks. Managed disks ensure that the disks for VMs within an availability set are isolated to avoid single points of failure.

> **NOTE**
> Currently the Resource Manager template for this reference architecture does not use managed disks. We are planning to update the template to use managed disks.

Use Premium managed disks for all SharePoint and SQL Server VMs. You can use Standard managed disks for the majority node server, the domain controllers, and the management server.

SharePoint Server recommendations

Before configuring the SharePoint farm, make sure you have one Windows Server Active Directory service account per service. For this architecture, you need at a minimum the following domain-level accounts to isolate privilege per role:

- SQL Server Service account
- Setup User account
- Server Farm account
- Search Service account
- Content Access account
- Web App Pool accounts
- Service App Pool accounts
- Cache Super User account
- Cache Super Reader account

For all roles except the Search Indexer, we recommended using the Standard_DS3_v2 VM size. The Search Indexer

should be at least the Standard_DS13_v2 size.

> **NOTE**
> The Resource Manager template for this reference architecture uses the smaller DS3 size for the Search Indexer, for purposes of testing the deployment. For a production deployment, use the DS13 size or larger.

For production workloads, see Hardware and software requirements for SharePoint Server 2016.

To meet the support requirement for disk throughput of 200 MB per second minimum, make sure to plan the Search architecture. See Plan enterprise search architecture in SharePoint Server 2013. Also follow the guidelines in Best practices for crawling in SharePoint Server 2016.

In addition, store the search component data on a separate storage volume or partition with high performance. To reduce load and improve throughput, configure the object cache user accounts, which are required in this architecture. Split the Windows Server operating system files, the SharePoint Server 2016 program files, and diagnostics logs across three separate storage volumes or partitions with normal performance.

For more information about these recommendations, see Initial deployment administrative and service accounts in SharePoint Server 2016.

Hybrid workloads

This reference architecture deploys a SharePoint Server 2016 farm that can be used as a SharePoint hybrid environment — that is, extending SharePoint Server 2016 to Office 365 SharePoint Online. If you have Office Online Server, see Office Web Apps and Office Online Server supportability in Azure.

The default service applications in this deployment are designed to support hybrid workloads. All SharePoint Server 2016 and Office 365 hybrid workloads can be deployed to this farm without changes to the SharePoint infrastructure, with one exception: The Cloud Hybrid Search Service Application must not be deployed onto servers hosting an existing search topology. Therefore, one or more search-role-based VMs must be added to the farm to support this hybrid scenario.

SQL Server Always On Availability Groups

This architecture uses SQL Server virtual machines because SharePoint Server 2016 cannot use Azure SQL Database. To support high availability in SQL Server, we recommend using Always On Availability Groups, which specify a set of databases that fail over together, making them highly-available and recoverable. In this reference architecture, the databases are created during deployment, but you must manually enable Always On Availability Groups and add the SharePoint databases to an availability group. For more information, see Create the availability group and add the SharePoint databases.

We also recommend adding a listener IP address to the cluster, which is the private IP address of the internal load balancer for the SQL Server virtual machines.

For recommended VM sizes and other performance recommendations for SQL Server running in Azure, see Performance best practices for SQL Server in Azure Virtual Machines. Also follow the recommendations in Best practices for SQL Server in a SharePoint Server 2016 farm.

We recommend that the majority node server reside on a separate computer from the replication partners. The server enables the secondary replication partner server in a high-safety mode session to recognize whether to initiate an automatic failover. Unlike the two partners, the majority node server doesn't serve the database but rather supports automatic failover.

Scalability considerations

To scale up the existing servers, simply change the VM size.

With the MinRoles capability in SharePoint Server 2016, you can scale out servers based on the server's role and also remove servers from a role. When you add servers to a role, you can specify any of the single roles or one of the combined roles. If you add servers to the Search role, however, you must also reconfigure the search topology using PowerShell. You can also convert roles using MinRoles. For more information, see Managing a MinRole Server Farm in SharePoint Server 2016.

Note that SharePoint Server 2016 doesn't support using virtual machine scale sets for auto-scaling.

Availability considerations

This reference architecture supports high availability within an Azure region, because each role has at least two VMs deployed in an availability set.

To protect against a regional failure, create a separate disaster recovery farm in a different Azure region. Your recovery time objectives (RTOs) and recovery point objectives (RPOs) will determine the setup requirements. For details, see Choose a disaster recovery strategy for SharePoint 2016. The secondary region should be a *paired region* with the primary region. In the event of a broad outage, recovery of one region is prioritized out of every pair. For more information, see Business continuity and disaster recovery (BCDR): Azure Paired Regions.

Manageability considerations

To operate and maintain servers, server farms, and sites, follow the recommended practices for SharePoint operations. For more information, see Operations for SharePoint Server 2016.

The tasks to consider when managing SQL Server in a SharePoint environment may differ from the ones typically considered for a database application. A best practice is to fully back up all SQL databases weekly with incremental nightly backups. Back up transaction logs every 15 minutes. Another practice is to implement SQL Server maintenance tasks on the databases while disabling the built-in SharePoint ones. For more information, see Storage and SQL Server capacity planning and configuration.

Security considerations

The domain-level service accounts used to run SharePoint Server 2016 require Windows Server AD domain controllers for domain-join and authentication processes. Azure Active Directory Domain Services can't be used for this purpose. To extend the Windows Server AD identity infrastructure already in place in the intranet, this architecture uses two Windows Server AD replica domain controllers of an existing on-premises Windows Server AD forest.

In addition, it's always wise to plan for security hardening. Other recommendations include:

- Add rules to NSGs to isolate subnets and roles.
- Don't assign public IP addresses to VMs.
- For intrusion detection and analysis of payloads, consider using a network virtual appliance in front of the front-end web servers instead of an internal Azure load balancer.
- As an option, use IPsec policies for encryption of cleartext traffic between servers. If you are also doing subnet isolation, update your network security group rules to allow IPsec traffic.
- Install anti-malware agents for the VMs.

Deploy the solution

The deployment scripts for this reference architecture are available on GitHub.

You can deploy this architecture incrementally or all at once. The first time, we recommend an incremental deployment, so that you can see what each deployment does. Specify the increment using one of the following *mode* parameters.

MODE	WHAT IT DOES
onprem	(Optional) Deploys a simulated on-premises network environment, for testing or evaluation. This step does not connect to an actual on-premises network.
infrastructure	Deploys the SharePoint 2016 network infrastructure and jumpbox to Azure.
createvpn	Deploys a virtual network gateway for both the SharePoint and on-premises networks and connects them. Run this step only if you ran the `onprem` step.
workload	Deploys the SharePoint servers to the SharePoint network.
security	Deploys the network security group to the SharePoint network.
all	Deploys all the preceding deployments.

To deploy the architecture incrementally with a simulated on-premises network environment, run the following steps in order:

1. onprem
2. infrastructure
3. createvpn
4. workload
5. security

To deploy the architecture incrementally without a simulated on-premises network environment, run the following steps in order:

1. infrastructure
2. workload
3. security

To deploy everything in one step, use `all`. Note that the entire process may take several hours.

Prerequisites

- Install the latest version of Azure PowerShell.
- Before deploying this reference architecture, verify that your subscription has sufficient quota—at least 38 cores. If you don't have enough, use the Azure portal to submit a support request for more quota.
- To estimate the cost of this deployment, see the Azure Pricing Calculator.

Deploy the reference architecture

1. Download or clone the GitHub repo to your local computer.
2. Open a PowerShell window and navigate to the `/sharepoint/sharepoint-2016` folder.
3. Run the following PowerShell command. For <subscription id>, use your Azure subscription ID. For <location>, specify an Azure region, such as `eastus` or `westus`. For <mode>, specify `onprem`, `infrastructure`, `createvpn`, `workload`, `security`, or `all`.

```
.\Deploy-ReferenceArchitecture.ps1 <subscription id> <location> <mode>
```

4. When prompted, log on to your Azure account. The deployment scripts can take up to several hours to complete, depending on the mode you selected.
5. When the deployment completes, run the scripts to configure SQL Server Always On Availability Groups. See the readme for details.

> **WARNING**
> The parameter files include a hard-coded password (`AweS0me@PW`) in various places. Change these values before you deploy.

Validate the deployment

After you deploy this reference architecture, the following resource groups are listed under the Subscription that you used:

RESOURCE GROUP	PURPOSE
ra-onprem-sp2016-rg	Simulated on-premises network with Active Directory, federated with the SharePoint 2016 network
ra-sp2016-network-rg	Infrastructure to support SharePoint deployment
ra-sp2016-workload-rg	SharePoint and supporting resources

Validate access to the SharePoint site from the on-premises network

1. In the Azure portal, under **Resource groups**, select the `ra-onprem-sp2016-rg` resource group.
2. In the list of resources, select the VM resource named `ra-adds-user-vm1`.
3. Connect to the VM, as described in Connect to virtual machine. The user name is `\onpremuser`.
4. When the remote connection to the VM is established, open a browser in the VM and navigate to `http://portal.contoso.local`.
5. In the **Windows Security** box, log on to the SharePoint portal using `contoso.local\testuser` for the user name.

This logon tunnels from the Fabrikam.com domain used by the on-premises network to the contoso.local domain used by the SharePoint portal. When the SharePoint site opens, you'll see the root demo site.

Validate jumpbox access to VMs and check configuration settings

1. In Azure portal, under **Resource groups**, select the `ra-sp2016-network-rg` resource group.
2. In the list of resources, select the VM resource named `ra-sp2016-jb-vm1`, which is the jumpbox.
3. Connect to the VM, as described in Connect to virtual machine. The user name is `testuser`.
4. After you log onto the jumpbox, open an RDP session from the jumpbox. Connect to any other VMs in the VNet. The username is `testuser`. You can ignore the warning about the remote computer's security certificate.
5. When the remote connection to the VM opens, review the configuration and make changes using the administrative tools such as Server Manager.

The following table shows the VMs that are deployed.

RESOURCE NAME	PURPOSE	RESOURCE GROUP	VM NAME
Ra-sp2016-ad-vm1	Active Directory + DNS	Ra-sp2016-network-rg	Ad1.contoso.local
Ra-sp2016-ad-vm2	Active Directory + DNS	Ra-sp2016-network-rg	Ad2.contoso.local
Ra-sp2016-fsw-vm1	SharePoint	Ra-sp2016-network-rg	Fsw1.contoso.local
Ra-sp2016-jb-vm1	Jumpbox	Ra-sp2016-network-rg	Jb (use public IP to log on)
Ra-sp2016-sql-vm1	SQL Always On - Failover	Ra-sp2016-network-rg	Sq1.contoso.local
Ra-sp2016-sql-vm2	SQL Always On - Primary	Ra-sp2016-network-rg	Sq2.contoso.local
Ra-sp2016-app-vm1	SharePoint 2016 Application MinRole	Ra-sp2016-workload-rg	App1.contoso.local
Ra-sp2016-app-vm2	SharePoint 2016 Application MinRole	Ra-sp2016-workload-rg	App2.contoso.local
Ra-sp2016-dch-vm1	SharePoint 2016 Distributed Cache MinRole	Ra-sp2016-workload-rg	Dch1.contoso.local
Ra-sp2016-dch-vm2	SharePoint 2016 Distributed Cache MinRole	Ra-sp2016-workload-rg	Dch2.contoso.local
Ra-sp2016-srch-vm1	SharePoint 2016 Search MinRole	Ra-sp2016-workload-rg	Srch1.contoso.local
Ra-sp2016-srch-vm2	SharePoint 2016 Search MinRole	Ra-sp2016-workload-rg	Srch2.contoso.local
Ra-sp2016-wfe-vm1	SharePoint 2016 Web Front End MinRole	Ra-sp2016-workload-rg	Wfe1.contoso.local
Ra-sp2016-wfe-vm2	SharePoint 2016 Web Front End MinRole	Ra-sp2016-workload-rg	Wfe2.contoso.local

Contributors to this reference architecture — Joe Davies, Bob Fox, Neil Hodgkinson, Paul Stork